НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Смолій В.М., Лісовиченко О.І.

НАВЧАЛЬНИЙ ПОСІБНИК

ОСНОВИ ПРОГРАМУВАННЯ

для студентів спеціальності 141 Електроенергетика, електротехніка та електромеханіка освітньої програми Інжиніринг електроенергетичних систем з відновлюваними джерелами

Київ-2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Кафедра інформаційних систем та технологій

Смолій В.М., Лісовиченко О.І.

ОСНОВИ ПРОГРАМУВАННЯ

Рекомендовано до друку Вченою радою Національного університету біоресурсів і природокористування України (протокол № 8 від 28 лютого 2024 р.)

Київ - 2024

УДК 004.42
П79

Основи програмування Навчальний посібник / Смолій В.М., Лісовиченко О.І. – К.: Видавництво, 2024 – 393 с.

Рекомендовано до друку Вченою радою Національного університету біоресурсів і природокористування України (протокол № 8 від 28 лютого 2024 р.)

Рецензенти:

Й.Й. Білинський, д.т.н., професор (Вінницький національний технічний університет, м. Вінниця);

І.М. Бондаренко, д.ф.-м.н., професор (Харківський національний університет радіоелектроніки, м. Харків).

О.Є. Коваленко, д.т.н., професор (Національний університет біоресурсів і природокористування України, м. Київ).

В навчальному посібнику представлені: прості елементи мови, структури даних і вирази, класи пам'яті, логічні вирази, керуючі структури, відношення, логічні операції, умовні вирази, керуючі структури, масиви та покажчики, функції та структури, директиви препроцесора, файли, функції обміну з потоками, керування буферизацією та завдання, індивідуальні варіанти і приклади розв’язання завдань з лабораторних і самостійних робіт. Наведено перелік базової і допоміжної літератури для поглибленого вивчення курсу та онлайн ресурси.

Посібник стане у нагоді для студентів, що вивчають основи програмування.

Призначений для студентів спеціальності 141 «Електроенергетика, електротехніка та електромеханіка» освітньої програми «Інжиніринг електроенергетичних систем з відновлюваними джерелами»

Видання здійснено за авторським редагуванням.

ЗМІСТ

ВСТУП

Навчальний посібник з дисципліни «Основи програмування» спрямований на формування сталої системи знань, вмінь і навичок програмування серед майбутніх фахівців з інжинірингу електроенергетичних систем з відновлюваними джерелами.

Вивчення навчальних матеріалів дисципліни сприяє отриманню майбутніми бакалаврами з інжинірингу електроенергетичних систем з відновлюваними джерелами сучасного рівня грамотності з основ програмування, опанування основ ефективної роботи у середовищі розробки програм; знайомство з методами, принципами та сучасними підходами до програмування та набуття практичних навичок створення програмного забезпечення для вирішення загальних інженерних задач у рамках спеціальності 141 «Електроенергетика, електротехніка та електромеханіка».

У результаті вивчення навчальної дисципліни студент повинен:

знати: особливості застосування сучасного програмного забезпечення з метою розв’язання загальних інженерних задач.

вміти: самостійно вчитися, опановувати нові знання і вдосконалювати навички роботи з сучасним прикладним програмним забезпеченням

Навчальна дисципліна забезпечує формування:

загальних компетентностей:

ЗК01. Здатність до абстрактного мислення, аналізу і синтезу.

фахових компетентностей:

СК19. Здатність виконувати загальні інженерні розрахунки із застосуванням сучасного програмного забезпечення.

У результаті вивчення навчальної дисципліни студент набуде певні програмні результати навчання, а саме:

ПРН06. Застосовувати прикладне програмне забезпечення, мікроконтролери та мікропроцесорну техніку для вирішення практичних проблем у професійній діяльності.

ПРН18. Вміти самостійно вчитися, опановувати нові знання і вдосконалювати навички роботи з сучасним обладнанням, вимірювальною технікою та прикладним програмним забезпеченням.

ПРН26. Знати особливості застосування сучасного програмного забезпечення з метою розв’язання загальних інженерних задач.

Силабус дисципліни наведено у Додатку 1 даного навчального посібника.

РОЗДІЛ І. ПРОСТІ ЕЛЕМЕНТИ МОВИ

1.1 Теоретичні відомості. Прості елементи мови

1.1.1 Історія мови

Мова програмування С++ створена на фірмі Bell Laboratories в 1972р. Денісом Рітчі (*Dennis MacAlistair Ritchie*). Попередником цієї мови є мова АЛГОЛ-60, на основі якої була розроблена мова CPL (мова комбінованого програмування) в 1963р. (Кембриджський і Лондонський університети) потім BCPL (базова мова комбінованого програмування) 1967р. (Кембриджський університет). Метою цих розробок було відродити мову АЛГОЛ, зберегти контакт із ЕОМ. Тобто поруч із можливостями універсальної процедурної мови були додані засоби роботи з апаратною частиною ЕОМ. Тому мова BCPL виявилася громіздким і в 1970р. Кеном Томпсоном (*Kenneth Lane Thompson*) в Bell Laboratories була розроблена мова B, яка була вузько спрямована на розробку системних програм. Історія створення мови наведена на рис. 1.1.

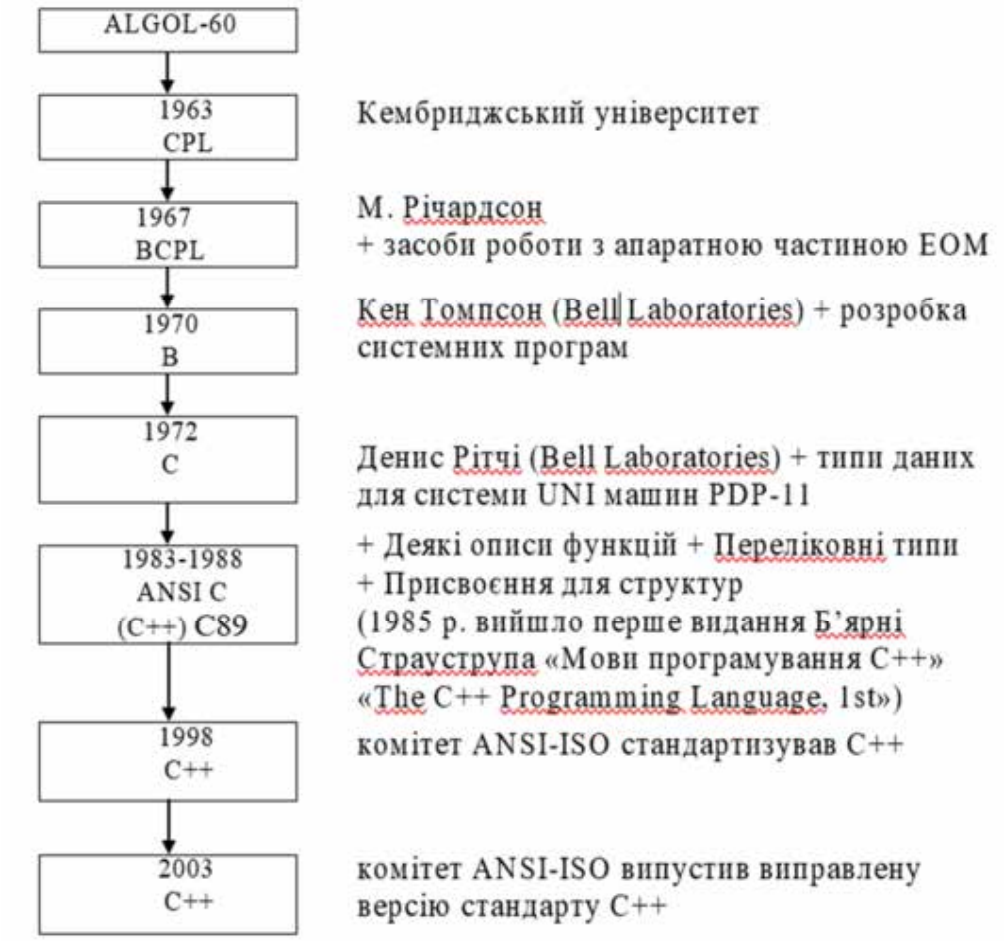


Рис. 1.1 - Історія створення мови С

Ця мова й стала основою для створення мови С++. Основна заслуга Д. Рітчі - це додавання вдалої системи типів даних. Початкове призначення

мови C++ - системне програмування. Сама мова була невід'ємною частиною системи UNI для ЕОМ типу PDP-11.

Так приблизно з 13 тис. рядків цієї системи лише 800 були написані на асемблері, інші наС++. Можна відзначити такі особливості мови:

1. Мова С належить до мов зі слабкою типізацією даних. Набагато ширше, ніж у мові Паскаль, передбачене неявне перетворення типів.

2. Передбачене використання покажчиків, які дають можливість оперувати з адресами й непрямою адресацією.

3. Клас об'єктів мови обмежений. Виключені такі структури даних як логічні, множини. Оскільки мова С тісно пов'язана з операційною системою UNI, то багато можливостей реалізуються функціями операційної системи. Це забезпечило компактність і високу ефективність даної мови.

Мова відзначається внутрішньою єдністю, що властива мовам "для людини" (Паскаль, Лисп). Має невеликий перелік вихідних засобів, з яких можна створювати досить складні конструкції.

Реалізує принципи структурного програмування, зокрема, блокову побудову. Усе це сприяло широкому поширенню мови не тільки для системних цілей, але й для розв'язку широкого класу прикладних завдань. Незважаючи на те, що С пов'язана із системою UNI PDP-11, відзначені позитивні властивості сприяли її широкому застосуванню на інших ЕОМ і операційних системах. Тому говорять про її високу мобільність і доброї переносимості.

Однак перші варіанти мови не були уніфіковані, через що питання перенесення все-таки виникали. Тому в 1983р. інститут американських національних стандартів (ANSI) створив комітет для розробки сучасної машинно незалежної мови С++. І в 1988р. ця робота закінчилася створенням мови "ANSI C". Ця версія широко використовується на персональних ЕОМ, зберігає більшість властивостей вихідної мови, відрізняється: деяким описом функцій, наявністю типу, що перелічується, дозволяє операцію присвоєння для структур (записів), розширеною й уніфікованою бібліотекою функцій.

Можна вважати, що мова С++, з одного боку, є зручної, виразної й гнучкої для програмування широкого класу завдань. З іншого боку - вона в достатній мірі наближена до ЕОМ, дає засоби контролю процесом реалізації програми, але зберігає певну дистанцію, яка дозволяє не враховувати всі особливості архітектури ЕОМ. Можна вважати, що це мова відносно низького рівня.

1985 р. вийшло перше видання Б’ярні Страуструпа «Мови програмування С++» «The C++ Programming Language, 1st»

Говорять, що мова С++ призначена для професіоналів- програмістів, а не для початківців. Це створює певні труднощі при вивченні мови.

На рис.1.2 наведено умовне представлення рівнів розгляду мови програмування С.

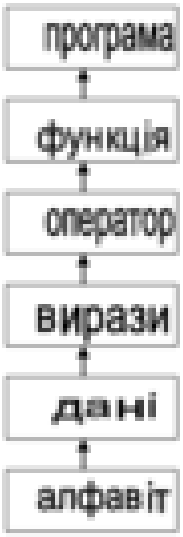


Рис. 1.2 – Умовне представлення рівнів розгляду мови програмування С

Спочатку розглянемо деяку мінімальну підмножину мови, яка дозволить писати прості програми.

1.1.2 Алфавіт

Символи мови складаються із трьох груп: **літер, цифр, спеціальних символів**. Літери – це букви латинського алфавіту (A-Z, a-z), а також кирилиці, які використовуються в коментарях і рядкових константах (А-я).

Спеціальні символи складаються з: 1) знаків арифметичних операцій (+ додавання; - віднімання; * множення; /ділення, остача від ділення x % y цілих чисел); 2) знаків відношень; 3) роздільників; 4) службових слів.

Арифметичні операції у мові С мають вигляд, представлений у табл. 1.1.

Таблиця 1.1. Арифметичні операції у мові С

Операція	Вигляд
Додавання	total = var1 + var2
Віднімання	total = var1 - var2
Множення	total = var1 * var2
Ділення	total = var1 / var2
Оператор + збільшення	total = total + 1
Оператор - зменшення	total = total - 1

(++)(--)variable - префіксний оператор збільшення; **variable(++)(--)** - постфіксний оператор збільшення.

У табл. 1.2 наведено операції і функції мови С.

Таблиця 1.2 Операції і функції мови С

Операція	Функція
%	Взяття по модулю або залишок
~	Додаток; інвертує біти значення
&	Побітове І (також &X – взяти адрес X)
|	Побітове включающе АБО
^	Побітовое виключаюче АБО
<<	Зсув вліво; зсуває біти значення вліво на зазначену кількість розрядів
>>	Зсув вправо; зсуває біти значення вправо на зазначену кількість розрядів

Операція взяття по модулю або залишок повертає залишок цілочисельного розподілу. Операції піднесення до ступеня у мові С не передбачено. Знаки відносин і логічних операцій зведено у табл. 1.3

Таблиця 1.3 Позначення логічних операцій

Позначення	Тип операції
>, <	Більше, менше
==	Дорівнює (2 знака рівності)
!=	Не дорівнює
&&	Логічне І
||	Логічне АБО
!	Логічне НЕ

До роздільників у мові С відносять: , ( ) [ ] { } ' " = : ;. Фігурні дужки мають значення операторних дужок (Begin - End мови Паскаль). Для коментарів використовуються пари знаків /*КОМЕНТАР*/.

Службові слова мови С наведено у таблиці 1.4.

Таблиця 1.4 Службові слова мови С

auto	break	case	char	continue	default	do	double
else	entry	enum	extern	float	for	goto	if
int	long	register	return	short	sizeof	static	struct
switch	typedef	union	unsigned	void	while		

Окрім наведених у таблиці службових слів існує всього 30 службових слів від auto до while.

1.1.3 Константи

Константами називаються перерахування величин у програмі. Розділяють константи таких типів:

1. цілі (243);
2. з плаваючою крапкою (дійсні) (543.8);
3. символьні ('A') - 1 байт;
4. строкові ("A") - 2 байта A + NULL;
5. переліковного типу (ANSI-C).

Цілі константи можуть записуватися в десятковій, вісімковій і шістнадцятковій системах числення.

Десяткова ціла константа подається звичайним чином (зі знаком або без нього): -2561; 458.

Ознакою вісімкової константи є наявність нуля ліворуч: Ø257.

Шістнадцяткова константа визначається двома початковими символами: ØX. Для представлення чисел від 10 до 15 у цій системі використовуються латинські літери: A - 10, B - 11, C - 12, D - 13, E - 14, F - 15.

Тому 3110 в цій системі буде записано ØX1F (1*16+ F=31).

Крім звичайних цілих констант можна використовувати цілі подвійної точності, які в пам'яті займають 2 машинних слова.

До них приєднується літера L: 371L. Діапазон: -2.147.483.648<=2.147.483.647. Константи з плаваючою крапкою (floating point) можуть подаватися у формі з фіксованою крапкою: 561.25, .5, 100.

При цьому в пам'яті розміщуються звичайним способом (2 слова).

Якщо ж константа задається у формі із плаваючою крапкою (експонентна форма)1E-06, 1.8E4, то в деяких трансляторах вона автоматично подається з подвійною точністю, тобто в 4 слова.

Символьні константи (CHAR) набувають значення одного символу й подаються в апострофах 'x', 'a', 'r' і займають 1 байт. Деякі коди команд наведено у табл. 1.5.

Таблиця 1.5 Коди команд

Перехід на наступну строку	\n'
Горизонтальна табуляція	\t\'
Перехід на попередню позицію	\b'
Переведення каретки	\r'
Перехід на наступну сторінку	\f'
Апостроф	\"

Звуковий сигнал	\a'
Вертикальна табуляція	\v'

Недруковані символи, які не мають графічного зображення, подаються умовно двома символами: перший - використовується зворотний слеш, а другий - який-небудь певний символ. Так звана зворотна послідовність перемикання коду (escape sequence або ескейп послідовність).

Строкові константи (строки, string) це послідовність символів, обмежена подвійними лапками: "string". Для переносу на інший рядок використовується зворотній слеш

"морозн
ий\ день" -
12+1 байт

Як і в мові Паскаль, константи можуть задаватися своїм іменем. Але в мові С++ немає спеціального розділу опису констант.

Тому визначення констант реалізується 3-а способами:

1. Процесором і має вигляд:

#define<им'я константи> <літерал або значення> #define<им'я константи ><вираз з констант>

Наприклад:

#define
LN 50 #define
PI 3.141592

Наприклад, наступний оператор створює макрокоманду CUBE:

#define CUBE(x) ((x)*(x)*(x))

Наприклад, наступний оператор створює макрокоманду DELAY:

#defin
e delay(x) { \
printf("Затримка
на %d", x); \ for (long int
i=0; i < x; i++) \
;
\ }

- ознака звертання до процесора. Для зручності константи позначаються більшими буквами. Перед трансляцією процесор скрізь у тілі програми заміняє ім'я константи її значенням.

2. За допомогою слова const: **const [тип] <ім'я> = <значення>** const float pi = 3.1415926;

const maxint = 32767;

3. Оголошення перерахування починається із ключового слова enum і має два формати:

Формат 1. **enum [ім'я-тега-перерахуванняя] {список-перерахування} визначник[,визначник...];**

Наприклад :

```
enum days { sun, mon, tues, wed, thur,
fri, }  anyday; enum day {sat, sun}  weekend,
superday;
```

Формат 2. **enum ім'я-тега-перерахування визначник [,визначник..];** Наприклад :

```
anyday = mon; // дозволено
 anyday = 1;// заборонено, хоча mon ==1
```

1.2 **Лабораторна робота №1.** Арифметичні основи побудови елементів і вузлів обчислювальних машин і систем

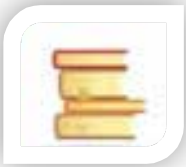
Тема: Арифметичні основи побудови елементів і вузлів обчислювальних машин і систем

Мета: вивчити системи числення (СЧ), позиційні і непозиційні системи числення, переклад чисел з однієї системи числення в іншу, вибір системи числення для ЕОМ, способи представлення чисел з фіксованою і плаваючою комою, форми представлення чисел в ЕОМ з фіксованою і плаваючою комою.

Хмарні сервіси: Gmail, Google Drive, Google Docs.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

У роботі слід ознайомить з арифметичними основами ЕОМ, тобто з тим, як представляються числа в машині і як виконуються арифметичні операції з кодами в двійковій системі числення. Подальший матеріал, а саме: побудова вузлів ЕОМ; арифметичних, запам'ятовуючих і управляючих пристроїв ЕОМ – заснований на матеріалі даної теми і не може зрозумілий при його недостатньому відпрацюванні. Всі питання теми повинні бути відпрацьовані не тільки теоретично. Особливо це торкається способів переведення чисел з однієї системи числення в іншу, виконання арифметичних операцій в двійковій системі числення і з числами в нормальній формі.

По величині основи одержує назву система числення: десяткова (основа 10), двійкова (основа 2), вісімкова (основа 8) і т.д. Якщо основа числа r перевищує 10, то цифри починаючи з 10, при записі позначають прописними літерами латинського алфавіту: A, B, …, Z. При цьому цифрі 10 відповідає знак «A», цифрі 11 – знак «B» і т.д.

В табл. 1.6 наведено відповідності чисел між різними СЧ.

Таблиця 1.6. Відповідності чисел між різними системами числення

r=10	*r=2*	*r=16*
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

При вивченні різних позиційних СЧ слід мати на увазі, що всі ці системи побудовані за одним принципом, так само, як і десяткова СЧ. Тому побудова чисел і виконання арифметичних операцій проводиться за тими ж правилами, що і побудова для десяткової СЧ.

У практиці рахунку застосовується *десяткова система* числення. У ній будь-яке число представляється в сумі певного числа одиниць, десятків, сотень і т.д. У десятковій системі числення основою є 10; будь-яке ціле число записується як сума значень 10°, 10^{1}, 10^{2} і т. д., кожна з яких може бути узята 1-9 разів.

Наприклад, в записі 343,31 число 3, що стоїть зліва на першому місці, означає число сотень, число 3 перед комою — число одиниць, а число 3 після коми — число десятих доль одиниць. Послідовність цифр 343,31 є скорочено записом виразу

$343{,}31 = 3*10^{2} + 4*10^{1} + 3*10^{\circ} + 3*10^{-1} + 1*10^{-2}$.

В обчислювальній техніці найчастіше використовується *двійкова система числення*. Це обумовлено тим, що двійкова система числення найлегше реалізується за допомогою двопозиційних елементів (наприклад, тригерів). Крім того, вона дає можливість спростити виконання арифметичних дій. При двійковій системі числення потрібні всього лише дві цифри 0 і 1 (нуль і одиниця). Основою цієї системи числення є число 2; воно записується двома цифрами: 1 і 0. У двійковій системі числення будь-яке число записується як сума значень 2°, 2^{1}, 2^{2},2^{3} і т. д., кожна з яких може бути узята 0 або 1 разів.

В загальному вигляді всяке число в двійковій системі числення може бути представлено як

$$A_r = \sum_{k=-m}^{n} a_k r^k = a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_0 2^0 + a_{-1} 2^{-1} + \ldots + a_{-m} 2^{-m}, a_k = 0,1$$

і записано у вигляді послідовності одиниць і нулів.

Для переведення десяткового числа в двійкове необхідно послідовно ділити десяткове число на 2 до тих пір, поки остача від ділення дасть число 1 або 0. При цьому сукупність залишків (0 або 1) від ділення на 2 утворює шукане.

Наприклад, представимо десяткове число 26 в двійковій системі числення. Переведення числа в двійкову систему числення проводиться за вказаним вище правилом:

Ділення	Залишок
26:2=13	0 молодший розряд
13:2=6	1
6:2=3	0
3:2=1	1
1:2=0	1 старший розряд

Залишки, одержані при діленні, починаючи з останнього, утворюють двійковий запис десяткового числа 26:

$$26(10) = 11010(2),$$

де 26—десятковий код числа; 11010 – двійковий код числа.

Це відповідає запису

$26 = 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^\circ = 16 + 8 + 0 + 2 + 0.$

Переклад чисел з однієї СЧ в іншу може бути здійснений різними способами. Найбільш зручні для ручного рахунку способи розподілу на основу і підсумовування ступенів основи.

В загальному випадку при основі *r* число представляється у вигляді:

$$A_r = \sum_{k}^{n} a_k r^k,$$

де a_k =0,1,2,...,(*r* -1) k=-*m* і записується як послідовність коефіцієнтів a_k.

Переклад чисел за способом розподілу на основу базується на наступному.

Правило. Для перекладу цілого числа в систему числення з основою *r* ділять задане число на основу цієї системи. Залишок від розподілу є молодшим розрядом цього числа. Потім приватне знову ділять на основу, і залишок дає другий розряд. Так повторюють до тих пір, поки не отримають приватне, рівне нулю. Останній залишок дає старший розряд числа в системі числення з основою *r*. Розподіл потрібно проводити в тій системі числення, в якій задано число.

A_{10}=37.

Приклад. Вимагається записати у двійковій системі числення число. Вказані в правилі дії напишемо у вигляді таблиці:

Ділення	Остача	Залишок
37:2	18	1 молодший розряд
18:2	9	0
9:2	4	1
4:2	2	0
2:2	1	0
1:2	0	1 старший розряд

A_2 =100101.

Як було вказано, залишок при першому розподілі дає молодший розряд, а при останньому – старший.

Перетворення дробових чисел здійснюється шляхом множення на основу.

Правило. Для перекладу правильного дробу в систему числення з основою *r*, необхідно помножити дріб на основу і виділити цілу частину; отримана цифра в цілій частині буде першою цифрою (після коми) шуканого числа. Помноживши дробову частину отриманого числа, що залишилася, знову на основу і узявши цілу частину нового числа, знайдемо другу цифру. Множення слід повторювати до тих пір, поки не буде отримана кількість розрядів шуканого числа, що задовольняє необхідній точності представлення цього числа. Множення проводити в тій системі числення, в якій задано число.

Приклад. Вимагається записати в двійковій системі числення дріб *A*10 = 0.732 з точністю 1/32.

При заданій точності вимагається визначити чотири розряди двійкового дробу. Запис дій проведемо у вигляді:

0.732 * 2 = 1.464
0.464 * 2 = 0.928
0.928 * 2 = 1.856
0.856 * 2 = 1.712

Для запису двійкового дробу необхідно записати цифри, що стоять в числі зліва від запису. А саме:

$$A_2 = 0.1011$$

При перекладі чисел з системи числення з якою-небудь основою в десяткову систему зручно використовувати спосіб підсумовування ступенів основи.

Правило. Для перекладу чисел з однієї системи числення з основою *r* в десяткову систему, потрібно перемножити розряди заданого числа на відповідні їм ступені *r* і скласти.

Приклад. Перевести число А=100101 з двійкової системи числення в десяткову.

$A_{10} = 2^5 + 2^2 + 2^0 = 37$

Старшому розряду відповідає п'ятий ступінь двійки.

Десяткові числа від 0 до 10 в двійковому коді записуються таким чином:

0→0000;	4→0100;	8→1000;
1→0001;	5→0101;	9→1001;
2→0010;	6→0110;	10→1010.
3→0011;	7→0111;	

Арифметичні дії над двійковими числами проводяться за тими ж правилами, що і над десятковими:

Складання	**Віднімання**	**Множення**
0 + 0 = 0;	0 – 0 = 0;	0 * 0 = 0;
0 + 1 = 1;	0 – 1 =0;	0 * 1 = 0;
1 + 0 = 1;	1 – 1 = 0;	1 * 0 = 0;
1 + 1 = 10.	10 – 1 = 1.	1 * 1 = 1.

Приклади арифметичних дій над числами в десятковій і двійковій системах числення приведені табл. 1.2.

У двійковій системі числення, так само як і в десятковій і двійковій, можуть бути записані дробові і змішані числа. Ціла і дробова частини в двійковій і в десятковій системах числення відділяються комою.

У десятковій системі числення в дробовій частині підсумовуються десяті, соті, тисячні і т.д. долі числа, а в двійковій – половинні, четвертні, восьмі, шістнадцяті і т.д.

Наприклад:

$101{,}011 = 1*2^2 + 0*2^1 + 1*2^\circ + 0*2^{-1} + 1*2^{-2} + 1*2^{-3}$.

Таблиця 1.2. – Арифметичні дії над числами в двійковій і десятковій системах числення

Система	Складання	Віднімання	Множення	Ділення	
Десяткова	8 + 4 **12**	8 – 4 **4**	8 *4 **32**	8 - 8 0	4 **2**
Двійкова	1000 + 100 1100 **1100**	1000 - 100 **100**	1000 * 100 0000 0000 + 1000 100 000 0000 0000 +1000 **100 000**	1000 -1000	100 **10**

Отже при використовуванні двійкової системи числення для представлення в ЕОМ одного розряду числа необхідно мати фізичний елемент з двома стійкими станами. Один стан відповідає нулю, а інший – одиниці. Для того, щоб представити в ЦВМ число, необхідно мати ряд двійкових елементів, для кожного розряду свій елемент.

У цифрових обчислювальних машинах окрім двійкової застосовуються і інші системи числення. Так, наприклад, при програмуванні використовується вісімкова система числення, а для введення і виведення даних — двійково-десяткова.

У вісімковій системі числення основою є число 8. Будь-яке число у вісімковій системі числення може бути представлено як сума чисел 8^3, 8^2, 8^1, 8°, 8^{-1}, 8^{-2}, 8^{-3} ..., кожне з яких може бути узяте 1—7 разів.

Аналогічно запису цілого числа в десятковій позиційній системі числення у вісімковій системі числення представляється у вигляді

$$\overline{abcd}_8 = a \cdot 8^3 + b \cdot 8^2 + c \cdot 8^1 + d \cdot 8^0 ,$$

де *a, b, c, d* менше 8.

Наприклад: $26_8 = 2*8^1 + 6*8^\circ = 22_{10}$.

Отже, число 22_{10} є десятковий вид вісімкового числа 26. Десяткове число 22 в двійковому коді має наступний запис: $22_{(10)}= 10110$.

Якщо розглянути дане двійкове число, то виявляється, що перші дві цифри означають 2, а інші три — 6, тобто це не що інше, як вісімковий код, десяткового числа 22.

Отже, для переведення вісімкового числа в двійкове необхідно кожну цифру вісімкового числа окремо представити в двійковому коді.

Наприклад:

15(8)= (001)(101)(2), де (001)2=(1)8; (101)2 = (5)8;

23(8)=(010)(011)(2), де (010)2=(2)8; (011)2=(3)8.

Таким чином, одному розряду вісімкової системи відповідають три розряди двійкової системи. Вісімкова система зручніша тим, що вона вимагає менше число розрядів, ніж двійкова. Тому вісімкова система застосовується при складанні програм для кодування адрес і команд. Після того, як програма складена, вона переводиться в двійкову систему числення і потім вводиться в машину.

Сукупність двійкових елементів, необхідних для представлення одного числа, утворює розрядну сітку машини. Залежно від того, як використовується розрядна сітка, розрізняють два способи представлення чисел: в природній і в нормальній формі.

При звичайній формі представлення чисел в розрядній сітці машини відводиться певна кількість розрядів для цілої частини, для дробової частини і один розряд (іноді два) для знаку числа. Звичайно кому розташовують після знакового розряду, і отже, всі числа представляються правильними дробами.

Схематичне зображення розрядної сітки з комою, фіксованою після знакового розряду, і представлення чисел наведено на рис.1.3.

+0.10010111 (а) і
-0.01011001 (б) приведено на рисунку 1.1:

а) зн. числова частина (дріб)

0	1	0	0	1	0	1	1	1

б) зн. числова частина (дріб)

1	0	1	0	1	1	0	0	1

Рис. 1.3 - Представлення чисел з фіксованою комою

Діапазон чисел, які можна представити в машині при природній формі, визначається виразом:

$$2^{-n} \le N \le 1 - 2^{-n},$$

де n – кількість розрядів числової частини.

Обмежене число розрядів розрядної сітки приводить до погрішності в представленні чисел.

Для найбільшого можливого числа N_{max} відносна похибка:

$$\delta_{\max_1} = \frac{\Delta_{\max}}{N_{\max}} = \frac{0.5*2^{-n}}{1-2^{-n}} \cong 0.5*2^{-n}.$$

Для якнайменшого уявного числа відносна похибка:

$$\delta_{\max_2} = \frac{0.5*2^{-n}}{2^{-n}} = 0.5.$$

В нормальній формі число представляється у вигляді:

$$N=M*10^{p},$$

де M- мантиса,

10 - основа системи числення, записане в цій системі, p - порядок.

$0{,}110101 *10^{+100}$ (*а*) $0{,}101100 *10^{-10}$ (*б*)

В розрядній сітці машини відводять розряди для мантиси, порядку, знака числа і знака порядку, схематичне зображення розрядної сітки і представлення двійкових чисел

Приведено на рисунку 1.4.

Рис.1.4 - Представлення чисел з плаваючою комою

Діапазон чисел, які можна представити в машині при нормальній формі, визначається виразом:

$$2^{-2^{p}} \le N \le (1-2^{-m})*2^{(2^{p}-1)}$$

де P – кількість розрядів порядку;

M – кількість розрядів мантиси.

Виконання арифметичних операцій з цілими числами в нормальній формі має певні особливості.

При складанні і відніманні необхідно провести вирівнювання, яке полягає в тому, що меншому числу привласнюється порядок більшого числа. Після цього проводиться дія з мантисою, а потім нормалізація результатів. В результаті

повинна бути отриманий мантиса у вигляді правильного дробу із значущим старшим розрядом. При множенні і розподілі проводиться окремо дія з мантисами і окремо з порядками, а потім нормалізується результат.

Для представлення двійкових чисел в ЕОМ застосовують пряму, зворотну і додаткову коди. У всіх цих кодах передбачається додатковий розряд для представлення знаку числа, причому знак (+) кодується цифрою 0, а знак (-) - цифрою 1. При допомозі цих кодів операція віднімання або складання алгебри зводиться до арифметичного складання. В результаті спрощуються арифметичні пристрої ЕОМ.

Позитивні числа в прямому, зворотному і додаткових кодах мають один і той же вигляд, негативні - різний.

Щоб представити двійкове негативне число в зворотному коді, треба поставити в знаковий розряд одиницю, а в усіх інших розрядах замінити одиниці нулями, а нулі – одиницями.

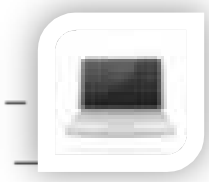
Наприклад, число:

$(-3.5)_{10}$-$(-11.1)_2$,

Представлене в зворотному коді, має вигляд: $(100.0)_{обр}$, де перша одиниця в знаковому розряді указує на те, що число негативне.

При записі числа в додатковому коді ставлять одиницю в розряд знаку, а цифрову частину числа замінюють доповненням модуля числа до цілої одиниці. Наприклад, число $(-0.01)_2$ в додатковому коді матиме вид $(1.11)_{дод}$. Тут перша одиниця указує на те, що число негативне.

При складанні алгебри двох цілих чисел з використанням зворотного коду позитивні доданки представляються в прямому коді, а негативні в зворотному, проводиться арифметичне підсумовування цих кодів, включаючи розряди знаків. При виникненні перенесення з розряду знаку одиниця перенесення додається до молодшого розряду суми кодів (таке перенесення називають круговим або циклічним). В результаті виходить сума алгебри в прямому коді, якщо ця сума позитивна, і в зворотному коді, якщо вона негативна.



Практична частина

Завдання

1. Перевести число з десяткової СЧ в шістнадцяткову, згідно варіанту

(табл. 1.3).

2. Перевести число з шістнадцяткової СЧ в десяткову, згідно варіанту (табл. 1.4).

3. Перевести число з двійкової СЧ в шістнадцяткову, згідно варіанту (табл. 1.5).

4. Перевести число з шістнадцяткової СЧ в двійкову, згідно варіанту (табл. 1.4).

5. Перевести правильний дріб з десяткової СЧ в двійкову та шістнадцяткову, згідно варіанту (табл. 1.6). Перевод виконувати до 4 значущих цифр після коми.

6. Перевести правильний дріб з двійкової СЧ в десяткову, згідно варіанту (табл. 1.7).

7. Перевести правильний дріб з шістнадцяткової СЧ в десяткову, згідно варіанту (табл. 1.8).

8. Перевести правильний дріб з двійкової СЧ в шістнадцяткову, згідно варіанту (табл. 1.7).

9. Перевести правильний дріб з шістнадцяткової СЧ в двійкову, згідно варіанту (табл. 1.8).

10. Перевести дрібне число з десяткової СЧ в шістнадцяткову, згідно варіанту (табл. 1.9). Переводи виконувати до 3 значущих цифр після коми.

Таблиця 1.3

Варіант	1	2	3	4	5	6	7	8	9	10
Число	23	54	41	100	33	75	58	68	112	202

Таблиця 1.4

Варіант	1	2	3	4	5	6	7	8	9	10
Число	15	22	33	61	42	77	82	40	25	34

Таблиця 1.5

Варіант	1	2	3	4	5	6	7	8	9	10
Число	01110	110010	001100	01010	11001	0101101	110110	001011	11100	000111

Таблиця 1.6

Варіант	1	2	3	4	5	6	7	8	9	10
Число	0,501	0,633	0,805	0,303	0,444	0,225	0,106	0,777	0,911	0,666

Таблиця 1.7

Варіант	1	2	3	4	5	6	7	8	9	10
Число	0,110	0,0011	0,1010	0,1100	0,0101	0,1111	0,0111	0,1110	0,1000	0,1101

Таблиця 1.8

Варіант	1	2	3	4	5	6	7	8	9	10
Число	0.A7E	0.3DC	0.C8F	0.B3D	0.2EA	0.5DF	0.DF3	0.B8A	0.EC6	0.D7B

Таблиця 1.9

Варіант	1	2	3	4	5	6	7	8	9	10
Число	20,404	33,02	150,843	100,115	44,201	22,444	18,041	51,002	22,761	31,052

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. В чому полягають основні властивості і особливості позиційних систем числення?

2. Що називають основою системи числення? 2^n *або* $-10 \le n \le 10$

3. Записати число в нормальній формі.

4. Сформулювати правило перекладу цілих чисел з однієї системи числення в іншу методом розподілу на основу.

5. Сформулювати правило перекладу дробових чисел з однієї системи числення в іншу методом множення на основу.

6. Сформулювати правило перекладу чисел з двійкової системи у вісімкову.

7. Як виконати над числами наступні арифметичні операції: A + B + C; A - B; C - D; E - F; B * E; C * D; F : E; A : E. Операцію віднімання виконувати як складання алгебри, представляючи негативні числа в зворотному коді

8. Чим викликано застосування додаткового і зворотного кодів?

9. В чому відмінності представлення чисел в природній і нормальній формах? Яка форма і коли зручніше?

10. Як провести складання чисел. $A=0.1100*10^{-101}$, $B=0.1001*10^{-001}$

1.3 **Лабораторна робота №2**. Алгоритмізація задач

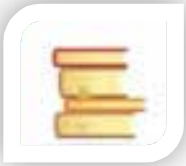
Тема: Алгоритмізація задач

Мета: опанування принципів розробки алгоритмів задач та описання алгоритмів з допомогою блок-схем.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, Draw io.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Алгоритмізація **–** це розділ науки про методи побудови алгоритмів, який містить докази правильності та реалізуємості алгоритмів, а також вивчення їх властивостей.

Алгоритмом називається кінцева сукупність точно сформульованих правил, тобто послідовність кроків (тактів) вирішення задачі, на кожному з яких виконується певна операція або дія.

Послідовність дій, яка описується в алгоритмі, може бути ***статичною*** (у порядку описання) та ***динамічною*** (у порядку виконання). Відповідність між статичною і динамічною послідовностями порушується у тих місцях алгоритму, де відбувається його розгалуження за певною умовою або звернення до допоміжного алгоритму

Базові структури алгоритму – це структури, з допомогою яких створюється алгоритм для вирішення певної задачі. Існують три основні алгоритмічні структури (три основні типи алгоритмів):

Лінійні (послідовний) алгоритм *(ланцюг)* – алгоритм, на якому статична та динамічна послідовності збігаються (дії виконуються послідовно, одна за одною, тобто лінійно) і який забезпечує отримання результату шляхом одноразового виконання послідовності дій, незалежно від вхідних даних та проміжних результатів.

Альтернативний (розгалужений) алгоритм *(умова, структура вибору)* – алгоритм, у якому передбачається можливість вибору рішення, залежно від заданої умови; розгалуження може бути повним, коли дії визначені, як під час

виконання, і при невиконанні умови, і неповним, коли дії визначені лише за виконання умови.

Циклічний алгоритм (цикл, структура повторення) – алгоритм, у якому передбачено багаторазове повторення певних дій; сукупність дій, що утворюють **тіло циклу**, може містити інший цикл, який називають вкладеним циклом; існують **два типи циклів**:

- *цикл із параметром (з лічильником)* – цикл із заздалегідь відомим числом повторень тіла циклу; перевірка умови роботи циклу з параметром здійснюється до виконання тіла циклу, тобто. він є циклом із передумовою;
- *ітераційний цикл* – цикл, число повторень якого визначається за умовою виконання чи завершення циклу, і може змінюватися у процесі виконання тіла циклу; до ітераційних циклів відносяться цикли з передумовою (коли умова перевіряється до виконання тіла циклу) та постумовою (коли умова перевіряється після виконання тіла циклу).

Блок-схема алгоритму – це його графічне представлення, що зображується у вигляді послідовності пов'язаних між собою за допомогою ліній зі стрілками (ліній переходу) функціональних блоків (графічних символів), кожен з яких відповідає виконанню однієї або декількох дій і всередині яких дається опис відповідної дії. Таким чином, графічні символи позначають виконувані дії, а лінії зі стрілками - послідовність їх виконання.

Основні символи блок-схем алгоритмів

1. ***Термінатор (пуск-останов)****: початковий* символ, що позначає початок алгоритму, *кінцевий* – його кінець (рис. 1.3).

Рис.1.3 - *Термінатор (пуск-останов)*

2. ***Процес*** (рис. 1.4)*: обчислювальний символ,* що позначає виконання однієї операції або групи операцій обробки даних, де операція є певною командою, яка називається *оператором*.

Рис.1.4 – *Позначення процесу*

3. ***Вирішення*** (рис. 1.5)*: умовний символ, що* позначає вибір напряму виконання алгоритму в залежності від логічної умови, яке може приймати лише два значення: «*1*» («+»), якщо умова виконується (т*ак*), «*0*» («–»), якщо умова не виконується (*ні*).

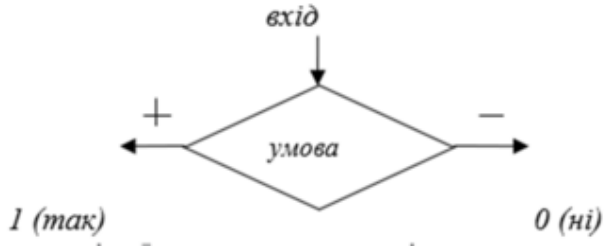


Рис. 1.5 – Позначення вирішення

4. ***Межа циклу*** (рис. 1.6): символ, що відображає початок та кінець циклу; умова завершення циклу міститься всередині символу, на початку чи кінці залежно від розташування операції, яка перевіряє умову.

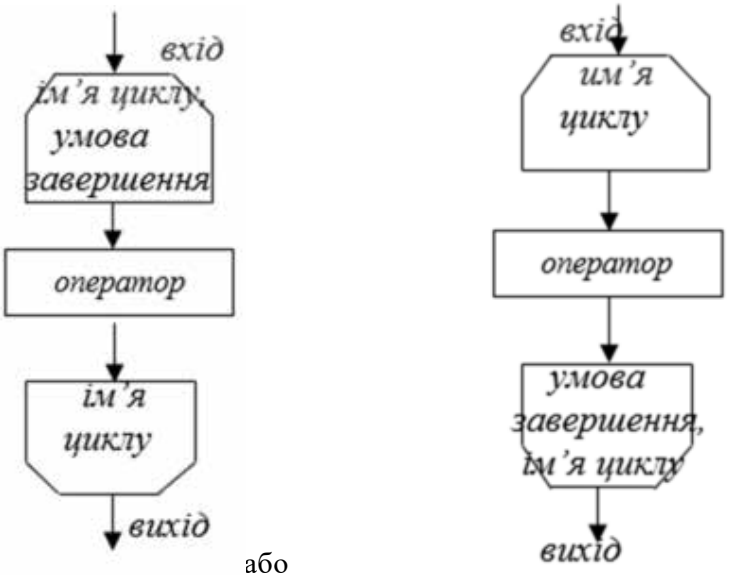


Рис. 1.6 – Позначення межі циклу

5. ***Підготовка*** (рис. 1.7)*:* символ, який відображає модифікацію команди або групи команд; можна використовувати як символ структури *циклу з параметром*.

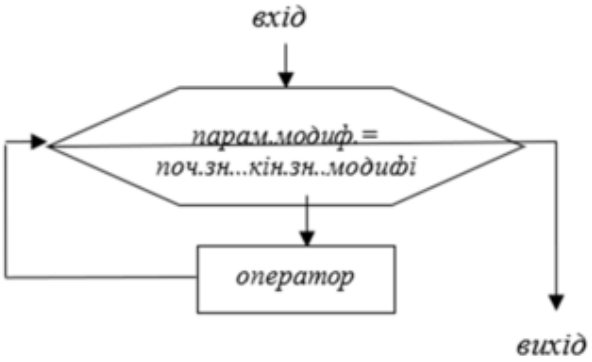


Рис. 1.7 – Позначення підготовки

6. Для структур ***ітераційних циклів*** можна використовувати символ *вирішення* (*умовний символ*).

Цикл з передумовою (рис. 1.8).

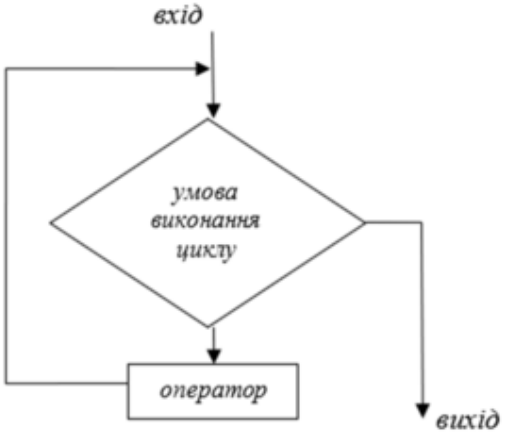


Рис. 1.8 – Позначення ітераційного циклу з передумовою

Цикл з постумовою (рис. 1.9).

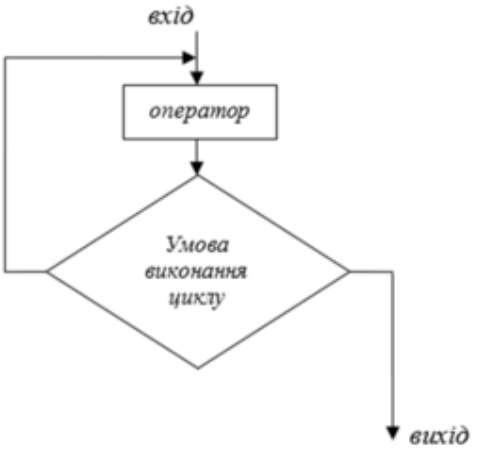


Рис. 1.9 – Позначення ітераційного циклу з постумовою

7. **Наперед визначений процес** (рис. 1.10): символ звернення до підпрограми використовується для звернення до допоміжного алгоритму, який є таким щодо основного алгоритму.

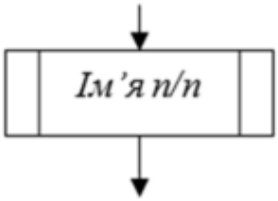


Рис. 1.10 – Позначення наперед визначеного процесу

8. ***Дані*** (рис. 1.11)***:*** символ відображає дані, носій яких не визначено; можна використовувати для позначення *вводу/виводу даних*.

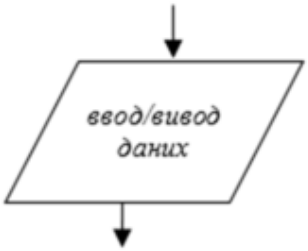


Рис. 1.11 – Позначення даних

9. ***Ручний ввод*** (рис. 1.12)***:*** символ відображає дані, які вводять вручну під час обробки з пристроїв будь-якого типу.

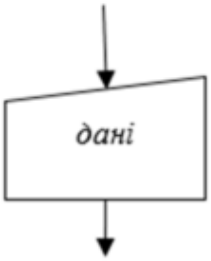


Рис. 1.12 – Позначення ручного вводу

10. ***Дисплей*** (рис. 1.13)***:*** символ відображає дані, що представлені на носії у вигляді пристрою виведення (екран монітору).

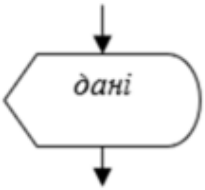


Рис. 1.13 – Позначення дисплею

11. ***З'єднувач*** (рис.1.14)*: символ переривання блок-схеми* використовують як вказівник зв'язку між перерваними лініями потоку, що пов'язують символи; відповідні символи-з'єднувачі повинні містити одне й теж унікальне позначення.

Рис. 1.14 – Позначення з'єднувача

Практична частина

1. Виконання загального завдання

Розібрати представлені приклади алгоритмізації задач.

Лінійний алгоритм

Задача

1. ***Постановка задачі:*** розробити алгоритм обчислення значення *a* за формулою:

$$a = \frac{\sqrt{|x-1|} - \sqrt{|y|}}{1 + \frac{x^2}{2} + \frac{y^2}{4}}$$

Значення *x, y* ввести з клавіатури; отримане значення *a* вивести на екран.

2. ***Математична модель (метод розробки алгоритму) та описання алгоритму задачі:***

- ввести значення *x, y*;
- обчислити значення *a за* заданою формулою;
- вивести отримане значення *a*.

Блок-схема алгоритму задачі (рис.1.15)***:***

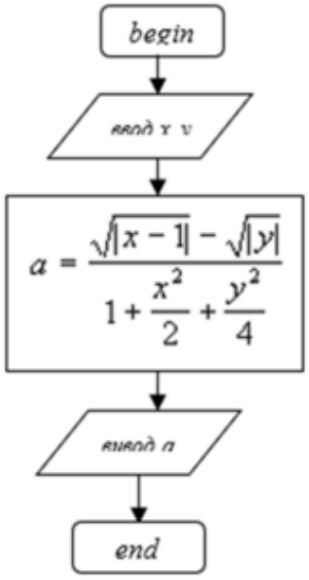


Рис.1.15 - *Блок-схема алгоритму вирішення задачі*

Розгалужений алгоритм

Задача 1

1. Постановка задачі: розробити алгоритм обчислення та виводу значення функції

$$y = \begin{cases} x^3 - 1{,}5, & \text{якщо } x<0; \\ x^2 + 2\cdot x, & \text{якщо } x \geq \pi/2; \\ \cos x + 0{,}2, & \text{якщо } 0 \leq x < \pi/2; \end{cases}$$

2. Математична модель та описання алгоритму задачі:

- ввести значення x;
- якщо $x<0$, то $y=x^3-1{,}5$;
- інакше, якщо $x \geq \pi/2$, то $y=x^2+2\cdot x$;
- інакше, якщо $0 \leq x < \pi/2$, тобто у всіх інших випадках, $y=\cos x+0{,}2$;
- вивести значення y.

3. Блок-схема алгоритму задачі (рис. 1.16)***:***

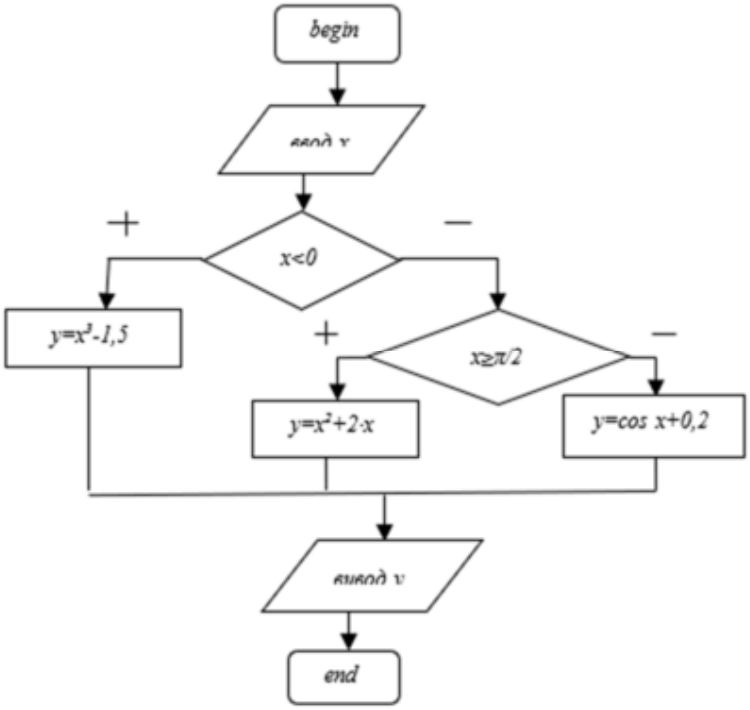


Рис.1.16 - *Блок-схема алгоритму вирішення задачі*

Задача 2

1. Постановка задачі: є два кола з центрами у початку координат і радіусами, які дорівнюють *2* та *4*; ввести координати точки та визначити, у коло якого радіусу воно потрапляє (вважати, якщо точка потрапляє у менше коло, то вона не належить більшому); вивести номер кола.

2. Математична модель и описання алгоритму задачі:

- ввести координати точки – *x* та *y*;
- використовуючи формулу кола: $x^2+y^2=r^2$, де *r* – радіус кола;
- нехай умова потрапляння у коло визначається значенням *k*, тоді, якщо $x^2+y^2\leq 4$, то значення $к=1$, тобто. точка потрапляє у перше коло;
- інакше, якщо $x^2+y^2\leq 16$, то $к=2$, тобто точка потрапляє у друге коло;
- інакше, потрапляння у коло нема, та $к=0$;
- вивести значення *k*.

3. Блок-схема алгоритму задачі (рис. 1.17)***:***

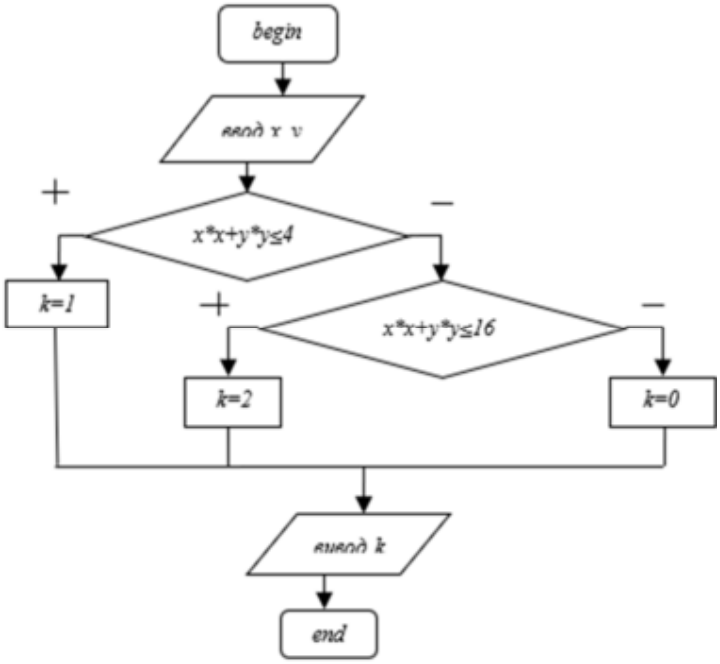


Рис.1.17 - *Блок-схема алгоритму вирішення задачі*

Циклічні алгоритми

Задача 1

1. Постановка задачі: розробити алгоритм сумування *n* введених чисел та виводу значень суми.

2. Математична модель та описання алгоритму задачі:

- використовуючи цикл з постумовою $n \leq 0$ (кількість введених чисел не може бути менше або дорівнювати нулю), ввести кількість введених чисел, щоб $n>0$;
- для формування суми значення *s* обнулити ($s=0$);
- у циклі з параметром (кількість повторів = *n*) кожен раз вводити значення *a* та формувати суму: $s=s+a$;
- після виходу з циклу вивести значення змінної *s*.

3. Блок-схема алгоритму задачі (рис.1.18)***:***

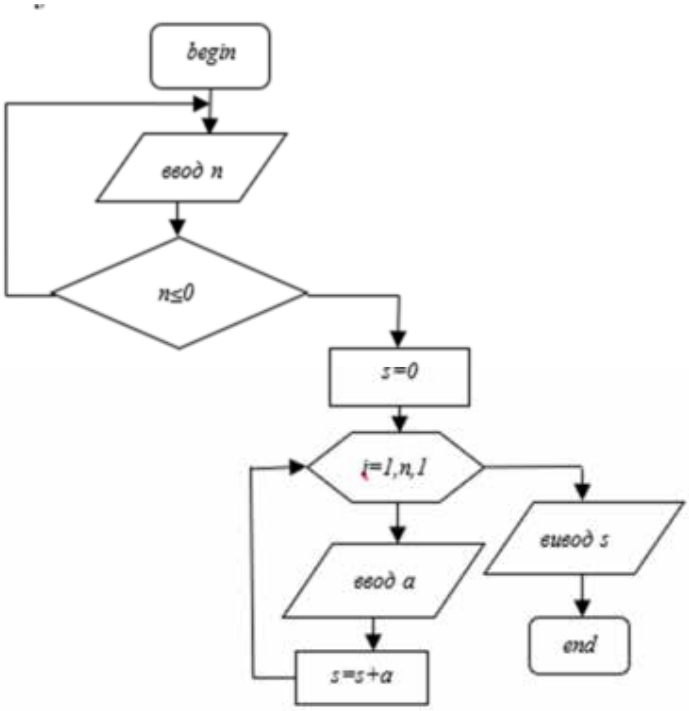


Рис.1.18 - *Блок-схема алгоритму вирішення задачі*

Задача 2

1. Постановка задачі: розробити алгоритм обчислення суми та добутку лише позитивних чисел з *n* введених; вивести значення суми та добутку.

2. Математична модель та алгоритм вирішення задачі:

- використовуючи цикл з постумовою $n \leq 0$, ввести кількість введених чисел, щоб $n>0$;
- для формування суму початкове значення $s=0$;
- для формування добутку початкове значення $p=1$;
- у циклі з параметром (кількість повторів = *n*) кожен раз вводить значення *a та* перевіряє: якщо $a>0$, то формувати суму $s=s+a$ а добуток $p=p \cdot a$;
- після виходу з циклу вивести значення *s* та *p*.

3. Блок-схема алгоритму задачі (рис. 1.19)***:***

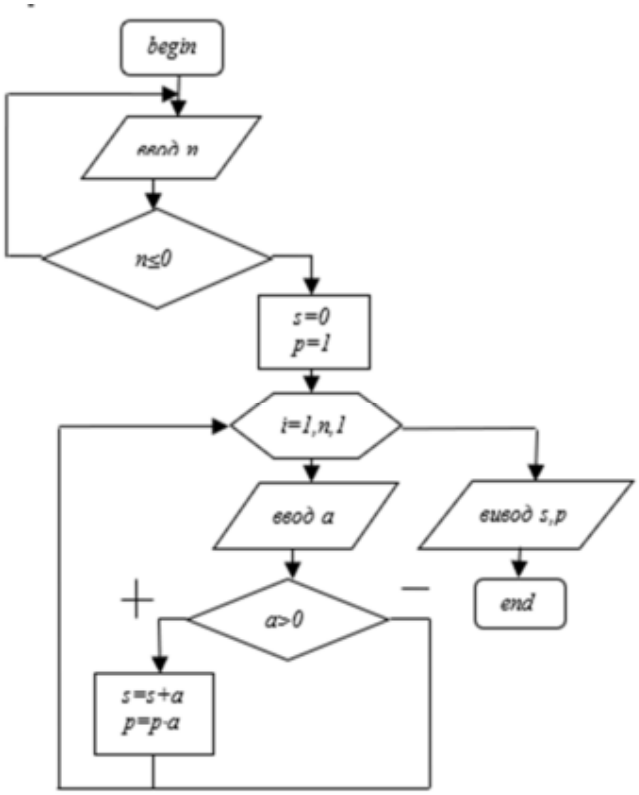


Рис.1.19 - *Блок-схема алгоритму вирішення задачі*

Задача 3

1. Постановка задачі: розробити алгоритм обчислення суми членів ряду: $S=\sum_{i=1}^{n} x^i = x+x^2+x^3+\ldots+x^{n-1}+x^n$ для $0<x\leq 4$, $n=7$.

2. Математична модель та алгоритм вирішення задачі:

- ввести значення x;
- перевірити умову: якщо $0<x\leq 4$, то продовжити виконання алгоритму, інакше, вийти з алгоритму;
- для формування суми значення $S=0$;
- для формування ступеня x початкове значення $xs=x$;
- у циклі з параметром (кількість повторів $n=7$) кожний раз S збільшується на xs, тобто $S=S+xs$, а xs збільшується в x разів, щоб отримати ступінь x, тобто $xs=xs\cdot x$;
- після виходу з циклу вивести значення S.

3. Блок-схема алгоритму задачі (рис.1.20)***:***

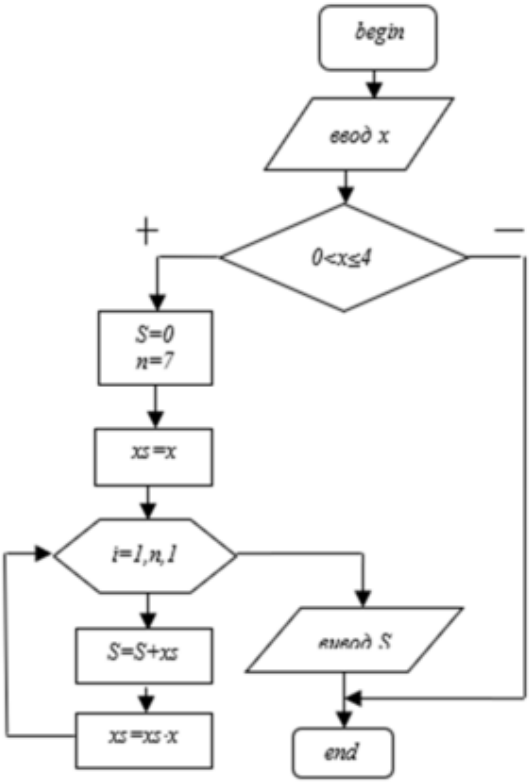


Рис.1.20 - *Блок-схема алгоритму вирішення задачі*

Задача 4

1. Постановка задачі: розробити алгоритм обчислення часткової суми членів ряду $S = \cos(x) + \frac{\cos(2 \cdot x)}{2} + \frac{\cos(3 \cdot x)}{3} + ... + \frac{\cos(n \cdot x)}{n} + ...$ для $\pi/3 < x \leq \pi$ с точністю $\varepsilon = 10^{-4}$, тобто. сумування продовжувати, поки поточний член ряду по модулю не буде $\leq 10^{-4}$.

2. Математична модель та описовий алгоритм задачі:

- використовуючи ітераційний цикл з передумовою $x \leq \pi/3$ або $x > \pi$, ввести значення, щоб $\pi/3 < x \leq \pi$; для входу в цикл $x=0$, оскільки 0 не входить в заданий діапазон значень x;
- для формування суми значення $S=0$;
- начальне значення коефіцієнта при x: $k=1$;
- використовуючи цикл з передумовою для обчислення суми членів ряду, поки поточний член ряду $|cos(k \cdot x)/k| > \varepsilon$, зформувати суму членів ряду $S=S+cos(k \cdot x)/k$ і коефіцієнт k збільшити на одиницю для наступної ітерації;
- після закінчення циклу вивести значення S та кількість ітерацій $k-1$ (скільки разів повторився цикл, тобто скільки членів ряду просумувалось, щоб отримати часткову суму для заданої умови).

3. Блок-схема алгоритму задачі (рис.1.21)***:***

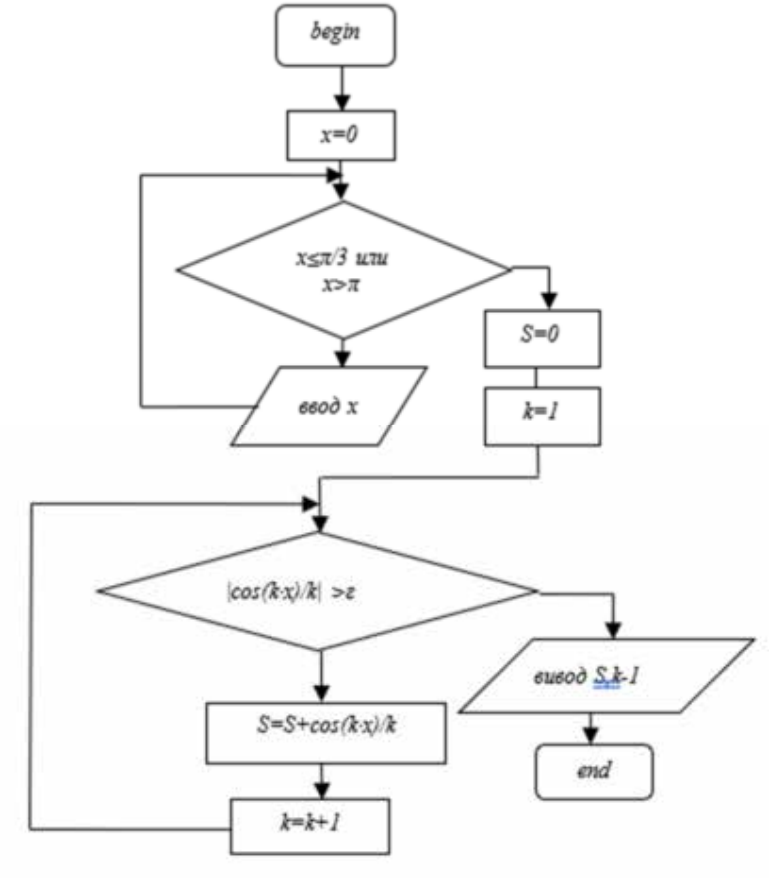


Рис.1.21 - *Блок-схема алгоритму вирішення задачі*

Завдання . Варіанти індивідуальних завдань

Варіант №1

Розробити алгоритм отримання суми *n* введених чисел, обчислення їх середнього арифметичного значення та виводу одержаних значень.

Варіант №2

Розробити алгоритм отримання суми позитивних з *n* введених чисел та виводу одержаного значення суми.

Варіант №3

Розробити алгоритм обчислення значення функції

$$y = \begin{cases} x^2+1, & \text{якщо } x \le 0; \\ 2 \cdot x, & \text{якщо } x > \pi/4; \\ \cos x + \sin x & \text{у інших випадках;} \end{cases}$$

та виводу одержаного значення.

Варіант №4

Розробити алгоритм обчислення суми членів ряду $S=x+2 \cdot x+3 \cdot x+\ldots+(n-1)x+nx$ для $-10<x<10$, $n=10$ та виводу її значення.

Варіант №5

Розробити алгоритм одержання суми та добутку *n* введених чисел та виводу отриманих значень суми і добутку.

Варіант №6

Розробити алгоритм одержання суми лише від'ємних з *n* введених чисел та виводу отриманого значення.

Варіант №7

Розробити алгоритм одержання суми *n* введених чисел, обчислення середнього арифметичного значення лише від'ємних з цих чисел та виводу одержаних значень.

Варіант №8

Розробити алгоритм обчислення суми членів ряду $S=x+2\cdot x^2+3\cdot x^3+\ldots+(n-1)x^{n-1}+nx^n$ для $-1<x<3$, $n=7$ і виводу отриманого значення.

Варіант №9

Розробити алгоритм одержання суми позитивних, кратних 3-ом, з *n* введених чисел та виводу отриманого значення.

Варіант №10

Розробити алгоритм обчислення добутку позитивних з *n* введених чисел та виводу одержаного значення.

Варіант №11

Розробити алгоритм обчислення середнього арифметичного значення позитивних з *n* введених чисел та виводу одержаного значення.

Варіант №12

Розробити алгоритм задачі: ввести три числа, знайти та вивести значення найбільшого з них.

Варіант №13

Розробити алгоритм задачі: ввести три позитивних числа, знайти та вивести значення найменшого з них.

Варіант №14

Розробити алгоритм обчислення значень функції

$$y=\begin{cases} x^2+3\cdot x, & \text{якщо } x<-1 \\ 2\cdot x+5, & \text{якщо } x>10 \\ x & \text{в інших випадках} \end{cases}$$

та виводу цього значення.

Варіант №15

Розробити алгоритм підрахунку окремо позитивних, окремо від'ємних з *n* введених чисел та виводу цих значень.

Варіант №16

Розробити алгоритм обчислення значення функції

$$y = \begin{cases} 2 \cdot x^2 + 3 \cdot x, & \text{якщо } x < -2 \\ 2 \cdot x + 7, & \text{якщо } x \geq 0 \\ -4 & \text{в інших випадках} \end{cases}$$

та виводу цього значення.

Варіант №17

Розробити алгоритм обчислення суми членів ряду $S=x+2(x+1)+3(x+2)+\ldots+(n-1)(x+n-2)+n(x+n-1)$ для $-2<x\leq 4$, $n=8$ та виводу отриманого значення.

Варіант №18

Розробити алгоритм одержання суми окремо від'ємних, окремо позитивних з n введених чисел та виводу отриманих значень.

Варіант №19

Розробити алгоритм обчислення добутку лише від'ємних з n введених чисел з відніманням з нього значення першого введеного числа та виводу одержаного значення.

Варіант №20

Розробити алгоритм обчислення значення функції

$$y = \begin{cases} x^4 - 4 \cdot x, & \text{якщо } x \leq -2 \\ 2 \cdot x - 10, & \text{якщо } x > 10 \\ 3 \cdot x, & \text{якщо } -2 < x \leq 10 \end{cases}$$

та виводу цього значення.

Варіант №21

Розробити алгоритм обчислення значення функції

$$y = \begin{cases} x^2 - 4 \cdot x - 1, & \text{якщо } x \leq 0 \\ 5 \cdot x - 1, & \text{якщо } x > 5 \\ x, & \text{якщо } 0 < x \leq 5 \end{cases}$$

та виводу цього значення.

Варіант №22

Розробити алгоритм обчислення добутку лише від'ємних парних чисел з n введених та вивід отриманого значення.

Варіант №23*

Розробити алгоритм обчислення часткової суми членів ряду $S = \sin(x) + \frac{\sin(3 \cdot x)}{2} + \frac{\sin(5 \cdot x)}{3} + \ldots + \frac{\sin((2n-1) \cdot x)}{n} + \ldots$ з точністю $\varepsilon=10^{-5}$ для $\pi/4<x\leq 2\pi$ та виводу її значення.

Варіант №24*

Розробити алгоритм обчислення часткової суми членів ряду

$S=cos(x)+cos(2x)+cos(3x)+…+cos(nx)+…$ з точністю $\varepsilon=10^{-4}$ для $\pi/6<x\leq\pi/3$ та виводу її значення.

Варіант №25*

Розробити алгоритм обчислення часткової суми членів ряду $S=\frac{\cos(x)}{2}+\frac{\cos(3\cdot x)}{4}+...+\frac{\cos((2n-1)\cdot x)}{2\cdot n}+...$ з точністю $\varepsilon=10^{-3}$ для $\pi/6<x\leq\pi$ та виводу її значення.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке *алгоритм*?
2. Перерахувати базові структури алгоритмів.
3. Визначити, що являють собою ланцюг та *розгалуження*.
4. Що таке *цикл* як базова структура алгоритму?
5. Які існують різновиди циклів?

РОЗДІЛ ІІ. СТРУКТУРА ДАНИХ ТА ВИРАЗИ

2.1 Теоретичні відомості. Структура даних та вирази

2.1.1 Типи і змінні

Після розгляду алфавіту мови С актуальним є дослідження різновидів типів даних (рис.2.1), використовуємих при програмуванні мовою С.

Рис. 2.1 - Місце даних в загальному уявленні про мову С

Змінні в мові С++ діляться на дві групи типів: основні й похідні. Варіант розподілу наведено на рис. 2.2

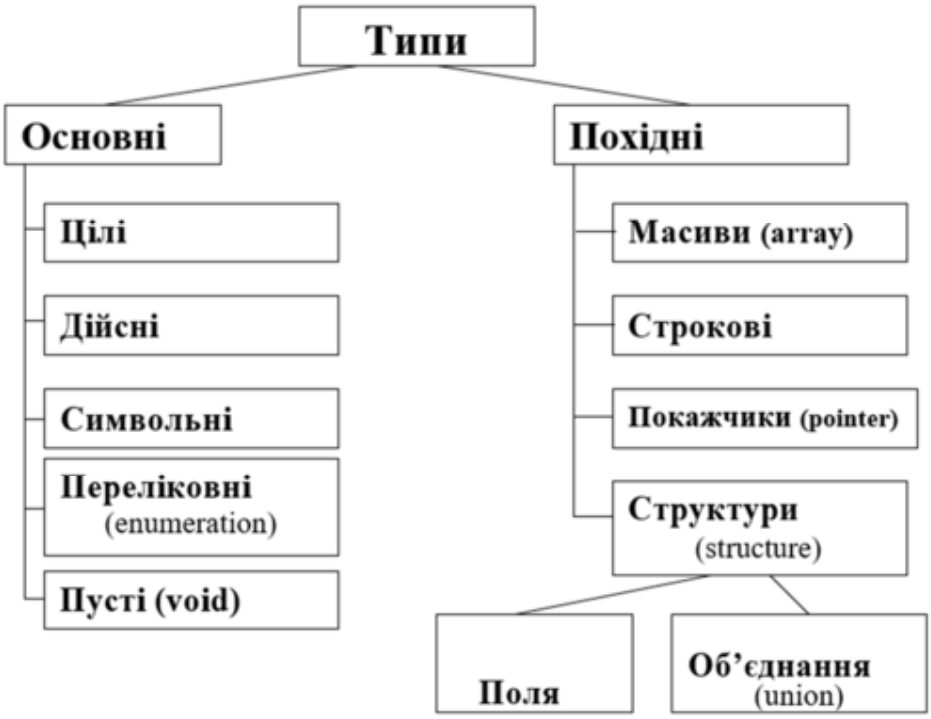


Рис. 2.2 – Типи змінних мови С

До основних належать ***5 типів даних***: цілі; дійсні (з плаваючою крапкою); символьні (char); переліковні (enumeration); порожні (void). З основних типів створюються похідні ***4- х типів***: масиви (array); покажчики (pointer); строкові (рядки); структури або записи (structure) і їх різновиди поля й об'єднання (union).

2.1.2 Цілий тип даних

Як вже відзначалось раніше, окрім стандартних цілих можна ще використовувати й деякі інші різновиди типів даних: short - короткі; long - довгі; unsigned - без знака.

Оголошення цілих змінних має виглд, наведений у табл. 2.1

Таблиця 2.1 Діапазон і розмір цілих змінних

Тип	Розмір	Діапазон
char	1	0 ÷ 256
int	2	-32768 ÷ 32767
short	2	-32768 до 32767
long	4	-2 147 483 648 ÷ 2 147 483 647
unsigned shar	1	0 ÷ 255
unsigned int	2	0 ÷ 65535
unsigned short	2	0 ÷ 65535
unsigned long	4	0 ÷ 4 294 967 295

До прикладу, розглянемо змінні:

```
int           i, n;
short int     low, hight;
long int      max;
unsigned int  kl;
```

В останніх трьох описах ключове слово int може бути пропущене. ЗАЛЕЖНО від транслятора short int можуть займати 1 байт або 2 байта.

В описах змінних можна задавати й початкове значення

```
long max = 32767L; short dogs, cats = 93;
```

В останньому випадку змінна cats має значення 93, а змінна dogs - ніякого.

2.1.3 Дійсний тип даних

Опис змінних дійсного типу з плаваючою крапкою має вигляд, наведений у табл. 2.2

Таблиця 2.2 Дійсний тип даних з плаваючою крапкою

Тип	Розмір	Діапазон	Точність
float	2 слова - 4 байти - 32 біта	10e-38 ÷ 10e38	5 знаків
double	4 слова - 8 байт - 64 біта	10e-308 ÷ 10e308	15знаків
long double	5 слів - 10 байт - 80 біт	10e-4932 ÷ 10e4932	19знаків

До прикладу наведено описання змінних дійсного типу даних

```
float pi_float;
double pi_double;
long double pi_long_double;
pi_float = 3.1415;
pi_double = 3.14159265358979;
pi_long_double = 3.141592653589793238;
```

2.1.4 Символьні змінні

Символьні змінні задаються описанням:

```
char c, ch='Y';
char esc ='\x1B';
```

У таблиці 2.3 наведено шістнадцятковий формат та найменування для певних управляючих послідовносей.

Таблиця 2.3 Шістнадцятковий формат та найменування для певних управляючих послідовносей

Управляюча послідовність	Найменування	Шістнадцятковий формат
\a	Звуковий сигнал	007
\b	Повернення на крок	008
\t	Горизонтальна табуляція	009
\n	Перехід на нову строку	00A
\v	Вертикальна табуляція	00B
\r	Повернення каретки	00C
\f	Перевод формата	00D
\"	Лапки	022
\'	Апостроф	027
\0	Ноль-символ	000
\\	Зворотня дробова риса	05C
\0ddd	Символ набору кодів в вісімковому представленні	
\xddd	набір кодів в шістнадцятковому представленні	

Як і для цілих змінних, початкові значення змінних інших типів також можуть бути задані в описанні.

2.1.5 Переліковний тип - ANSI - C

Переліковний тип (enum) використовується для описання об'єктів з певним

набором, наприклад {winter, spring, summer, autumn}. Тоді можна записати таке оголошення (декларацію):

enum seasons {wint, spring, sum, autumn}

І описати відповідні змінні:

enum seasons a, b, c.

Кожна із зазначених змінних може здобувати одне із чотирьох значень. І оголошення типу, і опис змінних можуть бути об'єднані:

enum seasons {wint, spring, sum, autumn} a, b, c;

Імена, які зазначені в дужках, це не рядок. Усередині ЕОМ вони набувають значення цілих чисел: перше - 0, друге - 1, і т.д. Можна присвоїти й інші значення, але за збільшенням:

enum month {JAN = 1, FEB, MAR, ..., DEC};

enum days {mon = 5, tues = 8, wed = 10, thur, fri, sat, sun}.

Переліковний тип також можна використовувати для надання константам імен:

enum escapes {BELL = '\a', BACKPASE = '\b', TAB = '\t', NEWLINE = '\n', VTAB = '\V', RETURN = '\r'}.

2.1.6 Арифметичні вирази та операції присвоювання

У мові С передбачене майже 40 операцій, але розглянемо спочатку ті, які найчастіше вживаються - арифметичні операції, операцію присвоєння, зменшення й збільшення, обчислення модуля. Місце виразів у мові С наведено на рис. 2.3.

Рис. 2.3 - Місце виразів у мові С

Операція присвоєння найчастіше використовується в програмах і має вигляд:

a = b, де a - певна змінна, b - вираз.

Результат виразу b присвоюється змінної a. Можна використовувати й багаторазове присвоєння

a = b = c = d = 1.

Таке присвоєння виконується ⇐ зправа наліво.

Якщо b є арифметичним виразом загального виду, то його результат залежить від прийнятої послідовності виконання операцій - їх старшинства. Враховуючи велику кількість операцій, існують таблиці, де обумовлений порядок виконання й старшинство.

Ієрархію операцій в С наведено в табл. 2.4.

Таблиця 2.4 Ієрархія операцій в С

Операція	Ім'я	Приклад
::	Діапазон області визначення	classname::classmember_name
::	Глобальний діапазон	::variable_name
.	Вибір елемента	object.member_name
->	Вибір елемента	pointer->membername
[]	Індексація	pointer[element]
()	Виклик функції	expression(parameters)
()	Будування значення	type(parameters)
sizeof	Розмір об'єкта	sizeof expression
sizeof	Розмір типу	sizeof(type)
++	Збільшення після	variable++
++	Збільшення до	++variable
--	Зменшення після	variable--
--	Зменшення до	-- variable
&	Адреса об'єкту	&variable
*	Розіменування	*pointer
new	Створення (розміщення)	new type
delete	Видалення	delete pointer
delete[]	Видалення масиву	delete pointer
~	Доповнення	~expression
!	Логічне НІ	! expression
+	Унарний плюс	+1
-	Унарний мінус	-1
()	Приведення	(type) expression
.*	Вибір елемента покажчика	object.*pointer
->	Вибір елемента покажчика	object->*pointer
*	Множення	expression * expression
/	Ділення	expression / expression
%	Взяття по модулю	expression % expression
+	Додавання (плюс)	expression + expression
-	Віднімання (мінус)	expression expression

Для зміни порядку виконання операцій у виразах використовуються круглі дужки. Проте тут є особливості.

Наприклад:

c = (d = 1) + 2; Тут компактно записано d =1; c = d + 2;

Ще один приклад a = (b = c / (d * e)) + f.

Такі записи можливі тому, що присвоєння в мові С розглядається не як оператор, а як операція.

Наприклад, операцію присвоєння i=i+2 можна записати i+=2. І взагалі, для багатьох бінарних операцій можливі операції виду op= , де op - операція. Тому діє правило e1 op=e2, де e1, e2 – вирази, а op - операція означає **e1=(e1) op (e2)**

Тут дужки необхідні.

Наприклад:

y*=z+1 еквівалентно **y=y*(z+1)**, а не **y=y*z+1**.

Знаки операцій збільшення й зменшення можуть стояти як ліворуч, так і праворуч від операнда **i++** або **++i**. Для одного операнда це рівнозначно. Але в операціях присвоєння при цьому з'являються певні відмінності.

Наприклад:

a=++b і **a=b++** це не тотожність.

1. b++ a=b

2. a=b b++

Операції збільшення й зменшення можна застосовувати лише до цілих змінних, а не до виразів **(a+b)++** - НЕ МОЖНА.

В арифметичних виразах операція розподілу виконується з урахуванням формату змінних.

Для цілих і символьних величин існує операція залишку від ділення %:

5%3 буде 2.

Відзначимо, що % **не можна** використовувати для чисел із плаваючою крапкою.

2.1.7 Особливості виконання арифметичних виразів і операції присвоєння

В операціях присвоєння із правої сторони у виразах можуть траплятися операнди різних типів і тип результату виразу може не збігатися з типом змінної з лівої сторони. Тому виникає питання, а яким буде остаточний результат?

Послідовність дій наступна:

1. Спочатку обчислюється вираз праворуч;

2. Остаточний результат визначається змінною ліворуч.

Для визначення результату виразів діють такі правила:

1. Якщо у виразі операнди тільки одного типу, то результат має той же самий тип;

2. Якщо операнди різного типу, то результат має "вищий тип".

Тобто у мові С типи даних мають певну ієрархію:

символьні (char)< цілі (int) < довгі (long int) < дійсні (float) < подвійної точності (double) < довгої подвійної точності (long double).

Це означає, що перед виконанням виразу всі операнди перетворюються в старший тип, тобто відбувається підвищення типу. Після обчислення результату виразу відбувається перетворення типів при виконанні операцій присвоєння. І яким би не був результат виразу, він перетворюється до типу зміною ліворуч. Відбувається начебто зниження типу.

Наприклад:

```
char c1, c2,c3;
int i1, i2, i3;
float f1, f2, f3;
c1='x'; // нема перетворень
c2=1000; // ціла відсікається до символу
c3=6.02e23; // плаваюча відсікається до символу
i3='x'; // символ розширюється до цілого
i3=6.02e23; // плаваюча відсікається до цілого
```

Розглянемо c2=1000; 100010= 17508=11111010002=1508=64+40=10410

Це код літери h.

Як бачимо, у деяких випадках, особливо при участі дійсних чисел одержуємо безглузді результати. Проте цей засіб можна використовувати для перетворення типів.

Відзначимо, що з погляду мови Паскаль ці присвоєння взагалі на можуть бути неможливі. Проте в мові С вони припустимі, але слід бути обережним при їхньому використанні. Крім неявного перетворення типів можна використовувати функцію явного перетворення, яка має вигляд:

(тип) <вираз>

наприклад:

```
int t1,t2;
float f1,f2=3.6;
t1=f2;// t1=
t2=(int) 1.6+7; // t2=8
f1=(int) 1.6; // f1=1.0
```

2.2 **Лабораторна робота №3.** Лінійні обчислювальні процеси

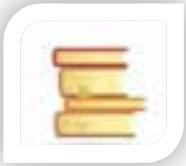
Тема: Лінійні обчислювальні процеси

Мета: отримання практичних навичок вирішення задач, пов’язаних з обчисленням значень по заданим формулам, використовуючи оператори простої послідовності та вводу/виводу.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Алгоритм – точно визначений опис способу вирішення задачі у вигляді кінцевої (у часі) послідовності дій.

Для представлення алгоритму у вигляді, зрозумілому комп'ютеру, є **мови програмування**, які є штучними мовами.

Спочатку розробляється алгоритм дій, потім він записується однією з мов програмування, в результаті, виходить текст програми, який є повним, закінченим і детальним описом алгоритму даною мовою програмування.

2. Состав та описання мови програмування

Алгоритмічна мова містить наступні елементи: ***символи***, ***елементарні конструкції*** (слова), ***вирази*** (словосполучення), ***оператори*** (речення).

Опис мови – це описання символів, елементарних конструкцій, виразів та операторів.

Опис символів полягає у переліку допустимих символів мови, які є основними неподільними знаками.

Під описом елементарних конструкцій розуміють правила їх утворення. Елементарні конструкції – це мінімальні одиниці мови, що мають самостійний зміст. Вони утворюються із основних символів мови.

Опис виразів – це правила утворення будь-яких висловів, які мають сенс у цій мові. Вираз в алгоритмічній мові складається з елементарних конструкцій та символів, він задає правило обчислення деякого значення.

Опис операторів складається з усіх типів операторів, допустимих у мові. Оператор визначає повний опис деякої дії, яку необхідно виконати. Для опису складної дії може знадобитися група операторів. У цьому випадку оператори об'єднуються у складний оператор або блок.

Дії, задані операторами, виконуються над даними. Речення алгоритмічної мови, в яких даються відомості про типи даних, називаються описами або операторами, що не виконуються.

Описання кожного елементу мови задається його синтаксисом та семантикою.

Тип даних – це форма представлення існуючих та оброблюваних у мові програмування даних (цілі, дійсні числа, символи, строки (рядки) тощо).

Тип даних визначає:

- ***можливий діапазон значень*** констант, змінних, виразів, функцій, які належать до даного типу;
- ***внутрішню форму представлення даних*** у комп'ютері: з фіксованою або плаваючою точкою;
- **обсяг пам'яті**, який займають дані;
- **операції та функції**, які можуть виконуватися над величинами, що належать до цього типу.

У програмі на основі типів даних описуються (оголошуються) змінні, що позначаються символами (ідентифікаторами), кожній з них виділяється пам'ять і ставиться у відповідність форма подання даних. Таким чином, змінна – це іменована область пам'яті програми, в якій розміщені дані у певній формі представлення (певного типу), для її використання необхідно виконати опис (оголошення) змінної.

Базові типи даних – це стандартні форми представлення даних у комп'ютері, які перенесено у мову програмування.

Цілочисельні типи даних (числа з фіксованою точкою).

int *i; // ціле число зі знаком, яке займає у пам'яті стандартне машинне слово – 2 байта, діапазон значень від -32768 до 32767*

long *l; // ціле число зі знаком, подвійне слово – 4 байта*

short *s; // ціле число зі знаком, коротке слово або байт*

char *c; // ціле число зі знаком, байт, діапазон значень від -128 до 127*

Дійсні типи даних (числа з плаваючою точкою).

Тип ***float*** використовується загалом при вводі-виводі, займає у пам'яті 4 байта.

Тип ***double*** (8 байт) забезпечує стандартну точність обчислень в арифметичних виразах, тому будь-яка змінна типу *float* перед використанням у

виразі автоматично перетворюється в *double*. Окрім того, якщо в операції присутня одна змінна типу *double*, а інша є цілим числом, то остання також перетворюється (приводиться) к *double*.

Представлення символьних даних

Представлення і обробка символьної інформації в *C* основані на використанні базового цілочисельного типу даних ***char***, кожний байт якого може зберігати або двійкове число, або символ тексту, за яким стандартами закріплено значення байту, яке називається ***кодом символу***.

Тип *void.*

Множина значень типу void порожня; використовується для визначення функцій, які не повертають жодного значення, для вказівки порожнього списку аргументів функції та операції приведення типів.

Програма на мові С є рядком символів, що складається з лексичних елементів (лексем): констант (літералів), зарезервованих слів, ідентифікаторів, знаків операцій та обмежувачів (розділювачів).

// коментарі

// Глобальні оголошення

#<директиви_препроцесора> // приклад, *#include <stdio.h> або "myfile.h"*

<прототипи_використаних_функцій>; // мають вигляд: *<тип_функції> <ім'я_функції>(список_формальних_параметрів);*

<оголошення_зовнішніх_змінних>; // має вид: *extern <им'я_змінної>;*

<тип_головної_функції> main(<список_формальних_параметрів>); // наприклад, *int main(void)* – функція, з якої починається виконання програми

```
{
  <оголошення_змінних>;
  <послідовність_операторів>;
}
<тип_функції> <им'я_функції_1>(<список _формальних_параметрів>)
{
  <оголошення_змінних>;
  <послідовність_операторів>;
}
...
```

Операції є елементарними конструкціями і є діями, які можуть бути виконані над змінними базових типів даних.

Група операцій, що послідовно виконуються (елементарних конструкцій) над змінними (позначаються символами) утворює вираз, який є правилом обчислення значення, наприклад, a+b.

Операндами називаються змінні, константи, вирази, що беруть участь в операції.

Унарною операцією називається операція над одним операндом.

Бінарною операцією називається операція над двома операндами.

Операції і вирази являють собою ***безумовну послідовність дій*** (табл. 2.5).

Таблиця 2.5. Знаки операцій

Операції	*Призначення операцій*
( ) [ ] -> .	операції найвищого пріоритету, які використовуються при виклику функцій, індексуванні елементів масиву, операції вибору компонентів структурованого об'єкта (прямий та опосередкований)
*! + - ++ -- & **	унарні операції: логічне заперечення, позитивне значення, зміна знаку, збільшення на одиницю, зменшення на одиницю, отримання адреси, звернення за адресою
** / %*	мультиплікативні бінарні операції: множення, ділення, отримання залишку від поділу цілих операндів
+ -	адитивні бінарні операції: додавання, віднімання
<< >>	операції порозрядного зсуву: ліворуч, праворуч
< <= >= > == !=	операції відношення: менше, менше або дорівнює, більше або дорівнює, більше, дорівнює, не дорівнює
& ^ |	логічні бінарні операції: порозрядні кон'юнкція, виключне АБО, диз'юнкція
&& ||	логічні бінарні операції: кон'юнкція, диз'юнкція
?:	умовна тернарна операція (три операнда)
*= *= /= %= += -= &= ^= |= <<= >>=*	операції присвоєння: просте, після відповідної операції
,	операції групування обчислень зліва направо

Операції присвоювання

До операцій присвоювання відносяться всі операції, які змінюють значення одного з операндів.

Групи операцій присвоювання:

- *звичайне присвоювання* (=);

- *присвоювання, поєднане з однією з бінарних операцій* (+=, -=, *=, /=, %=, <<=, >>=, &=, |=, ^=);
- *операції інкременту* (++) *і декременту* (--) – збільшення та зменшення на одиницю.

Наприклад, якщо оголосити змінні:

int a,b,c;

тоді можна використати такі оператори присвоювання:

a=b=c; // еквівалентно: b=c; a=b;

a +=c; // еквівалентно: a=a+c;

c++; // отримати c та збільшити на 1 після використання: c=c+1

++c; // збільшити c на 1 до використання: c=c+1

b--; // отримати b та зменшити на 1 після використання: b=b-1

--a; // зменшити a на 1 до використання: a=a-1

Арифметичні операції: *, /, % (остаток від ділення), **+, -**

Приклад,

```
a=(a+5)%3;        // a набуде значення остаток від ділення a+5 на 3
b=a--;
d=b/a;
```

Операції порівняння і логічні операції.

У мові С відсутній базовий тип даних для представлення логічних значень (*false, true*), тому використовують цілочисельні значення: *0* завжди є *хиба*, *1 – істина*. Такі значення дають операції відношення і логічні операції.

Операції порівняння: **<, <=, >, >=**

Логічні операції: **==, !=, &&, ||**

Всі операції порівняння дають в якості результату значення *1* або *0*, тому їх можна використати разом з арифметичними або іншими операціями:

```
a=b<c;  // запам'ятати результат порівняння
a=(b<c)*3  // набуває значення 0 або 3
```

Умовна операція дозволяє вбудувати у будь-який вираз умовний оператор, позначається як ***? :*** та означає:

<умова> ? <вираз_для_істини> : <вираз_для_хиби>

Якщо оголосити

int a,b,c,d;

тоді

```
c=d-a>b?a:b;  // c=a, якщо d-a>b, інакше, c=b
```

Операції явного перетворення типу. Операція перетворення (приведення) типу дозволяє перетворити значення операнду до заданого типу. В якості

операнду використовується унарний вираз, який може бути змінною, константою або виразом.

Формат операції перетворення типу:

(<тип>)<операнд>;

Наприклад,

```
int x;
float y;
y=(float)x/3;
```

Логіка алгоритму і оператори

Алгоритм – це формальне однозначне описання послідовності дій над даними. ***Дані*** є змінними, які створюються з урахуванням типів даних та обробляються алгоритмом.

Логіка алгоритму складається з операторів. **Оператор** є дію, що є програмною одиницею, тобто. це – структурна одиниця програми.

Одним із способів представлення алгоритмів є блок-схеми, в яких дії позначаються певними графічними символами, а їх виконання лініями зі стрілками.

Оператори простої послідовності **дій** не змінюють послідовності свого виконання, яка збігається з природним порядком проходження операторів у програмі. Операторами простої послідовності реалізуються лінійні алгоритми.

Основним джерелом операторів простої послідовності у програмі є вираз. Вираз, обмежений символом «;», перетворюється на оператор. Символ ";", що зустрічається в програмі, позначає порожній оператор, який не виконує жодних дій і передає керування наступному оператору.

Будь-яка послідовність операторів, які розташовані у фігурних дужках (*{}*), може виступати у довільній синтаксичній конструкції як один ***складений оператор (блок)***. Оператори, які складають блок, виконуються послідовно один за одним.

Функції форматного вводу/виводу даних

Для використання функцій форматного вводу і виводу даних необхідно застосувати ***директиву препроцесора*** (програмного інструменту, який змінює код програми для наступної компіляції і зборки), за якою в текст програми вставляється заголовочний файл, який містить описання відповідних функцій:
#include <stdio.h>

Функція форматного вводу даних

scanf*(<керуюча_строка>, <аргумент_1>, <аргумент_2>,...);*

Якщо в якості *аргументу використовує*ться змінна, то перед її іменем записується ***символ взяття адреси – &***, який вказує, що значення заноситься за

адресою змінної; *керуюча_строка* містить специфікації перетворення та використовується для встановлення кількості типів аргументів; кожна *специфікація перетворення* починається зі знаку *%* та закінчується певним символом, що задає перетворення; *символ перетворення* пов'язано з типом змінних.

Символи перетворень:

для цілих чисел

- *%d* (значення аргументу у десятковій формі);
- *%u* (значення аргументу у десятковій формі без знаку);
- *%o* (значення аргументу у вісімковій формі без знаку);
- *%x* (значення аргументу у шістнадцятковій формі без знака);

для дійсних чисел

- *%f* (значення аргументу у формі з плаваючою точкой);
- *%e* (значення аргументу в експоненціальній формі);

для символьних даних

- *%c* (значення аргументу – символ);
- *%s* (значення аргументу – строка символів);
- *%p* (значення аргументу – вказівник).

- **Функція форматного виводу даних**
- ***printf****(<керуюча_строка>, <аргумент_1>, <аргумент_2>, ...);*

Керуюча_строка містить об'єкти трьох типів: звичайні символи, які просто виводяться на екран, специфікації перетворення, кожна з яких визиває вивод на екран значення поточного аргументу з подальшого списку і керуючі символи-константи. Кожна *специфікація перетворення*, як і в функції форматного вводу, починається із знаку *%* і закінчується також *символом перетворення*. Якщо після знаку *%* записано не символ, то він виводиться на екран. Функція *printf* використовує керуючу строку, щоб визначити, скільки всього аргументів та які їх типи.

- **Приклад організації форматного вводу/виводу даних**

```
#include <stdio.h> // директива препроцесора підключення заголовочного файлу, який містить описання використаних у програмі функції вводу/виводу
#include <conio.h> // директива препроцессра підключення заголовочного файлу, який містить описання таких функцій, як getch() – очікування натискання будь-якої клавіші
int main()
{
int in; // оголошення змінної цілого типу
```

- *float fl; // оголошення змінної дійсного типу*
- *char ch; // оголошення змінної символьного типу*
- *printf("\nВвод цілого числа\n"); // переход на другу строку, вивод інформації та переход на другу строку*
- *scanf("%d", &in); // введене число заноситься у змінну in за її адресою*
- *printf("Ввод чисел\n");*
- *scanf("%f %c",&fl,&ch); // введені числа заносяться в переменные x и ch по их адресам*
- *printf("Вивод чисел %d %f %c",in,fl,ch); // вивід на екран вказаної строки с заміною специфікацією перетворення на значення числа*
- *getch(); // очікування натискання будь-якої клавіші*
- *return 0; // повернення з функції*
- *}*
- **Основні стандартні функції**

Стандартні функції консольного вводу/виводу

Консольний ввод/вивід (табл.2.6) реалізовано як функції та оголошено у *include*-файлі ***<conio.h>***.

Таблиця 2.6 Консольний ввод/вивід

ФУНКЦІЯ	ВИКОРИСТАННЯ
Cgets	зчитує строку з консолі
Cputs	записує строку на консоль
Getch	зчитує символ з консолі
Putch	записує символ на консоль

•

Стандартні математичні функції (табл.2.7)

Стандартні математичні функції існують для полегшення програмування математичних обчислень; функції працюють із значеннями, які представлено у формі з плаваючою точкою. *Основні математичні функції* оголошені у *include*-файлі ***<math.h>***.

Таблиця 2.7 Стандартні математичні функції

ФУНКЦІЯ	ВИКОРИСТАННЯ
acos(x)	обчислює *arccos x*
asin(x)	обчислює *arcsin x*
atan(x)	обчислює *arctg x*
atan2(y,x)	обчислює *arctg(y/x)*
ceil(x)	знаходить цілу частину *x*
cos(x)	обчислює *cos x*
exp(x)	обчислює експоненціальну функцію

fabs(x)	обчислює абсолютне значення *x*
floor(x)	знаходить найбільше ціле, яке менше або дорівнює *x*
fmod(x,y)	знаходить залишок з плаваючою точкою від ділення *x* на *y*
log(x)	обчислює натуральний логарифм *x*
log10(x)	обчислює десятковий логарифм *x*
modf(x,&n)	розділяє *x* на цілу (повертаєме значення) та дробну (*n*) частини
pow(x,y)	обчислює *x* ступеня *y*
sin(x)	обчислює *sin x*
sqrt(x)	знаходить квадратний корінь з *x*
tan(x)	обчислює *tg x*

Практична частина

Завдання

Задача 1.

1. Постановка задачі: розробити алгоритм обчислення і виводу на екран значення *a* за формулою

$$a = \frac{\sqrt{|x-1|} - \sqrt{|y|}}{1 + \frac{x^2}{2} + \frac{y^2}{4}}$$

Вхідні значення *x, y* вводяться з клавіатури. Написати програму, яка реалізує розроблений алгоритм.

2. Математична модель та описання алгоритму задачі:

- ввести значення у змінні *x, y*;
- обчислити значення змінної *a за* заданою формулою без перевірок введених значень *x, y*, оскільки значення *a(x,y)* визначено для будь-яких значень аргументів;
- вивести отримане значення змінної *a*.

3. Блок-схема алгоритму задачі (рис. 2.4)***:***

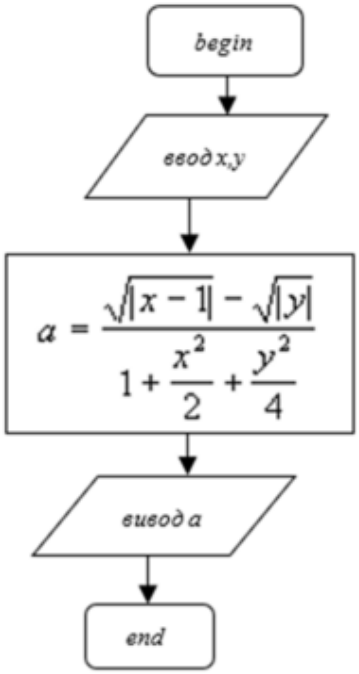


Рис. 2.4 - Блок-схема алгоритму вирішення задачі

4. Текст програми – реалізація алгоритму на мові С:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main()
{
int x, y; // описання змінної цілого типу
float a; // описання змінної дійсного типу
printf("\n input x=");
scanf("%d",&x);
printf("\n input y=");
scanf("%d",&y);
a=(sqrt(abs(x-1))-sqrt(abs(y)))/(1+(float)x*x/2+(float)y*y/4);          //
використовується перетворення типів для представлення цілих значень x, y у
формі з плаваючою крапкою
printf("\n output a=%f",a);
getch();
return 0;
}
```

5. Тестування:

Теоретично розраховані вихідні значення	*Практично отримані вихідні значенні*
Тест 1: *вхідні дані: x=1, y=-1*	
a≈-0,5714	*a=-0.571429*
Тест 2: *вхідні дані: x=-2, y=1*	
a≈0,2252	*a=0.225246*

Задача 2.

1. Постановка задачі: розробити програму обчислення і виводу на екран значень функції $y = f_1(x)$ та $z = f_2(y,a,b)$ для вхідних даних x, a, b робочого набору. Значення робочого набору, для якого обидві функції визначені, вводяться з клавіатури. Написати програму, яка реалізує розроблений алгоритм.

№	Функція $y = f_1(x)$	Функція $z = f_2(y,a,b)$	Робочий набір		
			x	a	b
1	$\frac{\lvert\lg x\rvert+5}{x+4}$	$\frac{\sin^2(a(2y^2+1))+29b}{\sin^2(a(2y^2+1))+b}$	0,2	2	1/2

2. Математична модель та опис алгоритму задачі:

- ввести значення робочого набору у змінні x, a, b;
- обчислити значення змінної y по формулі $y = \frac{|\lg x|+5}{x+4}$;
- формулу обчислення значення змінної $z = \frac{\sin^2\left(a\left(2y^2+1\right)\right)+29b}{\sin^2\left(a\left(2y^2+1\right)\right)+b}$ можна спростити, для чого використати допоміжну змінну $c=\sin^2(a\cdot(2\cdot y^2+1))$;
- обчислити значення змінної z через змінну c за формулою $z = \frac{c+29b}{c+b}$;
- вивести отримане значення змінних y, z.

3. Блок-схема алгоритму задачі (рис. 2.5):

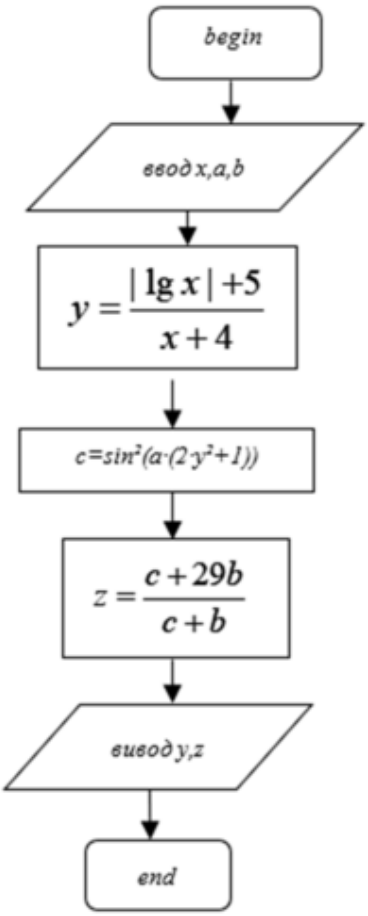


Рис. 2.5 - Блок-схема алгоритму вирішення задачі

4. Текст програми:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main()
{
float x, a, b, y, c, z; // описання змінних дійсного типу
printf("\n input x=");
scanf("%f",&x);
printf("\n input a, b:");
scanf("%f%f",&a,&b);
y=(fabs(log10(x))+5)/(x+4);
c=pow(sin(a*(2*y*y+1)),2);
z=(c+29*b)/(c+b);
```

```
printf("\n output y=%f\tz=%f",y,z);
getch();
return 0;
    }
```

5. Тестування:

Теоретично розраховані вихідні значення	*Фактично отримані вихідні значення*
Для робочого набору вхідних даних: x=0,2, a=2, b=12	
y≈1,3569; z≈28,991594	*y=1,356898; z=28,991587*

Виконати індивідуальне завдання

1. Постановка задачі

Розробити лінійний алгоритм та написати програму за індивідуальним завданням, використовуючи оператори простої послідовності дій, оператор вводу для вводу значень вхідних даних та оператор виводу для виводу результатів обчислення виразу.

2. Вхідні і вихідні дані.

Усі діючі у програмі змінні повинні бути оголошені.

Неприпустимим є задавати вхідні(початкові) дані з допомогою операторів присвоювання. Ввод даних з клавіатури повинен здійснюватися після виводу відповідного повідомлення (запрошення).

3. Математична модель та алгоритм задачі.

4. Блок-схема алгоритму.

Представити лінійний алгоритм у вигляді блок-схеми.

5. Текст програми.

Розроблений алгоритм реалізувати мовою програмування високого рівня С.

6. Тестування.

Результати тестування представити у вигляді таблиці.

Варіанти індивідуальних завдань (табл. 2.8)

1. Розробити лінійний алгоритм обчислення *a* або *b* за заданою формулою та вивід на екран отриманого значення; вхідні значення *x, y* або *x, y, z* вводяться з клавіатури. Написати програму, яка реалізує розроблений алгоритм, та здійснити її тестування.

Таблиця 2.8 Варіанти завдань:

1	$a=(1+y)\dfrac{x+\dfrac{y}{(x^2+4)}}{e^2+\dfrac{1}{(x^2+4)}}$	***2***	$b=1+\lvert y-x\rvert+\dfrac{(y-x)^2}{2}+\dfrac{(x-y)^2}{3}$
3	$b=\dfrac{1+\cos^3(y-x)}{\dfrac{x^2+1}{2}+\sin^2 z}$	***4***	$a=y+\dfrac{x}{y^2+\dfrac{x^2}{\lvert 1+y^2+x^2\rvert}+1}$
5	$a=\dfrac{2\cos^4(x-\frac{\pi}{6})}{\dfrac{1}{2}+\sin^2 y}$	***6***	$b=1+\dfrac{z^2}{3+\dfrac{z^2}{5}}$
7	$b=x-\dfrac{x^2}{3}+\dfrac{x+y}{x^2+1}$	***8***	$a=\dfrac{1+\sin^2(x+y)}{2+\left\lvert x-\dfrac{2x}{(1+x^2y^2)}\right\rvert}+x$
9	$a=\dfrac{x^2}{8+\dfrac{x^2}{3}+\dfrac{y^2}{6}}$	***10***	$b=x(\cos^3(x+z)+1)$
11	$b=\lvert y-4\rvert+\dfrac{(y-x)^2}{6}+\dfrac{(x-y)^2}{7}$	***12***	$a=(2+x)\dfrac{1+\dfrac{y}{(x^2+3)}}{y^2+\dfrac{1}{(z^2+4)}}$
13	$a=\dfrac{4\sin(x-\frac{\pi}{3})}{\dfrac{1}{3}+\sin^2 x}$	***14***	$b=(1+\cos^2\dfrac{z}{2})^2$
15	$b=\dfrac{z^2}{3+\dfrac{z^2}{5}}$	***16***	$a=\dfrac{\sqrt{\lvert x-1\rvert}-\sqrt{\lvert y\rvert}}{1+2x^2+\cos(x)}$
17	$a=\dfrac{1+\cos^2(x+y)}{2+\left\lvert x-\dfrac{2x}{(1+x^2z^2)}\right\rvert}+x$	***18***	$b=\dfrac{\sqrt{x^2+1}-2xy}{2-\lvert\sin(2x)\rvert}$
19	$b=(\sin^2 x-2\cos(2x))^3$	***20***	$a=\dfrac{\cos(x+\frac{\pi}{4})}{1-\sin(x)+\lvert 2x+1\rvert}$
21	$a=\sqrt{\dfrac{z^2+x^4+2}{5x^2+0{,}5}}$	***22***	$b=\dfrac{22z-3x^3+1}{\lvert 2x^2-3z^4-\lvert 2y\rvert\rvert+1}$
23	$b=\sqrt{1+\dfrac{\lvert 2xz-y\rvert}{4x^2+1}}-10^4$	***24***	$a=5\cos(3y)-\sin(z-2y)-\dfrac{1-3\cos(y)}{\lvert\sin^2(z)-2\rvert}$
25	$a=\dfrac{3xyz}{x^2+2}+\sqrt{2x^2+1}$		

2. Розробити лінійний алгоритм обчислення та виводу на екран значень функцій $y = f_1(x)$ та $z = f_2(y,a,b)$ для вхідних даних x, a, b робочого набору, для яких дані функції визначені і які вводяться з клавіатури (табл. 2.9). Написати програму, яка реалізує розроблений алгоритм, та здійснити її тестування.

Таблиця 2.9 Варіанти завдань

№	Функція $y = f_1(x)$	Функція $z = f_2(y,a,b)$	Робочий набір		
			X	A	B
1	$\frac{\sqrt{x^2+16}}{x+2}$	$\frac{y+\sqrt{\sin a+3}+b}{y^2+\sqrt{\sin a+3}}$	3,5	1,8	3,7
2	$\frac{e^{x-2,7}+3}{x+1,3}$	$\frac{y+0,75\lvert\cos b\rvert+a}{y^2+0,75\lvert\cos b\rvert}$	8,2	2,2	8,2
3	$\frac{\sin x+1,5}{2}$	$\frac{y^3+\sqrt{\sqrt{a}+3,3}}{b+\sqrt{\sqrt{a}+3,3}}$	8,1	0,8	1,2
4	$\frac{\ln(x-3)+4}{x^2+12}$	$\frac{\sqrt[3]{y+7}+a}{\sin b+\sqrt[3]{y+7}}$	4,7	7,6	8,1
5	$\frac{\lvert x\rvert+8}{x^3+18}$	$\frac{\sqrt[4]{y+15}+a}{\cos b+\sqrt[4]{y+15}}$	3,4	82	2,5
6	$\frac{\cos^2(x)+2}{3}$	$\frac{y\sqrt[5]{a}+1}{\sin b+y\sqrt[5]{a}}$	-8	8,7	1,3
7	$\frac{e^{x+3,1}+2}{x+6,1}$	$\frac{\sqrt[3]{a}+2y+tgb+3}{tgb+2y+3}$	2,5	8,7	1,8
8	$\frac{\sqrt{e^{x-2}+3}}{x}$	$\frac{\sqrt[4]{a}+\sqrt{5y+20}}{\sqrt{5y+20}+b}$	2,7	17	11
9	$\frac{tgx+3,73}{4}$	$\frac{7y+3\sin a+\sqrt{b^2+19}}{7y+\sqrt{b^2+19}+2}$	0,1	1,5	10
10	$\frac{\sin^3(x)+3,7}{5}$	$\frac{\sqrt{14y+2}+6}{\sqrt{14y+2}+\cos b+a}$	2,5	5	6,1
11	$\frac{\sqrt{x+12}}{2x^3+1}$	$\frac{\lvert y^2-a\rvert+6}{2\cos b+\lvert y^2-a\rvert+6}$	18	-3	8,1
12	$\frac{\sqrt[3]{x+8,3}}{x+0,3}$	$\frac{4+y^2+\lvert\sin x\rvert+a}{\lvert\sin x\rvert+y^2+0,2b}$	3,7	-2	8,1

13	$\frac{1+\ln(x+5,3)}{x+5,3}$	$\frac{\sqrt{y+15\sin a}}{\sqrt{y+15\sin a}+2b}$	*2*	*2*	*3*
14	$\frac{e^{x-1,5}+2}{2x+0,3}$	$\frac{\sqrt[4]{27y+54}+a}{\sqrt[4]{27y+54}+\lvert\cos b\rvert+1}$	*4,1*	*9*	*3,5*
15	$\frac{\lvert\sin x\rvert+2}{3}$	$\frac{\sqrt[3]{y+7a}+b}{\sin b+1+\sqrt[3]{y+7a}}$	*2,5*	*1,3*	*3,3*
16	$\frac{2}{\sqrt{\cos x+5}}$	$\frac{\sqrt[3]{y+13a}+5}{\cos b+\sqrt[3]{y+13a}}$	*6,1*	*2,3*	*2,6*
17	$\frac{\sqrt{\lvert\cos x+3\rvert}}{3}$	$\frac{\sin b+\sqrt[4]{y+15a}}{\sqrt[4]{y+15a}}$	*8*	*1,3*	*2,5*
18	$\frac{\sqrt[3]{x-3,1}}{x-27}$	$\frac{(y+1)^2+5a}{\sin b+(y+1)^2+5a}$	*80*	*0,8*	*-2*
19	$\frac{3e^{x-2}}{x+1}$	$\frac{a\sqrt[3]{y+2b}}{2-\cos b+\sqrt[3]{y+2b}}$	*6,1*	*8*	*9,2*
20	$\frac{\sin^2(x)+5}{5}$	$\frac{\sqrt[3]{ay+57}}{3+\lvert\cos b\rvert+\sqrt[3]{ay+57}}$	*-2*	*7,3*	*5,1*
21	$\frac{x-7}{\ln(x-2)+2}$	$\frac{\sqrt[4]{ay^2+3}+2}{\sqrt[4]{ay^2+3}+b}$	*10*	*23*	*1,1*
22	$\frac{\sqrt{\cos^2 x+10}}{5}$	$\frac{\lg(y^2+8)+5\sin a}{\lg(y^2+8)+\lvert\cos b\rvert}$	*5,2*	*2,5*	*7,2*
23	$\frac{\lg(17-2x)+2}{x+1}$	$\frac{\lvert\cos a\rvert(y+3)}{\lvert\cos a\rvert(y+3)-b}$	*0,6*	*5*	*2,1*
24	$\frac{e^{2x-7,4}+6}{x+4,3}$	$\frac{\lvert\sin a\rvert(y+7)}{\lvert\sin a\rvert(y+7)+2b}$	*5*	*-2*	*0,7*
25	$\frac{3\sin x+21}{\cos 2x+25}$	[illegible]	*3,5*	*14*	*7*

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.

5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке «базові типи даних»?
2. Які існують цілочисельні типи у мові С? Скільки байтів пам’яті займає тип *int*?
3. Назвати дійсні типи мови С? Скільки байтів пам’яті займає тип *float*?
4. Як представляються символьні дані у мові С?
5. Що являє собою тип *void*?
6. Що являє собою програма на мові С? Яка структура С-програми?
7. Що таке «операція» та що таке «операнд»? Яка операція називається унарною, а яка – бінарною?
8. Що являє собою вираз?
9. Які арифметичні операції використовуються у мові С?
10. Чи існує логічний тип у мові С? Які значення використовуються для представлення логічних значень?
11. Які використовують операції порівняння та які – логічні операції?
12. Які операції відносяться до операцій присвоювання?
13. Які групи операцій присвоювання існують у мові С? Навести приклади.
14. Чи є оператор програмною одиницею?
15. Які групи основних керуючих конструкцій являють собою оператори?
16. Що таке «пустий оператор» та для чого він використовується?
17. Які існують оператори простої послідовності?

РОЗДІЛ ІІІ. СТРУКТУРА І ПРИКЛАД ПРОГРАМИ

3.1 Теоретичні відомості. Структура і приклади програми

3.1.1 Функції та програми

Програма мовою С складається з окремих функцій. Функції знаходяться на певному рівні ієрархії в мові С (рис. 3.1). Кожна функція включає заголовок і тіло.

Рис.3.1 - Функції в мові С

Тип_функції ім'я_функції (тип змінна, тип змінна, ...)

{
Тіло функції;
[Return (вираз);]
}

де **тип змінна**– список формальних параметрів.

Заголовок функції - це ім'я й у круглих дужках список формальних параметрів. Поряд із усіма іншими функціями програма обов'язково повинна мати головну функцію main ().

Тому будемо завжди починати з головної функції **main ()**. По всьому тексту програми можна розміщати коментарі:

/* обчислення кореня */ - блоковий коментар або // закінчення циклу -

рядковий коментар.

Тіло функції завжди починається й закінчується операторними дужками { }, які є аналогом BEGIN і END у мові Паскаль.

Оскільки складений оператор це є також блок, то фігурні дужки використовуються й для складених операторів.

Тіло функції складається з окремих операторів, кожний з яких обов'язково закінчується роздільником ";", незалежно від того, що за ним іде далі. Згадаємо, що в мові Паскаль ; після оператора можна було й не ставити, якщо далі йде роздільник (END).

Як відомо, у Паскалі оператори розміщалися за розділом оголошень. Тут же описи можуть бути й до початкової дужки, і після й ще далі.

Складений оператор - це послідовність операторів, обмежених фігурними дужками. Іншими словами - це блоки. Розміщення операторів у рядку - довільне. Будемо розміщати дужки, які відкриваються й закриваються, на одній вертикалі, а внутрішні оператори - трошки праворуч. Одержимо текст із відступами, який краще сприймається.

Розташування програми в ієрархії мови С наведено на рис. 3.2.

Рис. 3.2 - Програма в мові С

Приклад програми наведено нижче.

```
/* арабське число пишеться римськими літерами */
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
//розділ опису глобальних змінних та функцій
long roman(long,int,char); //заголовок функції
void main(void)
{ long num;
  int rnum[7]={1000,500,100,50,10,5,1};
  char symb[7]={'M','D','C','L','X','V','I'};
  do
  {clrscr(); //очищення екрана
   printf("1000-M 500-D 100-C 50-L 10-X 5-X 1-
I\n");
   print(" Введіть натуральне число (арабське)->
");
   if(!(scanf("%ld",&num))||(num<0)) //
перевірка правильності введення
   { printf("\n Помилка вводу ");
     printf("\n Для продовження натисніть будь-
яку клавішу ");
     getch();
     num=0;
     fflush(stdin); // очищення буфера введення
   }
  } while (!num);
  printf(" Римське позначення числа -> ");
  // цикл визначення римских позначень
  for (int i=0;i<=6;i++)
  {   num=roman(num,rnum[i],rsymb[i]);  }
  getch();
} // завершення основної функції
// опис функції
long roman(long n1,int n2,char symb)
{
 while (n1>=n2)
 {
  putch(symb); // виводить символ на екран
  n1-=n2;
 }
 return (n1);
}
```

3.1.2 Директиви

1. Директива **#include**

Директива #include включає в текст програми зміст вказаного файлу. Ця

директива має дві форми:

#include "имя файла"

#include <имя файла>

2. Директива **#define**

Директива #define служить для заміни констант, що часто використовуються, ключових слів, операторів або виразів деякими ідентифікаторами.

Директива #define має дві синтаксичні форми:

#define ідентифікатор текст

#define ідентифікатор (список параметрів) текст

Приклад:

#define WIDTH 80

#define LENGTH (WIDTH+10)

#define MAX(x,y) ((x)>(y))?(x):(y) // t=MAX(i,s[i])
 t=((i)>(s[i])?(i):(s[i]);

3. Директива **#undef**

Директива #undef використовується для скасування дії директиви #define. Синтаксис цієї директиви наступний #undef ідентифікатор

Приклад:

#undef WIDTH #undef MAX

Ці директиви скасовують визначення іменованої константи WIDTH і макровизначення MAX.

3.2 Лабораторна робота 4. Умовні конструкції: оператори розгалуження

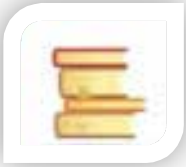
Тема: Умовні конструкції: оператори розгалуження

Мета: отримання практичних навичок вирішення задач з використанням умовних конструкцій.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Умовний оператор дозволяє перевіряти певну умову та, в залежності від результатів перевірки, виконати ту або іншу дію.

Структури умовного оператору:

1. ***неповна форма***

```
if (<умова>)
    <оператор>;
```

<умова> – логічний вираз; якщо він істина (*=1*), то виконується *<оператор>*, який стоїть після нього, інакше виконується наступний оператор після умовного оператору (після *;*).

2. ***полна форма***

```
if (<умова>)
    <оператор_1>
else
    <оператор_2>;
```

Якщо *<умова>* істина (*=1*), то виконується *<оператор_1>* (який стоїть після *умови*), інакше виконується *<оператор_2>*, який стоїть після *else*; далі виконується наступний оператор після умовного оператору (після *;*).

<оператор>, *<оператор_1>*, *<оператор_2>* можуть представляти собою один оператор або групу операторів, які заключено у фігурні дужки ({}).

Оператор множинного вибору (перемикач). ***Оператор вибору***, порівнюючи значення заданого виразу з набором перелічених значень, дозволяє вибрати одно з декількох можливих продовжень програми.

Структура оператора вибору:

switch (<вираз>)
{
case <значення_виразу_1>: <оператор_1>; break;
case <значення_виразу_2>: <оператор_2>; break;
...
[default: <оператор_n>; break;]
}

<вираз> в операторе *switch* – значення любого простого типу;

<значення_вираз>, які вказано в операторах *case,* сумісні за типом з *<вираз>* в операторі *switch* та обов'язково повинні відрізнятися одне від одного.

У випадку рівності *<вираз> <значення_виразу_1>*, виконується *<оператор_1>;* інакше, для випадку рівності *<вираз> <значення_выраз_2>*, виконується *<оператор_2>*; інакше, і так далі до останнього оператора *case*;

інакше, якщо *<вираз>* не відповідає жодного значення в операторах *case*, то керування передається *<операторy_n>* після ключового слова *default*, яке є необов'язковим.

Якщо відсутній оператор *default*, то виконується наступний оператор після оператору *switch* (після *}*).

Всередині оператору *switch* оператор *break* призводить до передачі керування наступному оператору, який стоїть після оператору *switch*, тобто використовується для виходу з перемикача. Якщо *break* відсутній, то після поточного розділу *case* буде виконуватись наступний оператор у операторі *switch* (*case* або *default*).

Практична частина

Завдання

Виконання загального завдання.

Задача 1.

1. Постановка задачі: розробити алгоритм обчислення і виводу на екран значення функції

$$y = \begin{cases} x+2, \text{ якщо } x<0; \\ x^3+5, \text{ якщо } x\geq\pi/2; \\ \sin x+0,5, \text{ якщо } 0\leq x<\pi/2; \end{cases}$$

Значення x вводиться з клавіатури. Написати програму, яка реалізує розроблений алгоритм.

2. Математична модель та опис алгоритму задачі:

- ввести значення змінної x;
- якщо $x<0$, то $y=x+2$;
- інакше, якщо $x\geq\pi/2$, то $y=x^3+5$;
- інакше, якщо $0\leq x<\pi/2$, тобто у всіх інших випадках, $y=\sin x+0,5$;
- вивести значення змінної y.

3. Блок-схема алгоритму задачі (рис. 3.3):

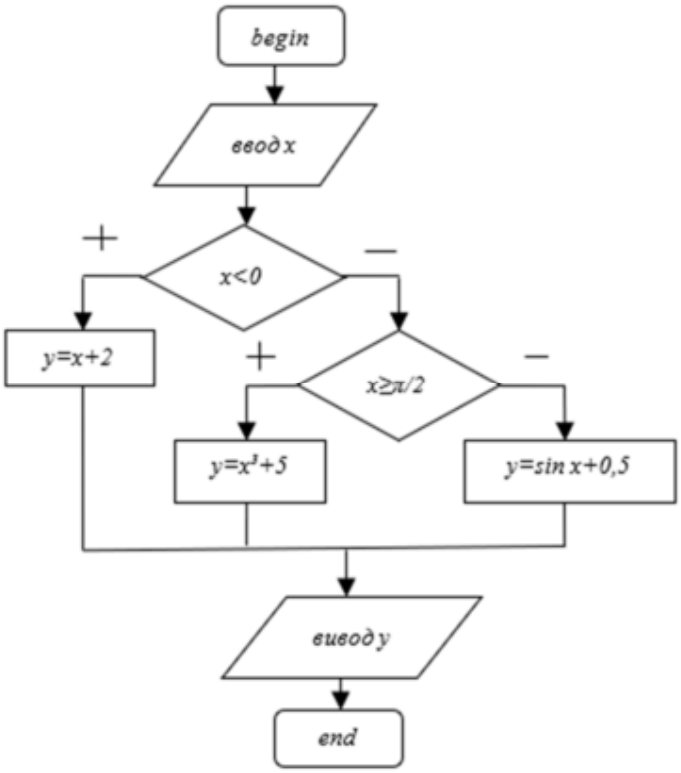


Рис. 3.3 - Блок-схема алгоритму вирішення задачі

4. Текст програми:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main()
```

```
{
 const float p=3.14;
 float x,y;
 printf("\n input x:");
 scanf("%f",&x); // ввод числа у формі з плаваючою точкою
 if (x<0)
    y=x+2;
 else
    if (x>=p/2)
       y=pow(x,3)+5;
    else
       y=sin(x)+0.5;
 printf("\n output y=%f",y);
 getch();
 return 0;
}
```

5. Тестування:

Теоретично розраховане вихідне значення	*Практично отримане вихідне значення*
Тест 1: *вхідні дані: x=-1 (умова x<0)*	
y=?	*y=?*
Тест 2: *вхідні дані: x=π≈3,14 (умова x≥π/2)*	
y=?	*y=?*
Тест 3: *вхідні дані: x=0 (умова 0≤x<π/2)*	
y=?	*y=?*
Тест 5: *вхідні дані: x=π/3≈1,0466 (умова 0≤x<π/2)*	
y=?	*y=?*

Задача 2.

1. Постановка задачі: розробити алгоритм обчислення та виводу на екран значення функції *b* за формулою $b = \frac{\sqrt{x^4 - 1} + 3xy}{1 - \cos(x)}$. Значення *x, y* вводяться з клавіатури. Написати програму, яка реалізує розроблений алгоритм.

2. Математична модель та описання алгоритму задачі:

- ввести значення у змінні *x, y*;
- оскільки функція *b(x,y)* визначена не на всіх значеннях *x*, *y*, то слід визначити область допустимих значень (ОДЗ): $x^4-1 \geq 0$ и $1-\cos(x) \neq 0$
- перевірити умови: якщо $x^4-1 \geq 0$ та $1-\cos(x) \neq 0$, то обчислити значення змінної

b за формулою $b = \dfrac{\sqrt{x^4 - 1} - 3xy}{1 - \cos(x)}$ та вивести отримане значення;

- інакшу вивести інформацію про те, що вхідні значення не задовольняють ОДЗ та вийти з алгоритму.

3. Блок-схема алгоритму задачі (рис. 3.4)***:***

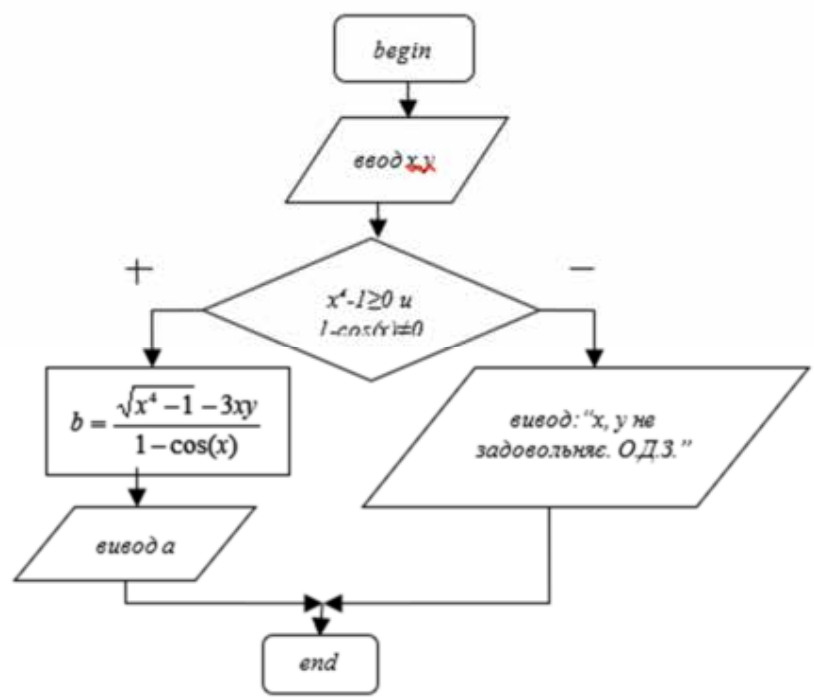


Рис. 3.4 - Блок-схема алгоритму вирішення задачі

4. Текст програми:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main()
{
float x,y,b;
printf("\n input x, y:");
scanf("%f%f",&x,&y);
if (pow(x,4)-1>=0 && 1-cos(x)!=0)
{
   b=(sqrt(pow(x,4)-1)-3*x*y)/(1-cos(x));
   printf("\n output b=%f",b);
}
else
   printf("\nx не задовольняє ОДЗ");
getch();
```

```
return 0;
    }
```

5. Тестування:

Теоретично розраховані вихідні значення	*Практично отримані вихідні значення*
Тест 1: *вхідні дані: x=1.5, y=3 (умова x^4-1≥0 та 1-cos(x)≠0)*	
b=?	*b=?*
Тест 2: *вхідні дані: x=0,3 (умова x^4-1<0 або 1-cos(x)==0)*	
x не задовольняє ОДЗ	*x не задовольняє ОДЗ*

Виконання індивідуального завдання

1. Постановка задачі.

Розробити розгалужений алгоритм та написати програму за індивідуальним завданням, використовуючи оператори розгалуження, оператор введення для введення значень вхідних даних та оператор виведення для виведення результату обчислення.

2. Вхідні і вихідні дані.

Усі змінні, що діють у програмі, мають бути оголошені.

Не можна задавати вихідні (вхідні) дані за допомогою операторів присвоювання. Введення даних з клавіатури слід здійснювати після виводу відповідного повідомлення.

3. Математична модель та описовий алгоритм вирішення задачі.

4. Блок-схема алгоритму.

Представити розгалужений алгоритм у вигляді блок-схеми.

5. Текст програми.

Розроблений алгоритм реалізується мовою програмування високого рівня С.

6. Тестування.

Результати тестування представити у вигляді таблиці.

Варіанти індивідуальних завдань

Розробити алгоритми вирішення задач, написати програми, які реалізують відповідні алгоритми, та здійснити їх тестування. Вхідні дані вводяться з клавіатури.

Варіант №1

1. Обчислити і вивести на екран значення функції

$$y = \begin{cases} \cos^2 x, & \text{якщо } 0<x<2 \\ 2\cdot\sin^2 x, & \text{якщо } x<0 \\ 1, & \text{якщо } x\geq 2 \end{cases}$$

2. Обчислити за формулою $b = \dfrac{1+\cos^3(y-x)}{\dfrac{x^2-1}{2}-\sin^2(z)}$ і вивести на екран значення b.

3. Обчислити і вивести на екран значення площі геометричної фігури, яке відповідає введеному значенню *n*:

$$S=\begin{cases} a\cdot b, & если \quad n=1 \\ a\cdot\dfrac{h}{2}, & если \quad n=2 \\ (a+b)\cdot\dfrac{h}{2}, & если \quad n=3 \\ \pi\cdot R^2, & если \quad n=4 \\ \pi\cdot R^2\cdot\dfrac{a}{360}, & если \quad n=5 \end{cases}$$

Варіант №2

1. Обчислити і вивести на екран значення функції

$$f(x)=\begin{cases} \dfrac{a}{|a-|b-x||} \\ 1, \quad если \quad знаменатель=0 \end{cases}$$

2. Вычислити за формулою $a=y+\dfrac{x}{y^2+\dfrac{x^2}{|1+y^2-x^2|}-1}$ і вивести на екран значення *a*.
3. Обрахувати і вывести на екран значення площі геометричної фігури, яка відповідає введеному значенню *k*:

$$S=\begin{cases} p\cdot l, & если \quad k=1 \\ p\cdot\dfrac{h}{2}, & если \quad k=2 \\ 2\pi\cdot R\cdot h, & если \quad k=3 \\ 2\pi\cdot R\cdot l, & если \quad k=4 \\ \pi\cdot R^2, & если \quad k=5 \\ \pi\cdot R\cdot(2h+a), & если \quad k=6 \end{cases}$$

Варіант №3

1. Обчислити і вивести на екран значення функції

$$y=\begin{cases} -2\cdot x+x+1, & якщо\ x\le 0 \\ -2\cdot x^2+3, & якщо\ 0<x<10 \\ 0, & якщо\ x\ge 10 \end{cases}$$

2. Обчислити за формулою $b=x-\dfrac{x^2}{y}+\dfrac{x+y}{x^2-1}$ і вивести на екран значення *b*.
3. Розрахувати остаток від ділення цілої частини значення функції $z=ln(x^2+a\cdot b)$ на *7*, в залежності від його значення, вивести повідомлення про один з днів тижня, пронумерувавши їх від *0* до *6*.

Варіант №4

1. Обчислити і вивести на екран значення функції *y(x)*, якщо при введеному $x>1$ $y=x^2$, а при $x\le 1$ $y=x$.
2. Розрахувати за формулою $b=\dfrac{\sqrt{x^2-1}-2xy}{1-\sin(2x)}$ і вивести на екран значення *b*.

3. Ввести три числа *a, b, c,* які задовольняють аксіомі трикутника і, в залежності від значення введеного *p*, вивести отримане значення або інформацію: при *p=1* обчислити периметр трикутника, при *p=2* – площа трикутника, при *p=3* – кут між сторонами *a* та *c*, інакше, вивести слово «*трикутник*».

Варіант №5

1. Знайти і вивести максимальне значення з двох введених чисел *a, b*; якщо числа рівні, то вивести відповідну інформацію.
2. Обчислити за формулою $a = \frac{\cos(x + \frac{\pi}{4})}{1 - \sin(x) + 2x}$ і вивести на екран значення *a*.
3. Розрахувати остаток від ділення цілого виразу *a=(c+d)·(2·k-m)* на *5*; при залишку, який дорівнює *0*, вивести значення *a*, при непарному залишку вивести «*непарне число*», при парному – «парне *число*».

Варіант №6

1. Є точка з введеними координатами *x, y*. Присвоїти *z=1*, якщо точка всередині еліпсу

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

та *z=0*, якщо точка зовні еліпсу; вивести значення *z*.

2. Обчислити за формулою $a = \sqrt{\frac{z^2 + x^3 - 1}{2x^2 - 0{,}5}}$ і вивести на екран значення *a*.
3. За введеним номером комп'ютеру вивести прізвище студента, який сидить за ним.

Варіант №7

1. Є два відрізки *[a,b]*, *[c,d]* на прямій; *a, b*, *c, d* вводяться з клавіатури. Встановити, чи мають відрізки спільні точки, та вивести «т*ак*» або «*ні*».
2. Розрахувати за формулою $b = \frac{22z - 3x^3 + 1}{2x^2 - 3z^3 - |2y| + 1}$ і вивести на екран значення *b*.
3. Обчислити залишок від ділення цілої частини виразу $z=cos(x^2+1)\cdot 10\cdot x$ на *4* і, в залежності від величини залишку, вивести повідомлення про одну з пор року, пронумерувавши їх від *0* до *3*.

Варіант №8

1. Ввести координати точки *x, y*; присвоїти *z=1*, якщо точка належить колу з введеним радіусом *R* і центром у точці з введеними координатами *a, b*, та присвоїти *z=0* – у протилежному випадку; вивести значення *z*.
2. Обчислити за формулою $b = \sqrt{1 - \frac{2xz - y}{4x^2 - 1}} - 10^4$ та вивести на екран значення *b*.
3. Розрахувати залишок від ділення цілого виразу *c=k(a+b)* на *4* та вивести на екран значення залишку і вираз: якщо залишок дорівнює *0*, то значення виразу залишити без змін, якщо – *1* або *3*, то зменшити на величину залишку, якщо – *2*, то збільшити на величину залишку.

Варіант №9

1. Обчислити і вивести на екран значення функції *f(x),* якщо при введенні *x≥0 f(x)=e^-x*, а при *x<0 f(x)=cos(x)*.
2. Розрахувати за формулою $a = 5\cos(3y) - \sin(z-2y) - \frac{1-3\cos(y)}{\sin^2(x)-1}$ та вивести на екран значення *a*.
3. За введеним номером дня тижня (*1,2,3,4,5,6,7*) вивести його назву з позначкою робочий чи вихідний день.

Варіант №10

1. Розв'яжіть рівняння $y = \frac{1}{x}$ та виведіть на екран значення *y*.
2. Обчисліть за формулою $a = \frac{3xyz}{x^2-2} + \sqrt{2x^2-1}$ та виведіть на екран значення *a*.
3. Розрахувати залишок від ділення цілої частини виразу *sin(a+b)0.5·c* на *4* та вивести на екран значення залишку і виразу: якщо залишок дорівнює *0*, то значення виразу замінити на його цілу частину, якщо остаток дорівнює *1* – замінити на його дробову частину, в інших випадках – залишити без змін.

Варіант №11

1. Розрахувати та вивести на екран значення кореня $\sqrt{4x-6}$.
2. Обчислити за формулою $a = \sin(2y) - \cos(z-y) + \frac{2-\sin(z)}{2\sin^2(2x)-1}$ та вивести на екран значення *a*.
3. В залежності від введеного номеру пори року (весна – 1, літо – 2, осінь – 3, зима – 4) вивести «*тепло*», «*жарко*», «*холодно*», «*дуже холодно*».

Варіант №12

1. Обчислити і вивести на екран значення функції *y=lg(3·x-6).*
2. Розрахувати за формулою $b = \sqrt{1 - \frac{xz+y}{2x^2-3}} - 2xy * 10^3$ та вивести на екран значення *b*.
3. Обчислити та вивести значення функції, в залежності від введеного *n*:

$$y = \begin{cases} ax^2 + bx + c, & при \quad n = 1 \\ ax^3 + 3bx + b^3, & при \quad n = 2 \\ 4\sin^2 x + \cos^2 x, & при \quad n = 3 \\ \sqrt{|x-y|} \cdot \cos^2 x, & при \quad n = 4 \\ ctg^2 x, & при \quad n = 5 \end{cases}$$

Варіант №13

1. Ввести координати *(a,b)* та *(c,d)* точок, вивести на екран координати тієї з точок, яка розташована ближче до початку координат.
2. Розрахувати за формулою $a = \frac{3xy-z}{x^2-y} + \sqrt{2x^2y-5}$ та вивести на екран значення *a*.
3. Обчислити та вивести значення функції, в залежності від введеного *k*:

$$y = \begin{cases} e^2 \sin x \cdot tg^2 x, & при \quad k = 1 \\ \pi \cdot R^2, & при \quad k = 2 \\ \dfrac{4}{3\pi R} + 2.1, & при \quad k = 3 \\ (a \cdot \cos(bx))^2 & при \quad k = 4 \end{cases}$$

Варіант №14

1. Вивести на екран повідомлення, в якому з квадрантів координатної площини знаходиться точка з координатами *x, y*, якщо *xy≠0*.
2. Розрахувати за формулою $b = \dfrac{2zy - x^3 + 1}{x^2 - 2z^4 - |y - 1|}$ та вивести на екран значення *b*.
3. Обчислити залишок від ділення цілої частини виразу *cos(a-b)є* на *3* та вивести на екран значення залишку та вирази: якщо залишок дорівнює *0*, то значення виразу замінити на його цілу частину, якщо залишок дорівнює *1* – замінити на його дробову частину, в інших випадках – залишити без змін.

Варіант №15

1. Якщо сума двох введених чисел менше одиниці, то найменше замінити напівсумою, у протилежному випадку – найменше замінити сумою; вивести на екран отримане значення.
2. Розрахувати за формулою $a = \dfrac{\sin(x + \dfrac{\pi}{3}) + 1}{1 - \sin^2(x) + 4x}$ та вивести на екран значення *a*.
3. Обчислити залишок від ділення цілого виразу *c=k(a-b)·a* на *3* та вивести на екран значення залишку та вирази: якщо залишок дорівнює *0*, то значення виразу залишити без змін, якщо – *1*, то зменшити на величину залишку, якщо – *2*, то збільшити на величину залишку.

Варіант №16

1. Ввести три дійсних числа; якщо сума чисел більше добутку на значення, менше одиниці, то вивести «*0»,* у протилежному випадку, вивести «*1»*.
2. Обчислити за формулою $b = \dfrac{\sqrt{x^2 - 2} + xy}{1 - \sin^3(2x)}$ та вивести на екран значення *b*.
3. Обчислити залишок від ділення цілого виразу *n(x+y)·2* на *5* та вивести на екран значення залишку і вирази: якщо залишок дорівнює *0*, то значення виразу залишити без змін, якщо – *1* або *3*, то зменшити на величину залишку, якщо – *2* або 4, то збільшити на величину залишку.

Варіант №17

1. Обчислить та вивести на екран значення функції *z*

$$y = \begin{cases} max(x,m), & якщо\ x<0 \\ min(x,m), & якшо\ x \geq 0 \end{cases}$$

2. Розрахувати за формулою $b = \dfrac{1 - \cos^2(y - x)}{\dfrac{x^2 + 1}{4} - \sin(z)}$ та вивести на екран значення *b*.

3. Ввести три числа *x, y, z,* які задовольняють аксіомі трикутника та, в залежності від значення введеного *k*, вивести отримане значення: при *k=1* обчислити периметр трикутника, при *k=2* – площа трикутника, при *k=3* – кут між сторонами *x* та *y*, при *k=4* – кут між сторонами *y* и *z*, інакше – кут між сторонами *x* и *z*.

Варіант №18

1. Є відрізки *[a,b]* та *[c,d]* і точка з координатою *x*. Вивести на екран повідомлення про приналежність даної точки одному або обом відрізкам, або вона лежить зовні відрізків.
2. Розрахувати за формулою $b = x - \frac{x^2 - 1}{y + 1} + \frac{x - 2y}{2x^2 - 1}$ та вивести на екран значення *b*.
3. Обчислити залишок від ділення цілої частини виразу $lg(y^3 - a \cdot b) - 1$ на *7*, в залежності від його значення, вивести повідомлення про один з днів тижня, пронумерувавши їх від *0* до

Варіант №19

1. Ввести два дійсних числа *x* та *y*; якщо найменше з них від'ємне, то замінити його нулем, у протилежному випадку – одиницею; вивести отримане значення.
2. Розрахувати за формулою $b = \frac{\sqrt{3x^2 - 2} + xy}{1 - \cos(x)}$ та вивести на екран значення *b*.
3. Ввести три числа *a, b, c,* які задовольняють аксіомі трикутника та, в залежності від значення введеного *n*, вивести отримані значення: при *n=1* обчислити кут між сторонами *a* та *b*, при *n=2* – площу трикутника, інакше – периметр трикутника.

Варіант №20

1. Вивести інформацію про існування трикутника з введеними сторонами *a, b, c*; якщо він існує, то визначити і вивести інформацію: чи є трикутник рівностороннім, рівнобедреним або різностороннім.
2. Розрахувати за формулою $a = \frac{\cos(2x + \frac{\pi}{2})}{1 + \cos(2x) + 2x}$ та вивести на екран значення *a*.
3. Обчислити залишок від ділення цілого виразу *(c+d)(k-m)* на *6*; при залишку, який дорівнює *0*, вивести значення виразу, при непарному залишку вивести «*непарне число*», при парному – «парне *число*».

Варіант №21

1. Ввести дійсні числа *x, y*; якщо *x* та *y* від'ємні , то *x* присвоїти модуль *x*; якщо одно з них – від'ємне, то збільшити *y* на *0.5*; якщо обидва числа – позитивні, то збільшити *x* у *10* разів; вивести змінене значення.
2. Розрахувати за формулою $a = \sqrt{\frac{3z^2 + x + 2}{x^2 - 1}}$ та вивести на екран значення *a*.
3. За введеним номер студентського квитка вивести прізвище студента.

Варіант №22

1. Є три дійсних числа *x, y, z* та відрізок *[a,b]*; замінити на нулі ті числа, які належать відрізку та на одиниці – інші; вивести значення *x, y, z*.
2. Розрахувати за формулою $b = \frac{2z - x^3}{x^2 - yz^3 - 2y} + 10^4$ та вивести на екран значення *b*.
3. Обчислити залишок від ділення цілої частини виразу $sin(3x^2+1)\cdot 20$ на *7* та, в залежності від величини залишку, вивести повідомлення про один з днів тижня, пронумерувавши їх від *0* до *6*.

Варіант №23

1. Ввести координати точки *x, y*; присвоїти *z=1*, якщо точка належить колу з введеним радіусом *r* та центром на початку координат, та присвоїти *z=0* – у протилежному випадку.
2. Розрахувати за формулою $b = \sqrt{1 - \frac{2xz^2 - 5y}{x^2 y - 1}} - 1$ та вивести на екран значення *b*.
3. Обчислити залишок від ділення цілого виразу $3\cdot k(2\cdot a+b)$ на *5* та вивести на екран значення залишку та вирази: якщо залишок дорівнює *0*, то значення виразу залишити без змін, якщо – *1* або *2*, то зменшити на величину залишку, якщо – *3*, то збільшити на величину залишку; якщо – *4*, то збільшити у 2 рази.

Варіант №24

1. Обчислити та вивести на екран значення функції *f(x),* якщо при введеному $x \geq 1$ $f(x)=e^{-x}+x$, при $0<x<1$ $f(x)=sin(2x)$, інакше $f(x)=1$.
2. Розрахувати за формулою $a = 2\sin(2y) + \frac{1 - \cos(y)}{\sin^2(xy) - 1}$ та вивести на екран значення *a*.
3. За введеним номер місяця (*1,2,3,4,5,6,7,8,9,10,11,12*) вивести його назву.

Варіант №25

1. Вивести мінімальне та максимальне значення з трьох введених чисел *x1, x2, x3*.
2. Розрахувати за формулою $a = \frac{xyz - 1}{x^2 - 3} - \sqrt{2x + 1}$ та вивести на екран значення *a*.
3. Обчислити залишок від ділення цілої частини виразу $cos(a-b)\cdot c$ на *3* та вивести на екран значення залишку та виразу: якщо залишок дорівнює *0*, то значення виразу замінити на його цілу частину, якщо залишок дорівнює *1* – замінити на його дробову частину, в інших випадках – залишити без змін.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.

4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке *умовна конструкція*?
2. Структури умовного оператору.
3. Як працює умовний оператор?
4. Що являє собою структура оператора множинного вибору?
5. Як працює оператор множинного вибору?

РОЗДІЛ IV. КЛАСИ ПАМ'ЯТІ. ЛОГІЧНІ ВИРАЗИ. КЕРУЮЧІ СТРУКТУРИ

4.1 Теоретичні відомості. Класи пам'яті. Логічні вирази. Керуючі структури

4.1.1 Класи пам'яті

Кожна змінна крім типу визначається своєю областю дії й часом існування (протягом виконання всієї програми, або лише в окремому блоці). Для підвищення ефективності використання пам'яті програма складається з 4 частин (сегментів):

1. Сегмента *коду*, де розміщаються ***коди програм***;

2. Сегмента *даних*, де розміщаються глобальні й статичні ***змінн***і, які існують протягом усього часу виконання програми;

3. Сегмента *стека*, де розміщаються локальні й реєстрові ***змінні***; 4. Додаткового *сегмента*.

Крім цього, є ще п'ята область - для запису ***динамічних змінних - "куча" (heap).***

Стек - це область пам'яті, у якій запис походить від більших адрес до менших, а зчитування у зворотному напрямку. Інакше говорять, що реалізується дисципліна - перший прийшов, останнім обробився.

Клас пам'яті визначає місце, де розташований об'єкт (внутрішні регістри процесора, сегмент даних, сегмент стека) і одночасно час існування.

У мові С існують 4 класу пам'яті:

1. Автоматична (auto);
2. Реєстрова (register);
3. Статична (static);
4. Зовнішня (extern).

ВІДПОВІДНО до класу пам'яті, де розміщається змінна, можна називати й змінну -автоматичної, реєстрової, статичної або зовнішньої.

4.1.2 Автоматичні змінні

Описуються в блоці таким чином

```
{ auto int a=123;
auto char b;
auto float c=45.28;
}
```

Зазвичай слово **auto** пропускається. Область дії автоматичних змінних - у межах блоку, після виходу із блоку їх значення стають невизначеними (усі що завгодно). Ініціалізація автоматичних змінних відбувається щораз при вході в блок, тобто в процесі виконання. Тому int **a=123** еквівалентно **int a; a=123**;

Як видно, автоматична змінна є локальною, існує тільки по необхідності й не може бути змінена іншими функціями.

Якщо в деякому блоці втримуються внутрішні блоки, то область дії автоматичних змінних, описаних у зовнішньому блоці, поширюється й на внутрішні. Тобто у внутрішньому блоці їх можна не описувати.

Якщо ж у внутрішньому блоці автоматична змінна вживається в іншому розумінні, то при виконанні дій внутрішнього блоку діють його опис, а не зовнішнього блоку.

```
main()
/ / Початок зовнішнього блоку { int x=1; char z='w';
if (x=1)
     { // внутрішній блок
        int y=2;
        float z=-345.5674;
        cout<<" y= "<<y<<" z= "<<z<<endl;
     }
printf (" x=%d zf=%e z=%c\n", x,z,z);
getch();
}
```

Буде написано: **y=2 z= -345.567413**

x=1 zf=-7.4204e+41 z=w // ‘м’

4.1.3 Реєстрові змінні

Як відомо, дані можуть розміщуватись як в оперативній пам'яті, так і в регістрах. Використання регістрів зменшує кількість пересилань і прискорює виконання програми. Тому можливе використання реєстрових змінних, опис яких має вигляд

```
{register int x=2;
}
```

Однак оскільки мовам високого рівня регістри недоступні, про їхній стан відомо тільки компіляторові, то використання регістрів буде можливо тільки тоді, коли є вільний регістр, і якщо регістр може вмістити відзначену змінну.

Інакше компілятор розмістить цю змінну в оперативній пам'яті й вона буде звичайною автоматичною змінною. Тому використання реєстрових змінних недоцільне.

Як видно, автоматичні й реєстрові змінні мають локальний характер, тому

їх розміщують у сегменті стека.

4.1.4 Статичні змінні

Статичні змінні задаються описанням

```
{ static int a;
static float b=3.15;
}
```

Як і для автоматичних, область дії статичних змінних є блоком. Відрізняються вони від автоматичних тим, що після виходу із блоку їх значення зберігається. Воно буде початковим при наступному вході в цей блок. Наприклад:

```
void lvars ()
{ int z=0;
      z++;
      printf (“ z=% d\n”,z);
 }
main ()
{lvars();
  lvars();
  lvars();
}
```

другий варіант (static int z=0**)**

```
void statvars ()
{ static int z=0;
   z++;
   printf (“ z=% d\n”,z);
 }
main()
{statvars ();
  statvars ();
  statvars ();
}
```

Ці дві функції **lvars()** і **statvars()** відрізняються тим, що в першої z є автоматичною, а в другій - статичною змінними. Це приведе до того, що в першому випадку змінна z щораз буде одержувати значення 0, і тому три рази буде надрукована одиниця.

У другому випадку z є статичною. Вона буде мати значення z=0 лише один раз і тому буде надруковано 1, 2, 3.

Відзначимо, що якщо автоматичній змінній початкові значення надаються щоразу при вході до блоку, то статична змінна набуває початкового значення лише один раз - у процесі компіляції. Якщо початкове значення не визначено, то за **замовчуванням** воно **дорівнює нулю.**

4.1.5 Глобальні змінні

Глобальні змінні є глобальними й доступні будь-яким функціям. Вони визначаються поза функціями. Нижче приводиться дуже невеликий приклад закінченої програми, у якій рядок описується в одному файлі, а її друк здійснюється в іншому. Файл vars.h визначає необхідні типи.

```
// vars.h
extern char* sms;
extern void global();
```

В файлі main.cpp знаходиться головна програма:

```
// main.cpp
#include “vars.h”
#include “gl.h”
char *sms = “Тест глобальних змінних”;
int main()
{ global(); }
```

а файл gl.h друкує рядок:

```
// gl.h
#include <iostream.h>
#include “ vars.h”
void global( )
{  cout << sms << “\n”;  }
```

Вивід на екран: **Тест глобальних змінних**

Однак, якщо опис глобальних змінних передував певній функції, то він може бути опущений. Таким чином, опис глобальних змінних діє на всі функції, які розташовані нижче. Як і для статичних змінних, ініціалізація зовнішніх змінних відбувається один раз під час компіляції. За замовчуванням привласнюється значення нуль.

Статичні й глобальні змінні розміщаються в сегменті даних. Якщо програми складається з декількох файлів, то дія статичних змінних поширюється лише на один файл, де вони описані. ***Зовнішні змінні діють у межах декількох файлів, але їх описи повинні бути продубльовані.***

У міру можливості слід уникати використання в програмах глобальних змінних. Однак якщо ваша програма повинна використовувати глобальну змінну, то може трапитися, що ім'я глобальної змінної конфліктує з іменем локальної змінної. При виникненні такого конфлікту С надає пріоритет локальній змінній. Інакше кажучи, програма припускає, що у випадку конфлікту кожне посилання на ім'я відповідає локальній змінній.

Однак можуть бути ситуації, коли вам необхідно звернутися до глобальної змінної, чиє ім'я конфліктує з іменем локальної змінної. У таких випадках ваші програми можуть використовувати глобальний оператор дозволу З++ (::), якщо ви прагнете використовувати глобальну змінну. Наприклад, припустимо, що у вас є глобальна й локальна змінні з іменем number. Якщо ваша функція прагне використовувати локальну змінну number, вона просто звертається до цієї змінної, як показано нижче:

```
number = 1001; // звернення до локальної змінної
```

З іншого боку, якщо ваша функція прагне звернутися до глобальної змінної, програма використовує глобальний оператор дозволу, як показано нижче:

```
::number = 2002; // Звернення до глобальної змінної
```

Наступна програма. На додаток до цього функція *show_numbers* використовує локальну змінну з іменем *number*. Ця функція використовує оператор глобального дозволу для звертання до глобальної змінної:

```
#include <iostream.h>
int number = 1001; //  Глобальна змінна
void show_numbers(int number)
{
   cout << " Локальна змінна number містить число " << number << endl;
   cout << " Глобальна змінна number містить число " << ::number << endl;
}
 void main(void)
 {
   int some_value = 2002;
   show_numbers(some_value) ;
}
```

4.2 Лабораторна робота 5. Структура програми, основні типи даних, ввод/вивід

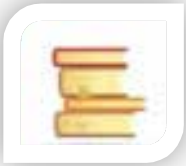
Тема: Структура програми, основні типи даних, ввод/вивід

Мета: вивчити структуру програми, навчитися використовувати змінні різних типів, опанувати функції форматованого вводу та виводу, арифметичні операції та операції присвоювання.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Програма на мові С є рядком символів, що складається з лексичних елементів (лексем): констант (літералів), зарезервованих слів, ідентифікаторів, знаків операцій та обмежувачів (розділювачів).

// коментарі

// Глобальні оголошення

#<директиви_препроцесора> // приклад, *#include <stdio.h> або "myfile.h"*

<прототипи_використаних_функцій>; // мають вигляд: *<тип_функції> <ім'я_функції>(список_формальних_параметрів);*

<оголошення_зовнішніх_змінних>; // має вид: *extern <им'я_змінної>;*

<тип_головної_функції> main(<список_формальних_параметрів>); // наприклад, *int main(void)* – функція, з якої починається виконання програми

```
{
  <оголошення_змінних>;
  <послідовність_операторів>;
}
<тип_функції> <им'я_функції_1>(<список _формальних_параметрів>)
{
  <оголошення_змінних>;
  <послідовність_операторів>;
}
```

...

Практична частина

Завдання

Набрати текст програми, яка наведена нижче. Проаналізувати призначення змінних після кожної операції присвоювання. Перевірити порядок виконання операцій у кожному виразі, який містить декілька операцій присвоювання, розділивши кожний оператор - вираз на декілька операторів, які виконуються послідовно. У функціях вводу та виводу змінити специфікатори формату, проаналізувати отримані результати.

```
#INCLUDE <STDIO.H>
    #include <stdlib.h>
INT MAIN (VOID)
{
    INT A, B = 5, C;
    DOUBLE X, Y = -.5, Z;
    PRINTF("A=");
    SCANF("%D", &A);
    X = C = A;
    PRINTF("X = C = A : A=%D C=%D X=%F\N", A, C, X);
    A += B;
    PRINTF("A += B : A=%D\N", A);
    X *= B+A;
    PRINTF("X *= B+A : X=%LF\N", X);
    B += A--;
    PRINTF("B += A-- : A=%D B=%D\N", A, B);
    X -= ++C;
    PRINTF("X -= ++C : C=%D X=%LF\N", C, X);
    C = A/B;
    PRINTF("C = A/B : C=%4D\N",C);
    C = A%B;
    PRINTF("C = A%%B : C=%D\N",C);
    Y += (A+1)/A++;
    PRINTF("Y += (A+1)/A++ : A=%D Y=%.3LF\TY=%.0LF\N", A, Y, Y);
```

```
    B = 3*(Y-=.6)+2*B+1;
    PRINTF("B = 3*(Y-=.6)+2*B+1 : B=%D Y=%.1LF\N", B, Y);
    Z = A/2;
    PRINTF("Z = A/2 : Z = A/2 : Z=%LF\N", Z);
    Z = (DOUBLE)A/2;
    PRINTF("Z = (DOUBLE)A/2 : Z=%LF\N", Z);
    Y = (X = 5.7)/2;
    PRINTF("Y = (X = 5.7)/2 : X=%LF Y=%LF\N", X, Y);
    Y = (INT)X/2;
    PRINTF("Y = (INT)X/2 : Y=%F\N", Y);
    Z = (B-3)/2 - X/5 +(C/=2) + 1/4*Z - Y++ + ++B/3.;
    PRINTF("Z = (B-3)/2 - X/5 +(C/=2) + 1/4*Z - Y++ + ++B/3. :\N\
    A=%D B=%D C=%D X=%LF Y=%LF Z=%LF\N", A,B,C,X,Y,Z);
    SYSTEM("PAUSE");
    RETURN 0;
}
```

2. Написати програму для обчислення значень наступних виразів:

a=5, c=5

a=a+b-2

c=c+1, d=c-a+d

a=a*c, c=c-1

a=a/10, c=c/2, b=b-1, d=d*(c+b+a)

Вирази, записані в одній строці, записувати одним оператором-виразом. Змінні c та d оголосити як цілі, змінні a та b – як дійсні. Значення змінних b та d слід вводити з клавіатури. Після обчислення кожного виразу виводити на екран значення всіх змінних.

3. Набрати текст програми, який наведено нижче. Проаналізувати видані програмою результати. Пояснити, чому вони саме такі.

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#include <float.h>
main()
{
   char c;
   unsigned char uc;
   int i;
   unsigned u;
```

```
    short s;
    long l;
    float f;
    double d;
    printf("sizeof(c)=%d\tsizeof(uc)=%d\nsizeof(i)=%d\tsizeof(u)=%d\t\
sizeof(s)=%d\tsizeof(l)=%d\nsizeof(f)=%d\tsizeof(d)=%d\n\n",
        sizeof(c),sizeof(uc),sizeof(i),sizeof(u),sizeof(s),
        sizeof(l),sizeof(f),sizeof(d));
    uc=c=CHAR_MAX;
    printf("CHAR_MAX : c=%d uc=%d\n", c, uc);
    c++; uc++;
    printf("CHAR_MAX+1 : c=%d uc=%d\n", c, uc);
    uc=c=CHAR_MIN;
    printf("CHAR_MIN : c=%d uc=%d\n", c, uc);
    c=uc=UCHAR_MAX;
    printf("UCHAR_MAX : c=%d uc=%d\n", c, uc);
    c++; uc++;
    printf("UCHAR_MAX+1 : c=%d uc=%d\n", c, uc);
    uc=c=-5;
    printf("-5 : c=%d uc=%d\n", c, uc);
    c=-5; uc=5;
    printf("char and unsigned char -5>5 : %d\n\n", c>uc);
    c=s=SHRT_MAX;
    uc=s;
    printf("SHRT_MAX : c=%d uc=%d s=%d\n", c, uc, s);
    s++;
    printf("SHRT_MAX+1 : s=%d\n", s);
    c=s; uc=s;
    printf("%d : c=%d uc=%d\n", SHRT_MIN, c, uc);
    s=0; c=s; uc=s;
    printf("0 : c=%d uc=%d s=%d\n", c, uc, s);
    i=INT_MAX;
    l=i; u=i;
    printf("INT_MAX : i=%d u=%u l=%ld\n", i, u, l);
    i++; l++; u++;
    printf("INT_MAX+1 : i=%d u=%u l=%ld\n", i, u, l);
    i=INT_MIN;
    l=i; u=i;
```

```
printf("INT_MIN : i=%d u=%u l=%ld\n", i, u, l);
u=UINT_MAX;
i=u; l=u;
printf("UINT_MAX : i=%d u=%u l=%ld\n", i, u, l);
u=i=-5;
printf("-5 : i=%d u=%u\n", i, u);
i=-5; u=5;
printf("int and unsigned int -5>5 : %d\n", i>u);
c=-5; u=5;
printf("char and unsigned int -5>5 : %d\n\n", c>u);
i=5.1;
printf("i=5.1 : i=%d\n", i);
i=5.9;
printf("i=5.9 : i=%d\n", i);
d=f=FLT_MAX;
printf("FLT_MAX : f=%g d=%g\n", f, d);
d=f=FLT_MIN;
printf("FLT_MIN : f=%g d=%g\n", f, d);
d=f=FLT_EPSILON;
printf("FLT_EPSILON : f=%g d=%g\n", f, d);
f=1e10;
printf("1e10 : f=%f\n", f);
f=1e11;
printf("1e11 : f=%f\n", f);
f=1234567890;
printf("1234567890 : f=%f\n", f);
d=DBL_MAX;
printf("DBL_MAX : d=%g\n", d);
d=DBL_MIN;
printf("DBL_MIN : d=%g\n", d);
d=DBL_EPSILON;
printf("DBL_EPSILON : d=%g\n", d);
d=1e15+1;
printf("1e15+1 : d=%lf\n", d);
d=1e16+1;
printf("1e16+1 : d=%lf\n", d);
f=10000*100000;
f+=1;
```

```
    f-=4*250000000;
    printf("1 : f=%f\n", f);
    f=10000*100000+1-4*250000000;
    printf("1 : f=%f\n", f);
    d=10000*100000+1-4*250000000;
    printf("1 : d=%lf\n", d);
    d=1e20*1e20+1000-1e22*1e18;
    printf("1000 : d=%lf\n", d);
    system("pause");
    return 0;
}
```

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Яка структура програми на мові С?
2. Навіщо потрібна директива #include?
3. Що таке main()?
4. Перерахувати скалярні типи даних мови С.
5. Що визначає тип даних?
6. Що таке void?
7. Що таке явне і неявне приведення типів? Як і коли воно використовується?
8. Що таке константа? Знайдіть константи у наведених програмах.
9. Що таке змінна?
10. Як проініціалізувати змінну?
11. Чим відрізняється оператор від операції?
12. Чим відрізняються унарні операції від бінарних?
13. Які операції відносяться до арифметичних? Який пріоритет кожної з них?

14. Який порядок виконання операцій у випадку їх однакового пріоритету?
15. Як виконується операція ділення у випадку цілочисельних операндів та у випадку, коли хоч би один з операндів дійсний?
16. Що таке вираз?
17. Яке значення обчислює операція присвоювання?
18. В якому порядку виконується присвоювання у випадку, якщо у виразі їх декілька?
19. Як і навіщо використовуються додаткові операції присвоювання?
20. Чим відрізняються префіксна форма операції інкремента або декремента від постфіксної?
21. Які функції використовуються для вводу інформації? Назвіть їх відмінні особливості.
22. Які функції використовуються для виводу інформації? Назвіть їх відмінні особливості.
23. Чому функції scanf() та printf() називаються функціями форматного вводу та виводу? Як вони працюють?
24. Чим відрізняється керуюча строка функції scanf() від керуючої строки функції printf()?
25. Що таке специфікатор формату? Навіщо він потрібен?
26. Які параметри вказуються функцією scanf() після керуючої строки? Скільки їх повинно бути?
27. Які наслідки невідповідності типу зчитуваної функцією scanf() змінної специфікатору типу?
28. Які параметри вказуються функції printf() після керуючої строки? Скільки їх повинно бути?
29. Які наслідки невідповідності типу виводимого функцією printf() значення специфікатору типу?
30. Що таке керуючі символи? Навіщо вони потрібні? Наведіть приклади.

4.3 Лабораторна робота 6. Циклічні конструкції: оператори циклу

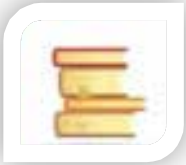
Тема: Циклічні конструкції: оператори циклу

Мета: отримання практичних навичок вирішення задач з використанням циклічних конструкцій.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

У циклічних конструкціях використовуються **оператори повторень**, за допомогою яких можна запрограмувати дії програми, що повторюються. Існують два **типи циклів**: з параметром (лічильником), який відноситься до циклів із передумовою, та ітераційні, які, у свою чергу, можуть бути як з передумовою, так і з постумовою.

Будь-який цикл можна розділити на 4 частини: ініціалізацію, умову, ітерацію та тіло циклу, яке складають від одного до декількох операторів.

Оператор циклу з параметром (лічильник) зазвичай використовується, коли число повторень тіла циклу наперед відоме. У мові С число повторень тіла циклу з параметром може також визначатися деякою умовою.

Таким чином, цикл з параметром працює поки значення параметра циклу, починаючи з початкового значення, не досягне кінцевого, заданого в умові або поки виконується певна умова. У будь-якому разі умова роботи циклу перевіряється до виконання операторів тіла циклу, тому цикл із параметром є циклом з передумовою.

В операторі циклу з параметром передбачені усі чотири частини циклу.

Структура оператору циклу з параметром:

for (<ініціалізація>; <умова>; <ітерація>)
<тіло_цикла>;

<ініціалізація> – оголошення та присвоювання початкових значень параметрам, що використовуються в циклі; *<умова>* – логічний вираз, який визначає умову

виконання циклу; <*ітерація*> – модифікація, яка виконується після кожного проходу циклу, слугує для зміни параметра циклу; <*тіло_циклу*> складають оператори, що виконуються в циклі.

Можна прибрати будь-яку частину циклу *for*, тоді <*ініціалізація*>, <*умова*> або <*ітерація*> будуть пустими операторами (*;*) без оператора або виразу перед ним.

Також одночасно можуть бути відсутніми всі частини циклу for (у тому числі і <тіло циклу>), тоді цикл буде нескінченним, і він ніколи не завершиться.

Всередині <ініціалізації> оператора for можна оголошувати змінні, які діятимуть у межах цього оператора.

У частинах <ініціалізація> та <ітерація> допускається вміст кількох операторів, тоді застосовується кома (,) всередині круглих дужок оператора for для поділу кількох операторів.

Ітераційні цикли використовуються, коли кількість повторень тіла циклу заздалегідь невідома і визначається умовою (виконання або завершення) циклу.

Оператор циклу з передумовою. Передумова обчислюється і перевіряється до виконання операторів тіла циклу, і якщо значення логічного виразу, що визначає умову роботи циклу, істинно (=1), цикл продовжується виконувати до тих пір, поки це значення не стане хибним *(=0)*.

Структура оператору циклу з передумовою:

[<ініціалізація>;]
while (<умова>)
{
<тіло_циклу>;
[<ітерація>;]
}

<умова> – логічний вираз, який визначає умову повторення тіла циклу; *<ініціалізація>* та *<ітерація>* – необов'язкові.

Якщо початкові умови такі, що при вході у цикл *<умова>* хибна, то *<тіло_циклу>* і *<ітерація>* не виконуються жодного разу.

Оператор циклу з постумовою. Постумова обчислюється і перевіряється після виконання операторів, що становлять тіло циклу, і якщо значення логічного виразу, що визначає умову роботи циклу (в мові Сі), істинно (=1), то цикл продовжує виконуватися до тих пір, поки це значення не стане хибним (= 0). У будь-якому випадку тіло циклу з постумовою виконається принаймні один раз.

Структура оператору циклу з постумовою:

[<ініціалізація>;]

```
do
{
   <тіло_циклу>;
   [<ітерація>;]
}
while (<умова>);
```

<умова> – логічний вираз, який визначає умову повторення тіла циклу; *<ініціалізація>* та *<ітерація>* – необов'язкові.

Практична частина

Завдання

Задача 1.

1. Постановка задачі: розробити алгоритм обчислення і виводу значення суми членів ряду $S=\sum_{i=1}^{n} x^i = x+x^2+x^3+...+x^{n-1}+x^n$ для $0<x\leq 5$ при $n=8$. Написати програму, яка реалізує розроблений алгоритм.

2. Математична модель та опис алгоритму задачі:

- ввести значення змінної *x*;
- перевірити умову: якщо $0<x\leq 5$, то продовжити виконання алгоритму, інакше вивести відповідне повідомлення і вийти з алгоритму;
- для формування суми використовувати змінну *S*, спочатку її слід обнулити – *S=0*;
- для отримання ступеня змінної *x* використати змінну *xs*, спочатку *xs=x*;
- у циклі з параметром (кількість повторень = *n*) кожен раз значення *S* збільшувати на значення *xs*, а значення *xs* збільшувати в *x* разів для отримання ступеня *x*;
- після виходу з циклу вивести значення змінної *S*.

3. Блок-схема алгоритму задачі (рис. 4.1)***:***

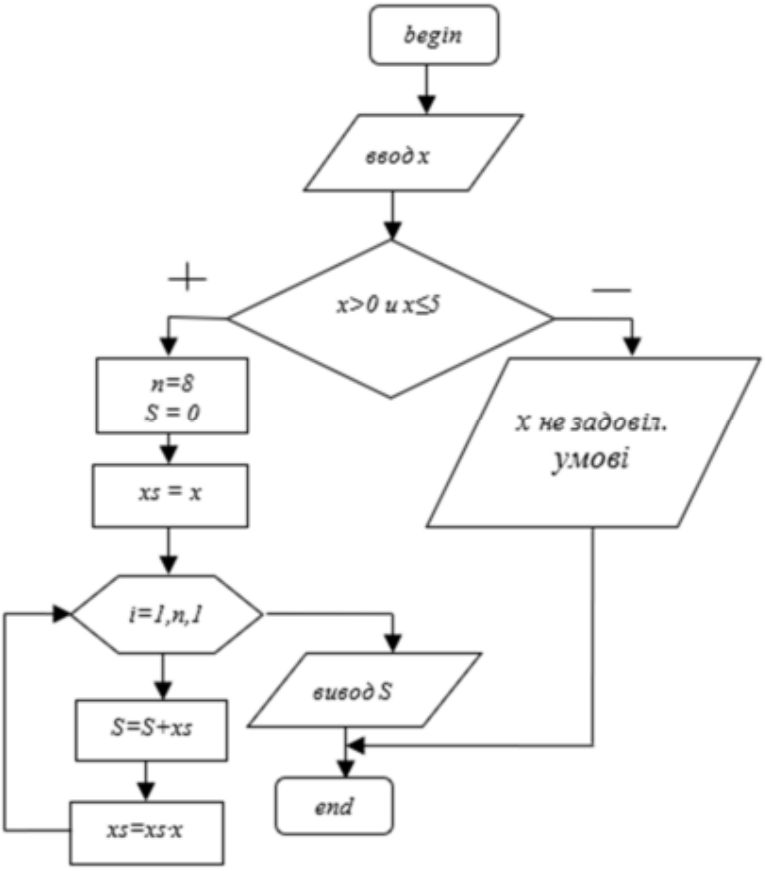


Рис.4.1 - Блок-схема алгоритму вирішення задачі

4. Текст програми:

```
#include <stdio.h>
#include <conio.h>
int main()
{
const n=8;
float x,xs,S;
printf("\nВвестти x=");
scanf("%d",&x);
 if (x>0 && x<=5)
{
   S=0; xs=x;
   for (int i=1;i<=n;i++)
   {
      S=S+xs; xs=xs*x;
   }
```

```
    printf("\nS=%d",S);
  }
  getch(); return 0;
     }
```

5. Тестування:

Теоретично розраховані вихідні значення	*Практично отримані вихідні значення*
Тест 1: *вхідні дані: x=-2 (не задовольняють умові 0<x≤5)*	
вихід з програми	*вихід з програми*
Тест 2: *вхідні дані: x=1*	
S=8	*S=8.000000*
Тест 3: *вхідні дані: x=10 (не задовольняють умові 0<x≤5)*	
вихід з програми	*вихід з програми*

Задача 2.

1. Постановка задачі: розробити алгоритм обчислення та виводу значення часткової суми членів ряду $S = \sin(x) + \frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} + ... + \frac{\sin(nx)}{n} + \ldots$ з точністю $\varepsilon = 10^{-4}$ (підсумовувати поки поточний член ряду по модулю не буде $\le 10^{-3}$) для $\pi/3 \le x \le 2\pi/3$. Написати програму, яка реалізує розроблений алгоритм.

2. Математична модель та опис алгоритму задачі:

- значення змінної *x* спочатку обнулити (*x=0)* перед входом в цикл з передумовою $x \le \pi/3$ або $x \ge 2 \cdot \pi/3$ з тим, щоб війти в цикл та вводити значення в змінну *x* до тих пір, поки воно не буде задовольняти заданій умові: $\pi/3 < x < 2 \cdot \pi/3$;
- для формування суми використовувати змінну *S*, спочатку її обнулити (*S=0*);
- спочатку коефіцієнт *i=1* при змінній *x*;
- у циклі з передумовою $|sin(i \cdot x)/i| \ge \varepsilon$ (перевіряється поточний член ряду по модулю) формується сума членів ряду – $S=S+sin(i \cdot x)/i$ і коефіцієнт *i* збільшується на 1 для наступної ітерації;
- після завершення циклу вивести значення змінної *S* і кількість ітерацій *i-1* (скільки разів повторився цикл, тобто скільки членів ряду підсумовувалось, щоб отримати часткову суму для заданих умов; кількість ітерацій дорівнює *i-1*, оскільки збільшення *i=i+1* відбувається для наступного проходу циклу, який, можливо, не виконується).

3. Блок-схема алгоритму задачі (рис.4.2)***:***

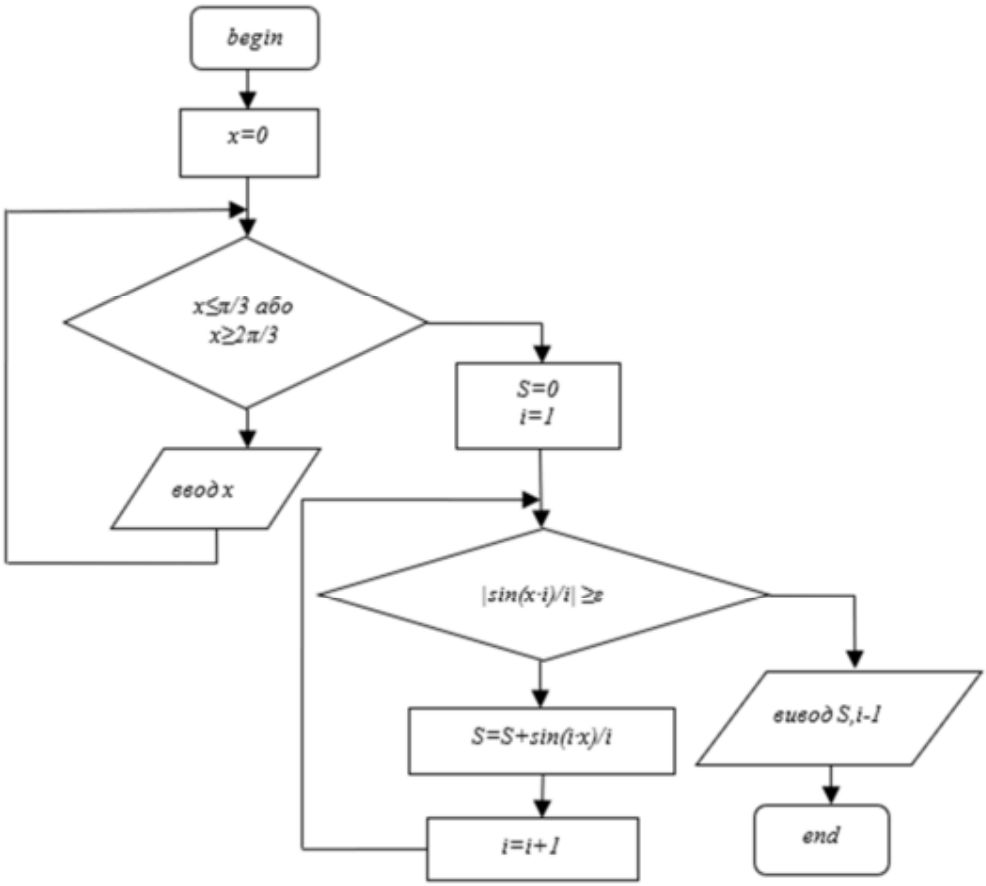


Рис.4.2 - Блок-схема алгоритму вирішення задачі

4. Текст программи

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
        int main()
        {
   const float p=3.14, e=.0001;
   int i=1;
   float S,x=0;
   while (x<=p/3 || x>=2*p/3)
   {
      printf("\nВвести x = ");
      scanf("%f",&x);
   }
    S=0;
    while (fabs(sin(i*x)/i)>=e)
    {
```

```
    S=S+sin(i*x)/i;
    i++;
  }
  printf("\n S = %f, кількість ітерацій = %d",S,i-1);
 getch();
 return 0;
     }
```

5. Тестування

Теоретично розраховане вихідне значення	*Практично отримане вихідне значення*
Тест: *вхідні дані: x=1,57 (≈π/2)*	
S=?; кількість ітерацій.=?	*S=?; кількість ітерацій.=?*

Задача 3.

1. Постановка задачі: розробити алгоритм обчислення значення часткової суми членів ряду Фібоначчі $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ... + \frac{x^{n-1}}{(n-1)!} + \frac{x^n}{n!} + \ldots$ для $-3<x\leq 2$ та оцінити швидкість збіжності, визначивши кількість доданків, необхідних для досягнення заданої похибки $\varepsilon=10^{-4}$. Вивести значення суми та кількість доданків. Написати програму, яка реалізує розроблений алгоритм.

2. Математична модель та опис алгоритму задачі:

- у циклі з постумовою $x\leq -5$ або $x\geq 5$ вводити значення у змінну x, поки не буде $-3<x\leq 2$;
- для введеного значення змінної x спочатку обчислюємо ліву частину рівності при використанні вбудованої функції *exp(x)*, потім обчислюється часткова сума ряду правої частини $s_n = \sum_{i=0}^{n} \frac{x^i}{i!}$ до тих пір, поки вона не буде відрізнятися від лівої частини менше, ніж на задану похибку ε (використовувати константу $e=10^{-4}$) або кількість ітерацій буде більше певного заданого значення *limit=30*;
- для формування суми використати змінну *S*, спочатку *S=1*, оскільки перший член ряду=1;
- для обчислення кожного доданку ряду $a_i = \frac{x^i}{i!}$ потрібно піднесення до ступеня (досить трудомістка операція) і обчислення факторіалу (додатковий цикл), тому у випадках функції ступеня і факторіала для спрощення обчислень суми ряду слід використовувати ***рекурентну формулу***, яка дозволяє отримати

наступний член ряду через попередній, тобто кожний поточний доданок можна рекурентно обчислити через попередній:

$$M=\frac{a_i}{a_{i-1}}=\frac{x}{i}\ ;\ a_i=M\cdot a_{i-1}=\frac{x}{i}\cdot a_{i-1}$$

- поточний член ряду – змінна *a=1* і кількість ітерацій – змінна *i=0*;
- для обчислення часткової суми ряду організувати цикл з постумовою $|y\text{-}S|\geq e$ та $i\leq limit$, в якому збільшити кількість ітерацій *i=i+1*, обрахувати наступний член ряду $a=a\cdot x/i$ та сформувати суму $S=S+a$;
- після виходу з циклу перевірити, за невиконання якої з двох умов завершився цикл: якщо $i>limit$, то вивести інформацію «точність не досягнуто»; інакше вивести значення *S* та кількість ітерацій *i*.

3. Блок-схема алгоритму задачі (рис. 4.3)***:***

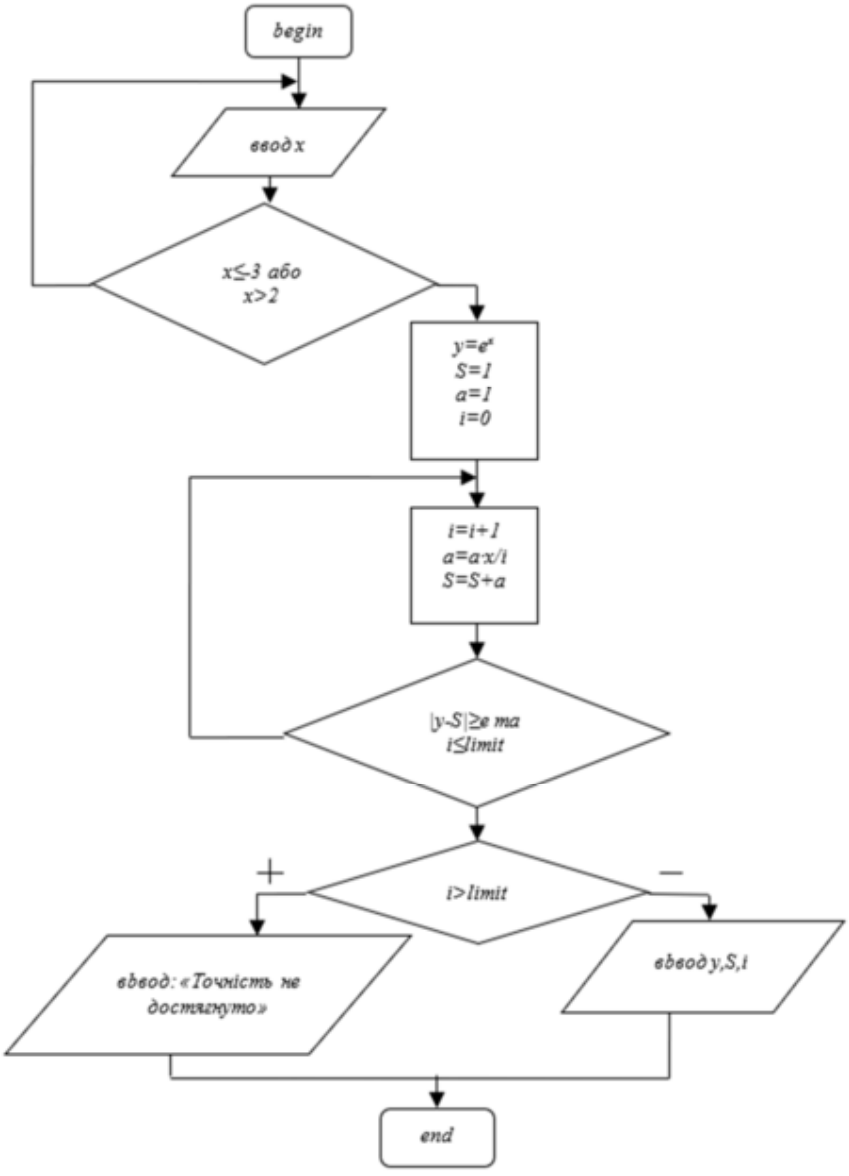


Рис.4.3 - Блок-схема алгоритму вирішення задачі

4. Текст програми:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
        int main()
        {
   const float e=.0001;
   const limit=30;
   int i=0;
    float y,a,S,x;
    do
    {
       printf("\nВвести x = ");
       scanf("%f",&x);
    }
   while (x<=-3 || x>2);
   y=exp(x);
   S=1; a=1;
   do
   {
      i++;
      a=a*x/i;
      S=S+a;
   }
   while (fabs(y-S)>=e && i<=limit);
   if (i>limit) printf("\n точність не досягнуто");
   else
   {
     printf("\n S = %f\t y = %f",S,y);
     printf("\n кількість ітерацій = %d",i);
   }
   getch();
   return 0;
        }
```

5. Тестування:

Теоретично розраховані вихідні значення	*Практично отримані вихідні значення*
Тест 1: *вхідні дані: x=-1.5 (-3<x≤2)*	

S=?; кількість ітерацій=?	*y=?; S=?; кількість ітерацій=?*
Тест 2: *вхідні дані: x=2 (-3<x≤2)*	
S=?; кількість ітерацій=?	*y=?; S=?; кількість ітерацій=?*
Тест 3: *вхідні дані: x=-3 (-3<x≤2)*	
точність не досягнуто	*точність не досягнуто*

Задача 4.

1. Постановка задачі: розробити алгоритм отримання суми *n* введених чисел та виводу значення суми.

2. Математична модель та описовий алгоритм задачі:

- використавши цикл с пост умовою *n≤0* (кількість введених чисел не може бути менше або дорівнювати нулю), ввести кількість введених чисел, щоб *n>0*;
- для формування суми значення *s* обнулити (*s=0*);
- у циклі з параметром (кількість повторів = *n*) кожен раз вводити значення *a* та формувати суму: *s=s+a*;
- після виходу з циклу вивести значення змінної *s*.

3. Блок-схема алгоритму задачі (рис. 4.4)***:***

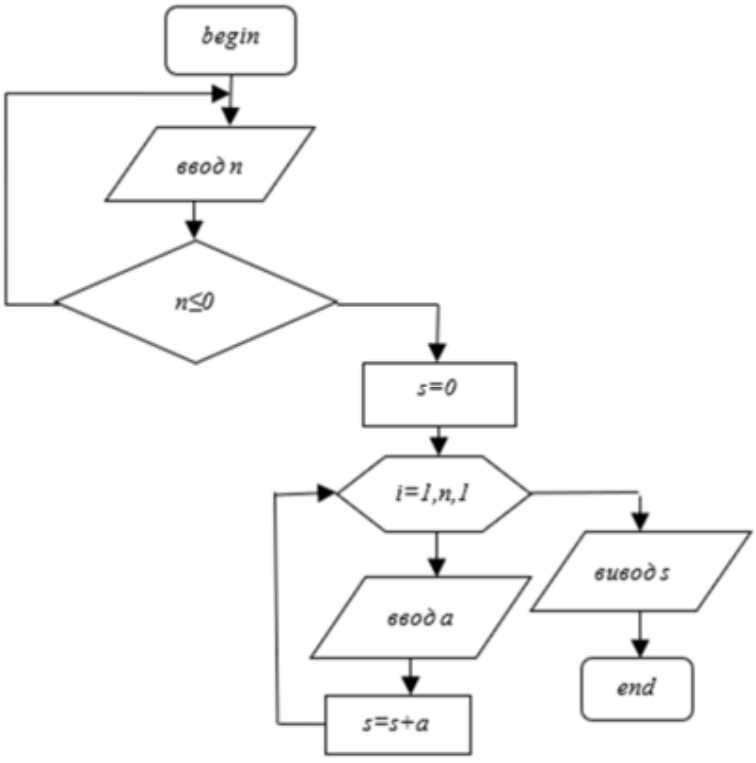


Рис.4.4 - Блок-схема алгоритму вирішення задачі

4. Текст програми: написати самостійно

5. Тестування: здійснити самостійно

Теоретично розраховане вихідне значення	*Практично отримане вихідне значення*
Тест 1: *вхідні дані: ?*	
S=?	*S=?*
Тест 2: *вхідні дані: ?*	
S=?	*S=?*

Виконання індивідуального завдання.

1. Постановка задачі.

Розробити циклічний алгоритм та написати програму за індивідуальним завданням, використовуючи оператори циклу, оператору вводу для вводу значень вхідних даних та оператор виводу для виводу результатів обчислень.

2. Вхідні і вихідні дані.

Усі задіяні у програмі змінні повинні бути оголошені.

Неприпустимо задавати вхідні (початкові) дані з допомогою операторів присвоювання. Ввід даних з клавіатури повинен здійснюватися після відповідного повідомлення.

3. Математична модель та описання алгоритму задачі.

4. Блок-схема алгоритму.

Представити циклічний алгоритм у вигляді блок-схеми.

5. Текст програми.

Розроблений алгоритм реалізується на мові програмування високого рівня С.

6. Тестування.

Результати тестування представити у вигляді таблиці.

Варіанти індивідуальних завдань

Розробити алгоритми вирішення задач, написати програми, які реалізують відповідні алгоритми, здійснити їх тестування. Вхідні дані вводяться з клавіатури. Друга задача для кожного варіанту відповідає номеру варіанту з таблиці 4.1.

Варіант №1

1. Знайти та вивести всі двозначні числа, в яких є цифра *n* або само число ділиться на *n*.

Варіант №2

1. З чисел от 10 до 99 вивести ті, сума цифр яких дорівнює *n*, де *0<n<18.*

Варіант №3

1. Вивести кількість тризначних натуральних чисел, сума цифр яких дорівнює заданому числу *n*.

Варіант №4

1. Серед двозначних чисел знайти та вивести ті, сума квадратів цифр яких ділиться на 13.

Варіант №5

1. Знайти та вивести двозначні числа такі, що якщо до суми цифр цього числа додати квадрат цієї суми, то буде це число.

Варіант №6

1. Знайти та вивести суму цілих позитивних чисел з інтервалу від *a* до *b*, які кратні 4.

Варіант №7

1. Для натурального *n* вивести всі його натуральні дільники.

Варіант №8

1. Суми цифр тризначного числа кратна 7, само число також ділиться на 7. Знайти і вивести всі такі числа.

Варіант №9

1. Серед чотиризначних чисел знайти та вивести ті, у яких всі 4 цифри різні.

Варіант №10

1. Визначити, чи є введене число *n* досконалим, тобто дорівнює сумі всіх своїх дільників, які не перевищують саме число; вивести відповідну інформацію.

Варіант №11

1. Визначити і вивести всі числа, що кратні введеним *a* та *b*, які менше $a \cdot b$.

Варіант №12

1. Визначити чи є введені натуральні числа *a* та *b* взаємно простими, тобто не мають спільних дільників, окрім одиниці.

Варіант №13

1. Обчислити і вивести для введеного натурального числа *n* суму $S=1+2^2+3^3+\ldots+n^n$, не використовуючи стандартну функцію піднесення до ступеня.

Варіант №14

1. Отримати та вивести всі прості числа *p*, які задовольняють нерівності $a<p<b$, де введені *a, b* ($a<b$) – натуральні числа.

Варіант №15

1. Для введеного натурального числа *s*, яке є площею, знайти та вивести сторони, які виражені натуральними числами, всіх таких прямокутників, площа яких дорівнює *s*.

Варіант №16

1. Знайти та вивести всі двозначні числа, в яких є цифра *n* або само число ділиться на *n*.

Варіант №17

1. З двозначних чисел вивести ті, суми цифр яких дорівнюють *n*, де *1<n<11*.

Варіант №18

1. Вивести кількість двозначних натуральних чисел, сума цифр яких дорівнює заданому числу *n*.

Варіант №19

1. Серед тризначних чисел знайти та вивести ті, сума квадратів цифр яких ділиться на 35.

Варіант №20

1. Знайти та вивести двозначні числа такі, що якщо до суми цифр кожного з них додати квадрат цієї суми, то можна отримати саме число.

Варіант №21

1. Знайти та вивести суму цілих позитивних чисел з діапазону від *a* до *b*, кратних 3.

Варіант №22

1. Для натурального *n* вивести всі його прості дільники.

Варіант №23

1. Знайти та вивести всі тризначні числа, сума цифр яких кратна 3, а само число ділиться на 3.

Варіант №24

1. Серед тризначних чисел слід знайти та вивести ті, у яких усі 3 цифри різні.

Варіант №25

1. Визначити, чи є введене число *n* досконалим, тобто дорівнює сумі всіх своїх дільників, які не перевищують саме число; вивести відповідну інформацію.

Таблиця 4.1 Обчислити часткову суму рядів з точністю ε:

N	Формула часткової суми ряду	N	Формула часткової суми ряду
1	$\sum_{n=1}^{\infty}\frac{(-1)^n x^n}{(n+1)!}$, $\varepsilon=10^{-5}$, $-1\le x\le 1$	***2***	$\sum_{n=1}^{\infty}\frac{(-1)^n}{nx^n}$, $\varepsilon=0.5\cdot10^{-5}$, $\lvert x\rvert>1$
3	$\sum_{n=1}^{\infty}\frac{(-1)^{n-1}x^n}{(n+1)!}$, $\varepsilon=10^{-4}$, $-1\le x\le 1$	***4***	$\sum_{n=1}^{\infty}\frac{(-1)^n x^n}{(2n)!}$, $\varepsilon=10^{-5}$, $-1\le x\le 1$
5	$\sum_{n=1}^{\infty}\frac{(-1)^n x^{n-1}}{(n+1)!}$, $\varepsilon=10^{-6}$, $-1\le x\le 1$	***6***	$\sum_{n=1}^{\infty}\frac{(-1)^n x^n}{(n+1)^2}$, $\varepsilon=10^{-4}$, $\lvert x\rvert<1$
7	$\sum_{n=1}^{\infty}\frac{(-1)^n x^n}{(2n+1)!}$, $\varepsilon=0.5\cdot10^{-5}$, $\lvert x\rvert\le 1$	***8***	$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}(x-1)^n}{(n-1)!}$, $\varepsilon=10^{-4}$, $0\le x\le 2$
9	$\sum_{n=1}^{\infty}\frac{(-1)^{2n-1}x^n}{(2n-1)!}$, $\varepsilon=10^{-3}$, $-1\le x\le 1$	***10***	$\sum_{k=1}^{\infty}\frac{(-1)^k}{(k-1)!}$, $\varepsilon=0.5\cdot10^{-5}$
11	$\sum_{n=1}^{\infty}\frac{(-1)^n x^{2n-1}}{(2n)!}$, $\varepsilon=0.2\cdot10^{-5}$, $\lvert x\rvert\le 1$	***12***	$\sum_{n=1}^{\infty}\frac{x+n}{x^n}$, $\varepsilon=10^{-3}$, $\lvert x\rvert>1$
13	$\sum_{n=1}^{\infty}\frac{(-1)^n x^{2n}}{2n!}$, $\varepsilon=10^{-4}$, $-1\le x\le 1$	***14***	$\sum_{n=1}^{\infty}\frac{(-1)^{2n-1}x^{2n-1}}{(2n-1)!}$, $\varepsilon=10^{-5}$, $-1\le x\le 1$
15	$\sum_{n=1}^{\infty}\frac{(-1)^{2n}(x-1)^n}{(2n+1)!}$, $\varepsilon=10^{-4}$, $0\le x\le 2$	***16***	$\sum_{n=1}^{\infty}\frac{(-1)^n(n-1)x^n}{(n+1)}$, $\varepsilon=10^{-4}$, $\lvert x\rvert<1$
17	$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}+1x^{n+1}}{(2n+1)!}$, $\varepsilon=10^{-5}$, $\lvert x\rvert\le 2$	***18***	$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}(x-1)^{n+1}}{(n-1)!}$, $\varepsilon=0.2\cdot10^{-4}$, $0\le x\le 1$
19	$\sum_{n=1}^{\infty}\frac{(-1)^{2n-1}x^{n+1}}{(2n-1)!}$, $\varepsilon=10^{-4}$, $-1\le x\le 1$	***20***	$\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{(k-1)!}$, $\varepsilon=10^{-5}$
21	$\sum_{n=1}^{\infty}\frac{(-1)^n x^{2n-1}}{(2n-1)!}$, $\varepsilon=10^{-5}$, $\lvert x\rvert\le 1$	***22***	$\sum_{n=1}^{\infty}\frac{x+n}{x^{n+1}}$, $\varepsilon=10^{-4}$, $\lvert x\rvert\ge 2$
23	$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n!}$, $\varepsilon=10^{-5}$, $-1\le x\le 1$	***24***	$\sum_{n=1}^{\infty}\frac{(-1)^{2n-1}x^{2n}}{(2n)!}$, $\varepsilon=10^{-4}$, $-1\le x\le 1$
25	$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n!}$, $\varepsilon=10^{-5}$, $-1\le x\le 1$		

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
    3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
    3.2 Написання програм, відналагодження програми.
    3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке *цикл*?
2. Які існують два типи циклів?
3. Структура оператору циклу з параметром.
4. Як працює цикл з параметром?
5. Вказати особливості оператора циклу с параметром у мові С?
6. Які цикли відносяться до ітераційних?
7. Навести структуру оператора циклу з передумовою.
8. Як працює цикл з передумовою?
9. Яку структуру має оператор циклу з постумовою?
10. Як працює цикл з постумовою?

4.4 Самостійна робота 1. Визначення стану принтеру у DOS

Тема: Розглянути операцію визначення стану принтеру у DOS.

Мета: Навчитися визначати стан принтеру.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, Draw io, https://www.onlinegdb.com.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом (лекції, додадкові матеріали до лекцій та перелік літературних джерел).

2. Виконати практичну частину.

2.1. Описати різновиди портів принтера.

2.2. Надати вміст адрес базових портів.

2.3. Пояснити програму зчитування стану принтера.

2.4. Визначити стан принтеру за його **запис**ом методом маскування (згідно варіанту завдання).

3. **Оформити** текстовий файл та **прикріпити у цьому розділі**.

Форма подання результатів виконаної роботи:

Звіт з самостійної роботи містить титульний лист, зміст з автоматичною нумерацією сторінок, хід виконання роботи з зазначенням завдання, підстановкою значень, отримані розрахункові значення, виконані індивідуальні варіанти завдання.

Кожне завдання містить: постановку задачі, математичну модель та опис алгоритму задачі, блок-схему алгоритму задачі, текст програми, тестування. Обов'язковими є висновок, який містить аналіз отриманих в результаті роботи результатів та відповіді на контрольні питання.

Звіт у форматі .docx, збережений транслітерацією прізвища та номеру роботи виставляється у зазначений ресурс у відповідну лабораторну роботу.

Перед здачею звіту обов'язкова демонстрація та тестування роботи розроблених програм викладачу, корекція звіту за результатами демонстрації, а вже потім здача лабораторних робіт у Єлерн.

Критерії оцінювання:

максимальна кількість балів - 15 балів, з них:

- створення блок-схем алгоритмів, математичних моделей та тестування - 3 бали;
- написання програм, відналагодження програми - 3 бали;
- демонстрація роботи програми та усунення зауважень – 3 бали;.
- відповіді на контрольні питання - 3 бали.
- грамотне оформлення звіту за вимогами - 3 бали.

РОЗДІЛ V. ЛОГІЧНІ ВИРАЗИ. ВІДНОШЕННЯ, ЛОГІЧНІ ОПЕРАЦІЇ, УМОВНІ ВИРАЗИ

5.1 Теоретичні відомості. Логічні вирази Відношення, логічні операції, умовні вирази

5.1.1 Логічні вирази

Існують чотири операції відносин >, >=, <, <= і дві операції рівності = = але !=.

Чотири перші операції між собою рівноправні, операції рівності мають менший пріоритет.

У відносинах арифметичні вираження виконуються раніше, тому у вираженні **i <a+b** додаткових дужок не потрібно. І взагалі на загал в С++ є 11 рівнів пріоритетів операцій, у мові Паскаль лише 4.

У логічних виразах використовуються 3 логічних операції: 1. **!** - “заперечення”

2. **&&** - “І”,

3. || - “АБО” з врахуванням старшинства (!, &&, ||).

Особливість виконання логічних операцій полягає в тому, що в мові немає логічних констант і логічних змінних. Тому логічні операції фактично виконуються над ціл показниками, що плавають і. При цьому FALSE (НЕПРАВДА) відповідає арифметичному нулю. Усе, що не є нулем відповідає TRUE (ПРАВДА).

5.1.2 Порозрядні логічні операції

Для роботи з окремими бітами поруч зі звичайними логічними операціями передбачено **6** порозрядних логічних операції (табл. 5.1):

1. **&** - порозрядне І;
2. | - порозрядне АБО;
3. ^ - порозрядне виключаюче АБО;
4. ~ - доповнення;
5. << - зсув вліво;
6. >> - зсув вправо.

Таблиця 5.1 Порозрядні логічні операції

x	y	x&y	x|y	x^y	x~	y~	y<<1	y>>2
0	0	0	0	0	1	1	*	*
0	1	0	1	1	1	0	*	*
1	0	0	1	1	0	1	*	*
1	1	1	1	0	0	0	*	*

Доповнення є унарною операцією, усі інші - бінарні НЕ можуть виконуватися над операціями типу *float* або *double.*

Операції **&**, **|** можна використовувати для керування окремими розрядами (бітами). Наприклад:

n=243 (11110011)

c=n&547 (1000100011)

// ***c*** буде присвоєно код результату **1000111=3510**

Потрібно визначити відмінність у виконанні звичайних логічних операцій і порозрядних.

Наприклад:

(1&&2 == 1) (0&&2 == 0) (1&&0 == 0) (1&&3 == 1)

У цьому випадку операція логічного І (&&) утворює значення 1, якщо обидва операнди мають НЕ нульові значення. Якщо один з операндів дорівнює 0, то результат також дорівнює 0. Якщо значення першого операнду дорівнює 0, то другий операнд не обчислюється.

(1&2 == 0) (0&2 == 0) (1&0 == 0) (1&3 == 1)

У другому випадку перевіряється, чи є хоча б в одній парі бітові одиниці. Оскільки код 1 і код 2 різні, то результат " НЕПРАВДА " або 0. Операція доповнення ~ у кожному біті змінює 1 на 0 і 0 на 1.

Операція зсуву виконується на таку кількість розрядів, яка відзначена в операнді праворуч **x<<2**. У правих бітах з'являється відповідна кількість нулів. Операція зсуву праворуч **y>>5** виконується в різних машинах по різному:

1. З фіксацією знакового розряду (як арифметичний зсув);
2. Без фіксації (логічний зсув).

5.1.3 Умовні вирази

У деяких випадках умовні оператори можна замінити умовним виразами, які мають вигляд:

e1 ? e2 : e3,

де **e1**, **e2**, **e3** - вираження.

Якщо вираження **e1** відмінне від 0 (ІСТИНА), то виконується вираз **e2**. Інакше - вираження **e3**. Блок-схема алгоритму для умовного оператору наведена на рис. 5.1.

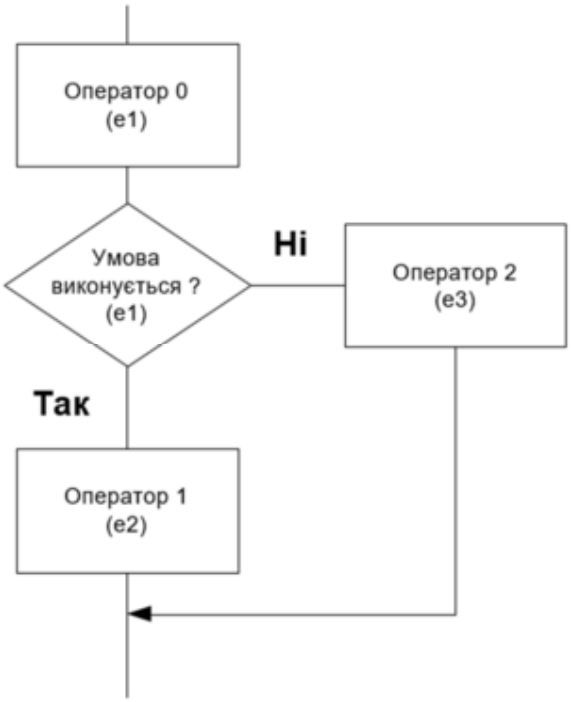


Рис. 5.1 - Блок-схема алгоритму умовного оператора

Наприклад, вибір максимального із двох чисел

```
z = a>b ? a : b;          /*z = max (a, b)*/
```

Виконання умовних виразів ефективніше, ніж операторів.

5.2 Лабораторні роботи 7, 8. Розгалуження і цикли (у двох частинах)

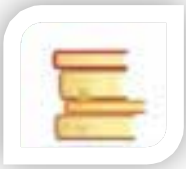
Тема: Розгалуження і цикли

Мета: познайомитися з функціями з математичної бібліотеки, освоїти операції відношення та логічні операції, навчитися грамотно програмувати базові алгоритмічні структури «розгалуження» та «цикл».

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

У програмуванні визначають, які операнди необхідно виконувати і в якому порядку. Інакше визначають потік управління у програмі. Як зазначалось у [1], що будь-який алгоритм може бути реалізований за допомогою трьох керуючих структур:

1. послідовне виконання;
2. умовне виконання;
3. цикл.

Умовний оператор *if* має вигляд:

if (вираз) оператор 1;
else оператор2; (рис.5.1)

Можливе використання й скорочений оператор:

if вираз) оператор1;

Якщо вираз має ненульове значення, то виконується оператор1. Інакше переходимо до виконання наступного за порядком оператора.

Ніякі обмеження на тип операторів не накладають. Тому можна використовувати й вкладені умовні оператори. Приходимо до конструкцій:

if вираз) оператор else if . else if.

У тих випадках, коли виникає вибір з багатьма можливими результатами, доцільно використовувати перемикач **switch**

Його структура:

switch (цілий вираз)
{case константа 1: оператор 1; [break]
case константа 2: оператор 2; [break]
.....
case константа n: оператор n; [break]
default :оператор n+1;
}

Слово **case** (рис. 5.2) вживається у кожному виборі, в інших випадках, замість констант можна вживати константний вираз.

Оператор виконується в такий спосіб:

1. Виконується вираз в круглих дужках. Його значенням може бути або ціле, або літера (символ);

2. Отримане значення рівняється з константами вибору. Якщо воно збігається з однієї з констант1 ... константа n, то виконується відповідний оператор і;

3. Далі виконуються всі наступні оператори від і+1 до default (n+1) (відсутність) включно;

4. Якщо значення виразу не збігається з ні однієї константою вибору, то виконується оператор з міткою default

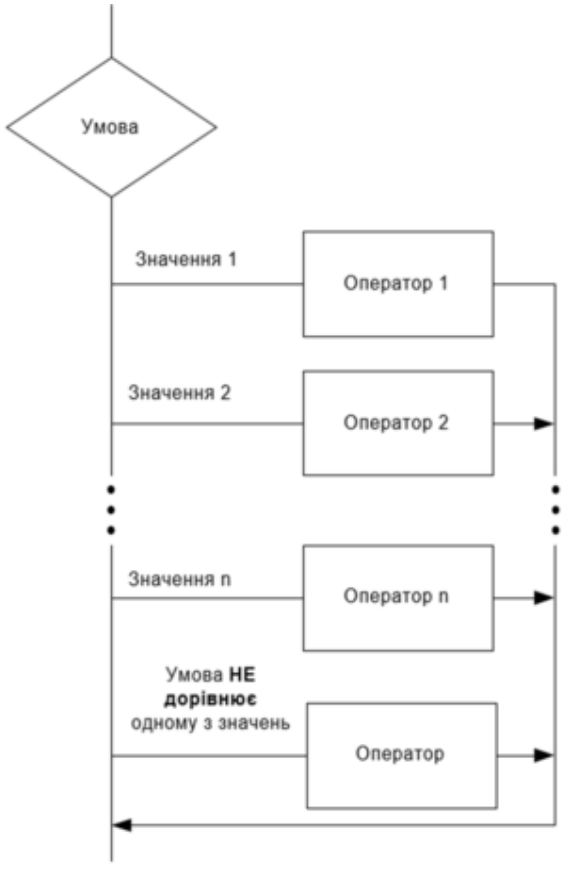


Рис. 5.2 – Блок-схема алгоритму умовного оператору

Цикл с передумовою **while** має структуру (рис. 5.3):
while (умова) оператор;

Рис. 5.3 – Блок-схема алгоритму циклу з передумовою **while**

Виконується доти , поки умова є істиною. Ясно, що замість одного оператора може бути складений оператор або блок.

Якщо необхідно перервати цикл до його завершення, то можна використовувати оператор break.

Цикл з післяумовою do-while (рис.5.4)**.** Якщо оператори циклу повинні виконуватися принаймні один раз, то можна використовувати інший варіант **do** оператор **while (условие);**
do { оператор} while (умова)

Рис.5.4 – Блок-схема алгоритму для цикл з післяумовою do-while

Оператор циклу **for** має блок-схему алгоритму, наведену на рис.5.5.
for (вираз 1; вираз 2; вираз 3) оператор;

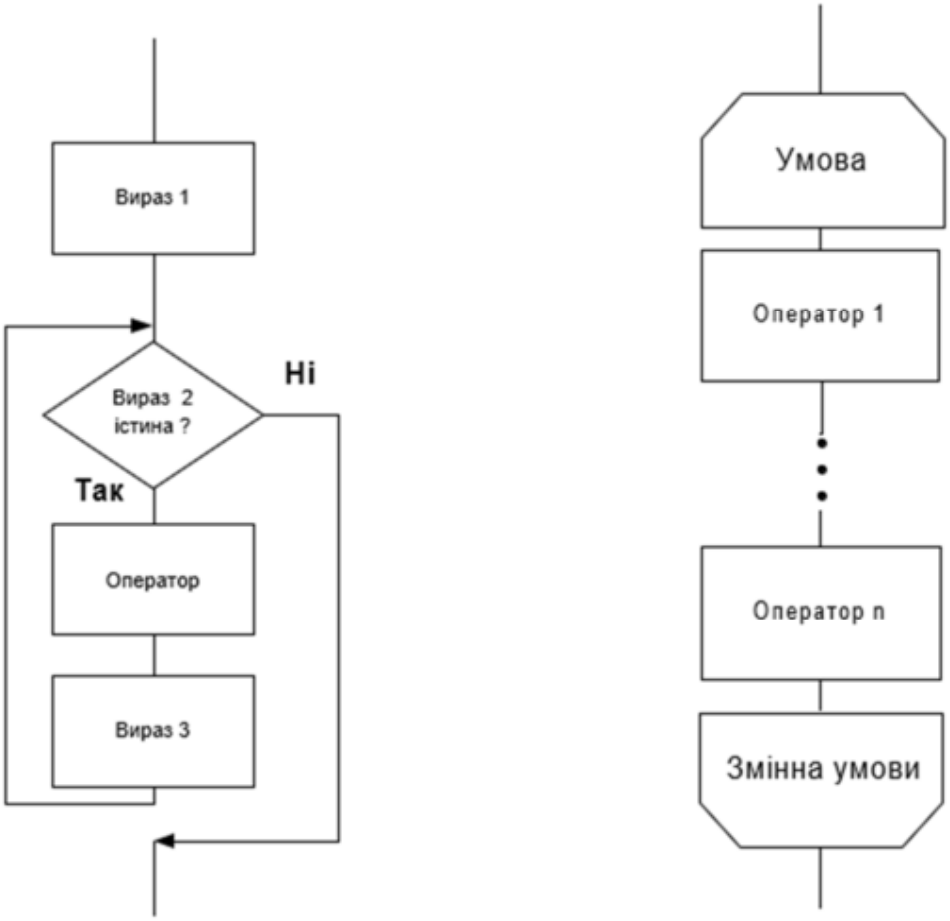


Рис.5.5 - Блок-схема алгоритму для оператору циклу **for**

Оператор циклу **for** виконується таким чином:

вираз 1; while (вираз 2) {оператор; вираз 3;}

Тобто вираз 1 виконується лише один раз. Вираз 2 еквівалентне умові закінчення циклу. Вираз 3 виконується щораз у циклі. Тому: вираз 1 - початкове значення параметра циклу, а вираз 3 - модифікація параметра циклу.

Практична частина

Перші чотири завдання - це сьома лабораторна робота, усі інші – восьма.

Завдання

Написати 9 програм згідно номеру індивідуального варіанту.

У першій програмі обчислити значення функції, використовуючи умовну операцію «*?:*».

У другій програмі обчислити значення за вказаною формулою, використовуючи функції математичної бібліотеки. Перед написанням програми потрібно обчислити область визначення функції (ОВФ), у програмі після вводу аргументів перевірити їх приналежність ОВФ.

У третій програмі використати вложений умовний оператор.

У четвертій програмі змоделювати арифметичний цикл з допомогою оператора циклу *for*.

У п’ятій та шостій програмах використати цикли *while* або *do … while*.

У сьомій та восьмій програмах обчислити нескінчену суму з заданою точністю, використовуючи рекурентні залежності.

У дев’ятій програмі використати конструкцію «цикл у циклі».

Варіанти індивідуальних завдань

Варіант 1

1. $f(x,y)=\begin{cases} x+y, \text{якщо } x>0 \\ xy, \text{якщо } x\le 0, y<0 \\ 5x, \text{в інших випадках} \end{cases}$

2. $z=x^2-\cos\dfrac{\ln\sqrt{|x|}}{tge^{-x}}$

3. Вивести на екран номер чверті, який належить точці з координатами (x,y), або вказати, до якої вісі належить ця точка.

4. Ввести з клавіатурі натуральне число N>2. Вивести на екран послідовність вигляду $S=\begin{cases} 2*4*6*\ldots*N, \text{ якщо } N \text{ парне} \\ 1*3*5*\ldots*N, \text{якщо } N \text{ непарне} \end{cases}$

5. Знайти перше число Фібоначчі, яке більше заданого n (n>1).

6. M та N – чисельник і знаменник звичайного дробу. Розробити програму, яка дозволяє скоротити цю дріб.

7. Обчислити значення суми нескінченого ряду з точністю до члена ряду, який за модулем менше $\varepsilon=10^{-4}$, $S=1-\frac{x}{1!}+\frac{x^2}{2!}-\ldots+(-1)^n\frac{x^n}{n!}+\ldots$, та значення функції (для перевірки) $f=e^{-x}$.

8. Обчислити значення суми нескінченого ряду з заданою точністю $\varepsilon=10^{-6}$ $f(x)=\frac{x\cos\frac{\pi}{3}}{1}+\frac{x^2\cos 2\frac{\pi}{3}}{2}+\ldots+\frac{x^n\cos n\frac{\pi}{3}}{n}+\ldots$, та значення функції (для перевірки) $y=-\frac{1}{2}\ln\left(1-2x\cos\frac{\pi}{3}+x^2\right)$ врахувати, що $0.1\le x\le 0.8$.

9. Дано ціле k>2. Надрукувати всі числа з діапазону [2,k], які не є простими.

Варіант 2

1. $$f(a,b)=\begin{cases}3a^2, если\ a>5\\ a/b, если\ 0<a\le 5, b\ne 0\\ b+a-1, в\ остальных\ случаях\end{cases}$$

2. $$D=\frac{-\sin a+\sqrt{\sin^2 a+12\left|\ln|b|\right|}}{(b-a)^2 e^{tg\frac{a}{b}}}$$

3. Є два цілих числа. Якщо вони обидва позитивні, то більше з них слід замінити їх середнім арифметичним; якщо обидва від'ємні, то змінити знак у меншого з них; інакше кожне з них подвоїти.

4. Дано натуральне n. Обчислити n сомножники добутку $\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\ldots$.

5. Представити натуральне число N у вигляді добутку простих множників.

6. Поміняти місцями цифри старшого і молодшого розрядів даного натурального числа (наприклад, з числа 3872 буде отримано 2873).

7. Обчислити значення суми нескінченого ряду $S=1+\frac{x\ln a}{1!}+\frac{(x\ln a)^2}{2!}+\ldots+\frac{(x\ln a)^n}{n!}+\ldots$ з точністю до члена ряду, який менше $\varepsilon=10^{-4}$, і значення функції (для перевірки) $f=a^x$, врахувати, що $0{,}1\le x\le 1$.

8. Обчислити значення суми нескінченого ряду $S=\frac{x-1}{x}+\frac{(x-1)^2}{2x^2}+\ldots+\frac{(x-1)^n}{nx^n}+\ldots$ з точністю до члена ряду, який менше $\varepsilon=10^{-5}$, та значення функції (для перевірки) $f=\ln x$, врахувати $x>1$.

9. Дано натуральне число *N*. Обчислити $S=1+\frac{1}{2^2}+\frac{1}{3^3}+\ldots+\frac{1}{N^N}$. Формулу

піднесення до ступеня не використовувати.

Варіант 3

1. $$f(m,n)=\begin{cases} m+n^2, если\, n<0, m>0 \\ m+2n, если\, m\le 0, n<0 \\ m+1, в\, остальных\, случаях \end{cases}$$

2. $$S=e^{\frac{\ln r-\cos r^2}{\sin^2 r+tg3r}}$$

3. Є два цілих числа. Якщо вони обидва парні, то больше з них поділити на 2; якщо обидва непарні, то кожне помножити на 2; у протилежному випадку непарнее з чисел збільшити на 1.

4. Розробити програму пошуку двозначних чисел таких, що якщо до суми цифр цього числа прибавити квадрат цієї суми, то буде це число.

5. Нехай A та B - позитивні дійсні числа, які більше 1, причому A>B. Розробити програму для пошуку такого найменшого натурального m, що $B^m>$ m*A.

6. Дано натуральне число *n*. Розробити програму для порівняння цифр старшого і молодшого розрядів цього числа.

7. Обчислити значення суми нескінченого ряду $S=1+\frac{x}{1!}+\frac{x^2}{2!}+...+\frac{x^n}{n!}+...$ з точністю до члена ряду, по модулю меншого $\varepsilon=10^{-4}$, та значення функції (для перевірки) $f=e^x$, врахувати, що $1\le x\le 2$.

8. Обчислити значення суми нескінченого ряду з заданою точністю $\varepsilon=10^{-5}$ $S=2\left[\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\frac{1}{7x^7}+...\right]$ та значення функції (для перевірки) $y=\ln\left(\frac{x+1}{x-1}\right)$, врахувати, що |x|>1.

9. Для кожного з 10 значень змінної *x*, яка змінюється від *–b* до *b* з постійним шагом, обчислити значення добутку $P=x*(x+0{,}2)*(x+0{,}4)*\ldots*(x+1{,}8)$.

Варіант 4

1. $$f(x,y)=\begin{cases} x+y, \text{якщо } x>0, y<0 \\ \frac{x}{y-1}, \text{якщо } x>0, y>1 \\ x-y, \text{в інших випадках} \end{cases}$$

2. $$F=\sqrt{\cos\frac{a^2+\sqrt{a}}{1+\frac{\sin^2 a}{2a}}+\frac{2{,}5}{2\ln a}}$$

3. Є 2 числа. Якщо вони обидва позитивні, то поміняти знак у більшого з них; якщо обидва від'ємні, то перше помножити на 2, а друге помножити на 3;

у протилежному випадку менше число замінити їх полусумою, а більше число зменшити на 1.

4. Вивести на екран через кому всі дільники натурального числа N.

5. Знайти суму цілих позитивних чисел з інтервалу від A до B, які кратні 4. Значення A та B вводяться з клавіатури.

6. Дано натуральне число *n*. Розробити програму для визначення кількості цифр у цьому числі.

7. Обчислити значення суми нескінченого ряду $S = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \ldots$ з точністю до члена ряду модуль якого менше ε=10^{-5}, та значення функції (для перевірки) $f = \frac{\sin x}{x}$, врахувати, що -π≤ x ≤ π.

8. Обчислити значення суми нескінченого ряду $S = \sum_{n=0}^{\infty} \left(1 + (-1)^n 2^{n+1}\right) x^n$ з точністю до члена ряду модуль якого менше ε=10^{-4}, та значення функції (для перевірки) $f = \frac{3}{(1-x)(1+2x)}$, врахувати, що |x|<0,5.

9. Написати програму для пошуку 100 перших простих чисел.

Варіант 5

1. $f(k,m) = \begin{cases} m/k, \text{якшо } k > 0 \\ |m+k|, \text{якщо } k \le 0, m < 0 \\ km, \text{в інших випадках} \end{cases}$

2. $D = \frac{e^{-\frac{1}{6}} \cdot \sqrt{a^2 + \ln|b|} - tga}{2\cos^2 a^3} \cdot 10^6$

3. Є координати двох точок на площині. Якщо хоча б одна з них лежить на будь-якій вісі, то вивести повідомлення про це; якщо вони обидві знаходяться в одній чверті, то найти і вивести відстань між ними; інакше знайти точку, яка найбільше віддалена від центру координат.

4. Обчислити $a(a\text{-}n)(a\text{-}2n)\ldots(a\text{-}n^2)$. Дійсні *a* та цілі *n* вводяться з клавіатури.

5. Обчислити за скільки рокі у банку при початковому вкладі *W* і відсотках річного приросту *Pr* буде накопичена сума *Sum* (відсотки капіталізуються щорічно).

6. Визначити чи міститься у десятковому записі натурального числа *N* цифра 3.

7. Обчислити значення суми нескінченого ряду с заданою точністю ε=10^{-5}, $f(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots$, значення функції (для перевірки) $shx = \frac{e^x - e^{-x}}{2}$.

8. Обчислити значення суми нескінченого ряду с заданою точністю ε=10^{-4}

$f(x) = 1 + 3x^2 + ... + \frac{2n+1}{n!}x^{2n} + ...$, та значення функції (для перевірки) $y = (1 + 2x^2)e^{x^2}$, врахувати, що $0.1 \le x \le 1$.

9. Послідовно вводяться *n* натуральних чисел(*n ≤10*). Обчислити суму тих з них, у яких перша цифра дорівнює останній.

Варіант 6

1. $f(a,b) = \begin{cases} a - b, \text{якщо } a > 10 \\ \frac{a-b}{a+b}, \text{якщо } 0 < a \le 10, b > 0 \\ b, \text{ в інших випадках} \end{cases}$

2. $G = \frac{\sqrt{|m-n|} - \sin m \cos n}{\ln tg\frac{m}{n} + e^{m^2}}$

3. Є три цілих числа. Якщо вони всі рівні, то залишити їх без змін; якщо вони утворюють монотонну (тобто або зростаючу, або спадаючу) послідовності, то замінити останнє число так, щоб задані числа утворювали арифметичну прогресію (вважаючи, що перші два числа є першими членами прогресії); інакше друге число замінити напівсумою першого і третього чисел.

4. У гусей та кроликів всього 2N лап. Скільки може бути гусей та кроликів (вивести всі можливі варіанти)?

5. Дано ціле *m* >1. Отримати найбільше ціле *k*, при якому $4^k < m$.

6. Найти число Фібоначчі, яке найближче до заданого натурального числа *n*.

7. Обчислити значення суми нескінченого ряду $S = 1 + \frac{\ln 3}{1!}x + \frac{\ln^2 3}{2!}x^2 + ... + \frac{\ln^n 3}{n!}x^n + ...$, з точністю до члена ряду, який за модулем менший $\varepsilon=10^{-4}$, значення функції (для перевірки) $f=3^x$, врахувати, що $0 \le x \le 1$.

8. Обчислити значення суми нескінченого ряду з заданою точністю $\varepsilon=10^{-5}$ $f(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - ... + \frac{(-1)^n x^{2n+1}}{2n+1} + ...$ та значення функції (для перевірки) f=arctg(x), врахувати, що $x^2<1$.

9. Дано *k* натуральних чисел. Визначити скільки з них досконалі. Досконалим називають число, яке дорівнює сумі всіх своїх дільників, включаючи 1 і не включаючи саме число.

Варіант 7

1. $f(x,y) = \begin{cases} x - |y|, \text{якщо } x > 0 \\ y^2 - x, \text{якщо } x \le 0, y < 0 \\ y + 1, \text{ в інших випадках} \end{cases}$

2. $S = \frac{ctgx - e^{\sqrt{x}}}{\ln|5x| + arctg^2 x}$

3. Дані 2 числа. Якщо вони обидва від'ємні, то перше зменшити на 1, а друге збільшити на 1; якщо обидва позитивні, то більше з них подвоїти, а менше потроїти; у протилежному випадку від'ємне з чисел замінити його абсолютним значенням.

4. Дано натуральне число n>10. Розробити програму для обчислення значення $y = n\cos x + (n-1)\cos 2x + (n-2)\cos 3x + ...2\cos(n-1)x + \cos nx$.

5. Дано натуральне n. Розробити програму для пошуку першої цифри цього числа.

6. Перевести задані натуральні число n з десяткової системи числення у двійкову, тобто отримати число, яке є двійковим записом числа n.

7. Обчислити значення суми нескінченого ряду с заданою точністю ε=10^{-6} $f(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + ... + \frac{x^{2n}}{(2n)!} + ...$, значення функції (для перевірки) $chx = \frac{e^x + e^{-x}}{2}$, врахувати, що 0.1≤x≤1.

8. Обчислити значення суми нескінченого ряду с заданою точністю ε=10^{-4} $f(x) = 1 + \frac{\cos\frac{\pi}{4}}{1!}x + ... + \frac{\cos n\frac{\pi}{4}}{n!}x^n + ...$, та значення функції (для перевірки) $f = e^{x\cos\frac{\pi}{4}}\cos\left(x\sin\frac{\pi}{4}\right)$, врахувати, що $0.1 \le x \le 1$.

9. Послідовно вводяться цілі позитивні числа. Для кожного числа з'ясувати чи є воно факторіалом будь-якого числа. Якщо да, то вивести число, факторіалом якого є введене, якщо ні, то вивести повідомлення про це. Ознакою кінця вводу є ввод нуля.

Варіант 8

1. $f(a,b) = \begin{cases} 3a + 2, \text{якщо } a < 0, b = 0 \\ \frac{3a}{2b}, \text{якщо } a < 0, b \neq 0 \\ 3a - 2b, \text{ в інших випадках} \end{cases}$

2. $Z = \frac{tg^2 x - \ln 3x^2}{\sqrt{\left|e^{\sin x} - \pi/7\right|}}$

3. Є сторони трикутника. Визначити його вид: рівносторонній, рівнобедрений або різносторонній,– та вивести повідомлення про це. Окрім того, якщо трикутник рівносторонній, то знайти його площу, якщо він рівнобедрений, то знайти його периметр, а в протилежному випадку знайти його найменшу сторону.

4. Вихідні дані – натуральне число *K*, яке виражає площу. Написати програму для знаходження всіх таких прямокутників, площа яких дорівнює *K* і сторони

виражені натуральними числами.

5. Спортсмен у перший день пробіг 10 км. Кожен наступний день він збільшував денну норму на 10% від попереднього дня. Через скільки днів спортсмен буде пробігати в день більше 20 км?

6. Визначити чи є дане натуральне число *N* паліндромом (паліндром зліва направо та справа наліво читається однаково, наприклад 32423).

7. Обчислити значення суми нескінченого ряду с заданою точністю $\varepsilon=10^{-5}$ $f(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots+\frac{(-1)^n x^{2n}}{(2n)!}+\ldots$, значення функції (для перевірки) $y=\cos x$.

8. Обчислити значення сумм нескінченого ряду с заданою точністю $\varepsilon=10^{-4}$ $f(x)=\frac{x-1}{x+1}+\frac{1}{3}\left(\frac{x-1}{x+1}\right)^3+\ldots+\frac{1}{2n+1}\left(\frac{x-1}{x+1}\right)^{2n+1}+\ldots$, значення функції (для перевірки) $y=\frac{1}{2}\ln x$, врахувати, що $0.2\le x\le 1$.

9. Знайти натуральне число в діапазоні від 2 до *n* ($n\le 100$) з максимальною сумою дільників (само число в суму не включати).

Варіант 9

1. $$f(t,s)=\begin{cases}3t-1, \text{якщо } s>4\\ \sqrt{|t-s|}, \text{якщо} s\le 4, t\ne 0\\ s+2, \text{ в інших випадках}\end{cases}$$

2. $$B=\frac{\sqrt{a_1 x\sin^2 2x+e^{-2x}(x+a_2)}}{x+\sqrt{a_1^2+a_2^2}}$$

3. Дано натуральне тризначне число. Якщо всі цифри в ньому однакові, то залишити задане число без змін; якщо всі цифри в ньому різні, то меньшу з них замінити в заданому числі нулем; якщо дві цифри в числі однакові, то отримати число зі зворотнім порядком цифр.

4. Дано натуральне число *N*. Обчислити добуток $\left(1+\frac{1}{1^2}\right)\cdot\left(1+\frac{1}{2^2}\right)\cdot\ldots\cdot\left(1+\frac{1}{N^2}\right)$.

5. Є натуральні числа *a* и *b*. Визначити всі числа, які кратні *a* и *b*, менші за *a***b* (*a* та *b* повинні бути більше 10).

6. Факторіал певного числа дорівнює р. Знайти це число.

7. Обчислити значення суми нескінченого ряду $S=\frac{\pi}{3}-\frac{\left(\frac{\pi}{3}\right)^3}{3!}+\frac{\left(\frac{\pi}{3}\right)^5}{5!}+\ldots+(-1)^n\frac{\left(\frac{\pi}{3}\right)^{2n+1}}{(2n+1)!}+\ldots$ с заданою точністю $\varepsilon=10^{-4}$ і значення функції (для перевірки) $f=\sin\left(\frac{\pi}{3}\right)$.

8. Обчислити значення суми нескінченого ряду

$S = x - \frac{x^2}{2} + \frac{x^3}{3} - ... + (-1)^{n-1} \frac{x^n}{n} + ...$ с заданою точністю ε=10^{-6} , значення функції (для перевірки) *f*=ln(1+*x*), врахувати -1<x≤1

9. Є натуральні числа *a*, *b* (*a*<*b*). Отримати всі прості числа *p*, які задовольняють нерівності *a*<*p*<*b*.

Варіант 10

1. $f(m,n) = \begin{cases} \frac{2m}{n-3}, \text{якщо } n \neq 3, m \leq 3 \\ \sqrt{m-3} - n, \text{якщо } 3 < m \\ m + n, \text{ в інших випадках} \end{cases}$

2. $K = \frac{2e^a}{3b^2\sqrt{b}} \cdot \frac{\cos^2 \frac{a}{2b+1}}{1 + \frac{1}{\ln a + b}}$

3. Напишіть програму для вирішення нерівності *ax*+*b*>0 відносно *x* для будь-яких значень *a* и *b*.

4. Для натурального числа n отримати всі його натуральні непарні дільники.

5. Визначити найбільше число, факторіал якого не перевищує 10^5.

6. Отримати число, яке утворене записом цифр вихідного числа *N* у зворотному порядку.

7. Обчислити значення суми нескінченого ряду $f(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - ... + \frac{(-1)^n x^{2n+1}}{(2n+1)!} + ...$ с заданою точністю ε=10^{-5} і значення функції (для перевірки) *y*=sin*x*.

8. Обчислити значення суми нескінченого ряду с заданою точністю ε=10^{-4} $f(x) = 1 + \frac{\cos x}{1!} + \frac{\cos 2x}{2!} + ... + \frac{\cos nx}{n!} + ...$, значення функції (для перевірки) *y*=$e^{\cos x}$*cos(sin*x*), врахувати, що $0.1 \leq x \leq 1$.

9. Дано натуральне число n. Обчислити S=1+2^2+3^3+…+n^n. Формулу піднесення до ступеня не використовувати.

Варіант 11

1. $f(x,y) = \begin{cases} y - x, \text{якщо } x < 0 \\ y^2 - x, \text{якщо } x \geq 0, y < 0 \\ \sin y, \text{ в інших випадках} \end{cases}$

2. $A = \frac{|x| + \cos \frac{y}{2x} - \sin z}{\sqrt{e^{xy}} + \ln|tgz|}$

3. Є три цілих числа: *K*, *M* та *N*. Визначити, скільки серед заданих чисел парних. Якщо жодного, то кожне число піднести у другу ступінь, якщо одне, то збільшити його на 1, якщо два, то непарне подвоїти. Якщо всі числа непарні,

змінняти їх не потрібно.

4. Розробити програму для обчислення значення $\underbrace{\sqrt{2+\sqrt{2+\sqrt{...+\sqrt{2}}}}}_{n}$ при заданому значенні *n*.

5. Перевести число з десяткової системи числення у вісімкову тобто отримати число, яке є вісімковим записом числа *n*.

6. Визначити, скільки разів в написанні введеного з клавіатури натурального числа зустрічається цифра 2.

7. Обчислити значення суми нескінченого ряду $S=\frac{\pi}{3}-\frac{\left(\frac{\pi}{3}\right)^3}{3!}+\frac{\left(\frac{\pi}{3}\right)^5}{5!}+...+(-1)^n\frac{\left(\frac{\pi}{3}\right)^{2n+1}}{(2n+1)!}+...$ з заданою точністю $\varepsilon=10^{-4}$, значення функції (для перевірки) $f=\sin\left(\frac{\pi}{3}\right)$.

8. Обчислити значення суми нескінченого ряду $f(x)=x+\frac{x^3}{2\cdot 3}+\frac{1\cdot 3\cdot x^5}{2\cdot 4\cdot 5}+...+\frac{(2n-1)!!x^{2n+1}}{(2n)!!(2n+1)}+...$ з заданою точністю $\varepsilon=10^{-5}$, значення функції (для перевірки) $f(x)=\arcsin x$, врахувати, що $0{,}05 \le x \le 1$.

9. Стверджується, шо різниця любого натурального числа і суми його цифр кратна 9. Перевірити цей факт для всіх чисел, які лежать між заданими *m* и *n.*

Варіант 12

1. $f(x,y)=\begin{cases} y/x, \text{якщо} x \neq 0 \\ y^2, \text{якщо } x=0, y<0 \\ 5 \text{ в інших випадках} \end{cases}$

2. $Q=\sqrt{\frac{\sin x+\cos^2 x}{2\ln(x^2+e^{-x})}}+\frac{1}{1+\frac{1}{2x}+|x|}$

3. Точки з координатами (x1, y1) та (x2, y2) є кінцями відрізку. Визначити чи перетинає цей відрізок графік функції $F(x)=x$. Якщо перетинає, то знайти відстань від кожної точки до даної прямої. Якщо не перетинає, то визначити, чи належить цей відрізок прямій, паралельній даній, або прямій, що перпендикулярна.

4. Дано натуральне число *n*. Обчислити F=1!+2!+…+*n!*

5. Вивести на екран значення першого від'ємного члена послідовності *cos(ctg(n)), n=1,2,3,....* Від'ємні члени у цій послідовності обов'язково є.

6. Підприємець, почавши справу, взяв кредит розміром *k* у.о. під *p* процентів річних та вложив його у свій бізнес. За прогнозами, його справа повинна давати прибуток *r* у.о. на рік Чи зможе підприємець накопичити суму, достатню для

погашення кредиту, а якщо да, то через скільки років?

7. Обчислити значення суми нескінченого ряду $S = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \ldots$ з точністю до члену ряду, який по модулю менше $\varepsilon=10^{-5}$, та значення функції (для перевірки) $f = \frac{\sin x}{x}$, врахувати, що $-\pi \le x \le \pi$.

8. Обчислити значення суми нескінченого ряду $S = 1 - \frac{1}{4}x + \frac{1*5}{4*8}x^2 - \frac{1*5*9}{4*8*12}x^3 + \frac{1*5*9*13}{4*8*12*16}x^4 - \ldots$ з точністю до члена ряду, який по модулю менше $\varepsilon=10^{-4}$, та значення функції (для перевірки) $f=(1+x)^{-1/4}$, врахувати, що $x^2<1$.

9. Знайти всі такі пари натуральних чисел *a* та *b*, що якщо число *a* піднести до другого ступеня та до отриманого числа дописати справа десятковий запис числа *b*, то буде число, яке більше добутку чисел *a* та *b* рівно у три рази.

Варіант 13

1. $$f(t,s) = \begin{cases} \sqrt[4]{t-s}, \text{якщо } t \ge s, 2 < s \le 4 \\ s^4 + 2t, \text{якщо } t < 0 \\ t + 2, \text{ в інших випадках} \end{cases}$$

2. $$S = \frac{\sqrt{\left|(\lg ctgx)^2 - \frac{\sqrt[4]{3x}}{\cos x}\right| + \sqrt{\frac{1}{2x}} + 1}}{e^{-3x} + e^{arctgx}}$$

3. Є натуральне тризначне число. Якщо всі цифри в ньому різні, то залишити задане число без змін; якщо всі цифри в ньому однакові, то першу зменшити на 1, а останню, якщо це не 9, збільшити на 1; якщо дві цифри у числі однакові, то отримати число зі зворотнім порядком цифр.

4. Дано натуральне n. Обчислити суму n доданків $\sin x + \cos\sin x + \sin\cos\sin x + \ldots$

5. Дано натуральне число N. Пояснити, чи є воно ступенем числа 5.

6. Відомий час початку і закінчення (наприклад, 6:00 та 24:00) роботи певного приміського автобусного маршруту з одним автобусом на лінії, а також тривалість маршруту у хвилинах (в один кінець) і час відпочинку на кінцевій зупинці. Скласти добовий розклад цього маршруту (час відправлення з кінцевих пунктів) без врахування часу на обід та перезмінку.

7. Обчислити значення суми нескінченого ряду з заданою точністю $\varepsilon=10^{-6}$ $f(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots + \frac{x^{2n}}{(2n)!} + \ldots$, значення функції (для перевірки) $chx = \frac{e^x + e^{-x}}{2}$, врахувати, що $0.1 \le x \le 1$.

8. Обчислити значення суми нескінченого ряду з заданою точністю $\varepsilon=10^{-5}$

$f(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - ... + \frac{(-1)^n x^{2n+1}}{2n+1} + ...$, значення функції (для перевірки) f=arctg(x), рахувати, що $x^2<1$.

9. Дано позитивне число *k*. Для кожного значення *x=2,3,4,…,8* знайти таке найменше ціле *n*, при якому x^n перевищує задане *k*.

Варіант 14

1. $f(m,n) = \begin{cases} \frac{5}{m} - \frac{n}{5}, \text{якщо } n > -5, m \neq 0 \\ 3m + n^2, \text{якщо } n \leq -5 \\ 2mn, \text{ в інших випадках} \end{cases}$

2. $S = \frac{\sin \alpha^3 + 2\cos^2 \beta}{\sqrt{2{,}5\alpha + 3\beta + \sqrt{2}} \ln \beta}$

3. У шашковому ендшпілі залишились біла дамка і два чорних пішака, позиції яких відомі. Хід білих. Чи зможе дамка вбити хоча б одного пішака?

4. Розробити програму для визначення, у яких двозначних числах подвоєна сума цифр дорівнює їх добутку.

5. Одноклітинна амеба кожні 3 години ділиться на 2 клітини. Вважаючи, що спочатку в замкненому об'ємі знаходиться 10 клітин, визначити, через який час в цьому об'ємі буде знаходитися 10^5 клітин.

6. Получити число, яке утворено записом цифри вихідного числа N у зворотному порядку.

7. Обчислити значення суми нескінченого ряду з заданою точністю $\varepsilon=10^{-5}$ $f(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - ... + \frac{(-1)^n x^{2n}}{(2n)!} + ...$, значення функції (для перевірки) *y*=cos*x*.

8. Обчислити значення суми членів нескінченого ряду $S = \ln x + 2\left[\frac{a}{2x+a} + \frac{a^3}{3(2x+a)^3} + \frac{a^5}{5(2x+a)^5} + ... + \frac{a^{2n-1}}{(2n-1)(2x+a)^{2n-1}} + ...\right]$ з точністю до члена ряду, меншого $\varepsilon=10^{-4}$ для $a^2<(2x+a)^2$ і значення функції (для перевірки) *f=ln (x+a).*

9. Обчислити $P = \prod_{i=1}^{10} \sum_{j=1}^{20} \frac{1}{i + j^2}$.

Варіант 15

1. $f(x,y) = \begin{cases} \frac{x-2}{y-4}, \text{якщо } y \neq 4, -y < x \leq y \\ \frac{y}{x}, \text{якщо } x \neq 0, y < x \\ yx, \text{ в інших випадках} \end{cases}$

2. $S = \frac{\cos^2 \frac{x}{2y} - 3tg \frac{y}{2}}{e^{xy} + \sqrt{12x^2 - \ln|y|}}$

3. На шаховій дошці стоять чорний король і білі тура і слон (тура б'є по

горизонталі та вертикалі, слон - по діагоналям). Перевірити, чи є загроза королю та якщо є, то від кого саме. Врахувати можливість захисту (наприклад, тура не б'є через слона).

4. Сума цифр тризначного числа кратна 7, само число також ділиться на 7. Зайти всі такі числа.

5. Визначити чи є натуральні числа *a* и *b* взаємно простими. Взаємно прості числа не мають спільних дільників, окрім одиниці.

6. Можна їхати на таксі зі швидкістю V_1 км/год та оплатою *p* р/км або йти пішки зі швидкістю V_2 км/год безкоштовно. Як з найменшими витратами подолати відстань S за час *t,* якщо це можливо? Які ці витрати?

7. Обчислити значення суми нескінченого ряду $f(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - ... + \frac{(-1)^n x^{2n+1}}{(2n+1)!} + ...$ з заданою точністю ε=10^{-5} і значення функції (для перевірки) *y*=sin*x*.

8. Обчислити значення суми нескінченого ряду з заданою точністю ε=10^{-4} $f(x) = 1 + 3x^2 + ... + \frac{2n+1}{n!}x^{2n} + ...$, значення функції (для перевірки) $y = (1+2x^2)e^{x^2}$, врахувати, що $0.1 \le x \le 1$.

9. Обчислити $s = \sum_{k=1}^{10} \frac{\sum_{n=1}^{k} \sin kn}{k!}$.

Варіант 16

1. $$f(p,q) = \begin{cases} \frac{p+q}{2q}, \text{якщо } q \ge 100 \\ p - 2q, \text{якщо } 0 < q < 100, p < 20 \\ p - 3q, \text{в інших випадках} \end{cases}$$

2. $$S = \frac{\cos x - 3ctgy}{e^{-xy} + \sqrt[6]{x^2 - \lg\left|\frac{y}{x}\right|}}$$

3. Визначити чи належать дві точки з координатами (*x*1, *y*1) та (*x*2, *y*2) одній чверті. Якщо так, то обчислити периметр трикутника, вершинами якого є початок координат та дані точки, якщо всі точки розташовані на одній прямій, то вивести повідомлення про це. Якщо точки в одній чверті не лежать, то визначити, чи знаходяться вони в одній напівплощині.

4. Ввести натуральне число N. Визначити, чи є воно досконалим (досконале число N дорівнює сумі всіх своїх дільників, які не перевищують саме N).

5. Розробити програму для визначення чи є натуральне число k ступенем числа 3.

6. Початковий об'єм деревини на ділянці лісу становить *p* кубометрів.

Щорічний приріст складає *k*%. Річний план заготівлі складає *t* кубометрів. Визначити, через скільки років в цьому лісі будуть рости лише опеньки?

7. Обчислити значення суми нескінченого ряду з заданою точністю ε=10^{-5}, $f(x)=x+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}+\ldots$, значення функції (для перевірки) $shx=\frac{e^x-e^{-x}}{2}$.

8. Обчислити значення суми нескінченого ряду з заданою точністю ε=10^{-4} $f(x)=1+\frac{\cos\frac{\pi}{4}}{1!}x+\ldots+\frac{\cos n\frac{\pi}{4}}{n!}x^n+\ldots$, значення функції (для перевірки) $f=e^{x\cos\frac{\pi}{4}}\cos\left(x\sin\frac{\pi}{4}\right)$, врахувати, що $0.1\le x\le 1$.

9. Дано натуральне N. Обчислити суму $\sum_{k=1}^{N}\sum_{i=1}^{k+1}\frac{1}{k^2+i}$

Варіант 17

1. $f(m,n)=\begin{cases} n-4, \text{якщо } n\ge 0 \\ \frac{m+n}{mn}, \text{якщо } m<0, n<0 \\ m+5, \text{в інших випадках} \end{cases}$

2. $A=\frac{\sin x\cos y-tg\frac{x}{y}}{\ln\left|4y^3\right|}e^{-x}$

3. Визначити, чи належать три точки с координатами (*x*1, *y*1), (*x*2, *y*2), (*x*3, *y*3) одній прямій. Екщо не належать, то обчислити відстань до кожної точки з початку координат. Якщо належить, то отримати загальне рівняння цієї прямої вигляду *Ax*+*By*+*C*=0, де A≥0.

4. Отримати таблицю температур за Цельсієм від 0° до 100° та їх еквівалентів за шкалою Фаренгейту, використовуючи для перерахунку формулу $t_F=\frac{9}{5}\cdot t_c+32$.

5. Обчислити суму кубів усіх парних чисел, які знаходяться у діапазоні від X до Y, де X та Y – натуральні числа, які вводяться з клавіатури.

6. З клавіатури вводиться число N. Визначити чи може воно бути двійковим (тобто складатися лише з 0 та 1).

7. Обчислити значення суми нескінченого ряду з заданою точністю ε=10^{-6} $S=\frac{1}{2}+\sum_{n=0}^{\infty}\frac{(-1)^n 2^{2n-1}x^{2n}}{(2n)!}$, значення функції (для перевірки) *y*=cos^2*x*.

8. Обчислити значення суми нескінченого ряду з заданою точністю ε=10^{-6}

$f(x)=\frac{x\cos\frac{\pi}{3}}{1}+\frac{x^2\cos 2\frac{\pi}{3}}{2}+\ldots+\frac{x^n\cos n\frac{\pi}{3}}{n}+\ldots$, значення функції (для перевірки) $y=-\frac{1}{2}\ln\left(1-2x\cos\frac{\pi}{3}+x^2\right)$ врахувати, що $0.1\le x\le 0.8$.

9. Знайти усі прості числа *p*, для кожного з яких існує таке число *a*, що дріб $\frac{a^4+12a^2-5}{a^3+11a}$ зменшується на *p*.

Варіант 18

1. $f(x,z)=\begin{cases}\frac{x-z}{z-1}, \text{якщо } x>0, z>1\\ 2xz, \text{якщо } x\le 0\\ x+1, \text{ в інших випадках}\end{cases}$

2. $Y=\frac{2\cos(x-\frac{\pi}{6})+\sqrt{2}}{\frac{1}{2\ln x}+\sin^2 x^2}\cdot e^{3x}$

3. Для заданого цілого *a* (0<*a*<100) , яке розглядається як вік людини, вивести фразу типу: «Мені 21 рік», «Мені 32 роки», «Мені 12 років».

4. Написати програму обчислення значення виразу при заданих x и n: $\sin x+\sin\sin x+\ldots+\underbrace{\sin\sin\ldots\sin x}_{n}$.

5. З клавіатури вводиться послідовність чисел, знайти максимальне з них. Ознака закінчення вводу – виводу 0 (в обчисленнях не враховується).

6. Визначити чи є сума цифр натурального числа N парною.

7. Обчислити значення суми нескінченого ряду $S=1+\frac{\ln 3}{1!}x+\frac{\ln^2 3}{2!}x^2+\ldots+\frac{\ln^n 3}{n!}x^n+\ldots$, з точністю до члена ряду, яке за модулем менше $\varepsilon=10^{-4}$, значення функції (для перевірки) $f=3^x$, врахувати, що $0\le x\le 1$.

8. Обчислити значення суми нескінченого ряду з заданою точністю $\varepsilon=10^{-5}$ $f(x)=1-\frac{1}{2}x+\frac{1\cdot 3}{2\cdot 4}x^2-\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}x^3+\ldots+(-1)^n\frac{(2n-1)!!}{(2n)!!}x^n+\ldots$ та значення функції (для перевірки) $y=\frac{1}{\sqrt{1+x}}$, врахувати, що $x^2<1$.

9. Дано *k* натуральних чисел. Визначити скільки з них є досконалими. Досконалим вважається число, яке дорівнює сумі всіх своїх дільників, включаючи 1 та не включаючи саме число.

Варіант 19

1. $f(x,y)=\begin{cases}\frac{x-1}{x+1}, \text{якщо } x>y, x\ne -1\\ xy, \text{якщо } x<y\\ 3y, \text{в інших випадках}\end{cases}$

2. $F = \dfrac{\sqrt{1+\sqrt{|x|}} + \ln x^3}{1 - \dfrac{2x^2 - 1}{2x}} + 10^4 \cos^2 x$

3. Є чотири числа: *a*, *b*, *c* та *d*. Визначити чи є серед них однакові по модулю. якщо знайдеться одна пара однакових по модулю значень, то обнулити їх. Якщо знайдеться три однакових по модулю значення, то змінити знак у кожного з них. Якщо всі значення однакові по модулю, то піднести їх до другого ступеня. Якщо знайдеться дві різні пари однакових за модулем значень, то більше за модулем значення подвоїти, а менше – потроїти.

4. Знайти всі двозначні числа, квадрати яких закінчуються на три однакові цифри, що відмінні від нуля.

5. Є два натуральних числа X та Y. Розробити програму для обчислення суми кубів усіх парних чисел, які належать діапазону [X, Y].

6. З клавіатури вводиться число N. Визначити чи є воно вісімковим (тобто складається лише з цифр, які менші за 8).

7. Обчислити значення суми нескінченого ряду $S = 1 + \dfrac{x \ln a}{1!} + \dfrac{(x \ln a)^2}{2!} + \ldots + \dfrac{(x \ln a)^n}{n!} + \ldots$ з точністю до члена ряду, який менше $\varepsilon=10^{-4}$, значення функції (для перевірки) $f=a^x$, врахувати, що $0{,}1 \le x \le 1$.

8. Обчислити значення суми нескінченого ряду з заданою точністю $\varepsilon=10^{-5}$ $S = 2\left[\dfrac{1}{x} + \dfrac{1}{3x^3} + \dfrac{1}{5x^5} + \dfrac{1}{7x^7} + \ldots\right]$, значення функції (для перевірки) $y = \ln\left(\dfrac{x+1}{x-1}\right)$, врахувати, що $|x|>1$.

9. Послідовність $a_1, a_2, a_3, \ldots$ яка утворена за законом $a_1=b$, $a_i = a_{i-1} - \dfrac{1}{\sqrt{i}}$, $i=2,3,\ldots$. Знайти перший від'ємний член послідовності для різних b=3,4,5,6,7,8,9.

Варіант 20

1. $f(x,y) = \begin{cases} 2x - y, \text{якщо } 0 \le x < 5 \\ x^2, \text{якщо } x < 0, y < 0 \\ 5y + 1, \text{в інших випадках} \end{cases}$

2. $N = \dfrac{xyz - 3.3\left|x + \sqrt{y}\right|}{10^7 + \sqrt{\ln z} - \cos^2 y}$

3. Визначити вид трикутника (прямокутний, гострокутний або тупокутний) для 3 значень відрізків. Якщо трикутник прямокутний, то знайти величини його гострих кутів, якщо він гострокутний, то знайти його периметр, а в протилежному випадку знайти його найбільшу сторону.

4. Розробити програму для визначення найменшого серед чисел $k^3 \sin\left(n + \dfrac{k}{n}\right)$,

(k=1, 2, …, n).

5. Визначити чи є дане натуральне число N факторіалом будь-якого числа, якщо «так», то якого.

6. Знайти всі шестизначні числа, які збільшуються втрое при перестановці першої цифри числа в кінець.

7. Обчислити значення суми нескінченого ряду з точністю до члена ряду, який за модулем менше $\varepsilon=10^{-4}$, $S = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \ldots + (-1)^n \frac{x^n}{n!} + \ldots$, значення функції (для перевірки) $f=e^{-x}$.

8. Обчислити значення суми нескінченого ряду $S = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \ldots + \frac{(x-1)^n}{nx^n} + \ldots$ з точністю до члена ряду, який менше $\varepsilon=10^{-5}$, та значення функції (для перевірки) $f=\ln x$, врахувати, що $x>1$.

9. Послідовно вводяться n натуральних чисел($n \le 10$). Обчислити суму тих з них, у яких перша цифра дорівнює останній.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Чим відрізняється умовна операція від умовного оператору?
2. Що таке повна і неповна форма умовного оператору?
3. Чи може існувати неповна форма умовної операції?
4. Чи потрібно писати "else", якщо при виконанні умови виконується оператор return?
5. Вирази якого типу можуть визначати умови?
6. Які значення виразів, що визначають умови, вважаються істинними, а які хибними?
7. Які операції відносяться до операцій відношення?

8. Чим відрізняються операції "= =" від операції "="?
9. Які операції відносяться до логічних? Який пріоритет їх виконання?
10. Якою операцією можна замінити операцію "&&" ?
11. Якою операцією можна замінити операцію "||" ?
12. Чому може дорівнювати значення виразу відношення або логічного виразу?
13. Як правильно порівняти на рівність дійсні числа?
14. Як правильно перевірити входження значення у певний діапазон?
15. Як перевірити певне цілочисельне значення на рівність нулю?
16. Як перевірити відмінність цілочисельного значення від нуля?
17. Коли використовується вложення умовних операторів?
18. Як правильно записати вложені умовні оператори?
19. Що являє собою оператор switch? Як їм користуватися?
20. Як записати оператор switch з допомогою умовних операторів?
21. Що таке цикл?
22. Які види циклів снують? Які оператори циклу існують у мові С?
23. Чим відрізняються цикл з передумовою від циклу з постумовою?
24. Коли необхідно використовувати цикл з передумовою, а коли з постумовою? Наведіть приклади.
25. Які цикли з передумовою існують у мові С?
26. Скільки операторів містять у собі тіло циклу у мові С?
27. Як правильно записати цикл з постумовою на мові С?
28. Як задати нескінчений цикл? Навіщо він потрібен? Як з нього вийти?
29. Яким повинно бути значення виразу, який визначає умову виконання циклу, для завершення циклу?
30. Яким повинно бути значення виразу, який визначає умову виконання циклу, для виконання тіла циклу?
31. До чого призведе неправильне завдання виразу, який визначає умови виконання циклу?
32. Чи може бути відсутнім тіло циклу? Якщо може, то наведіть приклади таких циклів.
33. Чим відрізняється оператор *while* від оператора *if*?
34. Який порядок дій при виконанні циклу *for*?
35. Як організувати арифметичний цикл з допомогою циклу *for*?
36. Запишіть алгоритм, який визначається циклом *for*, з допомогою циклу *while*.
37. Що таке вкладений цикл?
38. Скільки разів у загальній складності виконується тіло вкладеного циклу?
39. Як і коли використовуються оператори *break* то *continue*?
40. Що таке рекурентні обчислення? Коли вони використовуються? Як їх програмувати?

РОЗДІЛ VI. Керуючі структури

6.1 Теоретичні відомості. Керуючі структури

Керуючі структури визначають, які операнди необхідно виконувати і в якому порядку. Інакше визначають потік управління у програмі. Як зазначалось, Б'ом та Яконіні [3] показали, що будь-який алгоритм може бути реалізований за допомогою трьох керуючих структур: послідовне виконання; умовне виконання; цикл.

У мові С будь-який вираз стає оператором, якщо за ним поставити; наприклад: a = 0; i++; printf ("%d\n", x);

Послідовність операторів, обмежена { } є складеним оператором, або блоком. При цьому після правої дужки ; не вживається.

Різновиди умовних операторів детально розглянуто у п. 5.2 цього навчального посібника.

До прикладу наведено варіант використання умовних операторів у наступній програмі

```
// перевірка на можливість побудови трикутника

main ()
{ float a, b, c; int i;
  printf (" Введіть сторони a, b, c,\n"); scanf (" %f %f %f", &a, &b, &c);
  i = (a+b>c)&&(b+c>a)&&(a+c>b); if (i)
printf (" Трикутник можна побудувати \n"); else
   printf (" Трикутник побудувати неможливо !!! \n"); }
```

Можливе використання й скороченого оператору: **if вираз) оператор1;**

Якщо вираз має ненульове значення, то виконується оператор1. Інакше переходимо до виконання наступного за порядком оператора.

Ніякі обмеження на тип операторів не накладають. Тому можна використовувати й вкладені умовні оператори. Приходимо до конструкцій:

if вираз) оператор else if . else if

Оскільки в таких послідовностях можна використовувати як повні умовні оператори, так і скорочені, то виникають неоднозначності:

якщо до оператора віднести **else**

```
if (n>0)       if (a>b)  z=0;  else z=b;
```
або
```
if (n>0) {if (a>b)  z=0;} else z=b;
```

Діє правило, що саме внутрішнє else ставиться до найближчого ліворуч **if**. Якщо ж потрібно змінити приналежність, то необхідно використовувати фігурні дужки.

У тих випадках, коли виникає вибір з багатьма можливими результатами, доцільно використовувати перемикач. Його структура детально описана у п.5.2 цього посібника.

Оператор перемикач **switch** виконується в такий спосіб:

1. Виконується вираз в круглих дужках. Його значенням може бути або ціле, або літера (символ);

2. Отримане значення рівняється з константами вибору. Якщо воно збігається з однієї з констант1 ... константа n, то виконується відповідний оператор i;

3. Далі виконуються всі наступні оператори від i+1 до default (n+1) (відсутність) включно;

4. Якщо значення виразу не збігається з ні однієї константою вибору, то виконується оператор з міткою default.

Для прикладу наведено розв'язок задачі: ввести номер місяця й надрукувати його назву.

```
main ()
{ int month;
printf ("Введіть номер місяця \n");
scanf (" %2d",&month);
switch (month);
     { case 1: printf (" Це січень. \n");
       case 2: printf ("Це лютий. \n");
       ……….
       case 12: printf ("Це грудень. \n");
       default : printf ("Такого місяця немає. \n");
     }
}
```

Якщо ввести 1, то буде виведено:

Це січень. .....

Це грудень.

```
Такого місяця немає.
```

Якщо ввести 20, то буде надрукована лише остання позиція.

Для того, щоб, вибір був лише один, то в кожний варіант необхідно додати оператор break (вихід, обравши напрям потоку).

```
{ int month;
printf (“Введіть номер місяця \n”);
scanf (“ %2d”,&month);
switch (month);
      { case 1: printf (“ Це січень. \n”); break;
      case 2: printf (“Це лютий. \n”); break;
      ……..
      case 12: printf (“Це грудень. \n”); break;
      default : printf (“Такого місяця немає. \n”);
      }
   }
```

Константа може бути і символом: main ()

```
{ char c;
switch (c)
    { case ‘a’: printf (“Введено символ ‘a’. \n”); break; .....
      case ‘z’: printf (“Введено символ ‘z’. \n”); break;
      default : printf (“Такого символу не знаю. \n”); break;
    }
  }
```

Ясно, що варіанти можуть розташовуватися в довільному порядку, а не в певному. Дозволяється кілька констант вибору до одному операторові.

```
switch (month)
{ case 1:
   case 2:
   case 12: printf (“Це зима. \n”); break; case 3:
   case 4:
   case 5: printf (“Це весна. \n”);
}
```

Структура і різновиди операторів циклів детально описані у розділі 5.2 цього посібника.

Якщо необхідно перервати цикл до його завершення, то можна використовувати оператор break.

Для прикладу: необхідно визначити, чи є число n простим, тобто ділиться лише саме на себе і на 1:

```
main ()
{ int n, i=1;
printf (“Введіть ціле число \n”);
scanf (“ %5d”, &n);
while (++i<n)
        if (n%i==0) //остача від ділення
        {printf (“ число НЕ просте \n”); break;}
        if (i==n) printf (“Число просте \n”); }
```

З іншого боку, якщо певну послідовність операторів не треба виконувати та необхідно продовжити цикл (тобто повернутись до його заголовку), то в такому випадку можна використати оператор **continue**.

Для прикладу необхідно: для чисел від **1** до **n** надрукувати всі, не кратні 5, тобто ті, які діляться на 5 – не друкувати.

```
main ()
{ int n, i=0;
   printf (“Введіть число \n”);
   scanf (“%5d”,&n);
   while (++i<n)
        { if (i%5==0) continue;
        printf (“%5d \n”,i);
        }
}
```

Для прикладу: програма для перевірки коректності введення числа.

```
        main ()
{ int err;
   do{ err=0;
        printf (“Введіть ціле число \n”);
        if (!scanf (“%5d”,&n)) //не більше 5-ти символів
           { printf (Помилка вводу числа \n ”);
```

```
        err=1;
        }
    }  while (err);
}
```

Єдина його відмінність в тому, що умова перевіряється після виконання оператора, якщо умова істина, то все повторюється знову.

Для прикладу ілюстрації використання оператору циклу for: потрібно обчислити n!.

```
    nfactor ()
{ int i, n, nfact=1;
  printf (“Введіть ціле число \n”);
  scanf (“%2d”,n);
  for (i=1; i<=n; i++)    nfact*=i;
  printf (“ Факторіал %2d =%6d \n”, n, nfact);
}
```

На відміну від інших мов програмування лічильник циклу - це просто звичайна змінна. Тому можлива модифікація лічильнику в самому циклі. Після виходу із циклу значення лічильника запам'ятовується.

У загальному випадку, вирази тіла циклу мають більш широкий зміст. Наприклад, вони взагалі на можуть бути відсутніми, але два знаки "**;**" повинні зберігатися обов'язково:

```
for (i=1;;i++) printf(“Нескінченний цикл”);
```

Вийти з нескінченного циклу можна за допомогою оператора **break**. Замість виразів тіла циклу можна використовувати звертання до функцій:

```
for (putchar (‘a’); putchar (‘b’); putchar (‘c’))
putchar (‘d’);
```

Маємо нескінченний вивід abdcbdcbdcbdc...bdc. Допускається використання й вкладених циклів.

При цьому можна одержати дуже компактні конструкції, зрозуміти які буває дуже складно.

Для прикладу наведено швидке сортування Шела [2, 3].

```
        void main(void)
{ int v[]={5,-4,0,-17,68,36,90,-7,5,4,12,1,9};
 int gap, i, j, k,temp,n=13;
 for (gap=n/2; gap>0; gap/=2)
 for (i=gap; i<n; i++)
 for (j=i-gap; j>=0&&v[j]>v[j+gap]; j-=gap)
        {
        temp=v[j];
        v[j]=v[j+gap];
        v[j+gap]=temp;
        }
 for (k=0;k<n;k++) cout<<"  "<<v[k];
 getch();
 }
```

Масив розбивається на групи, кожна з яких складається із двох елементів. Відстань між парами елементів

d=n/2,

де n - кількість елементів масиву.

Елементи- Пари рівняються між собою і якщо потрібно, то міняються місцями. Потім групи попарно зливаються. Кожна нова група має 4 елемента, відстань між елементами d=d/2. У середині групи виконується сортування, потім групи зливаються. Процес триває доти , поки відстань між елементами не стане 1. На цьому етапі масив сортується методом вставки (пухирця).

Конструкції gap/=2 і j/=gap значить відповідно gap=gap/2 і j=j-gap. Зовнішній цикл змінює зрушення між парою елементів, які рівняються, від n/2, n/4 до 1. Наступний внутрішній цикл забезпечує перегляд масиву, починаючи з n/2, n/4 і 1- го до кінця.

Третій цикл (по j) до деякої міри є фіктивним: у ньому для певної пари для випадку, коли перший елемент більше другого, вони переставляються. Після цього j-=gap, тобто стає менше нуля й відбувається вихід із циклу.

Іноді в циклі for можна відразу використовувати два індекси. У цьому випадку одне вираз поєднує два за допомогою оператора ",": вираз 1, вираз 2. Кома розділяє два вирази, які виконуються зліва направо. Значення й тип операнду визначається правим виразом. Лівий вираз є ніби другорядним.

Наприклад: переставити символи рядка

```
reverse (char s[])
{ int c, i, j;
printf (“Ввести рядок \n”);
scanf (“ %s”,s);
for (i=0, j=strlen (s)-1; i<j; i++, j--)
      {  c=s[i];   s[i]=s[j];
      s[j]=c;
      }
}
```

Будемо використовувати два індекси: i- символ спочатку рядка, а j- й кінця, і буде збільшуватися, а j - зменшуватися. Обмін символами продовжуємо поки i<j. Функція strlen -визначає кількість букв у рядку.

Результати розв’язання:

```
5  -4  0  -17  68  36  90  -7  5  4  12  1  9
(5,90)(-4,-7)(0,5)(-17,4)(68,12)(36,1)

5  -7  0  -17  68  36  90  -4  5  4  12  1  9

5  -7  0  -17  12  36  90  -4  5  4  68  1  9

5  -7  0  -17  12  1  90  -4  5  4  68  36  9
5  -7  0  -17  12  1  9  -4  5  4  68  36  90
-17  -7  0  5  12  1  9  -4  5  4  68  36  90
-17  -7  0  5  -4  1  9  12  5  4  68  36  90
-17  -7 0  5  -4  1  4  12  5  9  68  36  90
-17  -7  0  4  -4  1  5  12  5  9  68  36  90
-17  -7  0  -4  4  1  5  12  5  9  68  36  90
 -17  -7  -4  0  4  1  5  12  5  9  68  36  90
-17  -7 -4  0  1  4  5  12  5  9  68  36  90
-17  -7  -4  0  1  4  5  5  12  9  68  36  90
-17  -7  -4  0 1  4  5  5  9  12  68  36  90
-17  -7  -4  0  1  4 5  5  9  12  36  68  90
```

Таким чином було проілюстровано алгоритм сортування.

6.2 Лабораторна робота 9. Змінні і константи, типи даних, ввод та вивод, оператори

Тема: Змінні і константи, типи даних, ввод та вивод, оператори

Мета: Закріпити навички написання програм мовою С за розділами: змінні і константи, типи даних, ввод та вивод, оператори.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.

Теоретичні відомості

Змінні в мові С діляться на дві групи типів: основні й похідні (рис. 6.1).

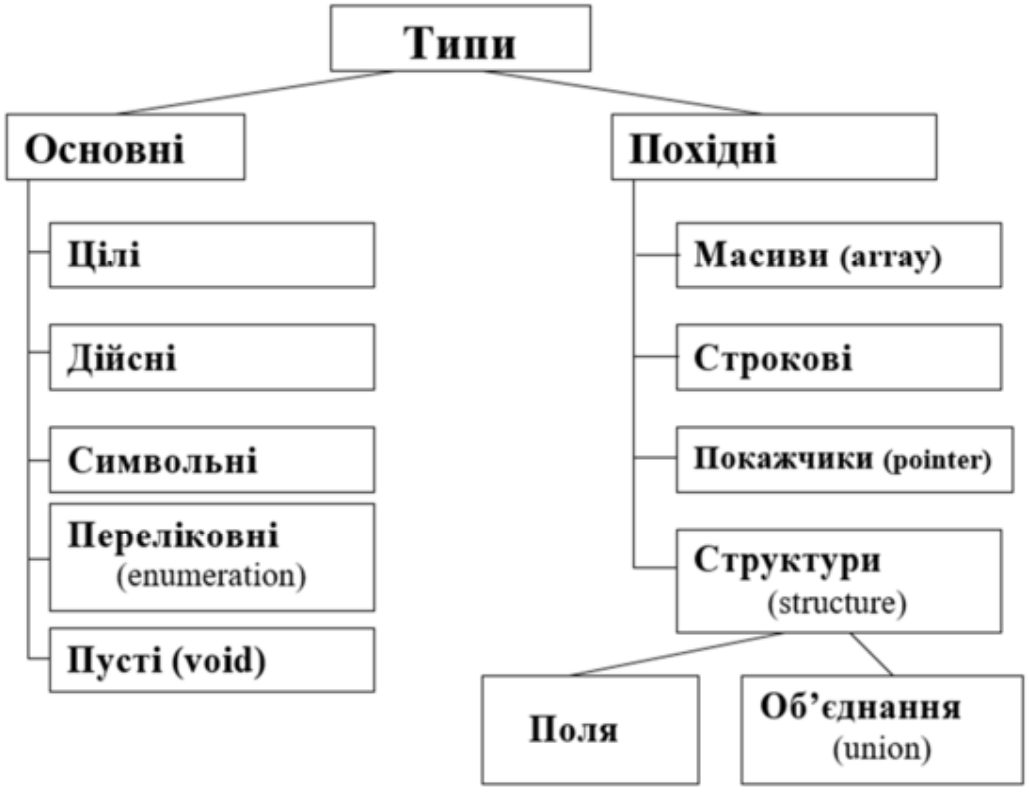


Рис.6.1 - Змінні в мові С

Константами називаються перерахування величин у програмі. Розділяють константи таких типів:

1. цілі (243);
2. з плаваючою крапкою (дійсні) (543.8);
3. символьні ('A') - 1 байт;
4. строкові ("A") - 2 байта A + NULL;
5. переліковного типу (ANSI-C).

Цілі константи можуть записуватися в десятковій, вісімковій і шістнадцятковій системах числення.

Десяткова ціла константа подається звичайним образом (зі знаком або без нього): -2561; 458.

Ознакою вісімкової константи є нуль ліворуч: Ø257.

Шістнадцяткова константа визначається двома початковими символами: ØX. Для представлення чисел від 10 до 15 у цій системі використовуються латинські букви: A - 10, B - 11, C - 12, D - 13, E - 14, F - 15.

Тому 3110 в цій системі буде записано ØX1F (1*16+ F=31).

Крім звичайних цілих констант можна використовувати цілі подвійної точності, які в пам'яті займають 2 слова.

До них приєднується літера L: 371L. Діапазон: -2.147.483.648<=2.147.483.647. Константи з плаваючою крапкою (floating point) можуть подаватися у формі з

фіксованою крапкою: 561.25, .5, 100.

При цьому в пам'яті розміщаються звичайним способом (2 слова).

Якщо ж константа задається у формі із плаваючою крапкою (експонентна форма)1E-06, 1.8E4, то в деяких трансляторах вона автоматично подається з подвійною точністю, тобто в 4 слова

Символьні константи (CHAR) набувають значення одного символу й подаються в апострофах 'x', 'a', 'r' і займають 1 байт.

Недруковані символи, які не мають графічного зображення (табл. 61), подаються умовно двома символами: перший - використовується зворотний слеш, а другий - який-небудь певний символ. Так звана зворотна послідовність перемикання коду (escape sequence або ескейп послідовність).

Таблиця 6.1 Недруковані символи

Перехід на наступну строку	\n'
Горизонтальна табуляція	\t\'
Перехід на попередню позицію	\b'
Переведення каретки	\r'
Перехід на наступну сторінку	\f'
Апостроф	\"
Звуковий сигнал	\a'
Вертикальна табуляція	\v'

Строкові константи (строки, string) це послідовність символів, обмежена подвійними лапками: "string". Для переносу на інший рядок використовується зворотний слеш

Визначення констант реалізується 3-а способами:

1. Процесором і має вигляд:

#define<им'я константи> <літерал або значення>

#define<им'я константи ><вираз з констант>

2. За допомогою слова const: **const [тип] <ім'я> = <значення>**

const float pi = 3.1415926;

const maxint = 32767;

3. Оголошення перерахування починається із ключового слова enum і має два формати вистави:

Формат 1. **enum [ім'я-тега-перелічення] {список-перелічення} опис[,опис...];**

Формат 2. **enum ім'я-тега-перелічення опис [,опис..];**

Практична частина

Завдання

Написати 9 програм згідно номеру індивідуального варіанту.

У першій програмі надрукувати на екрані трьома строками "факультет, групу, своє ПІБ".

У другій програмі ціле число, введене користувачем, зберегти у змінній і надрукувати на екран з нового рядка.

У третій програмі ввести два цілих числа, обчислити суму цих двох цілих чисел і відобразити на екрані.

У четвертій програмі обчислити добуток двох чисел з плаваючою комою, введених користувачем, та вивести на екран

У п’ятій програмі знайти значення ASCII символів свого прізвища та вивести їх на екран. Літери прізвища виводити в рядок, а під ним у рядок виводити значення відповідних ASCII символів.

У шостій програмі ввести два цілих числа, розрахувати частку та залишок від їх ділення та вивести на екран два результати на новій строчці, підписуючи що є що.

У сьомій програмі оцінити розмір змінних типу int, float, double та char за допомогою оператора sizeof.

У восьмій програмі для змінних типу int, long, long long, double, long double продемонструвати роботу ключового слова long та порівняти отримані результати з попереднім завданням.

У дев’ятій програмі ввести з клавіатури два числа, двома різними способами поміняти їх місцями, вивести результати на екран.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
    3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
    3.2 Написання програм, відналагодження програми.
    3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Визначення змінних.
2. Типи змінних
3. Типи констант.
4. Способи реалізації констант

5. типи даних
6. Ввод та вивод у мові С.
7. Оператори збільшення і зменшення у мові С.

РОЗДІЛ VII. МАСИВИ ТА ПОКАЖЧИКИ

7.1 Теоретичні відомості. Масиви та покажчики

7.1.1 Масиви

Масив - це послідовність однорідних даних, яка має фіксовану довжину. Цей тип належить до похідних, тому що складається із простих. Масиви, як і інші змінні, повинні бути описаними. Спеціальних ключових слів нема, тобто відзначається ім'я в прямокутних дужках кількість елементів:

int a[10] float x[100].

Такий опис може включатися у звичайні змінні
char c, d, f[20],
де c, d - звичайні символьні змінні.

Кожний елемент масиву визначається своїм номером a[2], x[50].

На відміну, від інших мов, нумерація елементів починається з нуля й закінчується номером N-1, де N - загальна кількість елементів. Тому a[2] - це третій елемент, а x[50] -п'ятдесят перший. Це зроблене тому, що повне ім'я масиву є базовою адресою, яка дорівнює адресі першого елемента. Тобто a==&a[0] Адреса другого елемента визначається як базова адреса плюс один &a[1]==a+1 Адреса N- го елемента рівняється базової плюс N-1. Ясно, що зсув по адресах фізичних змінних залежить від типу даних, тобто індекс необхідно помножити на відповідний коефіцієнт типу (для char це буде 1, для int 2, float 4 і т.д.). Тому, така система нумерації спрощує адресацію елементів масиву.

У деяких випадках можна не визначати кількість елементів масиву, наприклад int arr[]. Відповідний розмір визначає сам компілятор. Як і інші змінні масиви зв'язуються з відповідним класом пам'яті. Тому якщо масив відноситься до автоматичного, то його необхідно описувати в блоці. Коли масив є зовнішній і був описаний раніше, то в блоці його можна не описувати. Наприклад:

```
int arr[20]; .....
addum (arr, size) int arr[], size;
```

У початковій версії мови ініціалізувати можна було лише зовнішні й статичні масиви

```
Static int numb[5]={1, 2, 3, 4, 5};
```

Для автоматичних масивів цього робити не можна було. Версія ANSI-C зняла ці обмеження.

Для прикладу наведено програму обчислення скалярного добутку // скалярний добуток

```
main ()
{ float x[20], y[20], scal=0;
  int i, size;
  printf ("Введіть розмір масиву =");
  scanf (" %2d", &size);
  printf ("Введіть елементи масиву x \n");
    for (i=0; i<size; ++i)  scanf (" %f",&y[i]); // введення елементів масиву
    for (i=0; i<size; ++i)  scal+=x[i]*y[i]; // добуток
    printf (" scal=%f \n", scal); // вивід на екран результату
}
```

У цій програмі індекс задається цілою змінною i, але в загальному випадку він може визначатись цілим виразом, до якого входять цілі змінні та цілі константи.

7.1.2 Багатовимірні масиви

Масив може складатись не лише із даних простих типів, але і з похідних. Зокрема можна розглядати масив масивів. Приходимо до багатомірних масивів 2x, 3x, і т.д. Двомірний масив – це одновимірний масив, елементами якого є одновимірні масиви. Тому при описанні двовимірного масиву розмір по кожному виміру зазначається в окремих дужках int a[10][10]. Окремий елемент a[2][3]. Перший індекс – рядок, другий стовбця. Тому це буде елемент з третього рядка та четвертого стовбця.

Для позначення *float matr[m][n]* наочне представлення має вигляд

00	01	02	…	*0n*
10	11	12	…	*1n*
…	…	…	…	…
m0	*m1*	*m2*	…	*mn*

Порядок розташування багатомірних масивів у пам'яті такий, що першим змінюється самий правий індекс. Отже, для двовимірного масиву звичайне розташування - рядками.

A[2][2] a00 a01 a10 a11

Вище відзначалося, що іноді в деяких випадках можна не наводити розміри одномірних масивів. Але для багатомірних це не так. І насправді , якщо у функції записати int array[][], то буде незрозумілим, як такий масив ділити на рядки. Тому обов'язково потрібно відзначати кількість стовбців int array[][4], тобто в кожному рядку по 4 елемента. Початкові значення багатомірних масивів задаються в такий спосіб:

```
int z[3][2]={{1, 2},{4, 5},{7, 8}};
```

7.1.3 Оператор goto

Обробка елементів багатовимірних масивів зазвичай здійснюється за допомогою вкладених циклів. При цьому для завчасного припинення циклу по певних умовах можна використовувати оператор ***break***. Для того, щоб обійти деякі оператори не виходячи із циклу, застосовують оператор ***continue***.

Але оператор ***break*** здійснює вихід тільки з одного внутрішнього циклу. Тому при значній кількості вкладених циклів для повного припинення циклів краще використовувати оператор безумовного переходу.

З погляду структурного програмування слід уникати цього оператора. Але саме в такому випадку він є найбільше доцільним.

Нагадуємо, що мітка в мові С - це звичайне ім'я, а не ціле без знаку. Мітка спеціально не описується:

Приклад: у матриці розміром 20x20 цілих чисел знайти перше від'ємне число та надрукувати його координати.

```
main ()
{ int a[20][20], i, j, size1, size2;
  // визначення розмірів та значень елементів масиву
  for (i=0; i<size1; i++)
     for(j=0; j<size2; j++)
       if (a[i][j]<0) goto label;
  label: printf (" від'ємне число %4d має координати [%2d] [%2d] \n", i, j);
}
```

7.1.4 Строкові масиви

Рядок – це одновимірний масив символів, який закінчується

нульовим символом. Як зазначалось, рядок береться у дужки “Скоро весна!”. Рядок можна присвоювати певній змінній, якщо остання описана як символьний масив char string[]=”Скоро весна!”.

Такий масив можна вводити та виводити з використанням символу перетворення **s**. При цьому вивід триває до появи нуля.

Ніяких дій над такими масивами не передбачено. Для цього використовують стандартні функції мови (бібліотека string.h):

1. Приєднання рядка (конкатенація) char str1[], str2[];

strcat (str1, str2)

До рядка str1 приєднується рядок str2. Видаляється Ø після str1. Результат присвоюється str1. При цьому str2 не змінюється.

2. Порівняння двох рядків ***strcmp (str1, str2)***

strcmp (str1, str2)=str1-str2

strcmpi (str1, str2) та

stricmp (str1, str2) – вважає малі й великі букви однаковими Значення цієї функції менше Ø, якщо лексикографічно str1 раніше str2, дорівнює Ø, якщо вони співпадають та більше Ø – в іншому разі.

3. Копіювання рядка ***strcpy (str1, str2)***

strcpy (str1, str2)

рядок str2 копіюється в str1, str2 не змінюється.

4. Визначення кількості символів у рядку без обліку заключного нуля ***strlen (str1)***

strlen ("abcde") - 5.

5. Перетворення рядка str в число подвійної точності

double atof(char *str)

```
char str3[ ]="-345.575784876";
printf("\n\n6. str3=%s, atof(str1) %20.18f",str3,atof(str3));
strcpy (str3, "-345.54563782rt75");
printf("\n\n6. str3=%s, atof(str1) %20.18f",str3,atof(str3));
```

```
str3=-345.575784876, atof(str1) -345.575784876000000000

str3=-345.54563782rt75, atof(str1) -345.545637820000024000
```

Розпізнає символьна вистава числа із плаваючою крапкою, якщо символи

відповідають формату: [пробіли] [знак] [ddd] [.] [ddd] [e|e[знак]ddd], де [ddd] - числа;[e|e] - покажчик показника ступені;

Припиняє перетворення на першому неопізнаваному символі. У випадку переповнення ***atof*** повертає +(-)HUGE_VAL, глобальна змінна errno встановлюється в ERANGE.

6. Перетворення рядка str в число подвійної точності без втрати значення (бібліотека <stdlib.h>):

***strtod*(*str1,&str2*)**

```
char string[20], *stopstring;
float x;
string="3.1415926stop";
x=strtod (string, &stopstring);
printf (" x=%.7f  - %s\n", x, stopstring);
```

На екрані: x=3.1415925 – stop

7. Перетворення рядка str в десяткове ціле (бібліотека stdlib.h)

atoi (str1)

Розпізнає символьне представлення десяткового числа, якщо символи відповідають формату: [пробіли] [знак] [ddd], де [ddd] - числа;

Припиняє перетворення на першому не розпізнаному символі;

У випадку переповнення atoi повертає +(-)HUGE_VAL, глобальна змінна errno встановлюється в ERANGE.

8. Перетворення рядка str в десяткове довге ціле

strtol (*const char* ****str1,*** *char* *****error,*** *int* ***base)***

Перетворює рядки ***str1*** в довге ціле у відповідності з вказаним **base**, яке повинне знаходитися в діапазоні: 2- 36 включно або бути рівним нулю.

9. Перетворення цілого числа в рядок

itoa (int v, char *str, int baz)

Функція перетворить символи числа v у символьний рядок, що закінчується Null- символом, і запам'ятовує результат в str. Аргумент baz визначає основу системи числення для v; його значення може лежати в межах від 2 до 36. Якщо baz = 10 і v - негативне число, то першим символом у рядку результату буде знак мінус.

Самостійно переглянути: ***ltoa, utoa, ecvt, fcvt, gcvt.***

7.2 Лабораторна робота 10. Оператори та цикли

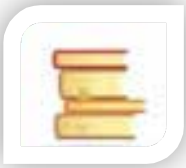
Тема: Оператори та цикли

Мета: Закріпити навички написання програм мовою С за розділами: оператори та цикли.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Спеціальні символи мови С складаються з:

1) знаків арифметичних операцій (+ додавання; - віднімання; * множення; /ділення, остача від ділення х % у цілих чисел);
2) знаків відношень;
3) роздільників;
4) службових слів.

Арифметичні операції:

	Додавання	total = var1 + var2
	Віднімання	total = var1 - var2
	Множення	total = var1 * var2
	Ділення	total = var1 / var2
+	Оператор збільшення	total = total + 1
-	Оператор зменшення	total = total - 1

(++)(--)variable - *префіксним оператором збільшення*; **variable(++)(--)** - *постфіксним оператором збільшення*;

Операція	Функція
%	Взяття по модулю або залишок; повертає залишок цілочисельного розподілу
~	Додаток; інвертує біти значення
&	Побітове І (також &X – взяти адрес X)
|	Побітове включающе АБО
^	Побітовое виключающе АБО
<<	Зсув вліво; зсуває біти значення вліво на зазначену кількість розрядів
>>	Зсув вправо; зсуває біти значення вправо на зазначену кількість розрядів

Операції піднесення до ступеня не передбачено

Знаки відносин і логічних операцій:

>, <	Більше, менше
==	Дорівнює (2 знака рівності)
!=	Не дорівнює
&&	Логічне І
||	Логічне АБО
!	Логічне НЕ

Різновиди та особливості циклів дивись попередні лабораторні роботи.

Практична частина

Завдання

Написати 8 програм згідно номеру індивідуального варіанту.

Варіант 1.

У першій програмі перевірити чи є введене число парним або непарним.

У другій програмі знайти корені квадратного рівняння.

У третій програмі перевірити, чи є введений користувачем символ алфавітом чи ні.

У четвертій програмі згенерувати таблицю множення числа, введеного користувачем.

У п’ятій програмі обчислити найменше спільне кратне двох чисел, введених користувачем.

У шостій програмі обчислити ступінь числа.

У сьомій програмі надрукувати всі прості числа між двома числами, введеними користувачем.

У восьмій програмі знайти всі доданки цілого числа, введеного користувачем.

Варіант 2.

У першій програмі перевірити чи є введений символ голосним або приголосним.

У другій програмі перевірити, чи введений користувачем рік є високосним чи ні.

У третій програмі обчислити суму натуральних чисел, введених користувачем.

У четвертій програмі відобразити послідовність Фібоначчі з перших n чисел (введених користувачем).

У п’ятій програмі підрахувати кількість цифр у цілому числі, яке введене користувачем.

У шостій програмі перевірити чи є число, введене користувачем, паліндромом чи ні.

У сьомій програмі перевірити чи є введене користувачем ціле число числом Армстронга чи ні.

У восьмій програмі роздрукувати напівпіраміди, перевернуті піраміди, повні піраміди, перевернуті повні піраміди, трикутники Паскаля та трикутники Флойда.

Варіант3.

У першій програмі знайти найбільше число серед трьох чисел, введених користувачем.

У другій програмі перевірити чи є число, введене користувачем, від’ємним або позитивним.

У третій програмі обчислити факторіал числа, введеного користувачем.

У четвертій програмі запропонувати два різних способи обчислення найбільшого спільного дільника двох цілих чисел, введених користувачем.

У п’ятій програмі інвертувати число, введене користувачем.

У шостій програмі обчислити чи є ціле число, введене користувачем, простим числом чи ні.

У сьомій програмі знайти всі числа Армстронга між двома цілими числами, введеними користувачем.

У восьмій програмі обчислити результат дії над двома операндами залежно від оператора, введеного користувачем.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.

3.3 Демонстрація роботи програми та усунення зауважень.

4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Які операції відносяться до операцій відношення?
2. Чим відрізняються операції "= =" від операції "="?
3. Які операції відносяться до логічних? Який пріоритет їх виконання?
4. Якою операцією можна замінити операцію "&&" ?
5. Якою операцією можна замінити операцію "||" ?
6. Чому може дорівнювати значення виразу відношення або логічного виразу?
7. Як правильно порівняти на рівність дійсні числа?
8. Як правильно перевірити входження значення у певний діапазон?
9. Як перевірити певне цілочисельне значення на рівність нулю?
10. Як перевірити відмінність цілочисельного значення від нуля?
11. Коли використовується вложення умовних операторів?
12. Як правильно записати вложені умовні оператори?
13. Що являє собою оператор switch? Як їм користуватися?
14. Як записати оператор switch з допомогою умовних операторів?
15. Які види циклів снують? Які оператори циклу існують у мові С?
16. Чим відрізняються цикл з передумовою від циклу з постумовою?
17. Коли необхідно використовувати цикл з передумовою, а коли з постумовою? Наведіть приклади.
18. Які цикли з передумовою існують у мові С?
19. Скільки операторів містять у собі тіло циклу у мові С?
20. Як правильно записати цикл з постумовою на мові С?
21. Як задати нескінчений цикл? Навіщо він потрібен? Як з нього вийти?
22. Яким повинно бути значення виразу, який визначає умову виконання циклу, для завершення циклу?
23. Яким повинно бути значення виразу, який визначає умову виконання циклу, для виконання тіла циклу?
24. До чого призведе неправильне завдання виразу, який визначає умови виконання циклу?
25. Чи може бути відсутнім тіло циклу? Якщо може, то наведіть приклади таких циклів.

26. Чим відрізняється оператор *while* від оператора *if*?
27. Який порядок дій при виконанні циклу *for*?
28. Як організувати арифметичний цикл з допомогою циклу *for*?
29. Запишіть алгоритм, який визначається циклом *for*, з допомогою циклу *while*.
30. Що таке вкладений цикл?
31. Скільки разів у загальній складності виконується тіло вкладеного циклу?
32. Як і коли використовуються оператори *break* то *continue*?
33. Що таке рекурентні обчислення? Коли вони використовуються? Як їх програмувати?

РОЗДІЛ VIII. ПОКАЖЧИКИ

8.1 Теоретичні відомості. Покажчики

8.1.1 Покажчики

Для всіх відзначених раніше типів даних ім'я - це символічне позначення місця в пам'яті. Як відомо, після компіляції вихідної програми ці імена заміщаються відповідними адресами. Під час виконання програми використовуються значення, які зберігаються за певними адресами. Наприклад, в операторі присвоєння var1=var2; значення змінної var2 копіюється в змінну var1.

Поруч із такими змінними в мові С передбачаються й такі, які зберігають адреси змінних або покажчики.

Покажчик - це змінна, що містить адресу іншої змінної. Відомо, що покажчики застосовуються й у мові Паскаль:

```
TYPE  POINT=^REAL;
```

Припустимо, що x - змінна, наприклад, типу INT, а px - покажчик, створений якимось ще не зазначеним способом.

Унарна операція & передає адресу об'єкта, так що оператор px=&x. присвоює адресу x змінній px; говорять, що px "вказує" на x. Операція & застосовується тільки до змінних і елементів масиву. Конструкції виду (&x-1) і &3 є некоректними. Не можна також одержати адресу реєстрової змінної.

```
int* p, y;     // int* p; int y; НЕ int* y;
int x, *p;     // int x; int* p;
int v[10], *p;// int v[10]; int* p;
```

8.1.2 Операції над покажчиками

Якщо визначені змінні ***x, i***

```
int x, i;
```

то поруч зі звичайними діями можна оперувати й з їхніми адресами:

```
px=&x; //& взяти адресу
```

за допомогою операції **&** (взяти адресу). Відзначимо, що праворуч від **&** може стояти тільки ім'я змінної або змінної з індексом, а не вираз загального вигляду.

Унарна операція * розглядає свій операнд як адресу кінцевої мети й звертається по цій адресі, щоб витягти вміст.

Змінні, які беруть участь у процесі адресації описують відповідним чином:

```
int x=10, y,*px; // x=10
```

```
px = &x;        // px – адреса змінної x
y = *px;        // y=x
```

Відзначимо, що в описі покажчика обов'язково визначається тип об'єкта, з яким визначений покажчик. Цей опис використовується при обчисленні виразів із покажчиками.

```
char c1 = 'a';
char* p = &c1; // в p зберігається адреса c1
char c2 = *p;    // c2 = 'a'
```

Покажчики можна використовувати в операторах присвоєння px=py, якщо px і py відповідають однаковим типам даних.

Єдиною константою, яку можна присвоїти покажчику, є нуль, або NULL, що знаходиться у файлі stdio.h : py=NULL.

До покажчиків можна додавати або віднімати ціле число:

py=px+2; pz=py-4 - виходить, що покажчик px необхідно збільшити на дві одиниці даних. Ясно, що коли px і py є покажчиками символів, те це буде два байти, або просто два. Коли ж px і py є покажчиками дійсних чисел з подвійною точністю double *px, *py; то це буде 16 байт або 16. тому, залежно від типу даних у виразах з покажчиками здійснюється відповідне масштабування.

Для покажчиків визначена й операція розрахунку різниці адрес: pz=px-py. Операція додавання покажчиків немає ніякого змісту. Оскільки визначені операції додавання й вирахування констант, те визначені й операції збільшення й зменшення px++; py--; px+=i.

Крім того **==, !, =, >, <.**

В арифметичних виразах замість імені змінної можна застосовувати операцію доступу за покажчиком. Коли:

```
int x,y,*px;
if (px==&x) y=*px+1; //y=x+1
else printf (“%d \n”, *px);.
```

Операції доступу за покажчиком можна вживати й у лівій частині оператора присвоєння:

```
px=0;
x=0;
```

або

```
*px+=1;
x=x+1;
```

Унарні операції * і **&** виконуються раніше арифметичних. Тому:

```
y=*px+1; y=x+1;
y=*(px+1); // привласнює змінній y значення комірки з покажчиком px+1
```

Між собою унарні операції рівноправні й виконуються праворуч ліворуч

```
*px++ // y значення комірки з покажчиком px+1
```

і

```
(*px)++  //x+1
```

також ++*px.

Для покажчиків можна застосовувати операцію взяття адреси, де перебуває сам покажчик &px.

От, наприклад, функція, що підраховує число символів у рядку (не враховуючи завершального 0):

```
int strlen(char* p)
{
int i = 0;
while (*p++) i++;
 return i;
}
```

Інший спосіб знайти довжину полягає в тому, щоб спочатку знайти кінець рядка, а потім відняти адресу початку рядка від адреси її кінця:

```
int strlen(char* p)
{  char* q = p;
while (*q++) ;
return q-p-1;
}
```

Наприклад, використання постфіксної операції збільшення робить наступні цикли **while** ідентичними:

```
while (*string) { cout << *string++;}
while (*string) { cout << *string;   string++;}
```

Крім того, визначені відносини: **==! = > <.**

В арифметичних виразах замість імені змінної можна застосовувати операцію доступу за покажчиком.

Коли:

```
int x,y,*px;
if (px==&x) y=*px+1; // y=x+1
else printf (“%d \n”, *px);
```

8.2 Лабораторна робота 11. Функції та рекурсія

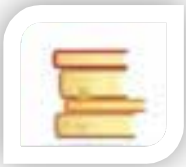
Тема: Функції та рекурсія

Мета: Закріпити навички написання програм мовою С за розділами: функції та рекурсія.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Програма мовою С складається з декількох функцій. Усі вони рівноправні, але обов'язково якась одна повинна мати ім'я main і компіляція починається саме із цієї функції. Для більшої ясності програму краще виконувати з невеликих функцій, а не з великих.

Імена функцій мають глобальний характер, тому вкладених функцій не передбачене. Послідовність обігу – будь-яка, може викликати функцію із цієї ж самої функції, тобто рекурсивні функції.

У загальному випадку програма містить кілька функцій. Тому необхідно визначити структуру функції - підпрограми.

Опис функції являє собою блок, тобто складається із заголовка й тіла. Заголовок має вигляд:

```
Тип_функції ім‘я_функції (тип змінна, тип змінна, ...)
{
Тіло функції;
return (вираз);
}
```

де ***тип змінна*** – список формальних параметрів.

На відміну від звертання до функції при описі функції наприкінці ";" не ставиться. З іменем функції можна зв'язувати одне або жодного значення. Наприклад, функція **puts** виводить тільки рядок і з її іменем не зв'язується ніяке значення.

Щоб ім'я функції одержало значення (повертало значення), у її тілі повинні бути присутній оператор (функція) **return (вираз)**. Дужки не обов'язкові.

Значення вираження привласнюється імені функції, тобто функція повертає значення змінній. Інше призначення цього оператора - передати керування відповідній до програми на оператор, який іде слідом за оператором виклику функції.

Функція може містити один, декілька операторів (функцій) **return** або жодного.

Функція уявляє собою певну закінчену процедуру і її можна розглядати як "чорну скриньку". Тобто програміста може не цікавити як вона реалізована, а тільки її призначення і вхідні параметри. Тому всі необхідні параметри краще передавати через список заголовку функції. Звичайно, при необхідності можна передавати параметри і через зовнішні змінні, але це гірше, бо відноситься до побічних ефектів і програє у неточності.

Виклик функції відбувається за ім'ям **absn (d)** і на відміну від інших мов, це звертання може входити до складу виразу чи бути окремим оператором (що залежить від призначення функції).

Викликаючи функцію, зазначають фактичні параметри.

Коли компілятор зустрічає виклик функції, то він робить копію значень фактичних параметрів. Ці копії передаються у функцію й привласнюються формальним параметрам. Відбуваються необхідні обчислення, по закінченню яких формальні параметри видаляються (стають невизначеними). Зрозуміло, що при цьому фактичні параметри не змінюються.

Отже, коли необхідно поміняти величину змінної у визваній функції або передати його із визваної функції, то необхідно передавати у функцію адресу параметра. Або передача параметрів за адресою здійснюється через покажчик. Це стосується й масивів.

У мові С++ передбачені посилання функції саму на себе, тобто рекурсивні функції.

Кожний виклик функції називається її активізацією. В даному випадку відбувається послідовність активізації, за якою слідує компілятор. Послідовність закінчується, коли **factor** поверне 1. Після цього перераховуються всі повертаємі значення до **factor (n-1)**, по якому і

обчислюється n!.

Окрім рекурсивних функцій дозволяється використовувати звертання до функцій як фактичні параметри.

Наприклад

printf (“ %d \n”, scanf (“ %f %e %s”, &a, &b, str));

Тут параметром є звертання до функції **scanf**. Тому вводимо три значення – два дійсних числа та рядок. Окрім вводу, функція **scanf** повертає ціле число, яке дорівнює кількості правильно виконаних вводів. Якщо вводимо три згаданих значення через пропуск, то функція **scanf** поверне число 3, яке і буде надруковано.

Практична частина

Завдання

Написати 6 програм згідно номеру індивідуального варіанту.

Варіант 1.

У першій програмі знайти прості числа між двома цілими числами, введеними користувачем.

У другій програмі перевірити чи може введене користувачем число бути виражено як сума двох простих чисел

У третій програмі знайти факторіал позитивного цілого числа, введеного користувачем, з допомогою рекурсії.

У четвертій програмі взяти речення від користувача та перевернути його за допомогою рекурсії.

У п’ятій програмі перетворити двійкове число в десяткове і навпаки вручну, створюючи користувальницьку функцію.

У шостій програмі перетворити двійкове число в вісімкове і навпаки вручну, створюючи користувальницьку функцію.

Варіант 2.

У першій програмі перевірити просте число або число Армстронга за допомогою функції, визначеної користувачем.

У другій програмі знайти суму натуральних чисел з допомогою рекурсії.

У третій програмі знайти найбільший спільний дільник двох позитивних цілих чисел, введених користувачем, з допомогою рекурсії.

У четвертій програмі обчислити ступінь числа з допомогою рекурсії.

У п’ятій програмі перетворити вісімкове число в десяткове і навпаки вручну, створюючи користувальницьку функцію.

У шостій програмі перетворити двійкове число в шістнадцяткове і навпаки вручну, створюючи користувальницьку функцію.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке функція?
2. Що таке функція типу void?
3. Що таке прототип функції?
4. Чим відрізняється прототип функції від виклику функції?
5. Чим відрізняється описання функції від визначення функції?
6. Критерії відповідності формальних і фактичних параметрів.
7. Відповідність типів формальних і фактичних параметрів.
8. Навіщо потрібен оператор return?
9. Чи може у функції бути декілька операторів return?
10. Коли необхідно писати оператор return у функції типу void?
11. Що таке побочний ефект функції?
12. Коли використовують формальні параметри-вказівники? Якими у цьому випадку повинні бути фактичні параметри?
13. Як передати масив у функцію? Як передати у функцію матрицю?
14. Як описати функцію, яка дозволяє працювати одночасно з одновимірними масивами, і з матрицями?
15. Що таке параметр-константа? Коли використовують такі параметри
16. Що таке вказівник на функцію? Як його описати?
17. Як передати функцію у функцію? Що буде формальним параметром, а що

фактичним? Як звернутися до функції, яку передано через параметри?

18. Що таке рекурсія? Коли вона використовується? Що таке глибина рекурсії?

19. Як описати функцію зі змінним числом параметрів? Як використовувати таку функцію?

20. Які параметри можуть бути у функції main()? Як налагоджувати і тестувати програму, яка містить функцію main() з параметрами?

РОЗДІЛ IX. МАСИВИ. ПОНЯТТЯ. ПРИКЛАДИ. ОСОБЛИВОСТІ ПРОГРАМУВАННЯ

9.1 Теоретичні відомості. Масиви. Поняття. Приклади. Особливості програмування

9.1.1 Масиви. Поняття. Приклади

Звичайно покажчики рідко вживаються для простих даних. Частіше вони зв'язуються зі складними типами, зокрема, з масивами.

У мовах високого рівня, наприклад Pascal, операції з елементами масивів реалізуються індексною арифметикою. У машинно-орієнтованих мовах ці операції реалізуються непрямою адресацією, яка є швидшою. Ця можливість існує й у мові С і полягає у використанні покажчиків. Тому покажчики й масиви тісно повзв'язані.

Приклад: посимвольно надрукувати рядок " Тестування !" ***у прямому порядку***

```
void main(void)
{ char *str=" Тестування !";
  while(*str) putchar(*str++);
}
```

Їхнє використання ефективніше, ніж з індексною арифметикою. Але зрозуміти такі програми важче.

Можна зробити деякі порівняння. Повне ім'я масиву є адресою базового елемента, покажчик також набуває значення адреси. Тому іноді говорять, що повне ім'я масиву є покажчиком. Але це не можна розуміти буквально.

У чому ж полягає різниця між ними?

Якщо задати рядок масивом char st[]="Тестування !", під час трансляції під масив буде виділена постійна ділянка пам'яті

st: “ Тестування!” ‘0’,

де повністю розміщений рядок разом із заключним нулем.

Ім'я масиву дорівнює базовій адресі цієї ділянки. Тобто на весь час виконання програми за ділянкою пам'яті закріплюється ім'я масиву. Тому воно є константою і його не можна змінювати під час виконання програми за допомогою оператора присвоювання.

Наприклад st="spring"; або st=pr,

де pr деякий покажчик на символ.

Такі спроби розцінюються як спроба змінити адресу, тобто константу, і кваліфікуються як помилка при виконанні. От чому в мові С не можна вживати

операції присвоювання для масивів.

Зміст масивів можна змінювати, наприклад, за допомогою функції копіювання або введення.

Якщо задати рядок покажчиком

char *pr=”Тест!”;

виходить, що під час трансляції рядковій константі десь виділена пам'ять і початкова адреса цього рядка записана у покажчику

pr

pr “Тест!”

Покажчик pr не є константою, а змінної. Тому його значення в процесі виконання можна змінювати *pr="spring";

Цей оператор не означає, що замість рядка "тест!" на його місце записується "spring". Просто в пам'яті виділяється нове місце під рядок "spring" і його базова адреса записується в покажчику pr

“Good st!”

Pr

“Spring”

Спроба змінити саму константу "good st!" *pr=:Spring; розглядається як помилка. Потрібно відзначити, що при роботі з масивами за допомогою покажчиків розмір масивів не контролюється й такі можливості не фіксуються виконуючою системою.

Якщо покажчик не буде ініціалізуватися, то його значення невизначене (кожне). Тому можливі помилки при використанні покажчиків.

Наприклад: main ()

{ int *a;

*a=25;

printf (“*a=%d\n”, *a);

}

У таких випадках необхідно спочатку виділити певна ділянка динамічної пам'яті (під час виконання). Це можна зробити за допомогою функції malloc:

(тип *) malloc (кіл-ть байт);

char *ptr;

ptr=(char*) malloc (100); // виділяється 100 байт, початкова адреса записується в ptr

Функція **malloc** повертає **char**- покажчик на закріплений простір.

У пам'яті, на яку вказує значення, що вертається, гарантоване вирівнювання для зберігання будь-якого типу об'єкта. Щоб одержати покажчик на тип, відмінний від char, використовується перетворювач типу значення, що вертається.

Вертається значення NULL, якщо вільної пам'яті залишилося мало.

```
#include <malloc.h>
int *intarray;
// виділяється місце для 20 цілих значень
intarray=(int*)malloc(20*sizeof(int));
```

Для звільнення динамічної пам'яті використовується функція:

free (pr),

де **pr** - ім'я покажчика.

Тому попередній фрагмент правильно треба було б написати так:

```
main ()
{ int *a;
a=(int *) malloc (sizeof(int));
*a=25;
printf (“*a=%d\n” *a);
free(a);
}
```

Ще одна функція виділення певної ділянка динамічної пам'яті (під час виконання)

(тип *) calloc (кількість елементів n, розмір елемента в байтах size);

Функція calloc виділяє простір для зберігання масиву з n елементів, кожний довжиною size байт. Кожний елемент ініціалізується в 0. Повертає значення NULL, якщо залишилося недостатньо пам'яті.

free(ім'я покажчика) - звільняє захоплене місце.

Ще одна функція виділення певної ділянка динамічної пам'яті (під час виконання) C .

ім'я покажчика = new тип[кіл-ть];- виділяє простір для зберігання масиву з [кількість] елементів

delete ім'я покажчика - звільняє виділене місце.

Функція ***realloc*** змінює розмір раніше захопленого блоку пам'яті. Вміст блоку не змінюється.

(тип *) realloc (новий розмір у байтах);

Функція **realloc** повертає покажчик на перезахоплений блок пам'яті.

```
#include <malloc.h>
#include <stdio.h>
char *t1;
t1=(char*) malloc(50*sizeof(char));
/ / Перевиділяє блок, який містить 100 символів
if (t1 != NULL)
t1=realloc(t1,100*sizeof(char));
```

9.1.2 Покажчики і багатовимірні масиви

Існує двовимірний масив:

```
int matr[3][2];
int *pr;
```

Ясно, що ім'я matr дорівнює адресі базового елементу matr==&matr[0][0]; тому після присвоєння pr=matr

```
pr+1==matr[0][1]
pr+2==matr[1][0]
pr+3==matr[1][1]
і т.д.
```

Отже за допомогою такого покажчика й відповідного зсуву можна адресувати будь-який елемент такого масиву.

Як відомо, елементи двовимірного масиву розташовуються в пам'яті рядками:

matr[0]	matr[0][0]	matr[0][1]
matr[1]	matr[1][0]	matr[1][1]
matr[2]	matr[2][0]	matr[2][1]

Двовимірний масив - це є масив, який складається з масивів, і повне ім'я масиву є адресою базового елементу. Тому перший рядок буде мати ім'я matr[0], другий matr[1], третій matr[2]. Ці імена мають значення адрес перших елементів відповідного рядка.

```
matr[0]=&matr[0][0]=matr
matr[1]=&matr[1][0] і т.д.
```

Із цього випливає, що поруч із повним іменем масиву matr, іменами окремих елементів matr[i][j] можна вживати й імена окремих рядків matr[i]. (В інших мовах, наприклад, Паскалі коли масив двовимірний, то вживання імені з одним індексом є помилкою).

Отже, відкривається можливість працювати із двовимірним масивом через одновимірний. Із цього випливає також, що двовимірний масив може бути заданий як одновимірний масив покажчиків.

При цьому кількість елементів у рядку не фіксується. Це зручно при роботі з масивами рядків. Наприклад, написати функцію, яка за номером місяця повертає його назву:

```
{char * name [] = {"Немає такого місяця", "Січень", "Лютий", "Березень",
"Квітень", "Травень", "Червень", "Липень", "Серпень", "Вересень", "Жовтень",
"Листопад", "Грудень"};
return (n <1 | | n> 12? name [0]: name [n]);
```

Тут кількість елементів у масиві покажчиків name визначається кількістю рядків у фігурних дужках.

Для того, щоб скопіювати рядок, зовсім не потрібно його фізично копіювати, а досить лише прирівняти відповідні покажчики. Також при сортуванні можна переставляти не рядки, а їх покажчики.

Розглянемо програму введення, сортування й виведення рядків. Використовуємо два масиви: рядків char **str і масив покажчиків char *tstr, який зв'яжемо з масивом str, де nums -максимальна кількість рядків, а numc - розмір рядків.

Сортування будемо проводити перестановкою покажчиків у масиві str. Сам же масив str буде залишатися незмінним.

Для порівняння рядків будемо використовувати функцію **strcmp (str1, str2)**, аргументами якої є імена рядків або відповідні покажчики.

Для сортування використаємо метод бульбашки із змінним зсувом, який розглядали раніше.

```
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream.h>
char **str,*tstr;
void main(void)
```

```
{ int a,b,nums,numc;
clrscr();
printf(" Введіть кількість рядків = "); scanf(" %d", &nums);
printf(" Введіть кількість символів = "); scanf(" %d", &numc); // виділення за допомогою new
/*  str= new char*[nums];
for(a=0;a<nums;a++) str[a]= new char[numc+1]; */
// виділення за допомогою calloc
str=(char**)calloc(nums,sizeof(char*)); виділення пам'яті для масиву рядків
for(a=0;a<nums;a++) str[a]=(char*)calloc(numc+1,sizeof(char));// для самих рядків
randomize();
for(a=0;a<nums;a++)
{
b=0;
do
{int ch=random(257);
if ((ch>64 && ch<91) || (ch>96 && ch<123))
{ str[a][b]= ch;
b++; }
}while (b<numc);
str[a][b]= '\0';
}
printf("\nСгенерований масив з [ %d ] рядків”, nums);
for (a=0;a<nums;++a)   printf("\n[%d] рядок -> %s", a+1, str[a]);
//сортування за зростанням
for(a=1;a<nums;++a)
for(b=nums-1;b>=a;--b)
{if(strcmp(str[b-1],str[b])>0)
{ tstr=str[b-1];
str[b-1]=str[b];
str[b]=tstr;
}
}
//   getch();
printf("\n\nВідсортований масив з [%d] рядків ", nums);
for (a=0;a<nums;a++) printf("\n [%d] рядок -> %s", a+1, str[a]);
// звільнення пам‘яті
for (a=0;a<nums;a++) free(str[a]);
```

```
free(str)
getch();
}
```

9.2 Лабораторна робота 12. Вказівники і одновимірні масиви даних

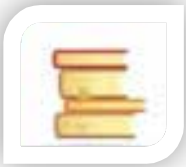
Тема: Вказівники і одновимірні масиви даних

Мета: отримання практичних навичок вирішення задач обробки масивів з використанням вказівників.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Вказівник. Змінна у програмі – об'єкт, що має ім'я та значення. На ім'я можна звернутися до змінної та отримати її значення. **Ім'я змінної** відповідає адресі ділянки пам'яті, виділеної для змінної, і не пов'язане з адресою. **Значення змінної** відповідає вмісту ділянки пам'яті. Щоб отримати адресу в явному вигляді, використовується унарна операція взяття адреси (&), яка застосовується лише до об'єктів, що мають ім'я та розміщені в пам'яті.

Адреси мають цілочисельні беззнакові значення, тому їх можна обробляти як цілочисельні величини, для цього використовуються змінні типу покажчик, що забезпечують безпосередній взаємозв'язок даних та можливість зміни цих зв'язків. Вказівник – змінна, що містить інформацію про розташування пам'яті іншої змінної, тобто. адреса об'єкта конкретного типу (адреса іншої змінної). Значенням покажчика може бути також нульова адреса, для позначення якого в ряді файлів заголовка (stdio.h) визначена спеціальна константа null.

Структура оголошення вказівника:

*<тип_вказуємих_данних> *<ім'я_вказівника>;*

Символ *«*»* – знак унарної ***операції косвенної адресації (розіменування)***, є операцією розкриття посилання (звернення за адресою), результатом якої є об'єкт, адресований покажчиком.

Наприклад,

*char *z; // вказівник на об'єкт символьного типу*

*int *k; // вказівник на об'єкт цілого типу*

*float *f; // вказівник на об'єкт дійсного типу*

Змінні *z, k, f* є вказівниками. **z* означає об'єкт типу *char*, на який вказує *z*. Позначення **z*, **k*, **f* мають права змінних відповідних типів. Оператор **z=' '* посилає символ *пробіл* у ту ділянку пам'яті, адресу якої визначає вказівник *z*.

Вказівник може посилатися на об'єкти того типу, який присутній в оголошенні вказівника. Виключенням є вказівники, в оголошенні яких використано тип *void*, тобто відсутнє значення. Такі вказівники можуть посилатися на об'єкти будь-якого типу, однак до них неможливо застосовувати операцію розіменовування.

Наприклад,

1. оголошення змінних:

int a,x; // ціла змінна

*int *p; // змінна – вказівник на іншу цілу змінну (оголошення змінної, при косвеному оголошенні до якої буде значення змінної цілого типу)*

2. операції присвоювання оголошеним змінним:

a=2000;

p=&a; // вказівник містить адресу змінної a (присвоювання змінной p адреси змінної a – призначення вказівника p на змінну a)

3. використання косвеного звертання:

*x=x+*p; // при косвеному звертанні на вказівник p береться значення змінної a, що еквівалентно: x=x+a*

Масив як статична структура даних. ***Статична (фундаментальна) структура даних*** – сукупність фіксованої кількості даних сталої розмірності з незмінним характером зв'язків між ними.

Змінні статичних структур можуть змінювати лише свої значення, а їх структура і множина допустимих значень залишаються незмінними; розмір пам'яті, яка займається такими змінними, залишається постійною.

До статичних структур відносять ***масив даних***, який являє собою впорядковану послідовність даних одного і того ж типів, які мають загальне ім'я; доступ до елементів масиву здійснюється за допомогою індексів, які є порядковими номерами елементів у послідовності. На етапе компіляції під елементи масиву виділяється фіксований об'єм пам'яті, який не змінюється в процесі виконання

програми. У пам'яті елементи масиву розміщуються послідовно у відповідності з ростом індексу.

Одномірний масив даних. Якщо для отримання доступу к елементів масиву потрібен один індекс, то ***масив*** є ***одномірним (лінійним)***. Математичним представленням одномірного масиву є вектор.

Структура оголошення одномірного масиву:

<тип_елементу_масиву> <ім'я_масиву>[<кількість_елементів];

Наприклад,

```
char v[10]; // оголошення одномірного масиву з 10-и символів
```

Наприклад, копіювання 10-и елементів одного масиву в інший можна організувати наступним чином:

```
int v1[10]={1,2,3,4,5,6,7,8,9,10},v2[10]; // оголошення змінних
// копіювання елементів одного масиву в інший
for (int i=0;i<10;i++) // для і, починаючи з 0 до 9,
  v2[i]=v1[i]; // копіювати і-ий елемент масиву v1 в  і-ий елемент масиву v2,
```

після чого і збільшується на 1 (інкремент змінної цілого типу)

При звертанні до елементу масиву відбувається звертання до його адреси, тому попередній фрагмент програми можна представити наступним чином, використовуючи вказівники:

```
int v1[10]={1,2,3,4,5,6,7,8,9,10},v2[10],*p1,*p2;  // оголошення змінних
 p1=&v1[0]; // у вказівник p1 заноситься адрес першого елементу масиву v1
p2=&v2[0]; // в p2 – адрес першого елементу масиву v2
// копіювання елементів одного масиву в інший
for (int i=0;i<10;i++)
{
   *p2=*p1;
   p2++; p1++;
}
```

Адресна арифметика. ***Операція адресної арифметики*** інтерпретується наступним чином:

- вказівник потенційно посилається на необмежену у обидві сторони області пам'яті, які заповнені змінними того типу, на який посилається вказівник;
- змінні в області пам'яті нумеруються від поточної змінної, на яку вказує вказівник і яка отримує відносний номер *0*; змінні у напряму зростання адрес пам'яті нумеруються позитивними значеннями (*1,2...*), у напрямі зменшення – від'ємними (*-1,-2…*);

- результатом операції *вказівник+i* є адрес *i*-й змінної у цій області пам'яті відносно поточного положення вказівника, тобто значення вказівника на *i*-у змінну.

Таким чином, в***казівник посилається на необмежений масив*** з відносною нумерацією елементів масиву від змінної, на яку показує вказівник.

p+i; // встановити вказівник на і-ю змінну після тієї, на яку вказує p

p-i; // встановити вказівник на і-ю змінну перед тією, на яку вказує p

**(p+i); // отримати значення і-ї змінної після тієї, на яку вказує p, що еквівалентно: p[i]*

p++; // отримати значення змінної, на яку вказує p, потім вказівник встановити на наступну змінну

p--; // отримати значення змінної, на яку вказує p, потім вказівник встановити на попередню змінну

p+=i; // перемістити вказівник на і змінних вперед відносно тієї, на яку вказує p

p-=i; // перемістити вказівник на і змінних назад відносно тієї, на яку вказує p

**p++; // отримати значення змінної, на яку вказує p, потім вказівник встановити на наступну змінну*

**(--p); // перемістити вказівник до змінної, яка передує тій, на яку вказує p, потім отримати її значення*

*(*p)++; // отримати значення змінної, на яку вказує p, потім збільшити її значення на 1*

*++(*p); // получити значення змінної, на яку вказує p, збільшити на 1*

Лінійний пошук і сортування в масивах даних. В задачах на ***лінійний пошук*** треба знайти в масиві даних елемент, групу елементів або фрагмент масиву, який відповідає заданим вимогам, при цьому нема необхідності сортувати масив або виділити додаткові робочі масиви того ж порядку, що і задані.

Сортування інформаційної структури (масиву, списку, файлу) називається перетворення вихідної структури шляхом перестановки її елементів для досягнення впорядкованості за заданою ознакою порядку.

Сортування за ознакою порядку

Одновимірний масив *a* з *n* елементів для будь-якого $i=2,3,\ldots,n$ називається впорядкованим:

- ***за зростанням***, якщо $a_i > a_{i-1}$;
- ***за незменшенням***, якщо $a_i \geq a_{i-1}$;
- ***за зменшенням***, якщо $a_i < a_{i-1}$;

- ***за незростанням***, якщо $a_i \leq a_{i-1}$.

Способи сортування у масивах даних.

Для реалізації алгоритму будь-якого способу сортування потрібно проходження не менше двох вложених циклів.

Бульбашкове сортування (попарні перестановки) виконується багаторазовими перестановками кожних двох сусідніх елементів, які порушують порядок; процес завершується, коли досягається впорядкованість масиву.

Алгоритм бульбашкового сортування можна реалізувати з допомогою двох циклів з параметром (зовнішнього і вложеного). Найбільший/найменший (в залежності від ознаки порядку) елемент масиву в останньому проході вложеного циклу «вспливає, як бульбашка», потрапляючи на своє місце на початок/кінець (або навпаки) масиву, після чого початкове/кінцеве значення параметру циклу змінюється на одиницю (збільшується/ зменшується), тим самим з кожним проходом зовнішнього циклу зменшується кількість проходів внутрішнього циклу.

Блок-схема алгоритму перестановок елементів масиву (сортування за зростанням) (рис.9.1)*:*

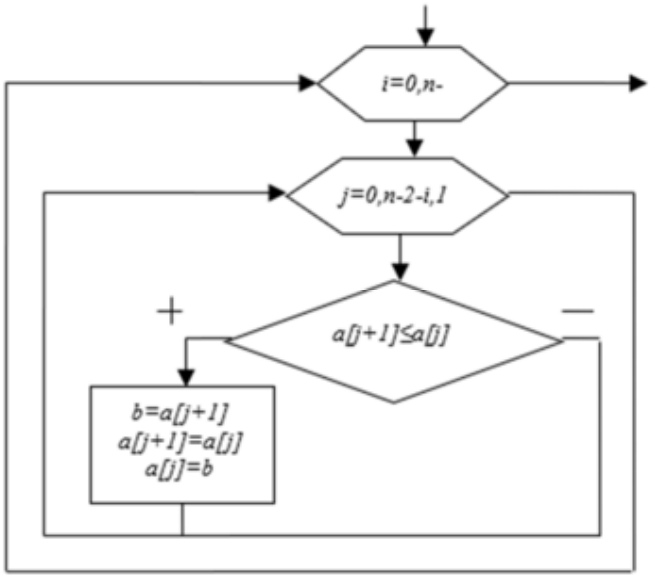


Рис. 9.1 - Блок-схема алгоритму перестановок

Також алгоритм бульбашкового сортування можна реалізувати, використовуючи в якості зовнішнього циклу цикл з постумовою (рис. 9.2) (змінна *f* використовується в якості ознаки перестановок):

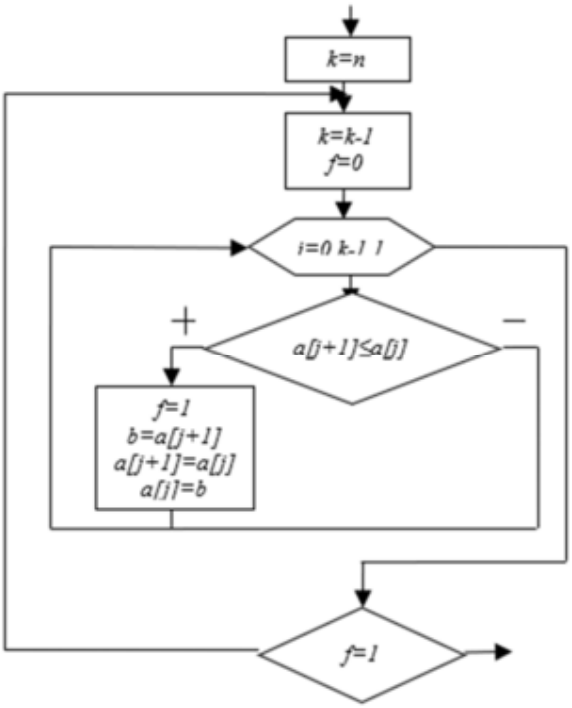


Рис. 9.2 - Блок-схема алгоритму з постумовою

Сортування простого вибору (мінімуму/максимуму) виконується перестановками кожного елементу з мінімальним/максимальним (в залежності від ознаки порядку) серед наступних за ним елементів.

Блок-схема алгоритму перестановок елементів масиву (сортування за зростанням) (рис.9.3)*:*

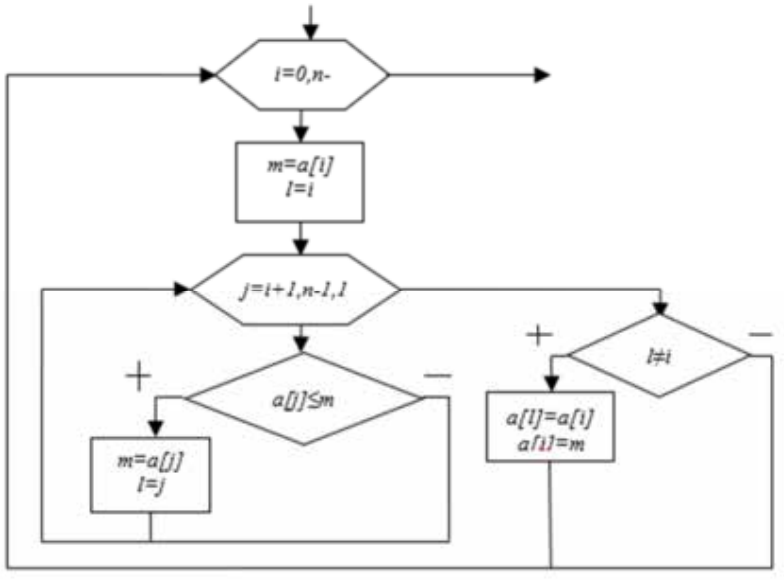


Рис. 9.3 - Блок-схема алгоритму сортування

Практична частина

Завдання

Задача 1.

Постановка задачі: розробити алгоритм створення масиву даних з позитивних елементів введеного масиву, який складається з *n* елементів, з наступним виводом створеного масиву. Написати програму, яка реалізує розроблений алгоритм.

Варіант 1 *(використовується звертання до елементів масиву за іменем)*

Математична модель та опис алгоритму: для введеного (вхідного) масиву даних: *1 -5 0 -9 5 7 0 -2 4 0*, вихідний масив: *1 5 7 4*.

- Оголосити вхідний масив, розмірність якого задана константою *c*: *int mas[c]*;
- вихідний масив може складатись з такої ж кількості елементів, як вхідний (якщо всі елементи позитивні), тому оголосити: *int mas_new[c]*;
- в циклі з постумовою $n \le 0$ або $n > c$ ввести кількість елементів масиву $0 < n \le c$;
- в циклі з параметром $0 \le i < n$ ввести елементи вхідного масиву *mas[i]*;
- змінна $j=0$ – лічильник кількості елементів вихідного масиву;
- у циклі з параметром $0 \le i < n$ для кожного елементу *mas[i]* перевірити умову: якщо $mas[i] > 0$, то $mas\_new[j] = mas[i]$ та $j = j+1$;
- якщо $j \ne 0$, то у вхідному масиві є позитивні елементи, значить, що вихідний масив створено, тоді у циклі з параметром $0 \le i < n$ вивести елементи *mas_new[i]*; інакше вивести «*нема елементів>0*».

Блок-схема алгоритму програми (рис.9.4)***:***

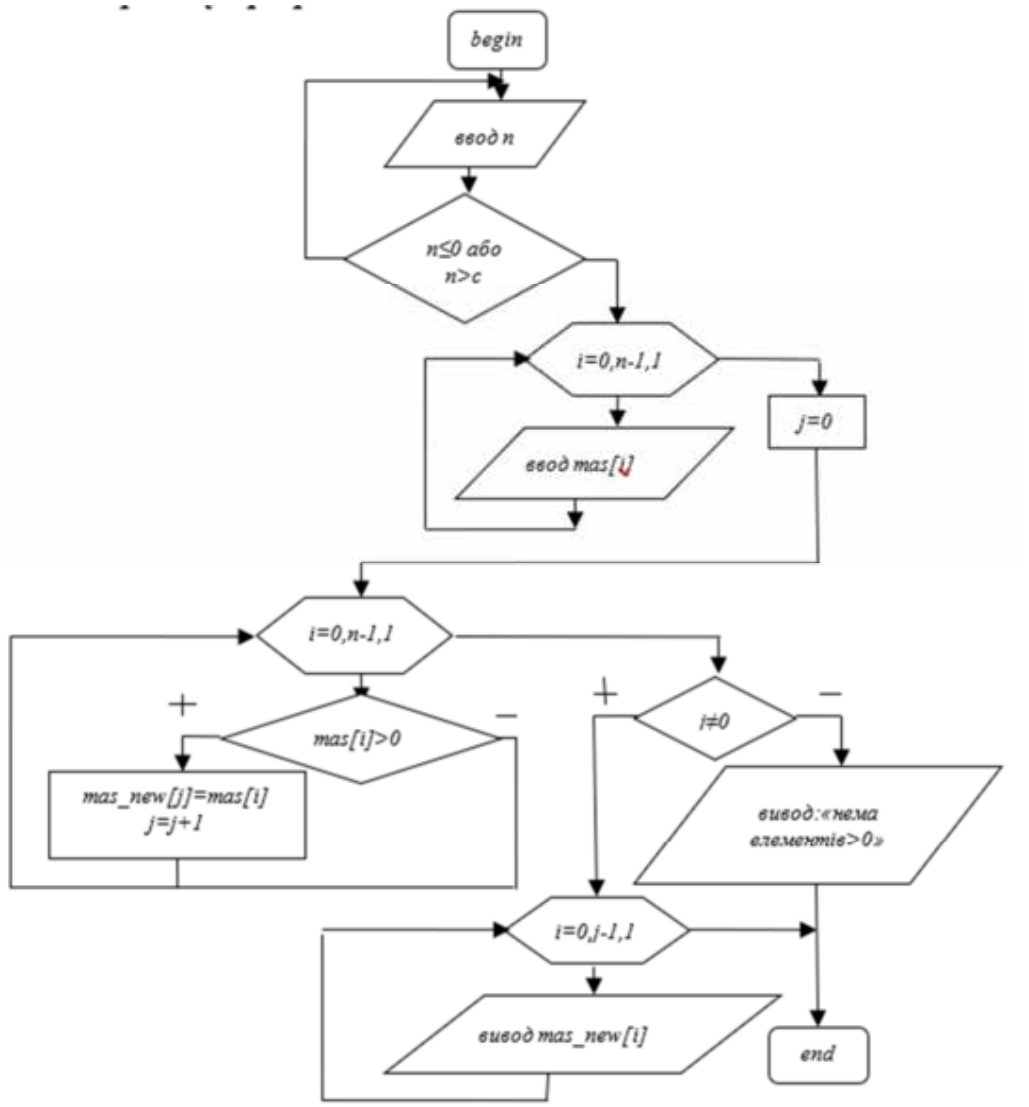


Рис. 9.4 - Блок-схема алгоритму програми

Текст програми

```
#include <stdio.h>
#include <conio.h>
int main()
{
   const c=15;
   int n,j=0,mas[c],mas_new[c];
   do
   {
      printf("\n input n=");
      scanf("%d",&n);  // ввод кількості елементів вхідного масиву
   }
   while (n<=0 || n>c);
   printf("\n input massiv:\n");
```

```
   for (int i=0;i<n;i++)
      scanf("%d",&mas[i]);  // ввод елементів вхідного масиву
   printf("\n output massiv:\n");
   for (i=0;i<n;i++)
      printf("%d ",mas[i]);  // вивод елементів вхідного масиву
   for (i=0;i<n;i++)
      if (mas[i]>0)
      {
         mas_new[j]=mas[i];  // створення вихідного масиву
         j++;
      }
   if (j!=0)
   {
      printf("\n output massiv:\n");
      for (i=0; i<j; i++)
      printf("%d ",mas_new[i]);  // вивод елементів вихідного масиву
   }
   else printf("\n there are no elements>0\n");
   getch();
   return 0;
}
```

Задача 2 *(використовується звернення до елементів масиву за адресою)*

Математична модель і опис алгоритму: вхідний масив елементів: *1 -5 0 -9 5 7 0 -2 4 0*, вихідний масив: *1 5 7 4*.

- Оголосити масиви, розмірність яких задано константою *c*: *int mas[c]*, *mas_new[c]*;
- у циклі з пост умовою $n \le 0$ або $n > c$ ввести кількість елементів масиву $0 < n \le c$;
- встановити вказівник на перший елемент масивів: *p=&mas[0]*, *p_new=&mas_new[0]*;
- у циклі з параметром $0 \le i < n$ ввести значення елементів вхідного масиву за адресою *p*, встановити після кожного проходу циклу вказівник на наступний елемент: *p= p+1*;
- *j=0* (лічильник елементів вихідного масиву), встановити вказівник знову на перший елемент вхідного масиву: *p=p-n*;
- у циклі з параметром $0 \le i < n$ перевірити умови: якщо **p>0*, то **p_new=*p*, *j=j+1* і *p_new=p_new+1*; *p=p+1*;
- якщо $j \ne 0$, то вихідний масив створено, тоді встановити вказівник на перший елемент вихідного масиву: *p_new=p_new-j*, у циклі з параметром $0 \le i < n$ вивести значення елементів вихідного масиву **p_new*; інакше вивести «*there are no elements>0*».

Блок-схема алгоритму програми (рис.9.5)

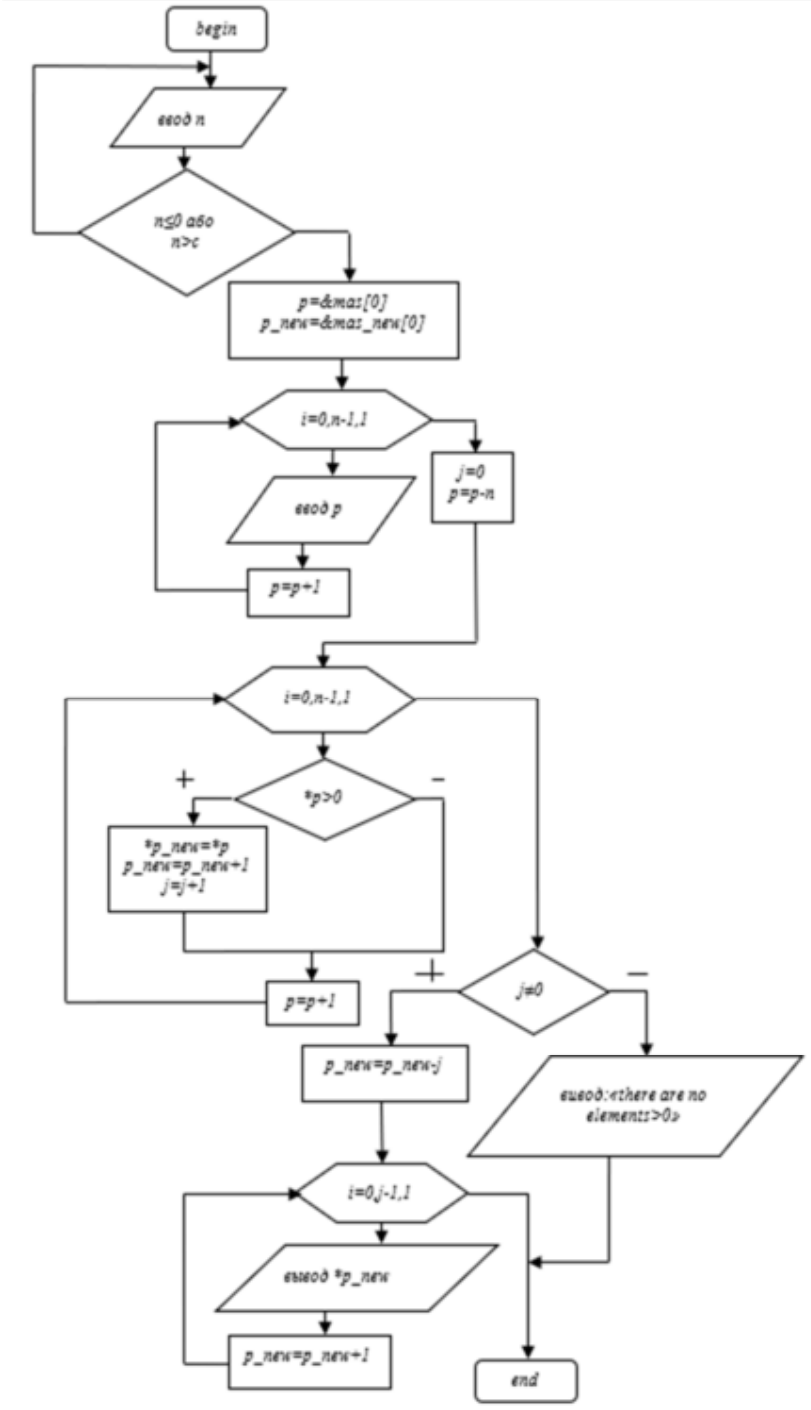


Рис. 9.5 - Блок-схема алгоритму програми

Текст програми

```
#include <stdio.h>
#include <conio.h>
int main()
{
   const c=15;
```

```
int n,j=0,mas[c],mas_new[c],*p,*p_new;
do
{
   printf("\n input n=");
   scanf("%d",&n);  // ввод кількості елементів в масиві
}
while (n<=0 || n>c);
p=&mas[0];
p_new=&mas_new[0];
printf("\n input massiv:\n");
for (int i=0;i<n;i++)
{
   scanf("%d",p);  // ввод елементів масиву
   p++;
}
p-=n;printf("\n output massiv:\n");
for (i=0;i<n;i++)
{
   printf("%d ",*p);  // вивод елементів масиву
   p++;
}
p-=n;
for (i=0;i<n;i++)
{
   if (*p>0)
   {
      *p_new=*p;  // створення нового масиву
      p_new++;
      j++;
   }
   p++;
}
if (j!=0)
{
   p_new-=j;
   printf("\n output massiv:\n");
   for (i=0;i<j;i++)
   {
```

```
        printf("%d ",*p_new);  // вивід елементів нового масиву
        p_new++;
      }
   }
   else printf("\n there are no elements>0\n");
   getch(); return 0;
}
```

Тестування

Теоретично розраховане вихідне значення	*Практично отримані вихідні значення*
Тест 1: *вхідний масив: n=10; mas[n]: 1 -5 0 -9 5 7 0 -2 4 0*	
j=4; mas_new[j]: 1 5 7 4	*j=4; mas_new[j]: 1 5 7 4*
Тест 2: *вхідний масив: n=10; mas[n]: -1 -5 0 -9 -5 -7 0 -2 -4 0*	
«нема елементів>0»	*«нема елементів>0»*

Задача 2.

Постановка задачі: розробити алгоритм знаходження в введеному масиві з *n* елементів останнього найменшого елементу, обміняти місцями його з третім елементом і вивід отриманого масиву. Написати програму, яка реалізує розроблений алгоритм.

Математична модель і опис алгоритму: вхідний масив *x*: *-5 8 0 4 -5 7 9 0 9 -1* (індекси елементів масиву: *0 1 2 3 4 5 6 7 8 9*); останній найменший елемент у масиві *x[4]=-5*; третій елемент у масиві *x[2]=0*; вихідний масив *x* після перестановки елементів: *-5 8 -5 4 0 7 9 0 9 -1* (*0 1 2 3 4 5 6 7 8 9*).

Оголосити масив, розмірність якого задати константою *c*: *int x[c]*;

- у циклі з постумовою *n≤0* або *n>c* ввести кількість елементів масиву *0<n≤c*;
- вказати *p* встановити на нульовий елемент: *p=&x[0]*;
- у циклі з параметром *0≤i<n* ввести значення елементів вхідного масиву за адресою *p*, *p=p+1*;
- вказівник *p* встановити знову на нульовий елемент: *p=p-n*; у циклі з параметром *0≤i<n* вивести значення елементів введеного масиву **p*, *p=p+1*;
- *p* зменшити на одиницю (*p=p-1*), тому що після виходу з циклу вивод значень елементів масиву *p=n*;
- змінній *min* присвоїти значення останнього елементу масиву, використовуючи операцію розіменування: *min=*p*,
- у циклі з параметром *i* від *n-2* та до *0* (*i=i-1*), починаючи з передостаннього елементу (*p=p-1*), порівняти *min* з поточним елементом: якщо **p<min*, то *min* та

індекс мінімального елементу *k* перевизначити: *min=*p*, *k=i*;

• щоб поміняти місцями третій елемент зі знайденим, необхідно перевизначити значення за адресами, використовуючи операцію розіменування: **(p+n-1-k) =*(p+2)*, **(p+2)=min*;

• у циклі з параметром *0≤i<n* вивести значення елементів вихідного масиву **p*, *p=p+1*.

Блок-схема алгоритму програми (рис.9.6)

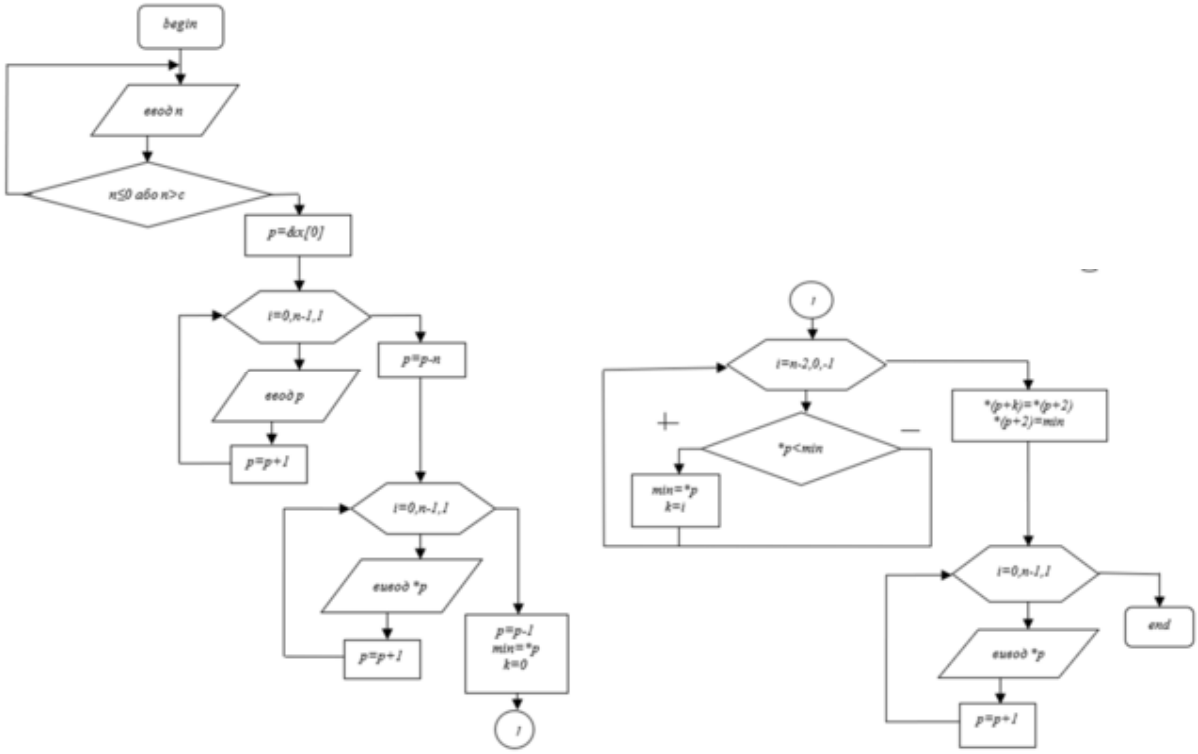


Рис. 9.6 - Блок-схема алгоритму програми

Текст програми

```
#include <stdio.h>
#include <conio.h>
void main()
{
   const c=20;
   int x[c],i,k=0,min,*p,n;
   do
   {
      printf("\nInput number of elements of massiv n=");
      scanf("%d",&n);
   }
   while (n<=0 || n>c);
   p=&x[0];
   printf("\nInput elements of massive:\n");
```

```
  for (i=0;i<n;i++)
  {
     printf("x[%d]=",i);
     scanf("%d",p);
     p++;
  }
  p-=n;
  printf("\nOutput massiv:\n");
  for (i=0;i<n;i++)
  {
     printf("%d ",*p);
     p++;
  }
  p--; min=*p;
  for (i=n-2;i>=0;i--)
  {
     p--;
     if (*p<min)
     {
        min=*p;
        k=i;
     }
  }
  *(p+k)=*(p+2);
  *(p+2)=min;
  printf("\nOutput massiv:\n");
  for (i=0;i<n;i++)
  {
     printf("%d ",*p);
     p++;
  }
  getch();
  return;
}
```

Тестування

Теоретично розраховані вихідні значення	*Практично отримані вихідні значення*
Тест 1: *вхідний масив: n=10; x[n]: -5 8 0 4 -5 7 9 0 9 -1*	
x[n]: -5 8 -5 4 0 7 9 0 9 -1	*x[n]: -5 8 -5 4 0 7 9 0 9 -1*
Тест 2: *вхідний масив: n=7; x[n]: 1 0 5 9 1 0 4*	
вих. масив x[n]: 1 0 0 9 1 5 4	*вих. масив x[n]: 1 0 0 9 1 5 4*

Задача 3.

1. Постановка задачі: розробити алгоритм впорядкування за не зменшенням значень елементів введеного масиву, який складається з *n* цілих чисел, використовуючи бульбашкове сортування, і вивод відсортованого масиву. Написати програму, яка реалізує розроблений алгоритм.

2. Математична модель і опис алгоритму: вхідний масив *a*: *9 0 -3 6 -3 2 8 9*; вхідний масив індексів *ai*: *0 1 2 3 4 5 6 7*; вихідний масив після перестановок (після сортування за не зменшенням, коли $a_{j+1} \geq a_j$): *-3 -3 0 2 6 8 9 9*; вихідний масив індексів: *2 4 1 5 3 6 0 7*.

У циклі з постумовою $n \leq 0$ або $n > c$ ввести кількість *n* елементів масиву;

- *p=&a[0]*, у вказівник *pi* занести адресу нульового елементу масиву індексів *pi=&ai[0]*;
- у циклі з параметром $0 \leq i < n$ ввести значення елементів за адресою *p*, *p=p+1* і сформувати масив індексів **pi=i*, *pi=pi+1*;
- *p=p-n*; у циклі з параметром $0 \leq i < n$ вивести значення елементів масиву **p*, *p=p+1*;
- для бульбашкового сортування масиву в зовнішньому циклі з параметром $0 \leq i < n-1$ встановити вказівник *p=p-n+i, pi=pi-n+i*; у вложеному циклі з параметром $0 \leq j < n-1-i$ переставити елементи, розташовані поруч, при порушенні ознаки порядку (незменшення – $a_{j+1} \geq a_j$): якщо **(p+1)<*p*, то поміняти елементи місцями, використовуючи додаткову змінну *b*, разом з елементами масиву переставляти елементи масиву індексів, щоб відслідкувати правильність сортування за заданою ознакою порядку, потім *p=p+1*, *pi=pi+1*;
- у циклі з параметром $0 \leq i < n$ вивести значення елементів відсортованого масиву **p*, *p=p+1*;
- у циклі з параметром $0 \leq i < n$ вивести значення елементів масиву індексів після відповідних перестановок **pi*, *pi=pi+1*.

Блок-схема алгоритму програми (рис.9.7)

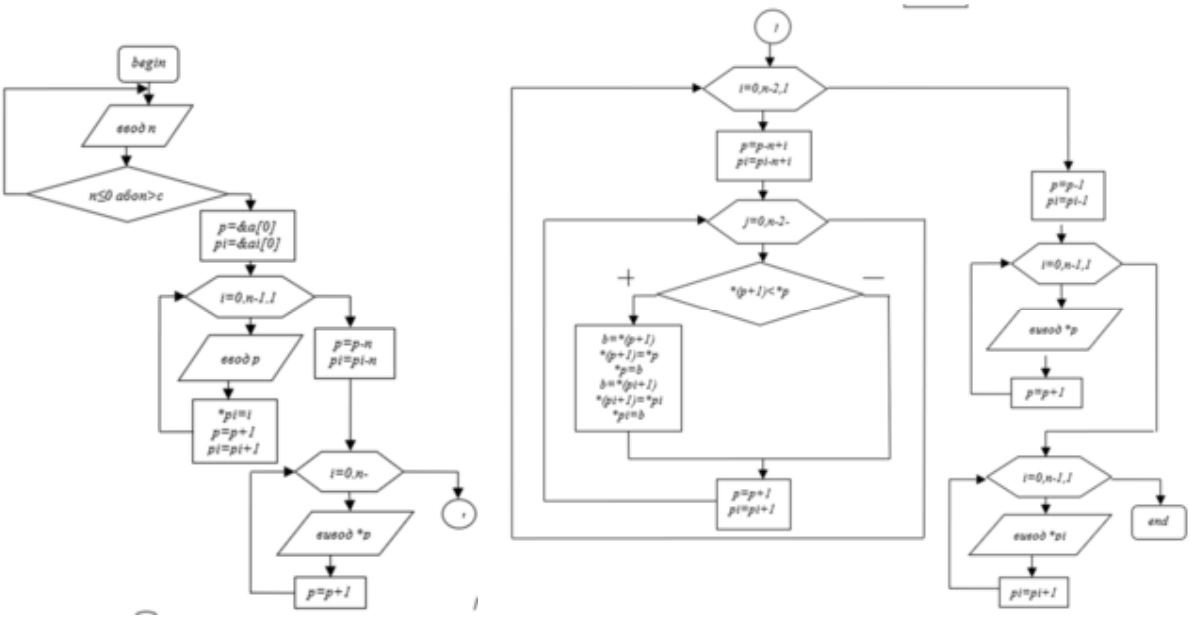


Рис. 9.7 - Блок-схема алгоритму програми

Текст програми

```
#include <stdio.h>
#include <conio.h>
void main()
{
   const c=10;
   int i,j,b,n,a[c],ai[c],*p,*pi;
   do
   {
      printf("\nInput number of elements of massiv n=");
      scanf("%d",&n);
   }
   while (n<=0 || n>c);
   p=&a[0]; pi=&ai[0];
   printf("\nInput elements of massive:\n");
   for (i=0;i<n;i++)
   {
      printf("a[%d]=",i);
      scanf("%d",p);
      *pi=i; p++; pi++;
   }
   p-=n; printf("\nOutput massiv:\n");
   for (i=0;i<n;i++)
   {
      printf("%d ",*p); p++;
   }
   for (i=0;i<n-1;i++)
```

```
   {
      p=p-n+i; pi=pi-n+i;
      for (j=0;j<n-1-i;j++)
      {
         if (*(p+1)<*p)
         {
            b=*(p+1);
            *(p+1)=*p; *p=b;
            b=*(pi+1);
            *(pi+1)=*pi; *pi=b;
         }
         p++; pi++;
      }
   }
   p--; pi--; printf("\nOutput massiv:\n");
   for (i=0;i<n;i++)
   {
      printf("%d ",*p); p++;
   }
   printf("\n");
   for (i=0;i<n;i++)
   {
      printf("%d ",*pi); pi++;
   }
   getch();
   return;
}
```

Тестування

Теоретично розраховане вихідне значення	*Практично отримане вихідне значення*
Тест: *вхідний масив: n=8; a[n]: 9 0 -3 6 -3 2 8 9*	
вих. масив a[n]: -3 -3 0 2 6 8 9 9 *масив індексів ai[n]: 2 4 1 5 3 6 0 7*	*вих. масив a[n]: -3 -3 0 2 6 8 9 9* *масив індексів ai[n]: 2 4 1 5 3 6 0 7*

Виконання індивідуального завдання.

1. Постановка задачі.

Розробити алгоритм і написати програму за індивідуальним завданням.

2. Вхідні і вихідні дані

Усі задіяні в програмі змінні повинні бути оголошені.

Неприпустимо задавати вхідні дані з допомогою операторів присвоювання. Вводу даних з клавіатури повинен передувати вивід відповідного повідомлення.

3. Математична модель і опис алгоритму задачі.

4. Блок-схема алгоритму

Представити алгоритм задачі у вигляді блок-схеми.

5. Текст програми

Розроблений алгоритм реалізується мовою програмування високого рівня С.

6. Тестування

Результати тестування представити у вигляді таблиці.

Варіант №1

1. Змінити знак у елементів масиву, номер яких кратний 3, вивести отриманий масив.

Знайти і вивести номер мінімального елементу серед елементів масиву, менших введеного значення *x*.

Варіант №2

1. Замінити всі позитивні парні елементи масиву одиницями, вивести отриманий масив.
2. Знайти і вивести номер першого максимального елементу серед від'ємних елементів масиву.

Варіант №3

1. Замінити всі позитивні непарні елементи масиву нулями, підрахувати їх кількість, вивести це значення у отриманий масив.
2. Знайти і вивести номер максимального елементу серед позитивних парних за значенням елементів масиву.

Варіант №4

1. Замінити всі позитивні елементи масиву мінімальним елементом, вивести отриманий масив.
2. Знайти і вивести номер першого максимального елементу серед від'ємних елементів масиву.

Варіант №5

1. Замінити всі позитивні елементи масиву максимальним елементом, вивести отриманий масив.
2. Знайти та вивести номер останнього максимального значення серед позитивних елементів масиву.

Варіант №6

1. Замінити кожен 5-й елемент масиву максимальним елементом, вивести отриманий масив.
2. Знайти і вивести номер останнього мінімального елементу серед елементів, які менше введеного значення *x*.

Варіант №7

1. Замінити знак у елементів масиву, кратних 5, вивести отриманий масив.

2. Знайти і вивести номер останнього елементу серед елементів масиву, які лежать в діапазоні введених значень [*c,d*].

Варіант №8

1. Замінити всі від'ємні непарні за значенням елементи масиву одиницями, вивести отриманий масив.
2. Знайти і вивести номер останнього мінімального елементу серед парних за значенням позитивних елементів масиву.

Варіант №9

1. Замінити всі позитивні непарні за значенням елементи масиву нулями, вивести отриманий масив.
2. Знайти і вивести номер останнього мінімального елементу серед елементів масиву, які менше введеного значення *x*.

Варіант №10

1. Замінити усі позитивні непарні за значенням елементи масиву мінімальним елементом, вивести отриманий масив.
2. Знайти і вивести номер першого максимального елементу серед елементів масиву, який лежить у діапазоні від *a* до *b*.

Варіант №11

1. Замінити кожен 3-й елемент масиву мінімальним елементом, вивести отриманий масив.
2. Знайти і вивести номер максимального позитивного елементу масиву, кратного 7.

Варіант №12

1. Замінити всі від'ємні парні елементи масиву нулями, підрахувати їх кількість, вивести це значення і отримати масив.
2. Знайти і вивести номер мінімального позитивного елементу.

Варіант №13

1. Замінити кожний 4-й елемент масиву мінімальним елементом, вивести отриманий масив.
2. Знайти і вивести номер першого максимального значення серед елементів, які менші введеного значення *t*.

Варіант №14

1. Замінити кожен 7-й елемент масиву мінімальним елементом, вивести отриманий масив.
2. Знайти и вивести номер останнього максимального елементу серед елементів масиву, які лежать у діапазоні введених значень [*c,d*].

Варіант №15

1. Переставити елементи масиву так, щоб спочатку були нульові значення у порядку їх слідування, і вивести отриманий масив. Для перестановок додатковий масив не використовувати.
2. Знайти і вивести номер останнього максимального елементу серед позитивних елементів масиву.

Варіант №16

1. Переставити елементи масиву так, щоб спочатку стояли від'ємні значення у порядку їх слідування, вивести від'ємний масив. Для перестановок додатковий масив не використовувати.
2. Знайти і вивести номер першого максимального елементу серед елементів масиву, які більше введеного значення *x*.

Варіант №17

1. Переставити елементи масиву так, щоб спочатку стояли позитивні значення у порядку їх слідування, вивести отриманий масив. Для перестановок додатковий масив не використовувати.
2. Знайти і вивести номер першого мінімального позитивного елементу масиву.

Варіант №18

1. Підрахувати суму елементів масиву, номер яких кратний 3-м, змінити знак цих елементів на протилежний, вивести значення суми в отриманий масив.
2. Впорядкувати масив за зростанням, використовуючи сортування-бульбашку, вивести отриманий масив і масив індексів після перестановок.

Варіант №19

1. Знайти і вивести номер максимального елементу серед парних за значенням елементів масиву.
2. Впорядкувати масив за зменшенням, використовуючи бульбашкове сортування, вивести отриманий масив і масив індексів після перестановок.

Варіант №20

1. Знайти та вивести номер максимального елементу масиву серед елементів, які кратні введеному значенню *k*.
2. Впорядкувати масив за зменшенням, використовуючи бульбашкове сортування, вивести отриманий масив і масив індексів після перестановок.

Варіант №21*

1. Підрахувати суму та добуток всіх від'ємних непарних елементів масиву, замінюючи їх нулями, вивести значення суми, добутку та отриманий масив.
2. Впорядкувати масив за збільшенням, використовуючи сортування простого вибору, вивести отриманий масив і масив індексів після перестановок.

Варіант №22*

1. Підрахувати суму парних за номером елементів масиву, замінити її значенням останній елемент, знайти номер першого мінімального позитивного елементу, вивести його і отриманий масив.
2. Впорядкувати масив за зростання, використовуючи сортування простого вибору, вивести отриманий масив і масив індексів після перестановок.

Варіант №23*

1. Знайти і вивести номер першого максимального елементу серед від'ємних парних за значенням елементів масиву.
2. Впорядкувати масив за зменшенням, використовуючи сортування простого вибору, вивести отриманий масив і масив індексів після перестановок.

Варіант №24*

1. Зробити *переворот* масиву так, щоб перший елемент став останнім, а останній – першим, вивести отриманий масив. Додатковий масив не використовувати.
2. Впорядкувати масив за зменшенням, використовуючи сортування простого вибору, вивести отриманий масив і масив індексів після перестановок.

Варіант №25*

1. Знайти і вивести номер останнього максимального елементу серед непарних за значенням елементів масиву.
2. Впорядкувати масив за збільшенням, використовуючи сортування простого вибору, вивести отриманий масив і масив індексів після перестановок.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Чим є змінна у програмі?
2. Що означає унарна операція «&», до яких об'єктів вона може застосовуватись?

3. Яке значення може набувати адреса?
4. Чим є вказівник, що він містить?
5. Для чого використовують вказівники?
6. Що означає унарна операція «*», що є результатом її виконання?
7. На об’єкт якого типу може посилатись вказівник?
8. Що означає вказівник на тип *void*?
9. Що таке *статична структура даних*?
10. Чим характеризуються змінні статичних структур?
11. Що являє собою масив даних?
12. На якому етапі виділяється пам’ять під елементи масиву?
13. Чи змінюється обсяг пам’яті, яка виділяється під масив, під час виконання програми?
14. Як занести у вказівник адресу першого елементу масиву?
15. На що посилається будь-який вказівник?
16. Що є результатом операції *вказівник+i*?
17. Чим відрізняються операції *p*++ та **p*++, якщо *p* – вказівник?
18. Чим відрізняються операції *(*p)*++ та **(++p)*, якщо *p* – вказівник?
19. Що означає лінійний пошук у масиві даних?
20. Що називається сортуванням інформаційної структури?
21. Які існують ознаки порядку?
22. У чому полягає суть бульбашкового сортування масиву даних?
23. Як виконати сортування масиву простим вибором?

9.3 Лабораторна робота 13. Вказівники

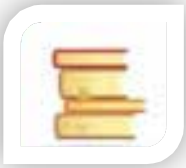
Тема: Вказівники (покажчики)

Мета: Познайомитися з адресацією пам’яті, навчитися правильно використовувати вказівники різних типів.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

У мові С передбачене використання покажчиків, які дають можливість оперувати з адресами пам’яті та здійснювати непрямою адресацією. ля всіх відзначених раніше типів даних ім'я - це символічне позначення місця в пам'яті. Як відомо, після компіляції вихідної програми ці імена заміщаються відповідними адресами. Під час виконання програми використовуються значення, які зберігаються по певних адресах. Наприклад, в операторі присвоєння var1=var2; значення змінної var2 копіюється в змінну var1.

Поруч із такими змінними в мові С++ передбачаються й такі, які зберігають адреси змінних або покажчики.

Покажчик - це змінна, що містить адресу іншої змінної. Якщо визначені змінні , то поруч зі звичайними діями можна оперувати й з їхніми адресами за допомогою операції **&** (взяти адресу). Відзначимо, що праворуч від **&** може стояти тільки ім'я змінної або змінної з індексом, а не вираження загального виду.

Унарна операція * розглядає свій операнд як адресу кінцевої мети й звертається по цій адресі, щоб витягти вміст. В описі покажчика обов'язково визначається тип об'єкту, з яким визначений покажчик. Цей опис використовується при обчисленні виразів із покажчиками.

Покажчики можна використовувати в операторах присвоєння px=py, якщо px і py відповідають однаковим типам даних.

Єдиною константою, яку можна присвоїти покажчику, є нуль, або NULL, що у файлі stdio.h : py=NULL.

До покажчиків можна додавати або віднімати ціле число, залежно від типу даних у виразах з покажчиками здійснюється відповідне масштабування.

Для покажчиків визначена й операція розрахунку різниці адресів: pz=px-py. Операція додавання покажчиків немає ніякого змісту. Оскільки визначені операції додавання й вирахування констант, та визначення й здійснення операцій збільшення й зменшення. В арифметичних виразах замість імені змінної можна застосовувати операцію доступу за покажчиком. Операції доступу за покажчиком можна вживати й у лівій частині оператора присвоєння. Унарні операції * і **&** виконуються раніше арифметичних. Між собою унарні операції рівноправні й виконуються праворуч ліворуч. Для покажчиків можна застосовувати операцію взяти адресу, де перебуває сам покажчик &px. Крім того, визначені відносини: **==**, **!=**, **>**, **<.** В арифметичних виразах замість імені змінної можна застосовувати операцію доступу за покажчиком.

Практична частина

Завдання

Набрати текст програми. Проаналізувати отримані від програми результати. Набрати текст іншої програми, знайти в ньому синтаксичні помилки і виправити їх, на початок програми додати вивід на екран адреси усіх змінних, а в кінець – значення всіх змінних, проаналізувати отримані результати і пояснити, чому вони саме такі.

```
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
int main(void){
//Змінні
int a=1;
float b=2;
double c=3;
//Вказівники
int *p1=&a;
float *p2=&b;
double *p3=&c;
void *p4; // Адреси змінних і розмір виділеної пам’яті
printf("a:      int: start address %p extent %d\n",&a,sizeof(a));
```

```
printf("b:    float: start address %p extent %d\n",&b,sizeof(b));
printf("c:       double: start address %p extent %d\n\n",&c,sizeof(c)); //Адреси вказівників і розмір виділеної пам’яті
printf("p1: pointer: start address %p extent %d\n",&p1,sizeof(p1));
printf("p2: pointer: start address %p extent %d\n",&p2,sizeof(p2));
printf("p3: pointer: start address %p extent %d\n\n",
      &p3,sizeof(p3)); //Значення, на які посилаються вказівники
printf("p1: %p related value %d\n",p1,*p1);
printf("p2: %p related value %f\n",p2,*p2);
printf("p3: %p related value %lf\n\n",p3,*p3);
//Використання вказівників у виразах
printf("a=%d\tb=%f\tc=%lf\n",a,b,c);
*p1=5;
*p2=*p2*(*p1);
*p3=sqrt(*p3);
printf("a=%d\tb=%f\tc=%lf\n",a,b,c);
printf("*p1=%d\t*p2=%f\t*p3=%lf\n\n",*p1,*p2,*p3); //Присвоювання вказівників
p1=(int*)p2;
p3=(double*)p2;
p4=p2;
printf("p1=%p\tp2=%p\tp3=%p\tp4=%p\n",p1,p2,p3,p4);
printf("*p1=%d\t*p2=%f\t*p3=%lf\t*(float*)p4=%f\n\n",
      *p1,*p2,*p3,*(float*)p4); //Зміна значень вказівників
p1++;
p3--;
printf("p1=%p\tp2=%p\tp3=%p\n",p1,p2,p3);
printf("*p1=%d\t\t*p2=%f\t*p3=%lf\n",*p1,*p2,*p3);
p1-=4;
p3=(double*)&a-1;
printf("p1=%p\tp2=%p\tp3=%p\n",p1,p2,p3);
printf("*p1=%d\t*p2=%f\t*p3=%lf\n",*p1,*p2,*p3);
system("pause");
return 0;
}
```

Набрати текст програми, знайти в ньому синтаксичні помилки і виправити їх, на початок програми додати вивід на екран адрес усіх змінних, а в кінець – значення всіх змінних, проаналізувати отримані результати і пояснити, чому

вони саме такі. Замінити оператор «m+=2;» оператором «m++;», проаналізувати результат.

```
#include<stdio.h>
#include<stdlib.h>
int main(void)
{
char *p, c;
int *a, b;
float *x, y = 3.5;
double *m, n;
a = &b;
printf("  Enter b = ");
scan("%d", a);
printf("a = %p\t*a = %d\tb = %d\n", a, *a, b);
p = a;
c=*p;
*p=*(p+3);
*(p+3)=c;
printf("p = %p\tc = %d\ta = %p\tb = %d\n", p, c, a, b);
x = &y;
printf("x = %p\t*x = %f\ty = %f\n", x, *x, y);
a = x;
*a = *x;
printf("a = %p\t*a = %d\tx = %p\t*x = %f\ty = %f\n",a,*a,x,*x,y);
a = &b;
y = 12345,6789;
printf("x = %p\t*x = %f\ty = %f\n", x, *x, y);
p = x;
c=*p;
*p=*(p+3);
*(p+3)=c;
printf("p = %p\tc = %d\tx = %p\ty = %f\n", p, c, x, y);
m = &n;
printf("m = %p\t*m = %lf\tn = %lf\n", m, *m, n);
n = 5.5;
printf("m = %p\t*m = %lf\tn = %lf\n", m, *m, n);
b = n = y = 1.7;
printf("b = %d\ty = %f\tn = %lf\n", b, y, n);
```

```
printf("*a = %d\t*x = %f\t*m = %lf\n", *a, *x, *m);
m += 2;
printf("n = %lf\tn = %p\tm = %p\n", n, &n, m);
*m = (float)*a - n + (int)*x;
printf("  m = %p\t*m = %lf\n", m, *m);
system("pause");
return 0;
}
```

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке вказівник?
2. Який об’єм пам’яті займає вказівник?
3. Що є значенням змінно-вказівника?
4. Як ініціалізувати вказівник?
5. Шо таке NULL?
6. Що таке вказівник на void? Навіщо потрібні такі вказівники?
7. Які операції допустимі при роботі з вказівниками?
8. Чим відрізняється унарна операція "&" від унарної "*" ?
9. Сумісність типів вказівників.
10. Чи можна отримати адресу вказівника?
11. Чи можна вказівнику присвоїти його ж адресу?
12. Чому до вказівника на void неможливо застосувати операцію розіменування?
13. Як працюють операції інкременту і декременту, при застосуванні до вказівників?
14. Який результат операції віднімання, яку застосовують до вказівників

одного типу?

15. Який специфікатор типу використовують при виводі адреси на екран з допомогою функції printf()?

16. У чому відмінність записів *(double *) a* від *(double) * a*, якщо *a* – вказівник на ціле число?

17. У чому відмінність записів **a*++ від (**a*)++, якщо *a* – певний вказівник, відмінний від void*?

18. Як описати вказівник на початок масиву?

19. Як описати вказівник на вказівник?

20. Коли і навіщо може повторно використовуватись операція розіменовування?

9.4 Лабораторна робота 14 та 15. Масиви. динамічне виділення пам'яті (у двох частинах)

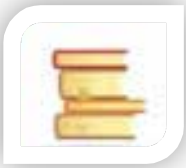
Тема: Масиви. Динамічне виділення пам’яті

Мета: Познайомитися з організацію масивів мовою С, вивчити принципи роботи з масивами, опанувати роботу з масивами через вказівники, навчитися грамотно виділяти та звільняти пам’ять у процесі роботи програми.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Масив - це послідовність однорідних даних, яка має фіксовану довжину. Цей тип належить до похідних тому, що складається із простих.

Масиви, як і інші змінні, повинні бути описаними. Спеціальних ключових слів нема, тобто відзначається ім'я в прямокутних дужках кількість елементів:

```
int a[10]
 float x[100].
```

Такий опис може включатися і у звичайні змінні

```
char c, d, f[20],
```

де c, d - звичайні символьні змінні.

Кожний елемент масиву визначається своїм номером a[2], x[50]. На відміну, від інших мов, нумерація елементів починається з нуля й закінчується номером N-1, де N - загальна кількість елементів. Тому a[2] - це третій елемент, а x[50] -п'ятдесят перший. Це зроблене тому, що повне ім'я масиву є базовою адресою, яка дорівнює адресі першого елементу. Тобто a==&a[0] Адреса другого елемента визначається як базова адреса плюс один &a[1]==a+1. Адреса N- го елемента дорівнює базовій плюс N-1. Ясно, що зсув по адресах фізичних змінних залежить від типу даних, тобто індекс необхідно помножити на відповідний коефіцієнт типу (для char це буде 1, для int 2, float 4 і т.д.). Тому, така система нумерації спрощує адресацію елементів масиву.

У деяких випадках можна не відзначати кількість елементів масиву, наприклад int arr[]. Відповідний розмір визначає сам компілятор. Як і інші змінні масиви зв'язуються з відповідним класом пам'яті. Тому якщо масив відноситься до автоматичного, то його необхідно описувати в блоці. Коли масив є зовнішній і був описаний раніше, то в блоці його можна не описувати.

Масив може складатись не лише із даних простих типів, але і з похідних. Зокрема можна розглядати масив масивів. Приходимо до багатомірних масивів 2х, 3х, і т.д. Двомірний масив – це одномірний масив, елементами якого є одномірні масиви. Тому при описі двовимірного масиву розмір по кожному виміру зазначається в окремих дужках int a[10][10].

Окремий елемент a[2][3]. Перший індекс – рядка, другий стовбчика. Тому це буде елемент з третього рядка та четвертого стовпчика.

Іноді в деяких випадках можна не наводити розміри одновимірних масивів. Але для багатовимірних це не так. І насправді , якщо у функції записати int array[][], то буде незрозумілим, як такий масив ділити на рядки. Тому обов'язково потрібно відзначати кількість стовбчиків int array[][4], тобто в кожному рядку по 4 елемента

Ім'я масиву дорівнює базовій адресі цієї ділянки. Тобто на увесь час виконання програми за ділянкою пам'яті закріплюється ім'я масиву. Тому воно є константою і його не можна змінювати під час виконання програми за допомогою оператора присвоєння. Зміст масивів можна змінювати, наприклад, за допомогою функції копіювання або введення.

Потрібно відзначити, що при роботі з масивами за допомогою покажчиків розмір масивів не контролюється й такі можливості не фіксуються виконуючою системою. Якщо покажчик не буде ініціалізуватися, то його значення невизначене (кожне). Тому можливі помилки при використанні покажчиків.

У таких випадках необхідно спочатку виділити певна ділянка динамічної пам'яті (під час виконання). Це можна зробити за допомогою функції malloc:

(тип *) malloc (кіл-ть байт);

Функція **malloc** повертає **char**- покажчик на закріплений простір.

У пам'яті, на яку вказує значення, що вертається, гарантоване вирівнювання для зберігання будь-якого типу об'єкта. Щоб одержати покажчик на тип, відмінний від char, використовується перетворювач типу значення, що вертається.

Вертається значення NULL, якщо вільної пам'яті залишилося мало.

Для звільнення динамічної пам'яті використовується функція: **free (pr),**
де **pr** - ім'я покажчика.

Ще одна функція виділення певної ділянка динамічної пам'яті (під час виконання)

(тип *) calloc (кількість елементів n, розмір елемента в байтах size);

Функція calloc виділяє простір для зберігання масиву з n елементів, кожний довжиною size байт. Кожний елемент ініціалізується в 0. Повертає значення NULL, якщо залишилося недостатньо пам'яті.

free (ім'я покажчика) - звільняє захоплене місце.

delete ім'я покажчика - звільняє виділене місце.

Функція ***realloc*** змінює розмір раніше захопленого блоку пам'яті. Вміст блоку не змінюється.

(тип *) realloc (новий розмір у байтах);

Отже, відкривається можливість працювати із двомірним масивом через одномірний. Із цього випливає також, що двомірний масив може бути заданий як одномірний масив покажчиків.

При цьому кількість елементів у рядку не фіксується. Це зручно при роботі з масивами рядків.

Практична частина

Завдання

Написати шість програм згідно індивідуальному варіанту. Використовувати звертання до елементів масиву з допомогою операції індексації і через вказівники. Ввод елементів масиву здійснювати з клавіатури. Під час відладки і тестування програми розмір масиву можна зменшити. При виводі матриці слідкувати за тим, щоб ширина всіх стовбців матриці була однаковою.
У другій задачі пам’ять під масив виділяти динамічно.
Одну з задач з матрицею (третю або четверту) вирішити двома способами:
1) матрицю записати у вигляді статичного двовимірного масиву;
2) організувати динамічну матрицю, використовуючи динамічне виділення пам’яті.
Варіанти завдань
Варіант 1
1. Визначити, чи являють собою елементи масиву A (20) зростаючою послідовністю.
2. Сформувати новий масив з елементів вихідного лінійного масиву, які зустрічаються в ньому лише один раз.

3. Обчислити $S = \frac{k_i + k_j}{m_i + m_j}$, де k_i де k_j – індекси рядка і стовбця мінімального позитивного елементу, а m_i та m_j – індекси рядка і стовбця першого позитивного елементу матриці Y (5x7).

4. Визначити, чи є дана квадратна матриця симетричною відносно своєї побочної діагоналі.

5. Книга - тривимірний масив букв (3 координати - сторінка, рядок, номер букви в рядку). Вивести на екран *n*-ю строку зі сторінки *p* книги, яка містить 450 сторінок, на кожній сторінці якої розміщується по 80 рядків, довжина кожного рядка – 64 символи.

Варіант 2

1. Записати елементи масиву C (20) у зворотному порядку $\{C_{20};C_{19};C_{18};\ldots;C_2;C_1\}$. Допоміжний масив не використовувати.

2. Визначити кількість елементів лінійного масиву, які більше середнього арифметичного значення елементів цього масиву.

3. Заповнити матрицю A (8x8) наступним чином: на головній діагоналі – «0», над діагоналлю – «1», під діагоналлю – «-1».

4. Поміняти місцями мінімальний елемент матриці P (9x11) і елемент, значення якого співпадає з заданим X. Якщо вказаний елемент в матриці відсутній, вивести повідомлення про це.

5. Куб складається з n^3 прозорих та непрозорих елементарних кубиків. Побудувати повністю непрозорий куб, використовуючи рівно n^2 непрозорих елементарних кубиків.

Варіант 3

1. В масиві A (45) знайти локальні максимуми, визначити їх місце знаходження. Локальним максимумом називають елемент масиву, значення якого більше, ніж значення двох сусідніх з ним елементів.

2. Видалити з одновимірного масиву всі елементи, значення яких у цьому масиві повторюються, залишивши по одному.

3. Дана матриця розміром 7x7. Поміняти місцями *k*-й стовбець з *k*-им рядком (*k* вводиться з клавіатури).

4. Обчислити середнє арифметичне значення елементів, які лежать на діагоналях квадратної матриці. Замінити цим значенням всі елементи матриці, які не розташовані на діагоналях.

5. Тривимірний масив описує журнал фіксації середньодобової температури протягом 10 календарних років. Кожна сторінка журналу описує один рік, номера строк відповідають місяцям року, а номер стовбця – дням місяця (вважати, що в кожному місяці 30 днів). Визначити самий спекотний і

самий холодний дні у кожному календарному році за весь період спостережень.

Варіант 4

1. З масивів A (20) та C (20) утворити новий масив X={a_1,c_1,a_2,c_2,…,a_{20},c_{20}}
2. Обчислити $Z = \frac{P_1 - P_2}{N_1 + N_2}$, де P_1 та P_2 – перший та другий позитивні елементи лінійного масиву, N_1 та N_2 – перший та другий від'ємні елементи того ж масиву.
3. Поміняти місцями максимальний і мінімальний елементи матриці A (8x12).
4. Обчислити середнє арифметичне значення елементів, які розташовані на діагоналях квадратної матриці. Замінити цим значенням усі діагональні елементи матриці.
5. Результати сесії, складаються з чотирьох іспитів, для трьох груп з 25 студентів представлені тривимірним масивом 3x25x4. Оцінка ставиться за чотирибальною системою; неявка позначена одиницею. Визначити, іспит з якої дисципліни викликав найбільші труднощі, тобто з якої дисципліни середній бал самий менший.

Варіант 5

1. В масиві A(27) найменший елемент поставити на перше місце, найменший із тих, що залишились – на останнє місце, наступний за величиною – на друге місце, наступний – на передостаннє і так далі до середини масиву.
2. Вставити число 100 після другого позитивного елементу лінійного масиву.
3. Обчислити $F = \frac{S_n + S_o}{S_n - S_o}$, де S_n – сума позитивних елементів у непарних рядках матриці Y(9x12), а S_o – сума від'ємних елементів в парних строках тієї ж матриці.
4. Дана матриця A(10x10). Відсортувати елементи, які лежать на головній діагоналі, у порядку зростання.
5. Тривимірний масив описує журнал викладача. Кожна сторінка журналу містить помітки щодо ступеня виконання N студентами однієї групи M лабораторних робіт (у кожному рядку – відмітки для одного студента, у кожному стовбці – відмітки по одній роботі). Всього існує 5 ступенів виконання роботи: 0 – нічого, 1 – отримано завдання, 2 – виконана робота, 3 – сформовано звіт по роботі, 4 – робота захищена. У журналі L сторінок – за кількістю студентських груп. Прийняти N=25, M=7, L=8. Порахувати, скільки студентів буде допущено до екзамену у кожній групі і скільки з них отримає екзамен «автоматом». До екзамену допускаються студенти, у яких є звіти по всіх лабораторних роботах.

«Автоматом» проставляється екзаменаційна оцінка студентам, які захистили всі роботи.

Варіант 6

1. Перетворити заданий масив наступним чином: з позитивних елементів відняти перший, до від'ємних прибавити останній елемент, перший і останній елементи, а також рівні 0 залишити без змін.
2. Вставити після кожного п'ятого елементу лінійного масиву значення, які дорівнюють сумі трьох попередніх елементів.
3. Відсортувати рядки матриць M(8x5) у порядку зростання.
4. Результати сесії, яка складається з трьох екзаменів, для групи з N студентів представлені матрицею K(Nx3). Оцінка ставиться за чотирибальною системою; неявка позначається одиницею. Підрахувати кількість неявок, незадовільних, задовільних, хороших та відмінних оцінок за кожним іспитом.
5. Куб складається з n^3 прозорих і непрозорих елементарних кубиків. Чи є хоча б один просвіт за кожним із трьох вимірів? Якщо так, вивести координати кожного просвіту.

Варіант 7

1. Поміняти місцями максимальний від'ємний і перший позитивні елементи масиву B (18).
2. Сформувати одновимірний масив простих двозначних чисел.
3. Дана матриця A(9x10). Розставити стовбці таким чином, щоб елементи у першому рядку були впорядковані за зростанням.
4. Видалити з матриці всі строки, які містять одиниці.
5. Тривимірний масив описує журнал відвідувань однієї групи вишу. Кожна сторінка журналу містить відомості про присутність N учнів на M заняттях за однією дисципліною (у кожному рядку – відомості про одного студента, у кожному стовбці – відомості про одне заняття). Якщо студент був присутнім на заняттях, у відповідну комірку записується 1, якщо відсутнім – 0. У журналі L сторінок – за кількістю дисциплін, що вивчаються. Нехай N=28, M=34, L=12. Визначити, скільки в групі студентів, які ніколи не пропускають заняття, і чи є такі, хто жодного разу не з'являвся на заняттях.

Варіант 8

1. Вставити число 0 в середину масиву M (20), попередньо зрушивши вправо значення елементів масиву, починаючи з 11-го. Виділити пам'ять одразу під 21 елемент.
2. Впорядкувати лінійний масив у порядку незростання значень його елементів.
3. У кожному рядку матриці A (7x9) поміняти місцями перший елемент і

максимальний за модулем.

4. Обчислити суму елементів квадратної матриці, які розташовані праворуч від побочної діагоналі.

5. Результати сесії, які складаються з чотирьох екзаменів для трьох груп з 25 студентів, представлені тривимірним масивом 3x25x4. Оцінка ставиться за чотирибальною системою; неявка позначена одиницею. Визначити, яка група краще підготувалась до сесії, отримавши більш високий середній бал.

Варіант 9

1. Обчислити середнє арифметичне значення елементів масиву M(25), значення яких розташовані у введеному з клавіатури діапазоні [X, Y].

2. Видалити з одновимірного масиву перший від'ємний елемент.

3. У кожному рядку матриці B (9x8) змінити знак максимального за модулем елементу на протилежний.

4. Відсортувати стовбці матриці у порядку зростання.

5. Яйця фасуються на лотки 5x6 і пакуються у коробки у 2 стопки по 6 лотків у кожній. Підприємець хоче купити пристрій, який буде перевіряти яйця на цілісність і повідомляти, який процент розбитих яєць знаходиться у коробці і яке їх місцезнаходження. Описати коробку яєць тривимірним масивом і змоделювати роботу такого приладу.

Варіант 10

1. Поміняти місцями максимальний і останній від'ємні елементи масиву A (40).

2. Сформувати новий масив з елементів заданого цілочисельного масиву, які кратні 7 або містять у записі числа цифру 7.

3. Визначити сідлові точки матриці M (9x10). Сідлова точка – елемент, який одночасно є максимальним у своєму рядку і мінімальним у своєму стовбці.

4. Обчислити суму позитивних елементів матриці Q (10x10), які розташовані зліва від побочної діагоналі.

5. Результати сесії, яка складається з чотирьох екзаменів, для трьох груп з 25 студентів представлена тривимірним масивом 3x25x4. Оцінка ставиться за чотирибальною системою; неявка позначена одиницею. Визначити, екзамен з якої дисципліни викликав найбільші складності, тобто з якої дисципліни отримано більше всього двійок.

Варіант 11

1. Замінити всі елементи масиву A (19), значення яких парні, на їх квадрати, а непарні - подвоїти. Перевірити, чи змінилась сума елементів масиву.

2. Видалити з лінійного цілочисельного масиву всі елементи, які кратні 3 або 5.

3. Перетворити матрицю P (7x7) таким чином, щоб мінімальні елементи рядків виявилися на побочній діагоналі.

4. Останній від'ємний елемент кожного стовбця прямокутної матриці замінити нулем.

5. Блокнот розміром 4x6 см містить 24 листа паперу у клітинку. Розмір клітинки 5x5 мм. Маленький хлопчик заповнив цей блокнот цифрами. Кожна цифра займає одну клітинку, вільних клітин хлопчик не залишив. Відомо, що найчастіше він писав свій вік. Який вік хлопця і на якій сторінці відповідна йому цифра зустрічається найбільшу кількість разів?

Варіант 12

1. Визначити чи є в масиві Q (20) задане число X і якщо ні, то знайти найближче до нього.

2. Видалити з масиву перший позитивний елемент.

3. Знайти кількість елементів у кожному рядку матриці C (11x8), значення яких більше середнього арифметичного елементів даного рядка.

4. У заданій цілочисельній матриці вказати індекси всіх елементів, які мають найбільше значення.

5. Тривимірний масив описує шкільний журнал одного класу. Кожна сторінка журналу містить оцінки N учнів за M уроків по одному предмету (у кожному рядку – оцінки одного учня, у кожному стовпчику – оцінки за один урок). У журналі L сторінок – зі кількістю вивчаємих школярами предметів. Нехай N=25, M=50, L=15. Визначити, чи є в класі учні, які залишаться на другий рік, і якщо є, то скільки їх. На повторне навчання залишають учнів, у яких середній бал хоча б з одного предмета нижче 2,8.

Варіант 13

1. Знайти суму елементів масиву A (45), які знаходяться між максимальним і мінімальним значеннями.

2. Сформувати новий масив з позитивних непарних елементів заданого лінійного цілочисельного масиву.

3. Дана дійсна матриця D (7x9). Впорядкувати (переставити) строки матриці за зростанням сум елементів строк.

4. Визначити середнє арифметичне значення елементів квадратної матриці, які знаходяться на головній діагоналі.

5. Створити масив B(3x4x2). Змінити індексацію таким чином, щоб останній елемент мав індекси 0,-1,2. Заповнити цей масив наступним чином: якщо всі індекси елементу негативні, то присвоїти цьому елементу значення -1, якщо хоча б один з індексів елементу дорівнює нулю, то присвоїти цьому елементу значення 0, іншим елементам присвоїти значення +1. Отриманий масив

вивести на екран із вказанням індексів і значень усіх елементів.

Варіант 14

1. Визначити місцезнаходження елементів масиву A (30), які не зустрічаються у масиві B (15).
2. Знайти найбільший позитивний елемент лінійного масиву, який розташований в діапазоні [X, Y], та видалити його, зсунувши залишені елементи до началу масиву.
3. Відсортувати строки матриці A(6x7) у порядку зменшення.
4. Визначити чи є дана квадратна матриця симетричною відносно своєї головної діагоналі.
5. Дитячі пластикові кубики трьох кольорів складено у прямокутний паралелепіпед розміром MxNxP кубиків. Визначити чи знайдеться у цьому паралелепіпеді хоча б одна площина, яка паралельна граням, та складена з кубиків одного кольору.

Варіант 15

1. Визначити, чи є в масиві M (15) пари сусідніх однакових елементів.
2. Знайти найбільший від'ємний елемент одновимірного масиву і видалити його.
3. Дана дійсна матриця D (7x9). Впорядкувати (переставити) рядки матриці за незменшенням найменших елементів рядків.
4. Обчислити середнє арифметичне значення елементів квадратної матриці, яка розташована ліворуч від головної діагоналі.
5. Тривимірний масив описує шкільний журнал одного класу. Кожна сторінка журналу містить оцінки N учнів за M уроків по одному предмету (у кожному рядку – оцінки одного учня, у кожній колонці – оцінки за один урок). У журналі L сторінок – за кількістю вивчаємих школярами предметів. Нехай N=26, M=48, L=14. Визначити чи є у класі учні, які вчаться без двійок.

Варіант 16

1. Визначити чи є масив M (20) перестановкою послідовності натуральних чисел від 1 до 20, тобто перевірити чи всі числа з цього діапазону входять у вказаний масив.
2. Поміняти місцями перший і останній від'ємні елементи одновимірного масиву.
3. Дана дійсна матриця A (8x8). Перетворити матрицю: поелементно відняти останній рядок з усіх стовбців, окрім останнього.
4. З матриці Q(6x8) сформувати одновимірний масив від'ємних чисел (перегляд за рядками).
5. Результати сесії, яка складається з чотирьох екзаменів, для трьох груп з

25 студентів представлені тривимірним масивом 3x25x4. Оцінка ставиться за чотирибальною системою; не з'явився позначається одиницею. Підрахувати кількість відмінників та хорошистів у кожній групі.

Варіант 17

1. Сформувати новий впорядкований за зменшенням масив з двох впорядкованих у тому ж порядку масивів A(12) та B(20).
2. Видалити з лінійного цілочисельного масиву усі нулі.
3. Є цілочисельний масив B (7x11). Визначити скільки в ньому пар сусідніх однакових елементів. Елементи вважаються сусідніми, якщо їх індекси у стовбцях та/або в строках розрізняються не більше ніж на одиницю.
4. Обчислити $Z = \frac{X_{\min}}{X_{\max} + X_{\min}}$, де $X_{\min}$ та $X_{\max}$ – мінімальний позитивний і максимальний елемент матриці.
5. Опишіть тривимірним масивом табуретку с перекладинами. Елементи, які «належать» табуретці, дорівнюють одиниці, інші обнулити.

Варіант 18

1. Визначити, скільки елементів масиву P(50) відрізняються від середнього арифметичного значення всіх елементів цього масиву не більше, ніж на 1.
2. Сформувати новий масив з елементів заданого цілочисельного лінійного масиву, значення яких зустрічаються в ньому неодноразово. У новому масиві всі значення повинні бути унікальними.
3. Дана матриця A(10x10). Розрахувати строку таким чином, щоб елементи у першому стовбці були впорядковані за зменшенням.
4. Знайти номера перших від'ємних елементів кожного стовбця матриці.
5. Тривимірний масив описує шкільний журнал одного класу. Кожна сторінка журналу містить оцінку N учнів за M уроків по одному предмету (у кожному рядку – оцінки одного учня, у кожній колонці – оцінки за один урок). У журналі L сторінок – за кількістю вивчаємих школярами предметів. Нехай N=20, M=64, L=12. Визначити, чи є у класі учні, які вчаться лише на «добре» та «відмінно».

Варіант 19

1. Визначити, чи є в масиві Q (10) задане число X, та якщо є, то видалити його (якщо зустрічається неодноразово, то видалити всі), а якщо ні, то добавити в кінець масиву.
2. Обчислити $Z = \frac{S_n + S_o}{S_n - S_o}$, де S_n та S_o – суми позитивних та від'ємних елементів масиву A (70).

3. Заповнити матрицю A (7x8) нулями або одиницями за наступним правилом: якщо сума індексів елементу парна, елемент буде нульовим, якщо непарна – одиничним.

4. Сформувати одновимірний масив, кожний елемент якого дорівнює кількості від'ємних елементів відповідного стовбця заданої цілочисельної матриці.

5. Є зошит у клітинку товщиною 12 листів, на кожній сторінці 50x30 клітинок. В цей зошит учнем записані приклади, які містять числа і знаки операцій, причому один символ займає одну клітинку. Підрахувати, скільки разів у цьому зошиті зустрічається цифра 5 і скільки з цих п'ятірок – оцінки (оцінки виставляються на полях зошиту, ширина поля 4 клітинки, врахувати різне розташування полів на парний і непарних сторанцах)

Варіант 20

1. Вставити число *X* після останнього позитивного елементу масиву *B* з *n* елементів($n \leq 20$).

2. Лінійний масив містить цілі числа. Змінити знак у елементів, значення яких закінчується на 2, на 5 або на 9, інші слід видалити.

3. Дана цілочисельна матриця B (7x7). Знайти номера стовбців, елементи кожного з яких утворюють зростаючу послідовність ($b_{1j}<b_{2j}<\ldots<b_{7j}$).

4. Транспонуванням квадратної матриці називається таке її перетворення, при якому рядки і стовбці міняються ролями: і-й стовбець становиться і-м рядком. Дана квадратна матриця розміру *n*x*n*. Отримати транспоновану матрицю.

5. Тривимірний масив описує шкільний журнал одного класу. Кожна сторінка журналу містить оцінки N учнів за M уроків по одному предмету (у кожному рядку – оцінка одного учня, у кожному стовбці – оцінки за один урок). У журналі L сторінок – за кількістю вивчаємих школярами предметів. Нехай N=25, M=40, L=15. Пизначити загальну середню оцінку учня X за всіма предметами.

Варіант 1**********

1. **Каса.** У масиві K(N) у порядку зменшення представлені номінали грошових знаків (купюр і монет) валютної системи певної країни. Реалізувати видачу у цій системі заданої суми S мінімальною кількістю грошових купюр.

2. **Равлик.** Матрицю A(MxN) заповнити натуральними числами від 1 до MxN по спіралі, починаючи у лівому верхньому куті і закручуючи за годинниковою стрілкою.

3. **Статистика.** Тривимірний масив описує журнал фіксації середньодобової температури протягом 10 календарних років. Кожна сторінка

журналу описує один рік, номера рядків відповідають місяцям року, а номера стовбців – дням місяця. Визначити самий спекотний та самий холодний дні у кожному календарному році та за весь період спостереження. Сформувати таблицю середньомісячних температур.

4. **Кубік Рубіка**. Кубік Рубіка описується тривимірним масивом 3x3x3, у якому кожен фрагмент кубіка задається комбінацією кольорів, закодованих цифрами від 1 до 6. Відповідно, фрагмент, який має одну кольорову грань, кодується однією цифрою, фрагмент, який має 2 кольорові грані,– двома цифрами, кутові фрагменти, які мають по 3 кольорові грані – трьома цифрами. Порядок цифр значення не має, тобто є кутовий фрагмент кубіка, який має грані кольорів 1, 2 та 3 і може бути заданий комбінаціями 123, 312, 213, 132, 321 або 231. Центральний фрагмент кодується нулем. Визначити чи є заданий користувачем тривимірний масив закодованим кубіком Рубіка.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке масив?
2. Що являє собою ім’я масиву?
3. Що являє собою індекс елементу масиву?
4. Як можна звернутися до елементу масиву?
5. Як отримати адресу елементу масиву?
6. Як описати вказівник на початок масиву?
7. Як звернутися до елементу масиву через вказівник?
8. Чи існує зв’язок між індексом елементу і його значенням?
9. Чому дорівнює індекс першого елементу масиву?
10. Як проініціалізувати масив?

11. Коли можна не вказувати кількість елементів масиву при описанні?
12. Якого типу можуть бути елементи масиву?
13. Які операції можна здійснювати над цілим масивом?
14. Чому при обробці масивів використовують цикли?
15. Як поміняти місцями два елементи масиву?
16. Чому дорівнює індекс останнього елементу масиву?
17. Чи виникає помилка при зверненні до елементу масиву, індекс якого більше індексу останнього елементу цього масиву?
18. Чи може існувати масив з одного елементу? Якщо може, то як його описати?
19. Чи є *p* вказівником на масив a[], якщо вони оголошені наступним чином: *int a[10], *p=a;* ?
20. Що таке сортування масиву?
21. Що таке двовимірний масив?
22. Як розташований двовимірний масив у пам'яті комп'ютера?
23. Як проініціалізувати двовимірний масив?
24. Чому при оголошенні двовимірного масиву з одночасною ініціалізацією перші скобки можна залишити порожніми, а другі - ні?
25. При звертанні до елементу двовимірного масиву спочатку вказується індекс рядку чи індекс стовбця?
26. Як вивести двовимірний масив таблицею з колонками однакової ширини?
27. Що являє собою ім'я двовимірного масиву?
28. Що являє собою перший індекс елемента двовимірного масиву?
29. Що являє собою другий індекс елемента двовимірного масиву?
30. Що являє собою ім'я строки матриці?
31. Як отримати адресу елементу двовимірного масиву?
32. Як отримати адресу початку двовимірного масиву?
33. Як отримати адресу рядка матриці?
34. Чи можна продивитись усі елементи двовимірного масиву в одному циклі?
35. Як звернутися до певного елементу двовимірного масиву через вказівник на перший елемент матриці?
36. Як звернутися до елементу двовимірного масиву через вказівник на початок масиву?
37. Як описати трьохвимірний масив і як з ним працювати?
38. Якими способами можна запрограмувати матрицю?
39. Чим відрізняється перегляд елементів матриці за стовбцями від перегляду за рядками?
40. Який зв'язок між індексами рядка і стовбця у елементів, які знаходяться на

головній діагоналі квадратної матриці?

41. Який зв’язок між індексами строки і стовбця у елементів, які лежать на побочній діагоналі квадратної матриці?

42. Чи можна подивитись всі елементи матриці в одному циклі?

43. В яких областях оперативної пам’яті можуть розташовуватись дані при виконанні програми?

44. Які змінні називають динамічними? Чим вони відрізняються від статичних змінних?

45. З якою метою використовують динамічні змінні?

46. Як звертаються до динамічних змінних?

47. Які бувають вказівники? Як їх описати?

48. Яким чином можна виділити пам’ять для динамічних змінних і звільнити її?

49. Як визначити, чи виділена пам’ять чи ні?

50. Чому неможливо забувати звільняти виділену пам’ять?

51. Чому дорівнює значення вказівника після звільнення області пам’яті, на яку він вказував?

52. Як розмістити в динамічній пам’яті масив?

53. Якими способами можна розмістити в динамічній пам’яті матрицю?

РОЗДІЛ X. ФУНКЦІЇ. СТРУКТУРИ

10.1 Теоретичні відомості. Функції. Структури

10.1.1 Функції та їх використання

Як відзначалося, програма мовою С складається з декількох функцій. Усі вони рівноправні, але обов'язково якась одна повинна мати ім'я **main** і компіляція починається саме із цієї функції. Для більшої ясності програму краще виконувати з невеликих функцій, а не з більших.

Імена функцій мають глобальний характер, тому вкладених функцій не передбачено. Послідовність обігу – будь-яка, може викликати функцію із цієї ж самої функції, тобто рекурсивні функції.

10.1.2 Структура функції

У загальному випадку програма містить кілька функцій. Тому необхідно визначити структуру функції - підпрограми.

Описання функції являє собою блок, тобто складається із заголовка й тіла. Заголовок має вигляд: ¶

Тип_функції ім‘я_функції (тип змінна, тип змінна, ...)

{

Тіло функції;

return (вираз);

}

де ***тип змінна*** – список формальних параметрів.

На відміну від звертання до функції при описі функції наприкінці ";" не ставиться. З іменем функції можна зв'язувати одне або жодного значення. Наприклад, функція **puts** виводить тільки рядок і з її іменем не зв'язується ніяке значення.

Щоб ім'я функції одержало значення (повертало значення), у її тілі повинні бути присутній оператор (функція) **return (вираз)**. Дужки не обов'язкові.

Значення виразу привласнюється імені функції, тобто функція повертає значення змінній. Інше призначення цього оператора - передати керування відповідній до програми на оператор, який іде слідом за оператором виклику функції.

Функція може містити один, декілька операторів (функцій) **return** або жодного. Приклад: використання функції визначення абсолютної величини:

```
main ()
{ int a=15, b=0, c=-32;
```

```
    int d, e, f;
    d=absn (a);
    e=absn (b);
    f=absn (c);
    printf (“ %d %d %d \n”, d, e, f);
}
/*функція абсолютного значення*/
absn (int x)
{  return (x<0 ? -x : x);}
```

Тут значення змінної **y** буде привласнено імені функції **absn**, яке передає це значення в визивну функцію. Відзначимо, що змінна **y** є локальною (автоматичною) і без неї можна обійтися.

10.1.3 Фактичні і формальні параметри

Функція являє собою певну закінчену процедуру і її можна розглядати як “чорну скриньку”. Тобто програміста може не цікавити як вона реалізована, а тільки її призначення і вхідні параметри. Тому всі необхідні параметри краще передавати через список заголовку функції. Звичайно, за необхідністю, можна передавати параметри і через зовнішні змінні, але це гірше, бо відноситься до побічних ефектів і програє у точності.

Виклик функції відбувається за ім'ям **absn (d)** і на відміну від інших мов, це звертання може входити до складу виразу чи бути окремим оператором (що залежить від призначення функції).

Викликаючи функцію, зазначають фактичні параметри.

Розглянемо програму перетворення маленьких латинських літер у великі. У кодах ASCII великі літери кодуються від A 65_{10} до Z 90_{10}, а маленькі - від a 97_{10} до z 122_{10}. Тому перехід від маленьких літер до великих здійснюється відніманням 32_{10}.

```
char test(char cf)
{
if ((cf<'a') || (cf>'z')) return(cf);
return(cf+'A'-'a');
}
main ()
{  char *st="test PrograM";
for(a=0;a< strlen(st);a++) cout<<test(st[a]);
}
```

У цій програмі формальним параметром є символьна змінна **'cf'**, а фактичним - покажчик **'st'**. Як відбувається їхня взаємодія?

Оскільки передача параметрів відбувається за значенням, у тілі функції не можна змінити значення змінних у визваній функції, що є фактичними параметрами.

Приклад: ¶

```
/ / Невірне використання параметрів void change (int x, int y)
{   int k=x;
     x=y;
     y=k;
}
```

Однак, якщо в якості параметра передати покажчик на деяку змінну, то використовуючи операцію переадресації можна змінити значення цієї змінної.

Приклад:

```
/ / Вірне використання параметрів void change (int *x, int *y)
{  int k=*x;
     *x=*y;
    *y=k;
}
```

При виклику такої функції в якості фактичних параметрів повинні бути використані не значення змінних, а їх адреси:

```
change (&a,&b);
```

Коли компілятор зустрічає виклик функції, то він робить копію значень фактичних параметрів. Ці копії передаються у функцію й привласнюються формальним параметрам. Відбуваються необхідні обчислення, по закінченню яких формальні параметри видаляються (стають невизначеними). Зрозуміло, що при цьому фактичні параметри не змінюються.

Розглянемо інший варіант цієї ж програми, коли фактичним параметром буде адреса:

```
char test2(char *cf2)
{
if ((*cf2<'a') || (*cf2 >'z')) return(*cf2);
return(*cf2+'A'-'a');
```

```
}
main ()
{   char *st="test PrograM";
for(a=0;a< strlen(st);a++) cout<<test(&st[a]); }
```

У цьому прикладі формальний параметр є покажчиком.

Отже, коли необхідно поміняти величину змінної в визваній функції або передати його із визваної функції, то необхідно передавати у функцію адресу параметра. Або передача параметрів за адресою здійснюється через покажчик. Це стосується й масивів.

10.1.4 Масиви як параметри функцій

Коли масиви використовуються як параметри функцій, то для уніфікації необхідно, щоб такі функції були придатними для масивів різної розмірності. Одним з можливих варіантів є створення динамічних масивів.

Існує наступна можливість: описати такий масив як зовнішній. Тоді у функції розміри можна не відзначати (двічі не визначати), а ще задати один або кілька параметрів - фактичні розміри масивів.

```
void arrv(int ar[], int size)
{  for(int a=0;a<size;a++) cout<<ar[a]; }
void main(void)
{ int a,arr[30];
   for(a=0; a<10; a++) arr[a]=a;
   arrv(arr,10);
   getch();
        }
```

10.1.5 Типи функції. Рекурсивні функції

Правила визначення типів функції ідентичні правилам опису змінних.

У мові С передбачені посилання функції саму на себе, тобто рекурсивні функції.

Наприклад, обчислення n!

```
long factor (n)
//обчислення  n-  факторіалу
int n;
{ if (n==1) return (1);
else return (n*factor (n-1));
```

```
}
```

Кожний виклик функції називається її активізацією. В даному випадку відбувається послідовність активізації, за якою слідує компілятор. Послідовність закінчується, коли **factor** поверне 1. Після цього перераховуються всі повертаємі значення до **factor (n-1)**, по якому і обчислюється n!.

Окрім рекурсивних функцій дозволяється використовувати звертання до функцій як фактичні параметри.

Наприклад

```
printf (“ %d \n”, scanf (“ %f %e %s”, &a, &b, str));
```

Тут параметром є звертання до функції **scanf**. Тому вводимо три значення – два дійсних числа та рядок. Окрім вводу, функція **scanf** повертає ціле число, яке дорівнює кількості правильно виконаних вводів. Якщо вводимо три згаданих значення через пропуск, то функція **scanf** поверне число 3, яке і буде надруковано.

10.1.6 Особливості побудови програм

Один з варіантів - розмістити всі функції в одному файлі. Але для більших програм краще створити кілька файлів: fil1.c, fil2.c. Тоді для їх об'єднання можна використовувати препроцесорні директиви ***#include "fil2.c".***

При відсутності формальних параметрів у дужках також потрібно визначити тип **void:** ***float fun(void)***

10.1.7 Особливості опису та використання функції в ANSI-C

У початкових версіях С функція могла повертати лише одне просте значення. В ANSI-C таке обмеження зняте. Тепер функція може повертати структуру або запис. Функції НЕ можуть повертати масив або функції, але дозволяється повертати покажчик на масив або функцію.

Наприклад:

```
int matr[10][10];
char ** ptr; / / покажчик на покажчик
int * arr [10]; / / масив покажчиків
int (* vect) [10] / / покажчик на масив
```

Для визначення типу даних діють такі правила пріоритетів модифікаторів:

1. чим більш ближче модифікатор до імені, тим вищий його пріоритет;

2. для модифікаторів одного рівня [], але () мають вищий пріоритет за *;

3. круглі дужки змінюють послідовність і мають вищий пріоритет.

Тому int *st[3][5] буде означати наступне:

Є модифікатор * та [3] перебувають безпосередньо біля імені, тому вони старші за [5]. Але у [3] пріоритет старший, ніж у *, тому: int *st[3][5] - масив покажчиків з трьох елементів на масив цілого типу з п'яти елементів.

Знаючи такі правила можна визначити й покажчики на функції. Вони мають наступний вигляд:

```
int (*func) (int, int);
```

Якщо круглі дужки вилучити, то int *func (int, int) означає функцію, яка передає покажчик на ціле.

Покажчики на функції реалізують процедурні змінні, масиви процедур і дозволяють передавати функції як параметри інших функцій.

Відзначимо деякі особливості роботи з покажчиками на функції: Припустимо маємо оголошення:

```
double (*fun_ptr) (int, int);
proc (int, int);
```

Як відзначалося раніше, покажчик fun_ptr поки ще формальний параметр, тобто не посилається на конкретну функцію. Щоб зв'язати його з певною функцією, потрібно привласнити йому ім'я функції без списку параметрів:

```
fun_ptr=proc.
```

Після цього звертання до функції через покажчик буде мати вигляд:

```
(*fun_ptr) (2, 5);
```

Ясно, що конкретний покажчик можна зв'язувати й з іншими функціями відповідного характеру.

Наприклад:

```
int difference (int a, int b) { return (a-b);} // функція різниці
int sum (int a, int b) { return (a+b);} // функція суми
void main (void)
{ int (*fun_ptr) (int, int);
int v1=15, v2=6,par;
fun_ptr=difference;
```

```
par=(*fun_ptr) (v1, v2);
printf (“ par=%d \n”, par);
fun_ptr=sum;
par=(*fun_ptr) (v1, v2);
printf (“ par=%d \n”, par);
}
```

Як бачимо покажчик fun_ptr тут реалізує процедурну змінну за аналогією Турбо Паскаля.

Як і звичайні змінні, покажчики на функції можуть поєднуватися в масиви.

```
int cmp_x (int, int);
int cmp_y (int, int);
int cmp_z (int, int);
int cmp_w (int, int);
int (*fcmp[4]) (int, int)={cmp_x, cmp_y, cmp_z, cmp_w};
```

Звертання до окремих функцій здійснюється як до звичайних елементів масиву:

```
int index=2;
(*fcmp[index]) (arg1, arg2);
```

Використання покажчиків на функції дає можливість задавати різні функції як фактичні параметри.

Наприклад, є стандартна бібліотечна функція:

```
void qsort (base, num, width, compare)
char * base; / / покажчик на перший елемент
int count; / / к-ть відсортованих елементів
int len; / / розмір елемента
int (* func) (); / / покажчик на функцію порівняння
```

Функція порівняння залежить від типу даних. Але вона повинна передавати ціле число:

<0 якщо перший елемент <другого;

=0 якщо елементи рівні;

>0 якщо перший елемент >другого.

У цієї функції два параметри - покажчики на 2 елемента. Коли порівнюємо рядки, то фактичний параметр:

```
extern int strcmp ();
qsort (base, count, len, strcmp);
```

```
void qsort (char *base, int num, int width, int (*compare) ());
```

Сортування дійсного масиву ***float f_mas [50];***
Потрібно зробити функцію порівняння у вигляді, необхідному для функції:
qsort

```
int f_comp (float *x, float *y)
{ return (*x-*y);}
```

Тоді звертання до функції qsort буде мати вигляд:
qsort (f_mas, size, sizeof (float), f_comp);

10.2 Лабораторна робота 16. Робота з масивами (закріплення навичок роботи з масивами)

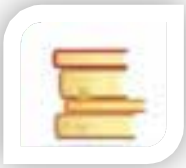
Тема: Робота з масивами (закріплення навичок роботи з масивами)

Мета: Закріпити навички роботи з масивами.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Існує двовимірний масив:

int matr[3][2];

int *pr;

Ясно, що ім’я matr дорівнює адресі базового елемента matr==&matr[0][0]; тому після присвоєння pr=matr

pr+1==matr[0][1]

pr+2==matr[1][0]

pr+3==matr[1][1] і т.д.

Отже за допомогою такого покажчика й відповідного зсуву можна адресувати будь-який елемент такого масиву.

Як відомо, елементи двомірного масиву розташовуються в пам'яті рядками. Двомірний масив - це є масив, який складається з масивів, і повне ім'я масиву є адресою базового елемента. Тому перший рядок буде мати ім'я matr[0], другий matr[1], третій matr[2]. Ці імена мають значення адрес перших елементів відповідного рядка.

Із цього випливає, що поруч із повним іменем масиву matr, іменами окремих елементів matr[i][j] можна вживати й імена окремих рядків matr[i].

Практична частина

Завдання

Написати десять програм згідно індивідуальному варіанту. Використовувати звертання до елементів масиву з допомогою операції індексації і через вказівники. Ввод елементів масиву здійснювати з клавіатури. Під час відладки і тестування програми розмір масиву можна зменшити. При виводі матриці слідкувати за тим, щоб ширина всіх стовбців матриці була однаковою.

Варіанти завдань

1. Обчислити середнє значення n елементів, введених користувачем з використанням масивів.
2. Визначити і вивести на екран найбільший елемент, введений користувачем у масиві.
3. Обчислити стандартне відхилення 10 чисел, які зберігаються у масиві.
4. Розробити програму для додавання двох матриць з використанням двовимірних масивів.
5. Розробити програму для перемноження двох матриць з використанням багатовимірних масивів.
6. Розробити програму для здійснення транспонування матриці.
7. Розробити програму для перемноження двох матриць шляхом передачі матриці у функцію.
8. Розробити програму для доступу до елементів масиву з використанням вказівників.
9. Розробити програму, в якій три числа, введені користувачем, міняються місцями у циклічному порядку з використанням виклику за посиланням.
10. Розробити програму знаходження найбільшого числа, введеного користувачем, у динамічно виділеній пам’яті.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.

4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке масив?
2. Що являє собою ім’я масиву?
3. Що являє собою індекс елементу масиву?
4. Як можна звернутися до елементу масиву?
5. Як отримати адресу елементу масиву?
6. Як описати вказівник на початок масиву?
7. Як звернутися до елементу масиву через вказівник?
8. Чи існує зв’язок між індексом елементу і його значенням?
9. Чому дорівнює індекс першого елементу масиву?
10. Як проініціалізувати масив?
11. Коли можна не вказувати кількість елементів масиву при описанні?
12. Якого типу можуть бути елементи масиву?
13. Які операції можна здійснювати над цілим масивом?
14. Чому при обробці масивів використовують цикли?
15. Як поміняти місцями два елементи масиву?
16. Чому дорівнює індекс останнього елементу масиву?
17. Чи виникає помилка при зверненні до елементу масиву, індекс якого більше індексу останнього елементу цього масиву?
18. Чи може існувати масив з одного елементу? Якщо може, то як його описати?
19. Чи є *p* вказівником на масив a[], якщо вони оголошені наступним чином: *int a[10], *p=a;* ?
20. Що таке сортування масиву?
21. Що таке двовимірний масив?
22. Як розташований двовимірний масив у пам’яті комп’ютера?
23. Як проініціалізувати двовимірний масив?
24. Чому при оголошенні двовимірного масиву з одночасною ініціалізацією перші скобки можна залишити порожніми, а другі - ні?
25. При звертанні до елементу двовимірного масиву спочатку вказується індекс рядку чи індекс стовбця?
26. Як вивести двовимірний масив таблицею з колонками однакової ширини?

27. Що являє собою ім’я двовимірного масиву?
28. Що являє собою перший індекс елемента двовимірного масиву?
29. Що являє собою другий індекс елемента двовимірного масиву?
30. Що являє собою ім’я строки матриці?
31. Як отримати адресу елементу двовимірного масиву?
32. Як отримати адресу початку двовимірного масиву?
33. Як отримати адресу рядка матриці?
34. Чи можна продивитись усі елементи двовимірного масиву в одному циклі?
35. Як звернутися до певного елементу двовимірного масиву через вказівник на перший елемент матриці?
36. Як звернутися до елементу двовимірного масиву через вказівник на початок масиву?
37. Як описати трьохвимірний масив і як з ним працювати?
38. Якими способами можна запрограмувати матрицю?
39. Чим відрізняється перегляд елементів матриці за стовбцями від перегляду за рядками?
40. Який зв’язок між індексами рядка і стовбця у елементів, які знаходяться на головній діагоналі квадратної матриці?
41. Який зв’язок між індексами строки і стовбця у елементів, які лежать на побочній діагоналі квадратної матриці?
42. Чи можна подивитись всі елементи матриці в одному циклі?
43. В яких областях оперативної пам’яті можуть розташовуватись дані при виконанні програми?
44. Які змінні називають динамічними? Чим вони відрізняються від статичних змінних?
45. З якою метою використовують динамічні змінні?
46. Як звертаються до динамічних змінних?
47. Які бувають вказівники? Як їх описати?
48. Яким чином можна виділити пам’ять для динамічних змінних і звільнити її?
49. Як визначити, чи виділена пам’ять чи ні?
50. Чому неможливо забувати звільняти виділену пам’ять?
51. Чому дорівнює значення вказівника після звільнення області пам’яті, на яку він вказував?
52. Як розмістити в динамічній пам’яті масив?
53. Якими способами можна розмістити в динамічній пам’яті матрицю?

10.3 Лабораторна робота 17. Робота зі строками (рядками)

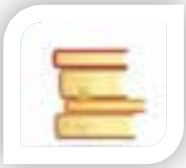
Тема: Робота зі строками (рядками)

Мета: Познайомитися з особливостями роботи зі строками, розглянути неформатований ввод зі стандартного потоку і вивод у стандартний потік

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

У мові програмування С строки – це одновимірні масиви символів (елементів базового типу **char**), які закінчуються нульовим значенням (‘\0’). До символів у строці, відповідно, можна звернутись за індексом у масиві – str[0], str[1] і т.д.

У той же час для роботи зі строками існує стандартна бібліотека функцій з заголовочним файлом **string.h**.

Найбільш поширеними функціями з цієї бібліотеки є:

strlen(str) – повертає ціле число – довжину строки str.

strcmp(str1,str2) – порівняння текстових строк str1 та str2, повертає нуль, якщо строки рівні, та ненульове значення у протилежному випадку.

strcpy(str1,str2) – копіює строку str2 у строку str1.

strstr(str1,str2) – пошук входження підстроки str1 у строку str2, повертає посилання на підстроку у строці; якщо не знайдено, то повертає NULL.

Для роботи з файлами використовується спеціальне оголошення типу. Оголошення FILE *fp оголошує змінну fp у якості *дескриптору* файлу. Відкрити файл можна з допомогою функції fopen. Приклад:

fp=fopen(“1.txt”, “r”); //відкриття файлу 1.txt на читання.

Для запису текстового файлу у якості другого аргументу fopen вказують “w”, для додавання інформації – “a”. Для випадку, якщо на запис або для додавання відкривається раніше не існуючий файл, він створюється.

У випадку успішного відкриття файлу fp вказує на файловий дескриптор а потім може використовуватись в операціях зчитування і запису fscanf(fp,…) та fprintf(fp,…), які відрізняються від стандартних функцій scanf та printf лише вказівником в якості першого аргументу файлового дескриптору, а в іншому працюють таким же чином.

Практична частина

Завдання

Завдання 1. Написати програму, яка з використанням оператора **switch** мови програмування С буде перетворювати введені з клавіатури цілі числа від двох до п’яти у оцінку за національною шкалою («незадовільно», «задовільно», «добре», «відмінно»), у випадку вводу іншого числа – виводити повідомлення про відсутність такої оцінки.

Текст програми:

```
#include <stdio.h>
int main() {
    int Mark;// Ввод оцінки у вигляді цілого числа
printf("Програма оцінювання\n Введіть ціле число->");
    scanf("%d",&Mark);// Вивід оцінки в строковой форме
    switch (Mark)
    {
      case 2:printf("незадовільно\n");
      break;
      case 3:printf("задовільно\n");
      break;
       case 4:printf("добре\n");
      break;
      case 5:printf("відмінно\n");
      break;
  default:printf("Нема такої оцінки!\n"); }
return 0;
}
```

Зверніть увагу на те, що в кінці кожної з розглянутих альтернатив оператора - перемикача switch обов’язково повинен стояти оператор **break**,

оскільки інакше внаслідок того факту, що конструкції “case” - значення розглядаються як мітки (labels) у програмі, здійсниться «провалювання» до наступної альтернативи і виконується наступна строка. Строка “default” слугує для заданої дії у тому що, якщо жодне з альтернативних значень, розглянутих у перемикачі, не підійшло до значення тестованої змінної.

Завдання 2. Переписати програму з попереднього завдання без використання оператора switch, але з використанням масиву строк.

Текст програми:

```
#include <stdio.h>
int main() {
  int i;
   char    Marx[4][12]={"незадовільно","задовільно","добре",
"відмінно"};// Ввод оцінки у вигляді цілого числа
   printf("Програма оцінювання \n Введіть ціле число->");
   scanf("%d",&i);
   if (i<2 || i>5) // Перевірка на допустимість введенного
значення
  printf("Нет такой оценки!");
  else
     printf(Marx[i-2]);
return 0; }
```

Зверніть увагу на те, як змінилась програма – центр тяжіння зараз перенесено з виконуваної частини (алгоритму) на дані, не дивлячись на збереження функціональності. Подібні можливості дуже часто існують і в більш професійних програмних комплексах. Marx - двовимірний масив, оскільки він являє собою одновимірний масив строк, які в свою чергу представляють собою масиви символів.

Завдання 3. Написати програму, яка буде здійснювати зворотне перетворення до попереднього завдання, тобто перетворювати введену з клавіатури оцінку у вигляді строки тексту у числове значення.

Текст програми:

```
#include <stdio.h>
#include <string.h>
int main() {
  int i,pr=0; // pr – ознака того, що оцінка-строка знайдена
  char  str[12],Marx[4][12]={"незд","здв",  "добр","відм"};//
Ввод оценки в виде строки
```

```
        printf("Программа оценивания по шкале вуза\n Введите
целое число->");
        scanf("%s",str);// Вывод оценки в строковой форме путём поиска в
массиве строк
        for (i=0;i<4 && pr!=1;i++)
          if (!strcmp(Marx[i],str)) {printf("%d", i+2); pr=1;}
        if (pr!=1)
        printf("Нет такой оценки!"); // Проверяем признак
return 0;
 }
```

Зверніть увагу на те, що при вводі текстової строки у функцію scanf, оскільки str і так являє собою строку – тобто масив символів (іншими словами, вказівник), використовується форматна строка “%s”і не використовується значок амперсанду (‘&’).

Ознака pr, яка спочатку встановлюється одразу при оголошенні у нуль, слугує в якості флагу, який показує чи було знайдено у масиві потрібне значення або не знайдено. У випадку, якщо значення знайдено, то не тільки друкується його номер у масиві (з додаванням 2, для забезпечення відповідності системі оцінювання), але і змінній - флагу pr присвоюється значення 1. Подібний прийом зазвичай використовується у програмуванні при вирішенні різноманітних задач широкого кола. Альтернативний варіант – переривання виконання циклу з допомогою оператора break.

Завдання 4. Написати програму, яка в деякому тексті файлу здійснює: підрахунок усіх входжень підстроки “пере”; заміну всіх входжень підстроки «абітурієнт» на слово «студент».

Текст програми:

```
 #include <stdio.h>
 #include <string.h>
int main() {
  FILE *fp,*fp1;
  char Str[80];
  int sc=0;
  printf ("Програма підрахунку числа входжень підстроки
\"пере\" у файлі test.txt\n");
printf ("і заміни слова \"абітурієнт\" на слова \"студент\"\n");// Відкриваємо
 вихідний файл
  if ((fp=fopen("test.txt","r"))==NULL) {
```

```
        printf("Помилка відкриття файлу test.txt");
        return 1;
      }
      // Створюємо проміжний файл для копіювання if
((fp1=fopen("test.bak","w"))==NULL)
      {
        printf("Помилка створення проміжного файлу");
        return 2;
      }
      while (!feof(fp)) // Поки не кінец вихідного файлу
      {
  fscanf(fp,"%s",Str); // Читаємо строку з файлу
  if (strstr(Str,"пере")!=NULL) sc++; // підстроки "пере"
        if (!strcmp("абітурієнт",Str))
  strcpy(Str,"студент"); // Заміна слів if (!feof(fp)) fprintf(fp1,"%s ",Str);
      }
      fclose(fp);
      fclose(fp1);
      if(remove("test.txt")) // Видаляємо вихідний файл
      {
        printf("Помилка видалення файлу");
        return 3;
      }
  if(rename("test.bak","test.txt")) //Перейменовуваєм проміжний файл
      {
   printf("Помилка перейменування проміжного файлу");
   return -1; }
      else
        printf("Було знайдено %d \"пере\"\n",sc);
 return 0; }
```

Файл test.txt (приклад)

абітурієнт зможе перейти через бар'єр і, стати справжньою людиною, яка вступає до вишу і буде переходити з курсу на курс, так, що абітурієнт кінець кінцем стане справжнім фахівцем − інженером!

Зверніть увагу на використання у цій програмі функцій **rename** та **remove** зі стандартної бібліотеки, які описуються заголовочним файлом stdio.h, для перейменування і видалення файлів. У програмі спочатку створюється

проміжний файл test.bak, у який копіюється з вихідного файлу вся інформація, але слова “абітурієнт” замінюються на слова “студент”, потім вихідний файл видаляється, а проміжний отримає ім’я вихідного файлу – test.txt.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке С-строка? Чим вона відрізняється від масиву символів?
2. Що таке нуль -термінатор?
3. Чим відрізняється описання *char *st* от *char st[N]*, де N – декотра константа?
4. Як ініціалізувати строку?
5. Чим відрізняється ввод строки з допомогою функції *scanf()* від вводу з допомогою функції *gets()*?
6. Як знайти довжину строки?
7. Чи можна використовувати операцію присвоювання для завдання значення строки?
8. Що таке конкатенація строк?
9. Які існують функції для роботи зі строками?

РОЗДІЛ XI. СТРУКТУРИ АБО ЗАПИСИ

11.1 Теоретичні відомості. Структури або записи

11.1.1 Опис і використання структур

При розв'язуванні багатьох інформаційних задач зазвичай доводиться використовувати дані різного типу про той же самий об'єкт. Наприклад, дані про книгу: автор, назва, видавництво, рік видання, кількість сторінок. Можна було б зберігати бібліографічні дані про книги у різних масивах: автори; назви і т.п., але це незручно. Тому виникає необхідність використання комбінованих типів, елементами яких можуть бути різні типи.

У мові С до такого типу належать структури або записи.

Описання структур має наступний вигляд:

struct < ім'я>

{ описи елементів } [ім'я змінної, …];

окремий опис елемента це фактично опис змінної – тип та ім'я.

Наприклад, для створення масиву "студенти", де про кожного треба ввести такі дані: прізвище, рік народження, стать – можна запровадити наступну структуру:

```
struct person
{ char fio[20];
int year;
char sex;
};
```

Ім'я структури інакше називається тегом (ярлик, етикетка). Тег дає назву структурі - коротким позначення тої частини структури, яка міститься в дужках.

Опис структури аналогічний опису типу у мові Паскаль, він є лише шаблоном, і не супроводжується виділенням місця у пам'яті.

У відповідності з наведеним шаблоном можна визначити змінні:

struct person stud1, stud2, stud3;

або person stud1, stud2, stud3;

Зрозуміло, що область дії і шаблону, і змінних залежить від їх розташування в програмі. Якщо опис структури навести до початку функції **main**, то він діятиме на всю програму, тобто матиме зовнішній характер. Якщо ж це зробити у функції після відкриття дужки – автоматичний.

Зазначимо, що слово структура має двоїстий характер. З одного боку його відносять до опису шаблона, а з другого – до змінних-структур.

Подібно до мови Паскаль, окремий опис шаблону надає ширші можливості. На нього можна посилатись у різних функціях, тобто використовувати багаторазово.

Якщо змінні-структури визначаються лише один раз, то і опис шаблону і визначення змінних можна сумістити:

```
struct person
{ char name[20];
int year;
char sex;
} stud1, stud2, stud3;
```

stud1, stud2, stud3; є змінними структурного типу, тобто кожна складається із трьох елементів: прізвища, рік народження та стать.

Оскільки структури належать до складних типів, то для них нема ні однієї операції. Тому у виразах повні імена структурних змінних використовувати НЕ можна.

В арифметичних виразах використовуються лише імена окремих елементів структури. Ім'я елемента складається із двох частин:

< ім'я структури>.< ім'я елемента>

Окремі елементи можуть бути різного типу як простого, так і складного. Допускається використання й елементів структур.

Наприклад, якщо замість року народження впровадити дату: рік, місяць, число, то дата сама є структурою:

```
struct birth
{ int year;
char month[10];
int day;
};
```

Тоді

```
struct person
{ char name[20];
birth date;
char sex; } stud1, stud2, stud3;
```

і ім’я елемента `stud1.date.day=25;`

Як і для масивів, початкові значення можна робити й для автоматичних структур
```
struct person stud1={ “Orlik”, 1972, ‘f’};
```

Самі структури можуть становити інші складні типи даних, наприклад, масиви структур:
```
struct person
{ char name[20];
int year;
char sex;
} student[30];
```
Тоді `student[0].name=”koval”;`

На базі структур і масивів можна створювати досить складні типи даних.

11.1.2 Структурні змінні та покажчики

На відміну від масиву, ім'я структурної змінної НЕ є адресою. Однак в окремих випадках використання покажчиків на структуру дозволяє реалізувати ефективну програму. Наприклад, надрукувати масив даних про студентів:
```
struct person
{ char name[20];
int year;
char sex;
};
main ()
{ struct person stud[30], *stpr;  // stpr – покажчик на структуру
// присвоєння значень всім елементам
printf (“Прізвище  рік  стать\n”);
for (stpr=stud; stpr<stud+30; ++stpr)
printf (“ %s %d %c \n“, (*stpr).name, (*stpr).year, (*stpr).sex);
}
```

Тут для звертання до елементів структур використані імена (*stpr).name. і насправді , при модифікації покажчика на структуру в stpr буде записана адреса наступної структури й звертання до неї можна здійснити за допомогою прямої адресації: ***stpr***. Круглі дужки тут потрібні тому, що операція **"."** має більший пріоритет, ніж *. І якщо дужок не буде, то вийде нісенітниця, тому що ***stpr*** - це покажчик, а не повне ім'я структури.

Бачимо, що звертання до елементів структури вийшло важким.

Допускається й більш просте звертання до елементів структури через покажчик за допомогою спеціальної операції **“->”** *(мінус, більше)*:

stpr->name еквівалентно ***(*stpr).name***

Тому інший варіант:

printf (“ %s %d %c \n“, stpr->name, stpr->year, stpr->sex);

У версії ANSI-C дозволено передавати структури як параметри функцій і передавати структури через ім'я функції. Крім того, для них впроваджена операція присвоєння. Інші арифметичні операції й відносини над структурами не виконуються, тому що в цьому нема сенсу.

Приклад: дії на крапками на площині. Кожна крапка задається своїми координатами (нехай буде цілими). Тоді необхідно впровадити операції над двома крапками, наприклад, додавання:

```
struct point
{ int x;
int y;
};
// додавання двох точок
struct point addpoint (
struct point p1, struct point p2)
{ p1.x+=p2.x
p1.y+=p2.y
 return p1;
}
```

Структури відіграють важливу роль у мові С . З їхньою допомогою можна створювати динамічні структури даних, наприклад, різні списки, дерева й т.п.

Приклад:

```
// Виведення даних, використовуючи масив, покажчик і масив покажчиків на
структуру
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
/ / створення структури
struct mybas
{ char fio[25];
int year;
```

```
char sex;
} peole;
mybas stud3[10]; // масив змінних типу структури
mybas *stud5,*studuk; // покажчики на сруктуру

void main(void)
{ int a,i;
clrscr();
cout<<"Масив структури"<<endl;
for(a=0;a<10;a++) {
for(i=0;i<10;i++)   stud3[a].fio[i]=random(26)+65;   stud3[a].fio[i]='\0';
stud3[a].year=random(3000);
stud3[a].sex=random(2)?'m':'f';
}
for(a=0;a<10;a++) cout<<a+1<<"->"<<stud3[a].fio<<" "<<stud3[a].year<<"
"<<stud3[a].sex<<endl;
getch();
// ----------------------------------------
cout<<" Покажчик "<<endl;
stud5=new mybas; // виділення пам'яті
for(a=0;a<10;a++) {
for(i=0;i<10;i++)   (*stud5).fio[i]=random(26)+65;   stud5->fio[i]='\0';
stud5->year=random(3000);
stud5->sex=random(2)?'m':'f';
cout<<a+1<<" -> "<<stud5->fio<<"   "<<stud5->year<<"   "<<stud5->sex<<endl;
}
delete(stud5);
getch();
//--------------------------------------------------
cout<<"Масив покажчиків "<<endl;
studuk=new mybas; // виділення пам'яті !!!!
for(a=0;a<10;a++) {
for(i=0;i<10;i++)   studuk[a].fio[i]=random(26)+65;
studuk[a].fio[i]='\0';
studuk[a].year=random(3000);
studuk[a].sex=random(2)?'m':'f';
}
for(a=0;a<10;a++) cout<<a+1<<"->"<<studuk[a].fio<<" "<<studuk[a].year<<"
```

```
"<<studuk[a].sex<<endl;
getch(); delete(studuk);
}
```

11.1.3 Поля

У деяких завданнях аналізу й класифікації шифр об'єкта повинен відображати наявність або відсутність певної якості. Наприклад, при синтаксичному аналізі таблиць імен змінних треба вирішувати, до якого класу пам'яті належить змінна: автоматичного, статичного або зовнішнього. Звичайно, можна було б використовувати структуру з такими трьома елементами, а при наявності такої властивості записувати 1, при відсутності 0.

При великій кількості змінних це приведе до нераціонального використання пам'яті. Тому краще під кожну таку ознаку відвести один біт і кодувати ці ознаки ступенем числа 2.

```
#define KEYWORD 01
#define EXTER 02
#define STATIC 04
#define AUTOM 08
```

Якщо необхідну ознаку занести до якоїсь змінної, то можна тоді використовувати порозрядні логічні операції:

```
int flag=3; // 0000 0011₂
flag=flag | STATIC;  або  flag |=STATIC
//  0000 0011.
//+
//  0000 0100
-----------------
//  0000 0111
```

Після цього в розряді №3 з'явиться одиниця. Тут незручність і, що для маніпуляції з окремими бітами потрібно підбирати спеціальну маску.

Тому для подібних завдань існує спеціальний тип даних - поля. Це тип є різновидом структури, і його опис має вигляд:

struct
{unsigned <ідентифікатор 1>: <довжина-поля 1>;
unsigned <ідентифікатор 2>: <довжина-поля 2>;
. . . . . .

} ім'я змінної;

Від шаблону звичайної структури поле відрізняється наявністю в кожному елементі :

№ - константи без знаку.

Ці константи визначають, скільки біт відводиться на відповідне поле.

Після цього кожне поле має своє ім'я і з ним можна оперувати як із звичайною змінною:

flag.stat=1 або if (flag.keyword==0 && flag.auto==1) оператор;

кількість біт під те чи інше поле може бути різною

```
struct { unsigned cod1 : 2;
unsigned cod2 : 4;
unsigned cod3 : 8;
} pcode;
```

Між окремими полями можна робити пробіли

```
struct { unsigned cod1 : 2;
unsigned cod2 : 4;
 : 2;
unsigned cod3 : 8;
} pncode;
```

1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768
0	0	0	0	0	0			0	0	0	0	0	0	0	0
cod1		cod2						cod3							

Тут між другим і третім полем пропущено 2 біта. Якщо всі поля не містяться в одне ціле, то переносяться в наступне. Але так, щоб поле не розривалося, а вирівнювалося по цілому.

Порожнє поле із цифрою 0 означає перехід до наступного цілого

```
struct { unsigned cod1 : 2;
unsigned cod2 : 4;
: 0;
unsigned cod3 : 8;
} prcode;
```

Третє поле буде записано з початку третього байту.

1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	2	4	8	16	32	64	128		1	2	4	8	16	32	64	128
0	0	0	0	0	0				0	0	0	0	0	0	0	0
cod1		cod2							cod3							
Перший байт									Другий байт							

Приклад:

```
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
struct mybas
{ char fio[25];
int year;
char sex;};
struct mybas stud3[10] ;
struct mybas *stud5,*studuk;
 // основна функція
void main(void)
{ int a,i;
clrscr();
cout<<"Масив структури"<<endl;
for(a=0;a<10;a++) {
for(i=0;i<10;i++)   stud3[a].fio[i]=random(26)+65;
stud3[a].fio[i]='\0';
stud3[a].year=random(3000);
stud3[a].sex=random(2)?'m':'f';
}
for(a=0;a<10;a++) cout<<a+1<<" -> "<<stud3[a].fio<<"   "<<stud3[a].year<<"
"<<stud3[a].sex<<endl;  getch();
// ----------------------------------------
cout<<" Покажчик "<<endl;
//  stud5=stud3;
stud5=(struct mybas*) calloc(10,sizeof(struct mybas));
for(a=0;a<10;a++) {
for(i=0;i<10;i++)   (*stud5).fio[i]=random(26)+65;
stud5->fio[i]='\0';
stud5->year=random(3000);
stud5->sex=random(2)?'m':'f';
cout<<a+1<<" -> "<<stud5->fio<<"    "<<stud5->year<<"    "<<stud5->sex<<endl; }
```

```
getch();
// ----------------------------------------
cout<<" Масив покажчиків "<<endl;
studuk=(struct mybas*) calloc(10,sizeof(struct mybas));
for(a=0;a<10;a++) {
for(i=0;i<10;i++)  studuk[a].fio[i]=random(26)+65;
studuk[a].fio[i]='\0';
studuk[a].year=random(3000);
studuk[a].sex=random(2)?'m':'f';
}
for(a=0;a<10;a++)  cout<<a+1<<"  ->  "<<studuk[a].fio<<"    "<<studuk[a].year<<"
"<<studuk[a].sex<<endl;
getch();
free(studuk);
}
```

11.1.4 Об’єднання

Інколи необхідно зберігати дані різних типів в одному й тому ж місці пам'яті. Наприклад, створити таблицю, елементами якої будуть цілі, дійсні числа та символи. Значення можуть з’являтись у довільному порядку.

Для таких випадків і передбачені об'єднання, опис яких подібних до структур і має вигляд:

union ім’я
{ <тип> <змінна>; …..;
} < змінна >;

Об'єднання можна розглядати як структуру, елементи якої мають 0-зміщення в пам'яті. Звертання до елементів об'єднання таке ж, як і до структур.

Поточне значення елемента об'єднання втрачається після того, як іншому елементові буде присвоєно значення

```
union uname
{ int digit;
double digf;
char letter;
} fit;
fit.digit=12;
```

```
fit.digf=2.56;
fit.letter='a';
```

Зрозуміло, що всі ці значення записуються з того ж самого місця в пам'яті. Тому об'єднання має своїм значенням останнє значення.

Транслятор НЕ контролює, якого типу було останнє значення. Це повинен робити сам розробник (програміст).

Як і для структур, з об'єднаннями можна працювати за допомогою покажчиків:

```
union uname *prf;
prf -> digf=5.61;
```

Об'єднання можуть входити в структури масивів і навпаки.
Наприклад:

```
struct
{   char *name;
int flag;
union
{ int digit;
double digf;
char *letter;
} fit;
} symt[N];
```

symt[N] - це масив структур, кожна з яких складається із трьох елементів. Третій елемент є об'єднанням. Тому можна записати:

symt[i].fit.bigf=3.14;

Ініціалізувати об'єднання можна лише першим елементом, тобто в попередньому випадку - цілим значенням.

Тому, крім того, що значення елементів об'єднання записуються на те саме

місце в пам'яті, усі інші властивості їх аналогічні структурам.

Слід зазначити, що реалізація об'єднань і полів залежить від особливостей архітектури ЕОМ. Тому можуть виникати питання перенесення цілих типів на різні ЕОМ.

11.1.5 Визначення типу

На відміну від мови Паскаль, у С поки ще не було можливості створювати нові типи даних, які можуть спростити програму.

Для цього можна використовувати визначення типу:

typedef <тип> <нова назва>

Наприклад, якщо прагнемо в програмі замінити тип float на ім'я REAL, то це можна зробити так:

```
typedef float REAL; // замінити тип float іменем REAL*
```

Далі в програмі можна писати:

```
REAL a, b, c;
```

Фактично нових типів тут не створюється, а надається нова назва існуючим типам. Ясно, що те ж можна зробити за допомогою директиви препроцесора

```
#define REAL float; /*REAL везде будет заменено на float*/
```

Однак існують і певні відмінності:

1. Визначення **typedef** виконується компілятором, а не препроцесором;
2. Функція **typedef** визначає тільки тип даних і не може визначати ім'я константі;
3. Функція **typedef** має більш широкі можливості ніж препроцесор.

Наприклад:

```
typedef char *STRING;
```

Без слова **typedef STRING** – це покажчик на символ, а з ним - це ім'я типу всіх покажчиків на символ.

Тому можна далі записати:

STRING m, mp[20]; // що еквівалентно ***char *m, *mp[20];***

typedef можна вживати й для структур:

```
typedef struct
{ float re;
float im
}; COMPLEX;
```

Тоді **COMPLEX c, d, e;** визначить структуру комплексного типу без

повторень слова **struct.**

11.1.6 Складні імена та покажчики на функції

При відсутності параметрів у дужках потрібно визначати тип **void**

```
float fun(void)
```

Функція може повертати й структуру або запис. Функції не можуть повертати масив або функції, але дозволяється повертати покажчик на масив або функцію.

```
Наприклад:  int matr[10][10];
    char **ptr; // покажчик на покажчик
    int *arr[10]; // масив покажчиків
    int (*vect)[10] // покажчик на масив
```

Для визначення типу даних діють такі правила пріоритетів модифікаторів:

1. чим ближчий модифікатор до імені, тим вищий його пріоритет;
2. для модифікаторів одного рівня **[]** та **()** мають вищий пріоритет за *****;
3. круглі дужки змінюють послідовність і мають вищий пріоритет.

Тому ***int *st[3][5]*** буде означати:

Модифікатори * і [3] перебувають безпосередньо біля імені, тому вони старші за [5]. [3] старше ніж *. Тому ***st[3]*** масив, а ****st[3]*** масив покажчиків на цілий масив з п'яти елементів. Знаючи такі правила можна визначити й покажчики на функції. Вони мають вигляд:

int (*func) (int, int);

Якщо круглі дужки вилучити, то ***int *func (int, int)*** означає функцію, яка передає покажчик на ціле.

Покажчики на функції реалізують процедурні змінні, масиви процедур і дозволяють передавати функції як параметри інших функцій.

Відзначимо деякі особливості роботи з покажчиками на функції: Допустимо наступне оголошення:

```
double (*fun_ptr) (int, int);
proc (int, int);
```

Як відзначалося раніше, покажчик fun_ptr поки ще не посилається на жоодну функцію. Щоб зв'язати його з певною функцією, потрібно привласнити

йому ім'я функції без списку параметрів fun_ptr=proc;

Після цього звертання до функції через покажчик буде мати вигляд

```
(*fun_ptr) (2, 5);
```

Ясно, що конкретний покажчик можна зв'язувати й з іншими функціями відповідного характеру.

Наприклад

```
int difference (int a, int b) { return (a-b);} // функція
int sum (int a, int b) { return (a+b);} // функція
void main (void)
{ int (*fun_ptr) (int, int);
int v1=15, v2=6,par;
fun_ptr=difference;
par=(*fun_ptr) (v1, v2);
printf (“ par=%d \n”, par);
fun_ptr=sum;
par=(*fun_ptr) (v1, v2);
printf (“ par=%d \n”, par);
}
```

Як бачимо покажчик fun_ptr тут реалізує процедурну змінну за аналогією мови Пасакаль. Як і звичайні змінні, покажчики на функції можуть поєднуватися в масиви.

```
int cmp_x (int, int);
int cmp_y (int, int);
int cmp_z (int, int);
int cmp_w (int, int);
int (*fcmp[4]) (int, int)={cmp_x, cmp_y, cmp_z, cmp_w};
```

Звертання до окремих функцій здійснюється як до звичайних елементів масиву:

```
int index=2;
(*fcmp[index]) (arg1, arg2);
```

Використання покажчиків на функції дає можливість задавати різні

функції як фактичні параметри.

Наприклад, є стандартна бібліотечна функція

```
void qsort (base, num, width, compare)
char *base;  // покажчик на перший елемент
int count;     // кількість впорядкованих елементів
int len;       // розмір елемента
int (*func) ();        // покажчик на функцію порівняння
```

Функція порівняння залежить від типу даних і повинна передавати ціле число:

1. <0, якщо перший елемент;
2. <, другого =0 якщо елементи рівні;
3. >0, якщо перший елемент > другого.

У цієї функції два параметри - покажчики на 2 елемента. Зрозуміло, коли порівнюємо рядки, то фактичний параметр:

```
extern int strcmp ();
qsort (base, count, len, strcmp);
void qsort (char *base, int num, int width, int (*compare) ());
```

Сортування дійсного масиву `float f_mas [50];`

Потрібно зробити функцію порівняння у вигляді, необхідному для функції qsort

```
int f_comp (float *x, float *y)
{ return (*x-*y);}
```

Тоді звертання до функції **qsort** буде наступним:

```
qsort (f_mas, size, sizeof (float), f_comp);
```

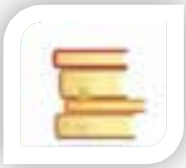
11.2 Лабораторна робота 18. Строки (рядки) закріплення матеріалу
Тема: Строки (рядки)

Мета: вивчення правил описання, вводу-виводу і основних функцій обробки символьних (строкових) даних.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Рядок – це одновимірний масив символів, який закінчується нульовим символом. Як зазначалось, рядок береться у дужки “Скоро весна!”.

Рядок (строку) можна присвоювати певній змінній, якщо остання описана як символьний масив char string[]=”Скоро весна!”.

Такий масив можна вводити та виводити з використанням символу перетворення **s**. При цьому вивід триває до появи нуля.

Ніяких дій над такими масивами не передбачено. Для цього використовують стандартні функції мови (бібліотека string.h). При обробці строк доцільно максимально використовувати бібліотечні функції і макроси.

1. Приєднання рядка (конкатенація) char str1[], str2[]; ***strcat (str1, str2).***

До рядка str1 приєднується рядок str2. Видаляється Ø після str1. Результат присвоюється str1. При цьому str2 не змінюється.

2. Порівняння двох рядків ***strcmp (str1, str2)***

strcmp (str1, str2)=str1-str2.

strcmpi (str1, str2) та ***stricmp (str1, str2)*** – вважає малі й великі букви однаковими Значення цієї функції менше Ø, якщо лексикографічно str1 раніше str2, дорівнює Ø, якщо вони співпадають та більше Ø – в іншому разі.

3. Копіювання рядка

strcpy (str1, str2)

strcpy (str1, str2) рядок str2 копіюється в str1, str2 не змінюється.

4. Визначення кількості символів у рядку без обліку заключного нуля ***strlen***

(str1)

strlen ("abcde") - 5.

5. Перетворення рядка str в число подвійної точності

double atof(char *str).

Розпізнає символьне представлення числа із плаваючою крапкою, якщо символи відповідають формату: [пробіли] [знак] [ddd] [.] [ddd] [e|e[знак]ddd], де [ddd] - числа;[e|e] - покажчик показника ступеня;

Припиняє перетворення на першому неопізнаному символі. У випадку переповнення ***atof*** повертає +(-)HUGE_VAL, глобальна змінна errno установлюється в ERANGE.

6. Перетворення рядка str у число подвійної точності без втрати значення (бібліотека <stdlib.h>):

strtod(***str1,&str2***).

7. Перетворення рядка str у десяткове ціле (бібліотека stdlib.h)

atoi (str1).

Розпізнає символьне представлення десяткового числа, якщо символи відповідають формату: [пробіли] [знак] [ddd], де [ddd] – числа.

Припиняє перетворення на першому не розпізнаному символі.

У випадку переповнення atoi повертає +(-)HUGE_VAL, глобальна змінна errno встановлюється в ERANGE.

8. Перетворення рядка str у десяткове довге ціле

strtol (*const char* ****str1,*** *char* *****error,*** *int* ***base).***

Перетворює рядки ***str1*** в довге ціле відповідності з указаним **base**, яке повинне знаходитися в діапазоні: 2- 36 включно або бути рівним нулю.

9. Перетворення цілого числа в рядок

itoa (int v, char *str, int baz).

Практична частина

Завдання

1. Вибрати завдання, що відповідає номеру варіанту.

2. Запропонувати алгоритм обробки заданої строки у відповідності до завдання.

3. Розробити програму, яка виконує:

– ввод початкової строки (початкова строка задається довільно);

– вивод початкової строки;

– обробка строки у відповідності з завданням;

– вивод результатів обробки з відповідними коментарями.

Варіанти завдань

1. У довільному тексті знайти й надрукувати усі слова, які починаються з літери *a*.

2. У довільному тексті знайти і надрукувати всі слова, які закінчуються буквою *r*.

3. У довільній строчці *S* підрахувати кількість входжень підстроки *S1*.

4. У довільному тексті вставити між першим і другим словом нове слово.

5. Визначити кількість символів у самому довгому слові строки. Слова визначаються знаком «пробіл».

6. У довільній строці *S* змінити усі входження підстрок *S1* на підстроки *S2*.

7. У довільному тексті знайти саме коротке слово.

8. У довільному слові визначити кількість символів, яка розташована у круглих дужках.

9. Вивести на екран друге і четверте слова довільного тексту.

10. У довільному тексті знайти та надрукувати слова, які містять букву *e*, але не містять букву *w*.

11. У довільному тексті вставити між другим і третім словом нове слово.

12. У довільному тексті знайти та надрукувати усі слова довжиною 5 символів.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   - 3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   - 3.2 Написання програм, відналагодження програми.
   - 3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке С-строка? Чим вона відрізняється від масиву символів?
2. Що таке нуль -термінатор?
3. Чим відрізняється описання *char *st* от *char st[N]*, де N – декотра константа?
4. Як ініціалізувати строку?
5. Чим відрізняється ввод строки з допомогою функції *scanf()* від вводу з допомогою функції *gets()*?
6. Як знайти довжину строки?
7. Чи можна використовувати операцію присвоювання для завдання значення строки?
8. Що таке конкатенація строк?
9. Які існують функції для роботи зі строками?

11.3 Самостійна робота 1. Робота зі строковими масивами

Тема: Робота зі строковими масивами

Мета: Закріпити навички роботи зі строковими масивами.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, Draw io, https://www.onlinegdb.com.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом (лекції, додаткові матеріали до лекцій та перелік літературних джерел).
2. Виконати практичну частину.

2.1.Записати своє прізвище ім'я та по батькові. Скориставшись основною та розширеною таблицями ASCII, записати масиви шістнадцяткових і десяткових кодів символів власних даних.

2.2. Записати своє прізвище ім'я та по батькові англійськими літерами. Скориставшись основною та розширеною таблицями ASCII, записати масиви шістнадцяткових і десяткових кодів символів власних даних.

2.3. Для двох рядків здійснити дії:

2.3.1 прикріплення рядків (конкатенація);

2.3.2 порівняння двох рядків та порівняння двох рядків без врахування реєстру символів;

2.3.3 копіювання рядків;

2.3.4 визначити кількість символів в рядку для української та англійської транслітерацій;

2.4 Результати навести з поясненнями визначення кодів символів.

3. Оформити текстовий файл та прикріпити у відповідному розділі Єльорну.

Форма подання результатів виконаної роботи:

Звіт з самостійної роботи містить титульний лист, зміст з автоматичною нумерацією сторінок, хід виконання роботи з зазначенням завдання, підстановкою значень, отримані розрахункові значення, виконані індивідуальні варіанти завдання.

Кожне завдання містить: постановку задачі, математичну модель та опис алгоритму задачі, блок-схему алгоритму задачі, текст програми, тестування. Обов'язковими є висновок, який містить аналіз отриманих в результаті роботи результатів та відповіді на контрольні питання.

Звіт у форматі .docx, збережений транслітерацією прізвища та номеру роботи виставляється у зазначений ресурс у відповідну лабораторну роботу.

Перед здачею звіту обов'язкова демонстрація та тестування роботи розроблених програм викладачу, корекція звіту за результатами демонстрації, а вже потім здача лабораторних робіт у Єлерн.

Критерії оцінювання:

максимальна кількість балів - 15 балів, з них:

- створення блок-схем алгоритмів, математичних моделей та тестування - 3 бали;
- написання програм, відналагодження програми - 3 бали;
- демонстрація роботи програми та усунення зауважень – 3 бали;.
- відповіді на контрольні питання - 3 бали.
- грамотне оформлення звіту за вимогами – 3 бали.

РОЗДІЛ XII. ДИРЕКТИВИ ПРЕПРОЦЕСОРА

12.1 Теоретичні відомості. Директиви препроцесора

12.1.1 Директиви препроцесора. Загальні відомості

Препроцесор С або текстовий процесор використовується для обробки тексту вихідного файлу на першій фазі компіляції. Директиви препроцесора відзначаються спеціальним знаком # (номера), який повинен бути в першій позиції відповідного рядка. ***Директиви препроцесора можуть розміщатися в будь-якому місці вихідного файлу, але діють тільки для частини програми, розташованої нижче директиви.***

Розглянемо деякі з них.

Директива #define має дві форми:

 # define ім'я текст_підстановки
 # define ім'я (список параметрів) текст_підстановки

Перша форма використовується, щоб зв'язати ім'я із часто використовуваними константами, ключовими словами, операторами й виразами:

```
#define BETA 3.56;
#define CUBE(x) ((x)*(x)*(x))
```

Імена, які заміняють константу, називаємо символічними константами, а імена, які пов'язані з операторами або виразами - макрокомандами. У тексті вихідного файлу, який іде слідом за #define всі імена заміняються на підстановку тексту. Якщо відповідне ім'я є частиною іншого, або частиною рядка, то заміна не виконується.

Якщо текст підстановки опущений, то з тексту вихідного файлу видаляється всі відповідні імена.

Корисне застосування **#define** для довгих констант, які використовуються кілька раз для звичних логічних операцій:

```
#define PI 3.141592
#define EQL ==
#define AND &&
#define OR ||
```

або для машинно-залежних констант:

```
#define INT_MAX 32767
#define INT_MIN –32768
```

Друга форма директиви зі списком параметрів передбачає, що кожне звертання до цьому імені зі списком аргументів заміняється на текст підстановки, де формальні параметри заміщаються фактичними:

```
#define ABS (x) ((x) <0 ? –(x): (x))
```

Після підстановки макрокоманди ***ABS (v)*** отримаємо підстановку ***v<0 ? – v : v***.

Коли текст підстановки не вміщається в одному рядку, для переносу можна використовувати зворотну дробову риску «\»

```
#define ZERO_ARRAY (array, size)
{ int i=0;  \
while (i<size)  \
array[i++]=0;  \
}  \
```

Звертання в програмі:

```
#define MAX 50
main ()
{ int values[MAX];
ZERO_ARRAY (values, MAX);
}
```

- це макрокоманда.

Така макропідстановка нагадує звертання до звичайної функції. Але відміна полягає в тому, що відбувається лише текстова підстановка.

Наприклад:

ABS (x+z) дає (x+z) <0 ? –(x+z): (x+z).

Перевага такої підстановки полягає в тому, що вона не залежить від типу змінних. У той час, як для функції потрібно вказувати їхній тип.

Враховуючи особливості такої підстановки, для правильної реалізації потрібно використовувати круглі дужки:

Наприклад `#define sqr (x)  (x)*(x)`

Для директиви ***#define*** з параметрами можна використовувати дві препроцесорні операції:

1. Операцію ***створення*** рядка, який відбивається одним знаком #,
2. Операцію ***об'єднання*** імен, яка відображається ##.

Якщо перед формальним параметром розміщений символ #, то в результаті підстановки на цьому місці буде розміщений аргумент у подвійних лапках.

Наприклад:

#define print (expr) printf (# expr “=%f \n”, expr).

При звертанні print (y/z); буде текст printf (“y/z” “=%f \n”, y/z);

Операція ## приведе до об'єднання двох аргументів в одне ім'я: якщо у визначенні використане arg1 ## arg2, то в тексті з фактичними параметрами n1, n2 буде одне ім'я **n1n2.**

Наприклад:

#define unite (arg1, arg2) arg1 ## arg2; і звертання unite (name, 2) створить ім’я name2. Визначення директиви **#define** може бути скасоване директивою **#undef <і’мя>** Це дає можливість у наступному тексті програми використовувати певне ім'я в іншому розумінні.

Наприклад:

#define NIL (char *) 0

#undef NIL

#define NIL (float *) 0

Тут спочатку ім'я **NIL** заміняється нульовим покажчиком на символ, потім визначення відміняється й далі це ім'я означає нульовий покажчик на дійсне.

Директива може скасовувати обидві форми директиви ***#define***. При цьому для другої форми наводити параметри не потрібно.

12.1.2 Директиви умовної компіляції

Це директиви *#if, #elif, #else, #endif*, які дозволяють не компілювати окремі частини вихідного файлу після перевірки константних виразів. Програма з директивами умовної компіляції має вигляд:

#if константний вираз
[текст програми]
#elif константний вираз
[текст програми]
.....
#else
[текст програми]
#endif

умовно компілюємий текст програми починається директивою ***#if*** і закінчується обов'язково директивою ***#endif***.

Між цими директивами може бути розміщено кілька директив ***#elif*** (це else - if) і одна директива ***#else.***

Константні вирази НЕ можуть містити операції sizeof, перетворення типу

й констант перерахування.

```
#define SIZE 16
#include stdio.h
main()
{
char c='A';
#if SIZE==16
int x=123;
printf("x=%d\n",x);
#else
static char x[SIZE]="інформатика";
printf("x=%s\n",x);
#endif
printf("%c\n",c);
}
```

Умовну компіляцію можна застосувати для програм, призначених для використання на різних ЕОМ та в різних ОС.

Існують і інші директиви препроцесора й деякі деталі виконання розглянутих директив, про які можна довідатися з довідкової літератури.

12.1.3 Директиви компілятора або прагми

Це інструкції компіляторові, які розміщаються в певних місцях програми й використовуються для керування діями компілятора, не впливаючи на програму в цілому. Формат директиви**:**

#pragma <послідовність символів>

Основні директиви**:**

#pragma check_pointer ([on/off]) встановлює контроль покажчиків або знімає для певної функції**;**

#pragma check_stack ([on/off]) - встановлює або забороняє контроль переповнення стека;

#pragma loop_opt ([on/off]) - знімає або встановлює оптимізацію циклів окремих функцій;

#pragma message (рядок повідомлень) – посилає рядок повідомлення в stdout без припинення компіляції;

#pragma pack ([1/2/4]) – для певних структур змінює спосіб упакування

елементів;

#pragma same_seg (змінна1, змінна2...) - усі наведені змінні розподіляються в тому самому сегменті даних.

Конкретний перелік наявних директив залежить від транслятора. Компілятор інтегрованої системи ВС сприймає наступну директиву: **check_pointer, check_stack, message, pack.**

12.2 Лабораторна робота 19. Робота зі строками (закріплення навичок роботи зі строками. частина 2)

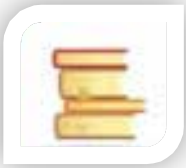
Тема: Робота зі строками (закріплення навичок роботи зі строками. Частина 2)

Мета: Закріпити навички роботи зі строками.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Дивись лабораторна робота 18.

Практична частина

Завдання

Написати вісім програм згідно індивідуальному варіанту. Уважно читати що слід зробити. Ввод елементів строки здійснювати з клавіатури. Під час відладки і тестування програми розмір строки можна зменшити.

Варіанти завдань

1. Знайти частоту появи символу у строці.
2. Підрахувати кількість голосних, приголосних, цифр та пробілів у строці, яку введено користувачем.
3. Введене користувачем речення перевернути з допомогою рекурсії.
4. Знайти довжину рядка вручну, не використовуючи функцію strlen().
5. Об'єднати дві строки вручну, не використовуючи функцію strcat().
6. Скопіювати строки без використання функції strcpy().
7. Видалити символи зі строки, окрім алфавітних.
8. Відсортувати елементи у лексикографічному порядку (за словником).

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
    3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
    3.2 Написання програм, відналагодження програми.
    3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке С-строка? Чим вона відрізняється від масиву символів?
2. Що таке нуль -термінатор?
3. Чим відрізняється описання *char *st* от *char st[N]*, де N – декотра константа?
4. Як ініціалізувати строку?
5. Чим відрізняється ввод строки з допомогою функції *scanf()* від вводу з допомогою функції *gets()*?
6. Як знайти довжину строки?
7. Чи можна використовувати операцію присвоювання для завдання значення строки?
8. Що таке конкатенація строк?
9. Які існують функції для роботи зі строками?

РОЗДІЛ XIII. ФАЙЛИ

13.1 Теоретичні відомості. Файли

13.1.1 Особливості файлів мови С

Як і в інших мовах, для роботи із зовнішніми обладнаннями пам'яті в мові С використовують файли.

Кількість елементів у файлі не фіксується. Кінець файлу зв'язується з іменованою константою End Of File (EOF).(-1)

Однак, на відміну від мови Паскаль внутрішня структура файлу не визначається - які саме елементи становлять файл. Вважається, що файл складається з послідовності байтів. А що саме являють собою ці байти і як їх коректно використовувати - це лежить на відповідальності програміста.

Такі поняття як "вікно" файлу, "файлова змінна" - не вживаються. Формально файл задається покажчиком

FILE *iot; // FILE в верхньому регістрі

Однак, якщо в мові Паскаль ім'я FILE є іменем типу даних: TYPE FIL=FILE OF REAL; то в мові С це не так.

Точніше кажучи **FILE** - це ім'я типу структури, де описуються властивості певного файлу. Тобто **FILE** - це тип структури у файлі stdio.h з використанням типу typedef:

```
typedef struct
{ short level;  // fill/empty level of buffer - рівень буфера
unsigned flags; // FILE status flags - прапорці статусу файла
char fd; // FILE descriptor - дескриптор файла
unsigned char hold; // ungets char if no buffer - попередній символ, якщо нема буфера
short bsize;  //buffer size - розмір буферу
unsigned char *buffer; // Data transfer buffer - буфер передачі даних
unsigned char *curp; // Current active pointer -  поточний активний покажчик
unsigned istemp; // Temporary file indicator - тимчасовий індикатор файлу
short token; // Used for validity - для перевірки коректності
} FILE
```

Тобто **iot** є покажчиком на структуру.

Звідси зрозуміло, що робота з файлами організована засобами операційної системи, а тип структури **FILE** є певною перехідною ланкою.

У мові С немає власних засобів роботи з файлами, а всі дії реалізуються за

допомогою функцій бібліотек С.

Ці функції поділяються на три класи:

1. верхнього рівня (з використанням терміну “потік”);

```
#include <fstream.h>
void main(void)
{
ofstream book_file("BOOKINFO.DAT");
book_file << "Вчимося писати програми мовою С," << "друга редакція" << endl;
book_file << "Jamsa Press" << endl;
book_file << "22.95" << endl; }
```

2. для консольного терміналу та порту з безпосереднім звертанням до них;

void main(char argc, char *argv[])

3. Нижнього рівня (з використанням терміну "дескриптор");

13.1.2 Поточний обмін даними

Усі дані можна розглядати як послідовність окремих байтів, або потік. Для користувача потік - це або файл на диску, або фізичне обладнання (дисплей або принтер).

Хоча потік є послідовністю окремих байтів, функції обміну для потоку дозволяють обробляти дані різного розміру й формату (від одного символу до великих структур). При цьому здійснюється форматний або безформатний обмін в буфері.

13.1.3 Відкриття потоків

Для виконання операцій з файлом його попередньо необхідно відкрити. Для цього використовується функція ***fopen ().*** Її параметрами є два рядки

```
char *name; // це ім'я файлу, наприклад «D: \ prog.c, prn»
char *mode; // режими використання файлу
```

fopen (name, mode)

Існує три режими доступу до файлу:

1. Читання (read):

"r"- відкрити файл для читання, файл повинен існувати;

"r+" - відкрити файл одночасно для читання й запису, файл повинен існувати;

2. Запис (write):

"w" - відкрити порожній файл для запису, якщо файл існує, він стирається;

"w+" - відкрити порожній файл для читання й запису, якщо файл існує, він стирається;

3. Доповнення (append):

"a"- відкрити файл для читання й додавання починаючи з кінця, якщо файлу немає він створюється для читання або запису (курсор наприкінці файлу).

Відзначимо, що коли файл відкривається для запису, то весь його зміст стає недоступним, тобто в логічному розумінні втрачається. При доповненні зміст файлу зберігається й покажчик встановлюється на кінець файлу.

Оскільки параметри **name** і **mode** є рядками, то вони беруться в лапки:

```
FILE *iot;
iot=fopen (“data.txt”, “r+”)
```

Функція **fopen** повертає покажчик на структуру **FILE,** яка описує даний відкритий файл.

Однак іноді, наприклад, при спробі відкрити неіснуючий файл для читання, функція **fopen** *цього зробити не може*. Тому, якщо файл не вдалося відкрити, функція **fopen** повертає значення ***NULL***, яке визначалося в stdio.h як 0, крім того, у глобальну змінну **errno** буде записано код помилки.

Ось чому в програмах роботи з файлами бажано перевіряти умову, чи відкрився файл, чи ні? Наприклад, так:

```
if ((iot=fopen (“data.txt”, “r+”)) ! =NULL
```

Існує два типи файлів:

t - відкрити в текстовому (перетворюючому) режимі, тобто при введенні комбінація "Повернення каретки - переклад рядка" (ПК- ПР) перетворюється до єдиного символу "перекладу рядка". При виведенні символ перекладу рядка перетворюється в комбінацію ПК-ПР;

b - відкрити у двійковому (неперетворюючому) режимі;

Якщо **t** або **b** у рядку **type** не задається, режим перетворення визначається змінною ***_fmode*** і режимом, встановлюваним за замовчуванням.

13.1.4 Стандартні покажчики потоків

З кожною задачею автоматично пов’язується 5 потоків:

1. стандартного вводу (stdin);
2. стандартного виводу (stdout);

3. стандартного виводу повідомлень про помилки (stderr);
4. додаткового потоку (stdaux);
5. стандартного потоку (stdprn).

Різновиди 1 – 3 – це потоки, пов’язані з консоллю користувача

Додатковий потік відносять до додаткового порту, до якого можна підключити допоміжну консоль, а стандартний друк – до друкуючого пристрою.

Покажчики **stdin, stdout, stderr, stdaux, stdprn** є константами, а не змінними. Тому їм не можна присвоювати інші значення покажчиків потоків.

Проте кожний з цих потоків можна переадресувати на інший пристрій або інший потік за допомогою функції freopen (“file.dat”, “w”, stdout)

Закривається потік для стандартного виводу і ім'я потоку **stdout** закріплюється за файлом file.dat у режимі запису. Після цього вивід у **stdout** означає виведення до файлу file.dat.

13.1.5 Закриття потоків

Закриття потоків здійснюється для:

- окремого потоку функцією **fclose (iot);**
- всіх потоків - функцією **fcloseall().**

Остання функція НЕ закриває стандартні 5 потоків.

При закритті потоків звільнюються всі буфери, ліквідуються покажчики на файл і, якщо потрібно, готується файл для зберігання.

Якщо потоки не закриваються явно, то вони закриваються автоматично при завершенні програми. Оскільки кількість відкритих потоків обмежена, то краще потоки в програмі закривати явно.

13.2 Лабораторна робота 20. Робота зі структурами

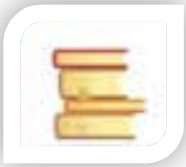
Тема: Робота зі структурами

Мета: Ознайомитись та набути навички роботи із структурами.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Описання структур має такий вигляд:

struct < ім'я>

{ описи елементів } [ім'я змінної, …];

окремий опис елемента це фактично опис змінної – тип та ім'я.

Ім'я структури інакше називається тегом (ярлик, етикетка). Тег дає назву структурі та є стислим позначенням тієї частини структури, яка міститься в дужках.

Опис структури є лише шаблоном, і не супроводжується виділенням місця у пам'яті.

У відповідності з наведеним шаблоном можна визначити змінні:

struct person stud1, stud2, stud3; або person stud1, stud2, stud3;

Зрозуміло, що область дії і шаблону, і змінних залежить від їх розташування в програмі. Якщо опис структури навести до початку функції **main**, то він діятиме на всю програму, тобто матиме зовнішній характер. Якщо ж це зробити у функції після відкриття дужки – автоматичний.

Зазначимо, що слово структура має двоїстий характер. З одного боку його відносять до опису шаблона, а з другого – до змінних-структур.

Подібно до мови Паскаль, окремий опис шаблону надає ширші можливості. На нього можна посилатись у різних функціях, тобто використовувати багаторазово.

Оскільки структури належать до складних типів, то для них нема ні однієї операції. Тому у виразах повні імена структурних змінних використовувати НЕ

можна.

В арифметичних виразах використовуються лише імена окремих елементів структури. Ім'я елемента складається із двох частин:

< ім'я структури>.< ім'я елементу>

Окремі елементи можуть бути різного типу як простого, так і складного. Допускається використання й елементів структур.

На відміну від масиву, ім'я структурної змінної НЕ є адресою. Однак в окремих випадках використання покажчиків на структуру дозволяє реалізувати ефективну програму.

Допускається й більш просте звертання до елементів структури через покажчик за допомогою спеціальної операції ***"->"*** *(мінус, більше)***:**

stpr->name еквівалентно ***(*stpr).name***

У версії ANSI-C дозволено передавати структури як параметри функцій і передавати структури через ім'я функції. Крім того, для них впроваджена операція присвоєння. Інші арифметичні операції й відносини над структурами не виконуються, тому що в цьому нема сенсу.

Практична частина

Завдання

Написати шість програм згідно індивідуальному варіанту. Уважно читати що слід зробити. Ввод елементів здійснювати з клавіатури. Під час відладки і тестування програми кількість елементів стуктури можна зменшити.

1. Розробити програму для зберігання інформації про учнів у структурі і відображення її на екрані.

2. З допомогою структури взяти дві відстані (у системі дюймів-футів), скласти їх і відобразити результат на екрані.

3. Навчитися використовувати два комплексних числа в якості структури і додавати їх, створюючи користувацьку функцію.

4. Розробити програму для обчислення різниці між двома періодами часу з допомогою визначеної користувачем функції.

5. Розробити програму для зберігання інформації про п'ятьох учнів, використовуючи масив структур.

6. Розробити програму для зберігання введеної користувачем інформації з допомогою динамічного виділення пам'яті.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.

3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.

3.2 Написання програм, відналагодження програми.

3.3 Демонстрація роботи програми та усунення зауважень.

4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Що таке структурний тип даних?
2. Які види структур даних існують?
3. Що таке вложені структури?
4. Яким чином можна звернутися до різних полів структури?
5. Як задати масив структур?

РОЗДІЛ XIV. ФУНКЦІЇ ОБМІНУ З ПОТОКАМИ

14.1 Теоретичні відомості. Функції обміну з потоками

14.1.1 Особливості і різновиди функцій обміну з потоками

Робота з файлами здійснюється за допомогою різних бібліотечних функцій.

У загальному випадку у файлах може зберігатися символьна інформація (форматовані файли) і двійкова (неформатовані файли). Тому існують потоки текстові, які складаються із символів, розділених на рядки. Для розподілу на рядки використовується керуючий символ '\n'. Текстові потоки добре переносяться на інші комп'ютери, якщо в них нема спеціальних символів псевдографіки фірми IBM.

Двійкові потоки - це послідовність значень типу **char**. Будь-які дані можна вважати послідовністю символів. Для визначення типу потоку при відкритті файлів додатково вказується буква **t** для текстових і **b** для двійкових потоків.

Наприклад: "r+b".

Вважається, що стандартні потоки **stdin**, **stdout**, **stderr** є текстовими, а **stdaux**, **stdprn** -ні.

Мова С має різноманітні функції обміну з потоками.

Об'єкт операції	Операції обміну					
	Зчитування			Запис		
	Із потоку ***stdin***	Із будь-якого потоку	Із рядку C++	В потік ***stdout***	В будь-який потік	В рядок C++
Послідовність байт		fread			fwrite	
Окремий символ	fgetchar getchar	fgets getchar		fputchar putchar	fputc putc	
Ціле типу int	getch	getw		ungets	fput	
Рядок	gets	fgets		puts	fputs	
Форматовані дані	scanf	fscanf	sscanf	printf vprintf	fprintf vfprintf	sprintf vsprintf

1. Обмін символами Для обміну символами передбачені такі функції:

```
FILE * point;
char ch;
```

```
ch = gets (point); // Зчитати символ з файлу
putc (ch, point); // Записати символ в файл
```

Коли прочитати символ не вдалося (помилка або кінець файлу), то функція повертає -1. Функція **fgets** і **fputc** діють аналогічно функціям **gets**, **putc**, але реалізовані інакше.

2. Обмін рядками

fgets (string, MAXLIN, point) - читання рядка з файлу з покажчиком **point string** - ім'я рядка, куди зчитуються символи рядка; ***MAXLIN*** - максимальна кількість символів, яку потрібно прочитати. ***третій параметр*** - це покажчик point на файл, звідки здійснюється зчитування, Функція ***fgets*** припиняє роботу, якщо прочитане ***MAXLIN-1*** символів, або до зчитування символу закінчення рядка **('\n')**. Повертає ***NULL***, якщо натрапили на кінець файлу. ***Status=fputs(string, point)*** - запис рядка у ***файл.*** Якщо відбулася помилка або кінець файлу, то вертається ***EOF.***

3. Форматний обмін

fscanf (point, "%d", &psi) – зчитування з файлу. Функція передає кількість правильно прочитаних значень, а також EOF, коли файл вичерпано

fprint (point, "psi=%d \n", psi***)*** - запис в файл

У файл **point** записується ціле значення **psi**. Функція передає кількість правильно записаних значень.

Для обміну цілими передбачено дві функції **getw(point)**, **putw(int, point)**;

```
int c;
FILE *point;
c=getw (point);
putw (c, point);
```

Функція **getw** зчитує два байти з потоку **point** і передає змінній с. Функція **putw** записує ціле значення змінної з у потік **point**. Але для різних систем може виникнути питання перенесення. Передбачено дві функції для обміну блоками байтів:

1. **fread (buffer, size, count, stream);**

```
struct mybas
{ char fio[25];
int year;
```

```
char sex;};
mybas stud3[NUM], stud5 ;
FILE *filw,*filr;
void main(void)
{ int i;
fwrite(&stud3[i],sizeof(stud3),1,filw);
fread(&stud5,sizeof(stud5),1,filw);// !!!!!
```

З потоку ***stream*** зчитується count блоків, кожний з яких має розмір ***size***, у буфер ***buffer***. Функція передає дійсну кількість лічених блоків і ніяких форматних перетворень НЕ відбувається.

2. fwrite(buffer, size, count, stream);

З буфера ***buffer*** записується count блоків, кожний з яких розміром ***size*** у потік stream. Функція передає кількість у дійсності записаних блоків. Відзначимо ще дві функції форматного обміну. Функція ***sscanf*** відрізняється від функції ***fscanf*** тим, що зчитування здійснюється з рядка string:

sscanf(string, format_string, list);

Функція sprintf відрізняється від ***fprintf*** тим, яке записує значення змінних не в потік, а в рядок:

sprintf(string, format_string, list).

Отже першу функцію можна використовувати для перетворення символів у внутрішній формат чисел, а другу навпаки - із внутрішнього формату до символьного.

```
char string[10];
float value=3.12;
sprintf (string, “%f”, value);
```

У рядку string одержимо символьний формат числа 3.12 у формі з фіксованою крапкою.

```
char string[10]=”3.12”;
float value;
sscanf (string, “%f” &value);
```

Рядок "3.12" перетвориться у внутрішній формат у змінній value.

Приклад: читати інформацію з файлу цілих, записувати в змінну t і виводити на екран. Алгоритм: якщо файл відкрився, то в циклі зчитувати значення в змінну ***t***, поки функція ***fscanf*** не дорівнює **EOF**, і виводити значення змінної ***t***. Після цього файл закрити. Якщо файл не відкрився, то вивести повідомлення: “ **файл не відкрито** ”:

```
main ()
{ int t;
FILE  *fp;
if (( fp=fopen (“DATA.TXT”, “r” )) ! =NULL)
{ while (fscanf (fp, “%d” & t) ! =EOF)
printf (“число t =%d \n”, t);
fclose (fp);
}
else printf (“файл не відкрито”);
getch ();}
```

14.1.2 Довільний доступ до потоку

У наведених вище прикладах потоки використовувалися як файли послідовного доступу. Операції читання або запису в потік починалися з поточної позиції потоку, яка визначається внутрішнім покажчиком потоку. При відкритті покажчик показує на початок потоку в режимі **"r"** і **"w"**, і на кінець потоку в режимі **"a"**.

Після виконання певної операції обміну покажчик змінюється й д о рівнює новій поточній позиції потоку. Наприклад, якщо з потоку прочитано 1 символ, то покажчик збільшується на 1.

Але існує можливість позиціонувати покажчик на будь-яку позицію в потоці, тобто реалізувати прямий доступ до файлу. Існує п'ять функцій для позиціонування покажчика потоку:

1. **ftell(point)** - передає поточну позицію покажчика потоку:

```
long ipos; ipos=ftell(fp);
```

2. fgetpos(point, long) – передає поточну позицію покажчика потоку, передає 0, якщо визначення відбулося правильно й -1 в іншому випадку:

```
long ipos; fgetpos(fp &ipos);
```

3. fsetpos(point, long) – встановлює відповідну позицію покажчика потоку;

```
long ipos;
fsetpos (fp, &ipos);
```

4. fseek(fp, count, start);

встановлення покажчика на будь-яку позицію покажчика потоку, де **fp** - покажчик на файл; **count** - кількість байт типу **long**, визначає абсолютну або відносну позицію у файлі; **start** - типу **int**, з якої позиції потрібно відраховувати кількість байт **count**:

start	0 – від початку файлу	SEEK_SET
	1 – від поточної позиції	SEEK_CUR
	2 – від кінця файлу	SEEK_END

```
Наприклад:
fseek (fp, 0L, SEEK_SET) // на початок потоку
fseek (fp, n, SEEK_CUR) // на n байт від поточної позиції
fseek (fp,-n, SEEK_CUR) // на n байт до поточної позиції
fseek (fp, 0L, SEEK_END) // на кінець потоку
```

5. ***rewind (fp)*** перейти на початок файлу, аналогічна ***fseek (fp, 0L, SEEK_SET).***

Якщо потік відкрито для додавання (режим **"a"**, **"a+"**), тоді запис завжди здійснюється в кінець файлу. Тобто хоч відповідними функціями можна знайти потрібну позицію, але при спробі запису покажчик попередньо буде встановлений на кінець файлу. Відзначені функції організації прямого доступу не можна застосовувати до стандартних файлів в текстовому форматі.

14.2 Лабораторна робота 21. Робота з файлами

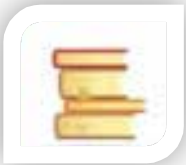
Тема: Робота з файлами

Мета: Ознайомитись та набути навичок роботи з файлами.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Для роботи із зовнішніми обладнаннями пам'яті в мові С використовують файли.

Кількість елементів у файлі не фіксується. Кінець файлу зв'язується з іменованою константою End Of File (EOF).(-1)

Однак, на відміну від мови Паскаль внутрішня структура файлу не визначається - які саме елементи становлять файл. Вважається, що файл складається з послідовності байтів. А що саме являють собою ці байти і як їх коректно використовувати - це лежить на відповідальності програміста.

Такі поняття як "вікно" файлу, "файлова змінна" - не вживаються. Формально файл задається покажчиком.

Точніше кажучи **FILE** - це ім'я типу структури, де описуються властивості певного файлу. Тобто **FILE** - це тип структури у файлі stdio.h з використанням типу typedef.

Тобто **iot** є покажчиком на структуру.

Звідси зрозуміло, що робота з файлами організована засобами операційної системи, а тип структури **FILE** є певною перехідною ланкою.

У мові С++ немає власних засобів роботи з файлами, а всі дії реалізуються за допомогою функцій бібліотек С++.

Ці функції поділяються на три класи:

1. верхнього рівня (з використанням терміну “потік”);
2. для консольного терміналу та порту з безпосереднім звертанням до них;
3. нижнього рівня (з використанням терміну "дескриптор").

Усі дані можна розглядати як послідовність окремих байтів, або потік. Для користувача потік - це або файл на диску, або фізичне обладнання (дисплей або принтер).

Хоча потік є послідовністю окремих байтів, функції обміну для потоку дозволяють обробляти дані різного розміру й формату (від одного символу до великих структур). При цьому здійснюється форматний або безформатний обмін в буфері.

Для виконання операцій з файлом його попередньо необхідно відкрити. Для цього використовується функція ***fopen ().***

Існує три режими доступу до файлу:

1. Читання (read):

"r"- відкрити файл для читання, файл повинен існувати;

"r+" - відкрити файл одночасно для читання й запису, файл повинен існувати;

2. Запис (write):

"w" - відкрити порожній файл для запису, якщо файл існує, він стирається;

"w+" - відкрити порожній файл для читання й запису, якщо файл існує, він стирається;

3. Доповнення (append):

"a"- відкрити файл для читання й додавання починаючи з кінця, якщо файлу немає він створюється для читання або запису (курсор наприкінці файлу).

Відзначимо, що коли файл відкривається для запису, то увесь його зміст стає недоступним, тобто в логічному розумінні втрачається. При доповненні зміст файлу зберігається й покажчик встановлюється на кінець файлу.

Функція **fopen** повертає покажчик на структуру **FILE,** яка описує даний відкритий файл.

Однак іноді, наприклад, при спробі відкрити неіснуючий файл для читання, Функція **fopen** *цього зробити не може*. Тому, якщо файл не вдалося відкрити, функція **fopen** повертає значення ***NULL***, яке визначалося в stdio.h як 0, крім того, у глобальну змінну **errno** буде записано код помилки.

Існує два типи файлів:

t - відкрити в текстовому (перетворюючому) режимі, тобто при введенні комбінація "Повернення каретки - переклад рядка" (ПК- ПР) перетворюється до єдиного символу "перекладу рядка". При виведенні символ перекладу рядка перетворюється в комбінацію ПК-ПР;

b - відкрити у двійковому (неперетворюючому) режимі.

Якщо **t** або **b** у рядку **type** не задається, режим перетворення визначається змінної ***_fmode*** і режимом, установлюваним за замовчуванням.

З кожною задачею автоматично пов’язується 5 потоків:

1. стандартного вводу (stdin);
2. стандартного виводу (stdout);
3. стандартного виводу повідомлень про помилки (stderr); 4. додаткового потоку (stdaux);
5. стандартного потоку (stdprn).

Додатковий потік відносять до додаткового порту, до якого можна підключити допоміжну консоль, а стандартний друк – до друкуючого пристрою.

Покажчики **stdin, stdout, stderr, stdaux, stdprn** є константами, а не змінними. Тому їм не можна присвоювати інші значення покажчиків потоків.

Проте кожний з цих потоків можна переадресувати на інший пристрій або інший потік за допомогою функції freopen (“file.dat”, “w”, stdout).

Закривається потік для стандартного виводу і ім'я потоку **stdout** закріплюється за файлом file.dat у режимі запису. Після цього вивід у **stdout** означає виведення до файлу file.dat.

Закриття потоків здійснюється для:

- окремого потоку функцією **fclose (iot)**
- всіх потоків - функцією **fcloseall().**

Остання функція НЕ закриває стандартні 5 потоків.

При закритті потоків звільнюються всі буфери, ліквідуються покажчики на файл і, якщо потрібно, підготовляється файл для зберігання.

Якщо потоки не закриваються явно, то вони закриваються автоматично при завершенні програми. Оскільки кількість відкритих потоків обмежена, то краще потоки в програмі закривати явно.

Практична частина

Завдання

Написати п’ять програм згідно індивідуальному варіанту. Уважно читати що слід зробити. Ввод елементів здійснювати з клавіатури. Під час відладки і тестування програми кількість елементів структури можна зменшити.

Варіанти завдань

1. Створити текстовий файл та записати у нього число, введене користувачем.
2. Зчитати з текстового файлу число, записане у завданні 1.
3. Створити бінарний файл та структуру з трьох чисел і записати у файл.
4. Працювати з бінарним файлом, зчитуючи записану у завданні 3 структуру.

5. Написати програму для читання імен і оцінок студентів і запису їх у файл.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   3.2 Написання програм, відналагодження програми.
   3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Відкриття і закриття файлів.
2. Читання з текстового файлу і запис в нього.
3. Запис у текстовий файл.
4. Зчитування з двійкового файлу і запис у нього.

РОЗДІЛ XV. КЕРУВАННЯ БУФЕРИЗАЦІЄЮ

15.1 Теоретичні відомості. Керування буферизацією

15.1.1 Функції роботи з буферами

Потоки за замовчуванням є буферизуємі, тобто при відкритті потоку з ним автоматично зв'язується область пам'яті, яку називають буфером. При читанні даних з потоку інформація розміщується у вхідному буфері й дані зчитуються з буфера. Коли весь буфер оброблений, у буфер зчитується наступний блок даних.

При записі вміст вихідного буфера записується у відповідний потік (буфер звільняється), коли буфер заповнений, або закривається відповідний потік, або коли програма успішно завершена.

Це підвищує ефективність програми, тому що обмін реально здійснюється не одним елементом даних, а цілими блоками. Буфери, створювані операційною системою, недоступні для користувача.

Але існує можливість створювати не системні буфери для обміну, або робити обмін небуферизуємим. Це можна здійснити за допомогою функцій:

1. ***void setbuf(stream, buffer);***
2. ***int setvbuf(stream, buf, type, size)***.

0 – виділено;

<>0 – Помилка при виділенні

З цими буферами можна поводитися як зі звичайними змінними.

Функція **setbuf** зв'язує потік з буфером, який повинен бути масивом символів довжиною **BUFSIZE** (ця константа у файлі **stdio.h** і дорівнює 512).

Наприклад**:**

```
#include <stdio.h>
FILE *fp;
char my_buf [BUFSIZE];
fp=fopen (“d:\data.txt”, “r+”);
setbuf (fp, my_buf);
```

Далі потік використовує буфер **my_buf**. Якщо потрібно зробити обмін без буферизаціїі, то замість імені буфера потрібно поставити ненульовий покажчик **NULL**.

```
setbuf (fp, NULL);
```

Функція **setvbuf** дозволяє встановити довільний розмір буфера, але НЕ більше 65535 байт ***(64K).***

Наприклад:

```
FILE *fp;
char *buf;
unsigned n;
setvbuf (fp, buf, _IOFBF, n);
```

Функція **setvbuf** дозволяє користувачеві керувати буферизацією і розміром буфера для потоку **stream**. **Stream** може посилатися на відкритий файл. Масив, на який вказує **buf**, використовується як буфер, якщо він не є **NULL**, тобто потік не є буферизованим. Якщо потік буферизований, використовується тип, вказаний по type; цей тип може бути або **_IONBF**, або **_IOFBF**, або **_IOLBF**. Якщо використовується тип **_IOFBF**, розмір буфера визначається по **size**; якщо використовується тип **_IOLBF** або **_IONBF**, потік є НЕбуферизованим, а **size** і **buf** ігноруються.

_IONBF Буфер НЕ використовується, незважаючи на присутність **size** і **buf**;

_IOFBF ПОВНА буферизація, якщо **buf** не є **NULL**; тому **buf** використовується в якості буфера, а **size** - його розміру;

_IOLBF Аналогічно _IOFBF (строкова буферизація).

```
setvbuf (fp, buf, _IONBF, n);  // не буферизуется
```

За допомогою цієї ж функції можна збільшувати розмір системного буфера: setvbuf (fp, NULL, _IOFBF, n);– розмір збільшується до n байтів.

Коли програма завершується аварійно, то вихідний буфер може бути не вивантаженим, що може привести до втрати інформації.

Для вивантаження буфера можна використовувати функцію: ***int fflush (FILE *stream)***

Якщо заданий потік **stream** відкритий для виводу, то вміст буфера, пов'язаного з потоком **stream** функції **fflush**, записується у відповідний файл. Якщо потік відкритий для введення, то функція **fflush** очищає вміст буфера. Після виклику функції потік залишається відкритим. Для небуферизованого потоку застосування цієї функції не ефективно.

int flushall() **-** функція записує вміст усіх буферів, пов'язаних з відкритими **input** потоками, у відповідні файли. Усі буфери, пов'язані з відкритими потоками, очищаються; наступна операція читання (якщо вона є) зчитує нові дані із вхідних файлів у буфер. Після виклику функції **flushall** усі потоки залишаються відкритими.

Дана функція повертає кількість відкритих потоків (вхідних і вихідних). У випадку помилки значення, що вертається, НЕМАЄ.

Слід зазначити, що потрібно обов'язково закривати файли, робота з якими проводиться через буфери користувача.

15.1.2 Текстовий та двійковий режим обміну

Як відзначалося раніше, можна відкривати потік у текстовому або двійковому режимі. Але ці режими мають певні особливості.

У текстовому режимі при введенні комбінація символів "повернути каретку - перевести строку" (\015 \012) перетвориться в один символ - перевести рядок. При виводі символ "перевести строку" перетвориться у два: \015, \012. Крім того, комбінація клавіш Ctrl+z (0x1a) сприймається як кінець файлу при виводі.

У двійковому режимі перетворення символів ***"ПК - ПС"*** не здійснюється.

Режим обміну з потоком можна змінювати після його відкриття. Для цього призначена функція нижнього рівня ***setmode***. Як відзначалося на рівні потоків функції використовують покажчик потоку.

На нижньому рівні використовується інше поняття - дескриптор файлу ***(handle)***. Це є ціле число, яке зв'язується з файлом і використовується для посилань на даний файл. Так стандартні потоки ***stdin, stdout, stderr, stdaux, stdprn*** мають відповідні дескриптори ***0,1,2,3,4.***

int setmode (int handle, unsigned mode);

(якщо функція повертає 0 - нормально, інакше –1)

handle – дескриптор файла, ***mode*** – режим.

Режим задається двома константами в файле <**fcntl.h**>

O_BINARYдвійковий

O_TEXT текстовий.

Отже, перед використанням функції setmode потрібно одержати дескриптор певного файлу, що й виконує функція

fileno ***(FILE *ptr)***, параметром якої є покажчик файлу.

Отже перехід від текстового до двійкового режиму можна здійснити так**:**

```
# include <stdio.h>
# include <io.h> //  для функції
# include <fcntl.h> // setmode
FILE * ptr;
if (setmode (fileno (ptr), O_BINERY) == 0)
printf ("Двійковий режим встановлено \ n");
```

else prinrf ("Двійковий режим встановити не вдалося \ n");

15.1.3 Інші функції обробки потоків

Для роботи з потоками можна використовувати й деякі інші функції. Для визначення кінця файлу призначена функція:

int feof (FILE *stream), яка передає нуль, якщо досягнуто кінець файлу при спробі прочитати символ, що випливає за останнім. Інакше її значення не нуль.

int ferror (FILE *stream) дозволяє встановити причину помилки читання або запису. Якщо 0 - помилки не було. Якщо трапилася помилка, то внутрішній індикатор помилки файлу буде встановлено доти, поки потік не буде закритим або **rewind**, або **clearrer**.

void rewind (stream), або **void cleareгr(stream**) - яка записує в внутрішній індикатор помилки файлу нуль. Це ж здійснюється й при закритті потоку.

Змінити назву файлу можна за допомогою функції:

int rename (char *oldname, char *newname) і поміняти **oldname** на нову назву **newname**. Якщо така процедура здійснилася успішно, то функція передає нуль. Цю функцію можна використовувати для запису файлу до іншого каталогу, але в межах одного обладнання.

Знищити файл можна за допомогою функції:

int remove (const char *filename). Наступна спроба відкрити файл із таким іменем буде помилкою.

int fstat(handle,buffer) - одержує інформацію про відкритий файл, пов'язаний з даним ***handle***, і запам'ятовує її в структурі, на яку вказує **buffer**.

buffer - структура, тип **stat** якої оголошений в <sys.h \ stat.h>, містить наступні поля:

Поле	Значення
st_mode	Бітова маска для інформації про режим файлу. Біт **S_IFCHR** встановлюється, якщо handle посилається на обладнання. Біт **S_IFREG** встановлюється, якщо handle посилається на звичайний файл.
st_dev	Номер обладнання диска, що містить файл, або **handle** - у випадку іншого обладнання.
st_rdev	Номер обладнання диска, щомістить файл, або **handle** - у випадку іншого обладнання (аналогічно **st_dev**).
st_nlink	Завжди 1.
st_size	Розмір файла в байтах.

st_atime	Час останньої модифікації файла.
st_mtime	Час останньої модифікації файла. (аналогічно **st_atime**).
st_ctime	Час останньої модифікації файла. (аналогічно **st_atime** та **st_mtime**).

Приклад.

```
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <stdio.h>
stat buf;
int fh, result;
fh=open("tmp/data", O_RDONLY); // дескриптор файлу
result=fstat(fh,&buf);
if (result==0)
printf("file size is %ld\n",buf.st_size);
```

15.2 Лабораторна робота 22. Використання потокових функцій для роботи з текстовими та бінарними файлами

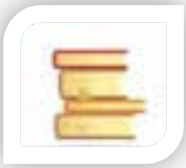
Тема: Використання потокових функцій для роботи з текстовими та бінарними файлами

Мета: Познайомитися з потоковими функціями мови С для роботи з текстовими та бінарними файлами.

Хмарні сервіси та інтернет-ресурси: Gmail, Google Drive, Google Docs, **Draw io, https://www.onlinegdb.com**.

Хід роботи:

1. Ознайомитися з теоретичним матеріалом.
2. Виконати практичну частину.



Теоретичні відомості

Дивись попередню лабораторну роботу.

Практична частина

Завдання

Написати програми згідно номеру індивідуального варіанту. Вихідні текстові файли можуть створюватись у будь-якому текстовому редакторі з використанням кодової сторінки, яка дозволяє безпосередньо обробляти у консольному додатку українські літери. Для створення початкового бінарного файлу до третьої задачі написати окрему програму, в програмі його обробки виводити на екран комп'ютера зміст файлу до та після зміни. Четверте завдання передбачає створення інформаційно-довідникової системи на базі бінарного файлу записів з наступними можливостями: створення файлу, перегляд змісту файлу, додавання, видалення і коректування даних, а також виконання запитів у відповідності до завдання. Пошук потрібних даних здійснювати за ключовим полем. Для організації інтерфейсу повинно використовуватись меню.

Завдання можуть бути виконані на трьох рівнях складності.

1) Низький. Початковий файл до першої задачі не містить українських літер, кожна фраза розташована на окремому рядку, словами вважаються групи символів між групами пропусків. Перший рядок вихідного файлу до другого завдання, якщо в ньому зберігається матриця, містить її розміри (кількість рядків та кількість чисел у кожному рядку). Виведення вмісту бінарних файлів на екран можна виконувати у будь-якому (головне, читабельному) вигляді.

2) Середній. Імена вхідних файлів повинні передаватися програмі під час її запуску (через параметри функції main()). Початковий файл до першого завдання може містити як латинські, так і українські літери, на одній строчці (рядку) може бути кілька фраз, можливе продовження фрази на наступному рядку. Фрази відокремлюються один від одної крапками, а слова – пробілами і розділовими знаками. Остання фраза у файлі може бути без крапки в кінці. Виведення вмісту файлу записів здійснювати у табличному вигляді з графленням візуально відповідними символами.

3) Високий. Імена вхідних файлів повинні передаватися програмі під час її запуску (через параметри функції main()). Якщо параметри користувача під час запуску програми не задані, імена файлів вводяться з клавіатури. Вихідний файл до першої задачі може містити як латинські, так і російські літери, фрази можуть бути будь-якої довжини, відповідно, одна фраза може розташовуватися на кількох рядках. Фрази відокремлюються одна від одної крапками, а слова – пробілами і розділовими знаками. Остання фраза у файлі може бути без крапки в кінці. Вивод вмісту файлу записів здійснювати посторінково у табличному вигляді з графленням візуально відповідними символами, передбачити можливість «гортання» сторінок як у прямому, так і у зворотному напрямку

Варіанти завдань

Варіант 1

1. Дан файл, який містить певний текст. Видалити з цього файлу всі фрази, які містять слово «мама».

2. У текстовому файлі зберігається цілочисельна матриця. Замінити в ній усі числа, які кратні 7, найбільшими значеннями матриці. Отриманий файл вивести на екран.

3. Компоненти бінарного файлу – дійсні числа. Змінити знак у кожному третьому числі на протилежний.

4. Дан файл, який містить відомості про асортимент іграшок у магазині. Структура запису: назва іграшки, ціна, кількість, вікові обмеження, наприклад від 2 до 5. Вивести назви іграшок, які підходять дітям певного віку і коштують не більше певної суми. Отримати відомості про самий вартісний конструктор.

Варіант № 2

1. Дан файл, який містить певний текст. Видалити з файлу всі фрази, які закінчуються та починаються на одну й ту ж літеру.

2. У текстовому файлі міститься цілочисельна матриця. Визначити кількість простих чисел у кожному рядку матриці і результати записати у новий текстовий файл із вказанням номерів строк вихідного файлу.

3. Компоненти бінарного файлу – дійсні числа. Поміняти місцями перший компонент файлу з мінімальним, а останній – з максимальним.

4. У файлі містяться відомості про спортсменів: прізвище, стать, вид спорту, рік народження, зріст. Знайти самого високого спортсмена, який займається плаванням, серед чоловіків. Вивести відомості про спортсменів, які виступають у юніорському розряді (14-17 років).

Варіант № 3

1. Дан файл, який містить певний текст. У кожній фразі знайти саме довге слово і записати його у другому текстовому файлі.

2. У текстовому файлі зберігається цілочисельна матриця. Замінити у ній усі від'ємні елементи мінімальним елементом, а позитивні – максимальним, а нульові – різницею максимального і мінімального елементів.

3. Бінарний файл містить дійсні числа. Видалити від'ємні, у кінці файлу записати кількість видалень.

4. Є файл, який містить відомості про наявність білетів на рейси певної авіакомпанії. Структура запису: номер рейсу, пункт призначення, час вильоту, ціна квитка, кількість вільних місць у салоні. Здійснити корегування даних у файлі при продажі квитків, вихідні дані – номер рейсу і кількість проданих квитків. Отримати відомості про наявність місць, ціні квитка і час відльоту для певного рейсу.

Варіант № 4

1. Дан файл, який містить певний текст. Видалити з цього файлу зайві пробіли, залишивши по одному між словами. Якщо слова розділяються розділовими знаками без пробілу, додати пробіл після розділового знаку. Для високого рівня складності додатково потрібно врахувати наступні правила. Одноразова зустріч символу «-» («HYPHEN-MINUS», «мінус», код символу дорівнює 45), оточеного ліворуч і праворуч літерами (або з одного боку межує з літерою, а з іншого – з цифрою) трактується як дефіс і не потребує вставки пробілів. При наявності пробілів між дефісом і символами праворуч або ліворуч – вони видаляються. Послідовність з двох поспіль символів «--» трактується як тире і відділяється від інших символів ліворуч і праворуч одним пробілом.

2. У текстовому файлі зберігається матриця. Записати в інший текстовий файл кількість позитивних, від'ємних і нульових елементів вихідної матриці, її середнє арифметичне значення, максимум и мінімум (з позиціями), найбільше від'ємне і найменше позитивне значення елементів матриці (з позиціями).

3. Компоненти бінарного файлу – дійсні числа. Поставити останнє число з цього файлу між 10-м та 11-м компонентами. Якщо у файлі менше одинадцяти чисел, то ніяких змін здійснювати не потрібно.

4. Бінарний файл містить інформацію про наявність насіння у магазині: назва рослин, час (місяць) висадки, кількість насінин в упаковці, вартість однієї упаковки. Вивести назви рослин, насіння яких можна висаджувати з березня до травня. Провести коригування ціни для насіння визначеного призначення.

Варіант № 5

1. Дан файл, який містить певний текст. Видалити з цього файлу фрази, які містять слова з двома літерами «О».

2. В текстовому файлі зберігається матриця дійсних чисел. У кожному рядку матриці поміняти мінімальний елемент з першим, а максимальний – з останнім і записати перетворену матрицю у інший текстовий файл.

3. Компоненти бінарного файлу – цілі числа. Добавити після кожного позитивного числа його квадрат, нулі видалити.

4. У бінарному файлі знаходяться відомості про тварин зоопарку: назва тварини, природня зона, витрати на корм за один день. Вивести кількість тварин певної природної зони, які знаходяться у зоопарку, і визначити, скільки грошей витрачається на утримання певної тварини на місяць.

Варіант № 6

1. Дан файл, який містить певний текст. Відредагувати файл: після кожної фрази у дужках вказувати кількість слів у ньому.

2. У текстовому файлі зберігається квадратна дійсна матриця. Замінити в ній елементи, які розташовані на головній діагоналі, значенням останнього елементу матриці.

3. Компоненти бінарного файлу – цілі числа. Видалити з цього файлу максимальне і мінімальне число.

4. Дан файл, який містить відомості про автомобілі, які надійшли у продаж. Записи містять наступні поля: марка автомобіля, країна - виробник, рік випуску, об'єм двигуна, витрата бензину на 100 км, ціна, кількість екземплярів. Скорегувати дані про певний автомобіль при зміні на нього ціни. Вивести марку автомобіля з певним об'ємом двигуна і найменшою витратою бензину.

Варіант № 7

1. Дан файл, який містить певний текст. Переписати в новий текстовий

файл фрази, які складаються з найбільшої кількості слів.

2. У текстовому файлі зберігається матриця дійсних чисел. Перетворити її наступним чином: у кожній парній строчці збільшити вдвічі елементи, які стоять на непарних місцях.

3. Компоненти бінарного файлу – цілі числа. Видалити з нього всі нулі. Добавити на початок файлу кількість від'ємних компонентів, а в кінець – кількість позитивних.

4. У файлі зберігається інформація про курорти світу: країна, місто, назва готелю, клас готелю, вартість проживання за одну добу, вартість проїзду в обидва боки. Вивести відомості про готелі певного класу, де вартість проживання за тиждень найменша. Визначити середню вартість туру на тиждень у певний клас готелю, включаючи вартість проживання і вартість проїзду.

Варіант № 8

1. Дан файл, який містить певний текст. Залишити в цьому файлі лише ті фрази, всі слова в яких містять літеру «а».

2. У текстовому файлі зберігається квадратна цілочисельна матриця. «Розірвати» цю матрицю по головній діагоналі, записав у інший файл спочатку елементи, які знаходяться над діагоналлю, а потім в одну строку діагональні елементи, і потім елементи, які знаходяться під діагоналлю. Форма трикутників повинна зберегтись.

3. Компоненти бінарного файлу – цілі числа. Додати на початок файлу значення -1, а в кінець файлу – значення, яке на 1 більше максимального в цьому файлі.

4. Дан файл, який містить відомості про хімічні елементи: назва, позначення символами, масу атома, заряд ядра. Вивести відомості про хімічний елемент за його символічною назвою. Знайти елемент з самою великою масою.

Варіант № 9

1. Дан файл, який містить певний текст. Залишити у цьому файлі лише ті фрази, в яких зустрічається не менше 4 різних голосних букв.

2. У текстовому файлі зберігається матриця дійсних чисел. Додати в цю матрицю стовбці, які містять суми елементів кожної строки, їх максимуми і мінімуми.

3. Компоненти бінарного файлу – цілі числа. Видалити з цього файлу всі числа, які розташовані між першим і останнім позитивними компонентами.

4. Розклад руху потягів зберігається у файлі і містить інформацію: пункт призначення, номер потягу, тип потягу (швидкий, експрес, пасажирський), час відправлення, час у дорозі. Вивести відомості про потяги, які відправляються з

Києва у певний часовий інтервал. Знайти потяг певного типу, який доїздить до Києва за найменший час.

Варіант № 10

1. Дан файл, який містить певний текст. Залишити в цьому файлі лише ті слова, які містять хоча б одну літеру «а» і не містять букв «е».

2. У текстовому файлі зберігається матриця дійсних чисел. Замінити в ній всі від'ємні числа нулями.

3. Компоненти бінарного файлу – беззнакові цілі числа. Видалити з цього файлу всі числа, які є ступенем числа 2.

4. У файлі зберігаються відомості про особисту бібліотеку: прізвище автора, назва, видавництво, рік видання, тематика книги. Вивести назву книг певного автору, які видані після 2000 року. Визначити долю книг у бібліотеці за темою «Програмування» від загальної кількості примірників.

Варіант № 11

1. Дан файл, який містить певний текст. Залишити в цьому файлі лише ті фрази, які містять не менше трьох слів.

2. У текстовому файлі у табличному вигляді зберігається інформація про кількість і ціни товарів на складі. Додати у таблицю графу із загальними сумами по кожному виду товару.

3. Компоненти бінарного файлу – цілі числа. Видалити з цього файлу всі позитивні числа, які кратні 3, додавши у кінці файлу їх кількість.

4. У файлі містяться відомості про співробітників лабораторії: прізвища, рік народження, стать, освіту (середня, вищі), рік вступу на роботу. Знайти самого старшого співробітника серед чоловіків. Вивести список молодих спеціалістів (до 28 років) з вищою освітою.

Варіант № 12

1. Дан файл, який містить певний текст. Видалити з нього всі фрази, у яких є слова, які містять великі літери (велику літеру на початку речення не враховувати).

2. У текстовому файлі міститься таблиця результатів здачі студентами сесії. У таблиці є шапка наступного вигляду: Прізвище, № залікової книжки, математика, фізика, філософія. Переписати в різні файли прізвища студентів - відмінників, студентів - хорошистів і студентів, які на іспиті отримали лише одну тройку.

3. Компоненти бінарного файлу – цілі числа. Обнулити компоненти, які відрізняються від середнього арифметичного значення компонент більше ніж втричі.

4. Є файл, який містить відомості про товари експорту: найменування

товару, країна-імпортер и обсяг поставленої партії в штуках. Вивести країни, в які експортується визначений товар і загальний обсяг цього експорту.

Варіант № 13

1. Дан файл, який містить певний текст. Залишити в цьому файлі лише ті фрази, в яких є числова інформація (тобто слова, які складаються лише із цифр, а для середнього і високого рівнів і скорочений запис числівників вигляду "1-й").

2. У текстовому файлі зберігається цілочисельна матриця. Перетворити її у дійсну і записати у інший файл з точністю до другого знаку після точки.

3. Компоненти бінарного файлу – дійсні числа. Змінити вміст файлу, додавши до кожного позитивного компоненту першу, а з від'ємних компонент відняти значення останньої.

4. Результати екзаменаційної сесії зберігаються у файлі, який містить наступні дані: прізвище студента і оцінки з фізики, математики і інформатики. Вивести кількість двійок по кожному з предметів і вивести список студентів, які мають двійки хоча б з одного предмету.

Варіант № 14

1. Дан файл, який містить певний текст. Залишити в цьому файлі лише ті фрази, в яких є слова, записані прописними літерами.

2. Дан текстовий файл, який містить цілі числа. Збільшити значення парних чисел цього файлу вдвічі, інші залишити без змін.

3. Компоненти бінарного файлу – дійсні числа. Поміняти місцями перший і останні від'ємні компоненти. У кінець файлу додати кількість від'ємних компонентів.

4. У файлі зберігаються відомості про вкладників банку: номер рахунку, паспортні дані, категорія вкладу, поточна сума вкладу, дата останньої операції. Зафіксувати (здійснити зміни) операції приймання і видачі будь-якої суми. Вивести найбільшу суму вкладу у категорії «терміновий».

Варіант № 15

1. Дан файл, який містить певний текст. Перевірити чи всі фрази починаються з прописної літери. Якщо ні – виправити.

2. У текстовому файлі зберігається таблиця синусів і косинусів різних кутів. У таблиці є шапка вигляду « x $\sin x$ $\cos x$ ». Додати в цей файл колонки з тангенсами і котангенсами цих кутів. Якщо значення тангенсу або котангенсу не визначено, у відповідній графі поставити прочерк.

3. Компоненти бінарного файлу – дійсні числа. Видалити з цього файлу кожне п'яте число.

4. У файлі містяться відомості про пацієнтів клініки ока. Структура

запису: прізвище пацієнта, стать, вік, місце проживання (місто), діагноз. Визначити кількість іногородніх пацієнтів, які прибули у клініку. Вивести відомості про пацієнтів пенсійного віку.

Варіант № 16

1. Дан файл, який містить певний текст. Переписати в новий текстовий файл фрази-паліндроми (фрази, які читаються однаково зліва направо та справа наліво).

2. У текстовому файлі зберігається таблиця з результатами сесії студентами одної групи. У таблиці є шапка наступного вигляду: Прізвище, № залікової книжки, математика, фізика, хімія, креслення. Додати у таблицю графу з середнім балом студента за сесію.

3. Компоненти бінарного файлу – цілі числа. Поміняти місцями перший компонент з останнім, другий – з передостаннім і т.д.

4. У файлі зберігаються відомості про архітектурні пам’ятники: назва, місце розташування, тип побудови, архітектор, рік побудови. Вивести відомості про споруду певного типу, наприклад, «собор», побудовану до 18 століття. Знайти самий старий архітектурний пам’ятник.

Варіант № 17

1. Дан файл, який містить певний текст. У новий файл записати саму довгу фразу і фразу з найбільшою кількістю слів.

2. У текстовому файлі зберігаються дійсні числа. Видалити з цього файлу кожне п’яте число.

3. Компоненти бінарного файлу – дійсні числа. Нормувати компоненти файлу відніманням середнього арифметичного всіх чисел з кожного числа.

4. Дан файл, який містить відомості про вступні іспити до вишу за результатами ЗНО з математики, української мови та літератури та англійської мови: прізвище, бали за предметами. Відомі: прохідна сума балів і мінімальна допустима кількість балів з кожної дисципліни. Вивести список абітурієнтів, які мають найбільшу суму балів, і процент абітурієнтів, які не пройшли за конкурсом.

Варіант № 18

1. Дан файл, який містить певний текст. В новий текстовий файл записати статистику по цьому файлу: кількість фраз, слів, знаків без пробілів, знаків з пробілами, – з поясненнями, що означає кожне число.

2. У текстовому файлі зберігається дійсна матриця. Змінити матрицю: округлити всі значення до першого знаку після коми.

3. Компоненти бінарного файлу – цілі числа. Видалити з цього файлу задане число (якщо зустрічається неодноразово, то видалити всі екземпляри). Якщо шуканого числа у файлі нема, то дописати його у кінець файлу.

4. Дан файл, який містить відомості про рух потягів далекого прямування з Київського вокзалу: номер потяга, станція призначення, час відправлення, час у дорозі, наявність квитків. Вивести номера потягів і час їх відправлення у певне місто у заданому часовому інтервалі. Отримати інформацію про наявність білетів на поїзд з певним номером.

Варіант № 19

1. Дан файл, який містить певний текст. Переписати його у новий файл по дві фрази на строку. Якщо кількість фраз непарна, то в останньому рядку залишиться одна фраза.

2. У текстовому файлі зберігаються впорядковані за зменшенням цілі числа. Вставити в файл введене з клавіатури число, не порушуючи впорядкованості.

3. Компоненти бінарного файлу – дійсні числа. Видалити з файлу ті числа, які менші середнього арифметичного усіх чисел файлу.

4. Є файл, який містить відомості про наявність білетів на вистави. Структура запису: назва вистави, назва театру, дата, кількість білетів, ціна. Здійснити корегування даних у файлі при продажу білетів, вихідні дані – назва вистави, назва театру, дата і кількість проданих білетів. Вивести назви вистав, на які є білети на зазначену дату.

Варіант № 20

1. Дан файл, який містить певний текст. Видалити з нього фрази, які містять слова, що складаються лише з голосних літер.

2. Компоненти текстового файлу – дійсні числа. Поміняти місцями перший і останній від'ємні компоненти. У кінці файлу додати середнє арифметичне від'ємних компонент.

3. У типізованому файлі зберігаються впорядковані за збільшенням дійсні числа. Вставити у файл введене з клавіатури число, не порушуючи впорядкованість.

4. Бінарний файл містить перелік монет нумізматичної колекції.

Структура запису: рік карбування, країна, метал, номінал, кількість, ринкова вартість. Визначити сумарну вартість колекції. Вивести відомості про монети, які випущені раніше вказаного століття.

Порядок виконання

1. Ознайомитися з теоретичними відомостями по лабораторній роботі.
2. Отримати варіант завдання.
3. Виконати завдання згідно варіанту.
   3.1 Створення блок-схем алгоритмів, математичних моделей та тестування.
   3.2 Написання програм, відналагодження програми.
   3.3 Демонстрація роботи програми та усунення зауважень.
4. Оформити звіт.
5. Дати відповіді на контрольні питання.
6. Захистити роботу.
7. Виставити звіт у Єлерн.

Контрольні запитання

1. Відкриття і закриття файлів.
2. Читання з текстового файлу і запис в нього.
3. Запис у текстовий файл.
4. Зчитування з двійкового файлу і запис у нього

ПИТАННЯ НА ПІДСУМКОВИЙ КОНТРОЛЬ

1. В чому полягають основні властивості і особливості позиційних систем числення?

2. Що називають основою системи числення?

$$2^n \quad або \ -10 \le n \le 10$$

3. Записати число в нормальній формі.

4. Сформулювати правило перекладу цілих чисел з однієї системи числення в іншу методом розподілу на основу.

5. Сформулювати правило перекладу дробових чисел з однієї системи числення в іншу методом множення на основу.

6. Сформулювати правило перекладу чисел з двійкової системи у вісімкову.

7. Як виконати над числами наступні арифметичні операції: A + B + C; A - B; C - D; E - F; B * E; C * D; F : E; A : E. Операцію віднімання виконувати як складання алгебри, представляючи негативні числа в зворотному коді

8. Чим викликано застосування додаткового і зворотного кодів?

9. В чому відмінності представлення чисел в природній і нормальній формах? Яка форма і коли зручніше?

10. Як провести складання чисел. $A = 0.1100*10^{-101}, B = 0.1001*10^{-001}$

11. Що таке *алгоритм*?

12. Перерахувати базові структури алгоритмів.

13. Визначити, що являють собою ланцюг та *розгалуження*.

14. Що таке *цикл* як базова структура алгоритму?

15. Які існують різновиди циклів?

16. Що таке «базові типи даних»?

17. Які існують цілочисельні типи у мові С? Скільки байтів пам'яті займає тип *int*?

18. Назвати дійсні типи мови С? Скільки байтів пам'яті займає тип *float*?

19. Як представляються символьні дані у мові С?

20. Що являє собою тип *void*?

21. Що являє собою програма на мові С? Яка структура С-програми?

22. Що таке «операція» та що таке «операнд»? Яка операція називається унарною, а яка – бінарною?

23. Що являє собою вираз?

24. Які арифметичні операції використовуються у мові С?

25. Чи існує логічний тип у мові С? Які значення використовуються для представлення логічних значень?

26. Які використовують операції порівняння та які – логічні операції?

27. Які операції відносяться до операцій присвоювання?

28. Які групи операцій присвоювання існують у мові С? Навести приклади.

29. Чи є оператор програмною одиницею?

30. Які групи основних керуючих конструкцій являють собою

операторі?

31. Що таке «пустий оператор» та для чого він використовується?

32. Які існують оператори простої послідовності?

33. Що таке *умовна конструкція*?

34. Структури умовного оператору.

35. Як працює умовний оператор?

36. Що являє собою структура оператора множинного вибору?

37. Як працює оператор множинного вибору?

38. Яка структура програми на мові С?

39. Навіщо потрібна директива #include?

40. Що таке main()?

41. Перерахувати скалярні типи даних мови С.

42. Що визначає тип даних?

43. Що таке явне і неявне приведення типів? Як і коли воно використовується?

44. Що таке константа? Знайдіть константи у наведених програмах.

45. Що таке змінна?

46. Як проініціалізувати змінну?

47. Чим відрізняється оператор від операції?

48. Чим відрізняються унарні операції від бінарних?

49. Які операції відносяться до арифметичних? Який пріоритет кожної з них?

50. Який порядок виконання операцій у випадку їх однакового пріоритету?

51. Як виконується операція ділення у випадку цілочисельних операндів та у випадку, коли хоч би один з операндів дійсний?

52. Що таке вираз?

53. Яке значення обчислює операція присвоювання?

54. В якому порядку виконується присвоювання у випадку, якщо у виразі їх декілька?

55. Як і навіщо використовуються додаткові операції присвоювання?

56. Чим відрізняються префіксна форма операції інкремента або декремента від постфіксної?

57. Які функції використовуються для вводу інформації? Назвіть їх відмінні особливості.

58. Які функції використовуються для виводу інформації? Назвіть їх відмінні особливості.

59. Чому функції scanf() та printf() називаються функціями форматного вводу та виводу? Як вони працюють?

60. Чим відрізняється керуюча строка функції scanf() від керуючої строки функції printf()?

61. Що таке специфікатор формату? Навіщо він потрібен?

62. Які параметри вказуються функцією scanf() після керуючої строки? Скільки їх повинно бути?

63. Які наслідки невідповідності типу зчитуваної функцією scanf() змінної специфікатору типу?

64. Які параметри вказуються функції printf() після керуючої строки? Скільки їх повинно бути?

65. Які наслідки невідповідності типу виводимого функцією printf() значення специфікатору типу?

66. Що таке керуючі символи? Навіщо вони потрібні? Наведіть приклади.

67. Що таке *цикл*?

68. Які існують два типи циклів?

69. Структура оператору циклу з параметром.

70. Як працює цикл з параметром?

71. Вказати особливості оператора циклу с параметром у мові С?

72. Які цикли відносяться до ітераційних?

73. Навести структуру оператора циклу з передумовою.

74. Як працює цикл з передумовою?

75. Яку структуру має оператор циклу з постумовою?

76. Як працює цикл з постумовою?

77. Чим відрізняється умовна операція від умовного оператору?

78. Що таке повна і неповна форма умовного оператору?

79. Чи може існувати неповна форма умовної операції?

80. Чи потрібно писати "else", якщо при виконанні умови виконується оператор return?

81. Вирази якого типу можуть визначати умови?

82. Які значення виразів, що визначають умови, вважаються істинними, а які хибними?

83. Які операції відносяться до операцій відношення?

84. Чим відрізняються операції "= =" від операції "="?

85. Які операції відносяться до логічних? Який пріоритет їх виконання?

86. Якою операцією можна замінити операцію "&&" ?

87. Якою операцією можна замінити операцію "||" ?

88. Чому може дорівнювати значення виразу відношення або логічного виразу?

89. Як правильно порівняти на рівність дійсні числа?

90. Як правильно перевірити входження значення у певний діапазон?

91. Як перевірити певне цілочисельне значення на рівність нулю?

92. Як перевірити відмінність цілочисельного значення від нуля?

93. Коли використовується вложення умовних операторів?

94. Як правильно записати вложені умовні оператори?

95. Що являє собою оператор switch? Як їм користуватися?

96. Як записати оператор switch з допомогою умовних операторів?

97. Які види циклів снують? Які оператори циклу існують у мові С?

98. Чим відрізняються цикл з передумовою від циклу з постумовою?

99. Коли необхідно використовувати цикл з передумовою, а коли з постумовою? Наведіть приклади.

100. Які цикли з передумовою існують у мові С?

101. Скільки операторів містять у собі тіло циклу у мові С?

102. Як правильно записати цикл з постумовою на мові С?

103. Як задати нескінчений цикл? Навіщо він потрібен? Як з нього вийти?

104. Яким повинно бути значення виразу, який визначає умову

виконання циклу, для завершення циклу?

105. Яким повинно бути значення виразу, який визначає умову виконання циклу, для виконання тіла циклу?

106. До чого призведе неправильне завдання виразу, який визначає умови виконання циклу?

107. Чи може бути відсутнім тіло циклу? Якщо може, то наведіть приклади таких циклів.

108. Чим відрізняється оператор *while* від оператора *if*?

109. Який порядок дій при виконанні циклу *for*?

110. Як організувати арифметичний цикл з допомогою циклу *for*?

111. Запишіть алгоритм, який визначається циклом *for*, з допомогою циклу *while*.

112. Що таке вкладений цикл?

113. Скільки разів у загальній складності виконується тіло вкладеного циклу?

114. Як і коли використовуються оператори *break* то *continue*?

115. Що таке рекурентні обчислення? Коли вони використовуються? Як їх програмувати?

116. Що таке функція?

117. Що таке функція типу void?

118. Що таке прототип функції?

119. Чим відрізняється прототип функції від виклику функції?

120. Чим відрізняється описання функції від визначення функції?

121. Критерії відповідності формальних і фактичних параметрів.

122. Відповідність типів формальних і фактичних параметрів.

123. Навіщо потрібен оператор return?

124. Чи може у функції бути декілька операторів return?

125. Коли необхідно писати оператор return у функції типу void?

126. Що таке побочний ефект функції?

127. Коли використовують формальні параметри-вказівники? Якими у цьому випадку повинні бути фактичні параметри?

128. Як передати масив у функцію? Як передати у функцію матрицю?

129. Як описати функцію, яка дозволяє працювати одночасно з одновимірними масивами, і з матрицями?

130. Що таке параметр-константа? Коли використовують такі параметри

131. Що таке вказівник на функцію? Як його описати?

132. Як передати функцію у функцію? Що буде формальним параметром, а що фактичним? Як 133. звернутися до функції, яку передано через параметри?

134. Що таке рекурсія? Коли вона використовується? Що таке глибина рекурсії?

135. Як описати функцію зі змінною кількістю параметрів? Як використовувати таку функцію?

136. Які параметри можуть бути у функції main()? Як налагоджувати і тестувати програму, яка містить функцію main() з параметрами?

137. Чим є змінна у програмі?

138. Що означає унарна операція «&», до яких об’єктів вона може

застосовуватись?

139. Яке значення може набувати адреса?

140. Чим є вказівник, що він містить?

141. Для чого використовують вказівники?

142. Що означає унарна операція «*», що є результатом її виконання?

143. На об’єкт якого типу може посилатись вказівник?

144. Що означає вказівник на тип *void*?

145. Що таке *статична структура даних*?

146. Чим характеризуються змінні статичних структур?

147. Що являє собою масив даних?

148. На якому етапі виділяється пам’ять під елементи масиву?

149. Чи змінюється обсяг пам’яті, яка виділяється під масив, під час виконання програми?

150. Як занести у вказівник адресу першого елементу масиву?

151. На що посилається будь-який вказівник?

152. Що є результатом операції *вказівник+і*?

153. Чим відрізняються операції *p*++ та **p*++, якщо *p* – вказівник?

154. Чим відрізняються операції *(*p)*++ та **(++p)*, якщо *p* – вказівник?

155. Що означає лінійний пошук у масиві даних?

156. Що називається сортуванням інформаційної структури?

157. Які існують ознаки порядку?

158. У чому полягає суть бульбашкового сортування масиву даних?

159. Як виконати сортування масиву простим вибором?

160. Що таке вказівник?

161. Який об’єм пам’яті займає вказівник?

162. Що є значенням змінно-вказівника?

163. Як ініціалізувати вказівник?

164. Шо таке NULL?

165. Що таке вказівник на void? Навіщо потрібні такі вказівники?

166. Які операції допустимі при роботі з вказівниками?

167. Чим відрізняється унарна операція "&" від унарної "*" ?

168. Сумісність типів вказівників.

169. Чи можна отримати адресу вказівника?

170. Чи можна вказівнику присвоїти його ж адресу?

171. Чому до вказівника на void неможливо застосувати операцію розіменування?

172. Як працюють операції інкременту і декременту, при застосуванні до вказівників?

173. Який результат операції віднімання, яку застосовують до вказівників одного типу?

174. Який специфікатор типу використовують при виводі адреси на екран з допомогою функції printf()?

175. У чому відмінність записів *(double *) a* від *(double) * a*, якщо *a* – вказівник на ціле число?

176. У чому відмінність записів **a*++ від (**a*)++, якщо *a* – певний вказівник, відмінний від void*?

177. Як описати вказівник на початок масиву?

178. Як описати вказівник на вказівник?

179. Коли і навіщо може повторно використовуватись операція розіменовування?

180. Що таке масив?

181. Що являє собою ім’я масиву?

182. Що являє собою індекс елементу масиву?

183. Як можна звернутися до елементу масиву?

184. Як отримати адресу елементу масиву?

185. Як описати вказівник на початок масиву?

186. Як звернутися до елементу масиву через вказівник?

187. Чи існує зв’язок між індексом елементу і його значенням?

188. Чому дорівнює індекс першого елементу масиву?

189. Як проініціалізувати масив?

190. Коли можна не вказувати кількість елементів масиву при описанні?

191. Якого типу можуть бути елементи масиву?

192. Які операції можна здійснювати над цілим масивом?

193. Чому при обробці масивів використовують цикли?

194. Як поміняти місцями два елементи масиву?

195. Чому дорівнює індекс останнього елементу масиву?

196. Чи виникає помилка при зверненні до елементу масиву, індекс якого більше індексу останнього елементу цього масиву?

197. Чи може існувати масив з одного елементу? Якщо може, то як його описати?

198. Чи є *p* вказівником на масив a[], якщо вони оголошені наступним чином: *int a[10], *p=a;* ?

199. Що таке сортування масиву?

200. Що таке двовимірний масив?

201. Як розташований двовимірний масив у пам’яті комп’ютера?

202. Як проініціалізувати двовимірний масив?

203. Чому при оголошенні двовимірного масиву з одночасною ініціалізацією перші скобки можна залишити порожніми, а другі - ні?

204. При звертанні до елементу двовимірного масиву спочатку вказується індекс рядку чи індекс стовбця?

205. Як вивести двовимірний масив таблицею з колонками однакової ширини?

206. Що являє собою ім’я двовимірного масиву?

207. Що являє собою перший індекс елемента двовимірного масиву?

208. Що являє собою другий індекс елемента двовимірного масиву?

209. Що являє собою ім’я строки матриці?

210. Як отримати адресу елементу двовимірного масиву?

211. Як отримати адресу початку двовимірного масиву?

212. Як отримати адресу рядка матриці?

213. Чи можна продивитись усі елементи двовимірного масиву в одному циклі?

214. Як звернутися до певного елементу двовимірного масиву через вказівник на перший елемент матриці?

215. Як звернутися до елементу двовимірного масиву через вказівник

на початок масиву?

216. Як описати трьохвимірний масив і як з ним працювати?

217. Якими способами можна запрограмувати матрицю?

218. Чим відрізняється перегляд елементів матриці за стовбцями від перегляду за рядками?

219. Який зв'язок між індексами рядка і стовбця у елементів, які знаходяться на головній діагоналі квадратної матриці?

220. Який зв'язок між індексами строки і стовбця у елементів, які лежать на побочній діагоналі квадратної матриці?

221. Чи можна подивитись всі елементи матриці в одному циклі?

222. В яких областях оперативної пам'яті можуть розташовуватись дані при виконанні програми?

223. Які змінні називають динамічними? Чим вони відрізняються від статичних змінних?

224. З якою метою використовують динамічні змінні?

225. Як звертаються до динамічних змінних?

226. Які бувають вказівники? Як їх описати?

227. Яким чином можна виділити пам'ять для динамічних змінних і звільнити її?

228. Як визначити, чи виділена пам'ять чи ні?

229. Чому неможливо забувати звільняти виділену пам'ять?

230. Чому дорівнює значення вказівника після звільнення області пам'яті, на яку він вказував?

231. Як розмістити в динамічній пам'яті масив?

232. Якими способами можна розмістити в динамічній пам'яті матрицю?

233. Що таке С-строка? Чим вона відрізняється від масиву символів?

234. Що таке нуль -термінатор?

235. Чим відрізняється описання *char *st* от *char st[N]*, де N – декотра константа?

236. Як ініціалізувати строку?

237. Чим відрізняється ввод строки з допомогою функції *scanf()* від вводу з допомогою функції *gets()*?

238. Як знайти довжину строки?

239. Чи можна використовувати операцію присвоювання для завдання значення строки?

240. Що таке конкатенація строк?

241. Які існують функції для роботи зі строками?

242. Що таке структурний тип даних?

243. Які види структур даних існують?

244. Що таке вложені структури?

245. Яким чином можна звернутися до різних полів структури?

246. Як задати масив структур?

247. Відкриття і закриття файлів.

248. Читання з текстового файлу і запис в нього.

249. Запис у текстовий файл.

250. Зчитування з двійкового файлу і запис у нього.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Базова

1. Пол Іре. Об'єктно-орієнтоване програмування з використанням C++: Пер. з англ. - Київ: НИПФ "ДіаСофт Лтд, 1995.

2. Bjarne Stroustrup The C++ Programming language, Addison Weasley, 1986.

3. Фейсон Т. Об'єктно-орієнтоване програмування на Borland C++ 4.5: Пер. з англ. - Київ: Діалектика, 1996. 544с.

4. Сван Т. Опанування Borland C++ 4.5: Пер. з англ. - Київ: Діалектика, 1996. 544с.

5. Крис Паппас \ Крис Паппас, Уильям Мюррей. Программирование на C и C++. Серия «Библиотека студента». – «Ирина», BHV, Киев, 2000. – 320с.

6. ANSI, American National Standart for Information Systems – Programming Language C. – New York, 1990.

Допоміжна

1. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

Інтернет-джерела:

1. Draw io. Free online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams [Електронний ресурс] - https://app.diagrams.net

2. Online compiler and debugger for c/c++ [Електронний ресурс] - https://www.onlinegdb.com

3. The Top 11 Computer Science Books for Self Study [2023] In today's modern, fast-paced world, we look to StackOverflow, Reddit, and hands-on courses to learn about computer science. [Електронний ресурс] - https://blog.boot.dev/computer-science/computer-science-books/

4. Data Structures - Full Course Using C and C++ Learn about data structures in this comprehensive course. We will be implementing these data structures in C or C++. [Електронний ресурс] –https://www.youtube.com/watch?v=B31LgI4Y4DQ

5. Top 10 Must-Read Books for Computer Science Majors - Computer Science Degree Hub++. [Електронний ресурс] – https://www.computersciencedegreehub.com/top-10-mustread-books-computer-science-majors/.

Додаток 1. Силабус дисципліни

СИЛАБУС ДИСЦИПЛІНИ «ОСНОВИ ПРОГРАМУВАННЯ»

Ступінь вищої освіти – Бакалавр
Спеціальність 141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА
Освітньо -професійна програма «Інжиніринг електроенергетичних систем з відновлюваними джерелами» Рік навчання 1, семестр 2
Форма навчання денна
Кількість кредитів ЄКТС 6
Мова викладання українська

Лектор дисципліни

Контактна інформація лектора (e-mail)

Смолій Вікторія Миколаївна, д.т.н., професор
кафедра інформаційних систем і технологій,
корпус. 15, к.212, тел. (044) 527-87-32
e-mail vmsmolij@nubip.edu.ua

Сторінка курсу в elearn https://elearn.nubip.edu.ua/course/view.php?id=802

ОПИС ДИСЦИПЛІНИ

Вивчення матеріалу дисципліни призводить до формування фундаментальних теоретичних знань з основ програмування, які використовуються при формулюванні алгоритмів вирішення спеціалізованих задач; визначенні або виборі оптимальної структури алгоритму; визначенні покажчиків продуктивності та часу виконання алгоритму; виявлені слабких місць програмної реалізації системи; врахуванні впливу технічних засобів на побудову і функціонування програм; вибору оптимальних мов програмування для реалізації алгоритмів функціонування системи. Здобуті у процесі вивчення дисципліни знання з формування теоретичних знань та практичних навичок у галузі програмування є базою для вивчення дисциплін професійно-орієнтованого циклу.

Навчальна дисципліна забезпечує формування ряду загальних та фахових компетентностей:

інтегральна компетентність (**ІК**): Здатність розв'язувати професійно-практичні задачі під час провадження професійної діяльності в сфері електричних мереж та електроенергетичних систем або у процесі навчання, що характеризується невизначеністю умов і вимог.

- *загальні компетентності*:

ЗК01. Здатність до абстрактного мислення, аналізу і синтезу.

- *спеціальні(фахові, предметні) компетентності:*

СК19. Здатність виконувати загальні інженерні розрахунки із застосуванням сучасного програмного забезпечення

У результаті вивчення навчальної дисципліни студент повинен показати певні програмні результати, а саме:

ПРН06. Застосовувати прикладне програмне забезпечення, мікроконтролери та мікропроцесорну техніку для вирішення практичних проблем у професійній діяльності.

ПРН18. Вміти самостійно вчитися, опановувати нові знання і вдосконалювати навички роботи з сучасним обладнанням, вимірювальною технікою та прикладним програмним забезпеченням.

ПРН26. Знати особливості застосування сучасного програмного забезпечення з метою розв'язання загальних інженерних задач.

СТРУКТУРА ДИЦИПЛІНИ

Тема	Години (лекції/лабораторні, практичні, семінарські)	Результати навчання	Завдання	Оцінювання
1 семестр				
Модуль 1				
Тема 1. ПРОСТІ ЕЛЕМЕНТИ МОВИ. ВСТУП Алгоритми. Загальні відомості. Різновиди **Лабораторна робота 1.** Арифметичні основи побудови елементів і вузлів обчислювальних машин і систем **Лабораторна робота 2.** Алгоритмізація задач	2/8/3	Знати арифметичні основи побудови елементів і вузлів та підходи до алгоритмізації задач	Виконання, налагодження, тестування, демонстрація функціонування, оформлення звіту, відповіді на контрольні питання та захист лабораторної (самостійної) роботи за означеною темою (в.т.ч. в Elearn).	**10**
Тема 2. СТРУКТУРА ДАНИХ ТА ВИРАЗИ **Лабораторна робота 3.** Лінійні обчислювальні процеси	2/4/3	Вміти організовувати лінійні обчислювальні процеси		**5**
Тема 3. СТРУКТУРА І ПРИКЛАДИ ПРОГРАМИ **Лабораторна робота 4.** Умовні конструкції: оператори розгалуження	2/4/3	Аналізувати умовні конструкції		**5**
Тема 4. КЛАСИ ПАМ'ЯТІ.	2/4/3	Розуміти структуру програми,		**10**

ЛОГІЧНІ ВИРАЗИ. КЕРУЮЧІ СТРУКТУРИ **Лабораторна робота 5.** Структура програми, основні типи даних, ввод/вивід **Лабораторна робота 6.** Циклічні конструкції: оператори циклу		основні типи даних, циклічні конструкції		
Самостійна робота 1. Визначення стану принтеру у DOS	0/0/5	Розрізняти, зчитувати та визначати стани принтеру		**15**
Тема 5. ЛОГІЧНІ ВИРАЗИ. ВІДНОШЕННЯ, ЛОГІЧНІ ОПЕРАЦІЇ, УМОВНІ ВИРАЗИ **Лабораторні роботи 7, 8.** Розгалуження і цикли (у двох частинах)	2/8/3	Застосовувати розгалуження і цикли	Виконання, налагодження, тестування, демонстрація функціонування, оформлення звіту, відповіді на контрольні питання та захист лабораторної (самостійної) роботи за означеною темою (в.т.ч. в Elearn).	**10**
Тема 6. КЕРУЮЧІ СТРУКТУРИ **Лабораторна робота 10.** Оператори та цикли	2/4/3	Використовувати оператори та цикли		**5**
Тема 7. МАСИВИ ТА ПОКАЖЧИКИ **Лабораторна робота 9.** Змінні і константи, типи даних, ввод та вивод, оператори	2/4/3	Знати що таке, як вводити і чим відрізняються змінні, константи, типи даних, ввод та вивод, оператори		**5**
Тема 8. ПОКАЖЧИКИ **Лабораторна робота 11.** Функції та рекурсія	2/4/3	Вміти використовувати вказівники (покажчики), функції та рекурсії		**5**

Модульне тестування				**15**
Модульне завдання				**15**
Модуль 2				
Тема 9. МАСИВИ. ПОНЯТТЯ. ПРИКЛАДИ. ОСОБЛИВОСТІ ПРОГРАМУВАННЯ **Лабораторна робота 12.** Вказівники і одновимірні масиви даних **Лабораторна робота 13.** Вказівники **Лабораторна робота 14 та 15.** Масиви. динамічне виділення пам'яті (у двох частинах)	2/16/6	Аналізувати вказівники, масиви, динамічне виділення пам'яті	Виконання, налагодження, тестування, демонстрація функціонування, оформлення звіту, відповіді на контрольні питання та захист лабораторної (самостійної) роботи за означеною темою (в.т.ч. в Elearn).	**20**
Тема 10. ФУНКЦІЇ. СТРУКТУРИ. **Лабораторна робота 16.** Робота з масивами (закріплення навичок роботи з масивами) **Лабораторна робота 17.** Робота зі строками (рядками)	2/8/5	Розуміти особливості роботи з масивами та строками		**10**
Тема 11. СТРУКТУРИ АБО ЗАПИСИ. **Лабораторна робота 18.** Строки (рядки) закріплення матеріалу	2/4/3	Знати що таке структури і як з ними працювати Розрізняти символи, строки і масиви символів		**5**
Самостійна робота 1. Робота зі строковими масивами	0/0/5	Застосовувати роботу зі строковими масивами		**15**
Тема 12. ДИРЕКТИВИ ПРЕПРОЦЕСОРА **Лабораторна**	2/4/3	Використовувати навички роботи зі строками		**5**

робота 19. Робота зі строками (закріплення навичок роботи зі строками. частина 2)				
Тема 13. ФАЙЛИ **Лабораторна робота 20.** Робота зі структурами	2/4/3	Вміти працювати з файлами	Виконання, налагодження, тестування, демонстрація функціонування, оформлення звіту, відповіді на контрольні питання та захист лабораторної (самостійної) роботи за означеною темою (в.т.ч. в Elearn).	**5**
Тема 14. ФУНКЦІЇ ОБМІНУ З ПОТОКАМИ **Лабораторна робота 21.** Робота з файлами	2/4/3	Аналізувати використання потокових функцій для роботи з текстовими та бінарними файлами		**5**
Тема 15. КЕРУВАННЯ БУФЕРІЗАЦІЄЮ **Лабораторна робота 22.** Використання потокових функцій для роботи з текстовими та бінарними файлами	2/4/3	Здійснювти керування буферизацією		**5**
Разом за курс	**30/90/60**			
Модульне тестування				**15**
Модульне завдання				**15**
Всього за 1 семестр				**70**
Екзамен (Підсумкове тестування)				**7**
Екзамен (Підсумкове завдання 1)				**7**
Екзамен (Підсумкове завдання 2)				**7**
Співбесіда				**4**
Неформальна освіта				**5**
Всього за курс				**100**

ПОЛІТИКА ОЦІНЮВАННЯ

Політика щодо дедлайнів та перескладання:	Дедлайни визначені в ЕНК. Роботи, які здаються із порушенням термінів без поважних причин, оцінюються на нижчу оцінку. Перескладання модулів відбувається із дозволу лектора за наявності поважних причин (наприклад, лікарняний).
Політика щодо академічної доброчесності:	Списування під час самостійних робіт, тестування та екзаменів заборонені (в т.ч. із використанням мобільних девайсів).
Політика щодо відвідування:	Відвідування занять є обов'язковим. За об'єктивних причин (наприклад, хвороба, міжнародне стажування) навчання може відбуватись індивідуально (в дистанційній on-line формі за погодженням із деканом факультету)

ШКАЛА ОЦІНЮВАННЯ СТУДЕНТІВ

Рейтинг здобувача вищої	**Оцінка національна за результати складання екзаменів заліків**	
	Екзаменів	**Заліків**
освіти, бали 90-100	Відмінно	зараховано
74-89	Добре	
60-73	Задовільно	
0-59	незадовільно	не зараховано

Література

Література для вивчення курсу

Базова

1. Пол Іре. Об’єктно-орієнтоване програмування з використанням C++: Пер. з англ. - Київ: НИПФ “ДіаСофт Лтд, 1995.

2. Bjarne Stroustrup The C++ Programming language, Addison Weasley, 1986.

3. Фейсон Т. Об’єктно-орієнтоване програмування на Borland C++ 4.5: Пер. з англ. - Київ: Діалектика, 1996. 544с.

4. Сван Т. Опанування Borland C++ 4.5: Пер. з англ. - Київ: Діалектика, 1996. 544с.

5. Козак Л. І., Костюк І. В., Стасевич С. П. Основи програмування: навчальний посібник – Львів: «Новий Світ-2000», 2020. – 328с.

Допоміжна

1. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

Інтернет-джерела:

1. Draw io. Free online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams [Електронний ресурс] - https://app.diagrams.net

2. Online compiler and debugger for c/c++ [Електронний ресурс] - https://www.onlinegdb.com

3. The Top 11 Computer Science Books for Self Study [2023]

In today’s modern, fast-paced world, we look to StackOverflow, Reddit, and hands-on courses to learn about computer science._[Електронний ресурс] - https://blog.boot.dev/computer-science/computer-science-books/

4. Data Structures - Full Course Using C and C++

Learn about data structures in this comprehensive course. We will be implementing these data structures in C or C++. [Електронний ресурс] – https://www.youtube.com/watch?v=B31LgI4Y4DQ

5. Top 10 Must-Read Books for Computer Science Majors - Computer Science Degree Hub++. [Електронний ресурс] – https://www.computersciencedegreehub.com/top-10-mustread-books-computer-science-majors/.

Додаток 2. Презентація до лекції 1

Дисципліна: “Основи програмування”

для груп IEC, TE

Тема 1: “Основи програмування”

ALGOL-60
↓
1963 CPL
↓
1967 BCPL
↓
1970 B
↓
1972 C
↓
1983-1988 ANSI C (C++) C89
↓
1998 C++
↓
2003 C++

Мова програмування С++ створена на фірмі Bell Laboratories в 1972р. Денісом Ритчі (*Dennis MacAlistair Ritchie*). Попередником цієї мови є мова АЛГОЛ-60, на основі якої була розроблена мова CPL (мова комбінованого програмування) в 1963р. (Кембриджський і Лондонський університети) потім BCPL (базова мова комбінованого програмування) 1967р. (Кембриджський університет). Метою цих розробок було відродити мову АЛГОЛ, зберегти контакт із ЕОМ. Тобто поруч із можливостями універсальної процедурної мови були додані засоби роботи з апаратною частиною ЕОМ. Тому мова BCPL виявилася громіздким і в 1970р. Кеном Томпсоном (*Kenneth Lane Thompson*) в Bell Laboratories була розроблена мова B, яка була вузько спрямована на розробку системних програм.

Ця мова й стала основою для створення мови С++. Основна заслуга Д. Ритчі - це додавання вдалої системи типів даних. Початкове призначення мови С++ - системне програмування. Сама мова була невід'ємною частиною системи UNI для ЕОМ типу PDP-11.
Так приблизно з 13 тис. рядків цієї системи лише 800 були написані на асемблері, інші наС++.

Можна відзначити такі особливості мови:

1. Мова С++ належить до мов зі слабкою типізацією даних. Набагато ширше, ніж у мові Паскаль, передбачене неявне перетворення типів.

2. Передбачене використання покажчиків, які дають можливість оперувати з адресами й непрямою адресацією.

3. Клас об'єктів мови обмежений. Виключені такі структури даних як логічні, множини. Оскільки мова С++ тісно пов'язана з операційною системою UNI, то багато можливостей реалізуються функціями операційної системи. Це забезпечило компактність і високу ефективність даної мови.

Мова відзначається внутрішньою єдністю, що властива мовам "для людини" (Паскаль, Лісп). Має невеликий перелік вихідних засобів, з яких можна створювати досить складні конструкції.

2

Однак перші варіанти мови не були уніфіковані, через що питання перенесення все-таки виникали. Тому в 1985р. інститут американських національних стандартів (ANSI) створив комітет для розробки сучасної машинно-незалежної мови С++. І в 1988р. ця робота закінчилася створенням мови "ANSI C". ця версія широко використовується на персональних ЕОМ, зберігає більшість властивостей вихідної мови, відрізняється:

1. деяким описом функцій,
2. наявністю типу, що перелічується,

3. дозволяє операцію присвоєння для структур (записів), розширеною й уніфікованою бібліотекою функцій.

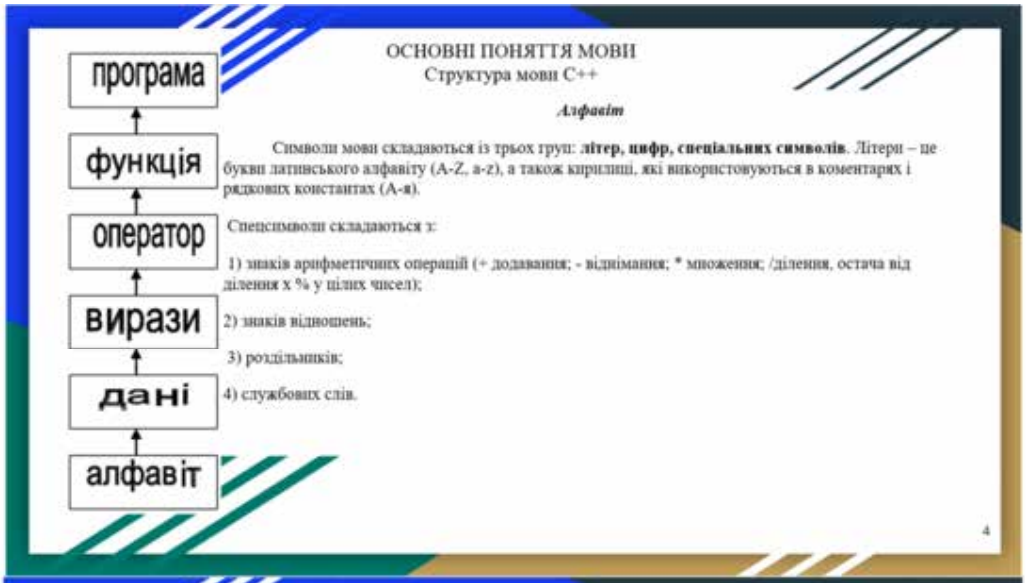
Можна вважати, що мова С++, з одного боку, є зручної, виразної й гнучкої для програмування широкого класу завдань. З іншого боку - вона в достатній мірі наближена до ЕОМ, дає засоби контролю процесом реалізації програми, але зберігає певну дистанцію, яка дозволяє не враховувати всі особливості архітектури ЕОМ. Можна вважати, що це мова відносно низького рівня.

1985 р. вийшло перше видання Бьярні Страуструпа «Мови програмування С++»

«The C++ Programming Language, 1st»

Говорять, що мова С++ призначена для професіоналів- програмістів, а не для початківців. Це створює певні труднощі при вивченні мови.

3



ОСНОВНІ ПОНЯТТЯ МОВИ
Структура мови С++

Алфавіт

Символи мови складаються із трьох груп: **літер, цифр, спеціальних символів**. Літери – це букви латинського алфавіту (A-Z, a-z), а також кирилиці, які використовуються в коментарях і рядкових константах (А-я).

Спецсимволи складаються з:

1) знаків арифметичних операцій (+ додавання; - віднімання; * множення; /ділення, остача від ділення х % у цілих чисел);

2) знаків відношень;

3) роздільників;

4) службових слів.

4

1. Арифметичні операції:

+	Додавання	total = var1 + var2
-	Віднімання	total = var1 - var2
*	Множення	total = var1 * var2
/	Ділення	total = var1 / var2
++	Оператор збільшення	total = total + 1
--	Оператор зменшення	total = total - 1

(++)(--)variable- префіксним оператором збільшення (зменшення);
intA= 7, B=5;
printf("Початкове значення А дорівнює: %d ", A);A += ++B;printf(" Кінцеве значення А дорівнює:%d ", A);
?
variable(++)(--)-постфіксним оператором збільшення (зменшення);
intA= 7, B=5;
printf("Початкове значення А дорівнює: %d ", A);
A +=B++;printf(" Кінцеве значення А дорівнює:%d ", A);
?

5

Операція	***Функція***
%	Повертає залишок цілочисленного розподілу
~	Доповнення; інвертує біти значення
&	Побітове множення I (також &X – взяти адресу X)
|	Побітове додавання АБО
^	Побітовое виключаюче додавання АБО
<<	Зсув вліво; зсуває біти значення вліво на зазначену кількість розрядів
>>	Зсув вправо; зсуває біти значення вправо на зазначену кількість розрядів

Операції піднесення до степеня НЕ передбачено

6

2. Знаки відносин і логічних операцій:

>, <	Більше, менше
==	Дорівнює (2 знака рівності)
!=	Не дорівнює
&&	Логічне І
||	Логічне АБО
!	Логічне НІ

3. Роздільники: , ( ) [ ] { } ' " = ; :

Фігурні дужки мають значення операторних дужок (Begin - End мови Паскаль).

Для коментарів використовуються пари знаків /*КОМЕНТАР*/.

4. Службові слова:

auto	break	case	char	continue	default	do	double
else	entry	enum	extern	float	for	goto	if
int	long	register	return	short	sizeof	static	struct
switch	typedef	union	unsigned	void	while		

Всього 30 службових слів відauto до while.

7

Константи

Константами називаються перелічення величин у програмі.

Розділяють константи таких типів:

1. Цілі (243)
2. З плаваючою крапкою (дійсні) (543.8)
3. Символьні ('A') -1 байт
4. Строкові ("A") -2 байта A + NULL
5. Переліковного типу (ANSI-C).

8

Цілі константи

можуть записуватися в десятковій, вісімковій і шістнадцятковій системах числення.

- Десяткова ціла константа подається звичайним чином (зі знаком або без нього): -2561; 458.
- Ознакою вісімкової константи є нуль ліворуч:Ø257.
- Шіснадцяткова константа визначається двома початковими символами: ØX.(ØX1F16=)

Крім звичайних цілих констант можна використовувати цілі подвійної точності, які в пам'яті займають 2 слова.

До них додається літера L: 371L.

Діапазон: -2147483.648 ÷2147483647

9

Константи дійсного типу (floating point)

можуть подаватися у формі з фіксованою крапкою: 561.25, .5, 100.

При цьому в пам'яті розміщуються звичайним чином (2 слова).

Якщо ж константа задається у формі з плаваючою крапкою, то в деяких трансляторах вона автоматично подається з подвійною точністю, тобто в 4 слова.

Стандартна форма
Експоненціальна форма
-576000,0
-5.76E+05
0,000078
7.8E-05

10

Символьні константи

Символьні константи (CHAR) набувають значення одного символу й подаються в апострофах 'x', 'a', 'r' і займають 1 байт.

Недруковані символи, які не мають графічного зображення, подаються умовно двома символами: перший - використовується зворотний слеш, а другий - певний символ.

Так звана зворотна послідовність перемикання коду (escape sequence або ескейп послідовність).

Перехід на наступний рядок	'\n'
Горизонтальна табуляція	'\t'
Перехід на попередню позицію	'\b'
Переведення каретки	'\r'
Перехід на наступну сторінку	'\f'
Апостроф	'\''
Звуковий сигнал	'\a'
Вертикальна табуляція	'\v'

11

Рядкові константи

Рядкові константи (рядки, string) це послідовність символів, обмежена подвійними лапками: "string". Для переносу на інший рядок використовується зворотний слеш

"морозный\
день+ NULL" -12+1 байт

У мові С кожний рядок автоматично закінчується Ø-байтом, що використовується при обробці масивів рядків. Тому 'z'та "z"це не те саме, бо у другому випадку рядок складається з двох символів 'z'та 'Ø'.

12

Визначення констант реалізується 3-а способами:

1. Процесором :

#define <ім'я константи> <літерал чи ім'я константи>
#define <ім'я константи ><вираз із констант>

Наприклад:

```
#define LN 50
#define PI 3.141592
```

Наприклад, наступний оператор створює макрокоманду CUBE:

```
#define CUBE(x) ((x)*(x)*(x))
```

або

```
#define delay(x)
{ cout << "Задержка на" << x<< endl; \
for (long int i=0; i < x; i++) ; \
}
```

13

2. За допомогою службового слова const:

const[тип]<ім'я >= <значення>

```
const float pi = 3.1415926;
const maxint = 32767;
```

14

3. Оголошення перерахування

Формат 1

enum [ім'я –тега-перерахування] {список-перерахування } описувач[, описувач...];

```
enum days { sun, mon, tues, wed, thur, fri, } anyday;
enum days {sat, sun} weekend, superday;
```

Формат 2

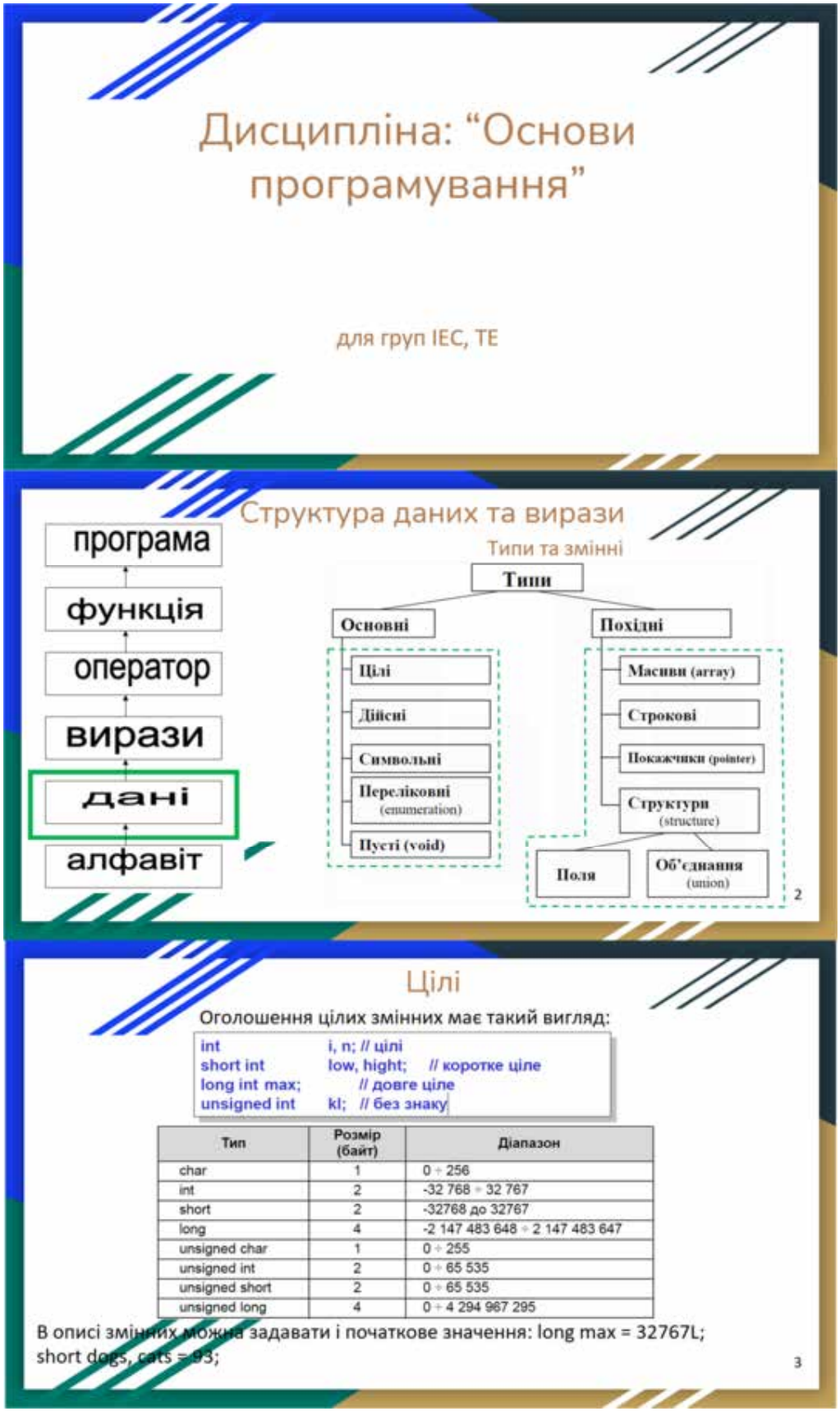
```
enum ім'я –тега-перерахування описувач[, описувач...];
enum days { sun, mon, tues, wed, thur, fri, };
enum days anyday;
```

anyday = mon; // можно

anyday = 1; // ПОМИЛКА, хоча mon ==1, але if (mon==1)

15

Додаток 3. Презентація до лекції 2



Дійсні з плаваючою крапкою

Тип	Розмір	Діапазон	Точність
float	2 слова - 4 байта 32 біта	10e-38 ÷ 10e38	5 знаків
double	4 слова - 8 байт 64 біта	10e-308 ÷ 10e308	15 знаків
long double	5 слов - 10 байт 80 біт	10e-4932 ÷ 10e4932	19 знаків

Приклад:

```
float pi_float; // pi_float = 3.1415

double pi_double; // pi_double = 3.14159265358979

long double pi_long; // pi_long = 3.141592653589793238
```

4

Символьні змінні

Символьні змінні (CHAR) приймають значення одного символу і (займають 1 байт):

```
char c, ch='Y';
char esc ='\x1B';
```

Управляюча послідовність	Функція	Шістнадцятковий формат
\a	Звуковий сигнал	007
\b	Повернення на крок	008
\t	Горизонтальна табуляція	009
\n	Перехід на нову строку	00A
\v	Вертикальна табуляція	00B
\r	Повернення каретки	00C
\f	Перевод формата	00D
\"	Лапки	022
\'	Апостроф	027
\0	Ноль-символ	000
\\	Зворотня дробова риса	05C
\0ddd	Символ набору кодів в вісімковому представленні	
\xddd	Набір кодів в шістнадцятковому представленні	

5

Переліковний тип - ANSI - C

Переліковний тип (enum) використовується для опису об'єктів з певні множини, наприклад:

```
enum seasons {winter, spring, summer, autumn}
```

І описати такі змінні можливо наступним чином:

```
enum seasons a, b, c;
```

або

```
enum seasons {wint, spring, sum, autumn} a, b, c;
```

Можна присвоїти значення, але по зростанню номера:

```
enum month {JAN = 1, FEB, MAR, ..., DEC} real_month;
enum days {mon = 5, tues = 8, wed = 10, thur, fri, sat, sun};
```

Змінна real_month буде приймати значення в діапазоні від 1 до 12

```
enum escapes {BELL='\a', BACKPASE='\b', TAB='\t', NEWLINE='\n'}
```

6

Приклад :

```
enum seasons {winter, spring, summer, autumn}
```

Опис таких змінних :

```
enum seasons day, b, c;
```

```
// змінна day буде приймати значення в діапазоні від 0 до 3
day= winter;          // day - змінна переліковного типу
int test= winter;     // test = 0
day= seasons (1);     // seasons (1) дорівнює значенню spring
```

7

3. Вирази

8

Арифметичні вирази та операції присвоювання

Операція присвоєння використовується найчастіше і має вигляд :

```
a = b
```

де a - певна змінна, b - вираз. Значення виразу b присвоюється змінної a. Можна використовувати й багаторазове присвоєння

```
a = b = c = d = 1
```

Таке присвоєння виконується ⇐ з права на ліво.

Якщо b є арифметичним виразом загального виду, то його результат залежить від прийнятої послідовності виконання операцій - їх старшинства.
Враховуючи велику кількість операцій, існують таблиці, де обумовлено порядок виконання й ієрархія.

9

Особливості використання круглих дужок

Наприклад: c = (d = 1) + 2;
Тут компактно записано d =1; c = d + 2;
або
a = (b = c / (d * e)) + f

1. d =d * e
2. b = c / d
3. a = b+f

Наприклад операцію присвоєння **i=i+2** можна записати **i+=2**.

Для бінарних операцій можливе використання виразу типу:
e1 op = e2
де ***e1***, ***e2*** - вираз, а ***op*** – операція

e1=(e1) op (e2) ⇒ ***e1 op = e2***

Наприклад:

y*=z+1 *еквівалентно* ***y=y*(z+1)***, *а НЕ* ***y=y*z+1***.

10

Оператори

програма

функція

оператори

вирази

дані

алфавіт

11

Ієрархія операцій в С

Операція	Ім'я	Приклад
::	Діапазон області визначення	classname::classmember_name
::	Глобальний діапазон	::variable_name
.	Вибір елемента	object.member_name
->	Вибір елемента	pointer->membername
[]	Індексація	pointer[element]
()	Виклик функції	expression(parameters)
()	Будування значення	type(parameters)
sizeof	Розмір об'єкта	sizeof expression
sizeof	Розмір типу	sizeof(type)
++	Збільшення після	variable++
++	Збільшення до	++variable
--	Зменшення після	variable--
--	Зменшення до	-- variable
&	Адреса об'єкту	&variable

12

Ієрархія операцій в С

&	Адреса об'єкту	&variable
*	Розіменування	*pointer
new	Створення (розміщення)	new type
delete	Видалення	delete pointer
delete[]	Видалення масиву	delete pointer
~	Доповнення	~expression
!	Логічне НІ	! expression
+	Унарний плюс	+1
-	Унарний мінус	-1
()	Приведення	(type) expression
.*	Вибір елемента покажчика	object.*pointer
->	Вибір елемента покажчика	object->*pointer
*	Множення	expression * expression
/	Ділення	expression / expression
%	Взяття по модулю	expression % expression
+	Додавання (плюс)	expression + expression
-	Віднімання (мінус)	expression expression

13

Особливості виконання арифметичних виразів і операції присвоєння

Як відомо, послідовність дій така:

1. Спочатку обчислюється вираз праворуч;
2. Остаточний результат визначається змінної ліворуч.

В мові С типи даних мають певну ієрархію:

довгої подвійної точності (long double)
подвійної точності (double)
дійсні (float)
довгі (long int)
цілі(int)
символьні (char)

14

Приклад

```
char  c1, c2,c3;
int    y1, y2, y3;
float  f1, f2, f3;
c1='x';
c2=1000;
c3=6.02e23;
y1='x';
y2=1000.0;
y3=6.02e23;
f1=c1;
f2=c3;
f3=6.02e23;
```

Крім неявного пер з можна використовувати функцію явного перетворення, яка має вигляд :

(тип) <вираз>

---- наприклад: ---- **(int) 21.645**

15

Приклад

```
char  c1, c2,c3;
int    y1, y2, y3;
float  f1, f2, f3;
c1='x'; // нема перетворень-  'x'
c2=1000;  // ціла відсікається до символу -   -- - 104 10 - 'ш' або 'h'
c3=6.02e23;  // плаваюча відсікається до символу -  '\x0'
y1='x'; // символ розширюється до цілого –   120
y2=1000.0;  // плаваюча відсікається до цілого -   1000
y3=6.02e23;  // плаваюча відсікається до цілого -   0
f1=c1;  //символ розширюється до дійсного -   120.0
f2=c3;  // ціле розширюється до дійсного -   0.0
f3=6.02e23;  // // нема перетворень -   6.02e23
```

Крім неявного перетворення типів можна використовувати функцію явного перетворення, яка має вигляд :

(тип) <вираз>

---- наприклад: ---- (int) 21.645 ➡ 21

16

Цілі

1000_{10}

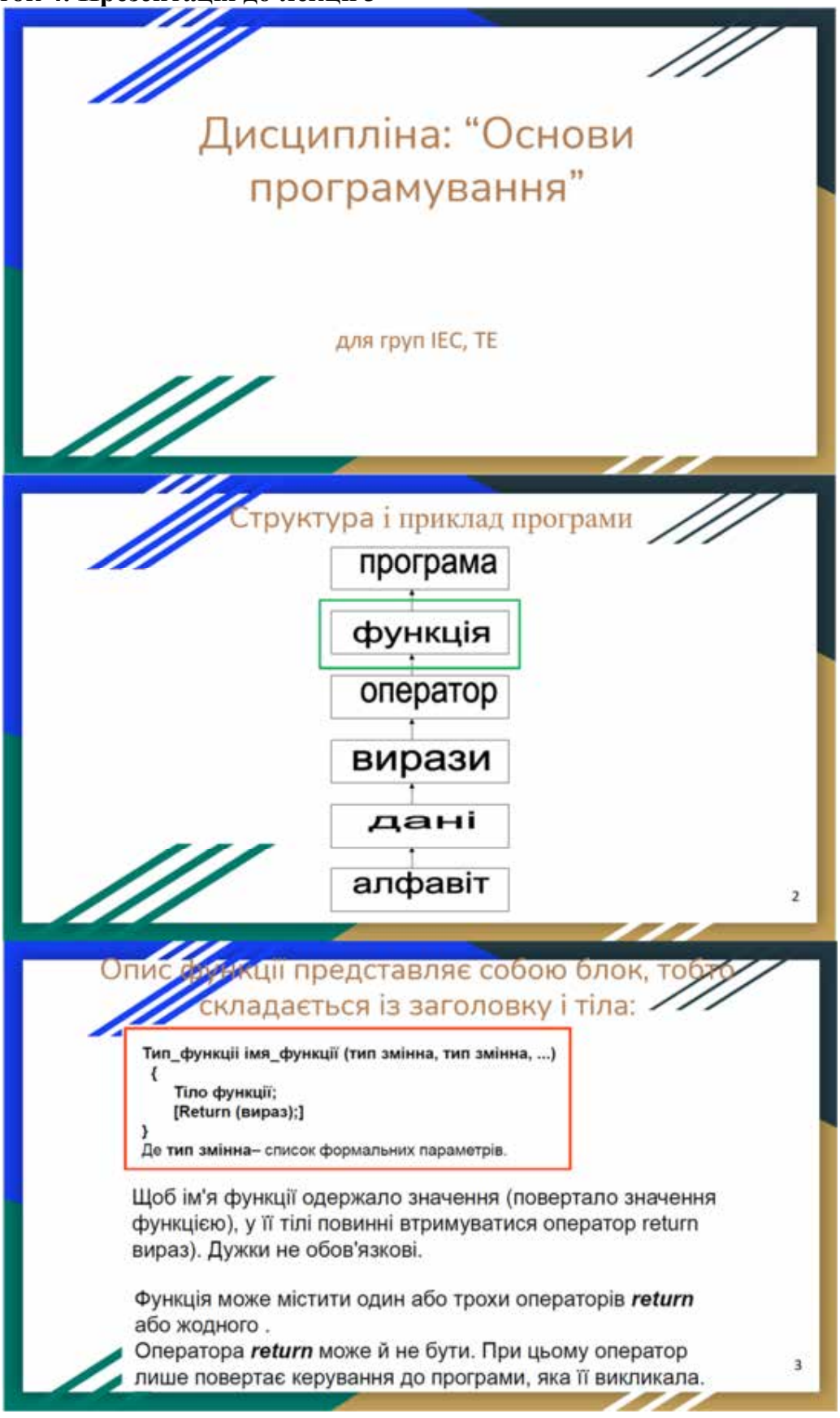
00000011 11101000$_2$ - 2 байта

11101000 - 1 байт **= -24_{10}**

01101000 - 1 байт **= 104_{10}** 'ш' або 'h'

17

Додаток 4. Презентація до лекції 3



Фактичні та формальні параметри

Виклик функції відбувається по імені **test(d)** і на відміну від інших мов, це звернення може входити до складу виразу або бути окремим оператором (це залежить від призначення функції).

Викликаючи функцію, визначають фактичні параметри.

Приклад: Розглянемо програму перетворення рядкових латинських букв в ПРОПИСНІ. У кодах ASCII великі літери кодуються від A - 65_{10} до Z -90_{10}, а маленькі від a - 97_{10} до z -122_{10}. Тому для переходу від маленьких до великих літер – треба відняти 32_{10}.

```
char test(char cf)
{
  if ((cf<'a') || (cf>'z')) return(cf);
  return(cf+'A'-'a'); // 65-97 = -32
}

void main (void )
{  char *st="test PrograM";
    for(int a=0;a< strlen(st);a++) printf(test( st[a]) );
}
```

У цій програмі формальним параметром є символьна змінна cf, а фактичним - покажчик st.

4

Як відбувається їх взаємодія?

Оскільки передача параметрів відбувається за значенням, в тілі функції не можна змінити значення змінних в викликаємої функції, які є фактичними параметрами.

Приклад:

```
/* Невірне використання параметрів */
   void change (int x, int y)
   {   int k=x;
         x=y;
         y=k;
   }
```

Однак, якщо в якості параметра передати покажчик на деяку змінну, то використовуючи операцію взяття адреси (**&**) можна змінити значення цієї змінної.

Приклад:

```
/* Вірне використання параметрів */
   void change (int *x, int *y)
   {  int k=*x;
       *x=*y;
       *y=k;
   }
```

При виклику такої функції в якості фактичних параметрів повинні бути використані не значення змінних, а їх адреси change (&a,&b);

5

Структура і приклад програми

6

Приклад програми

7

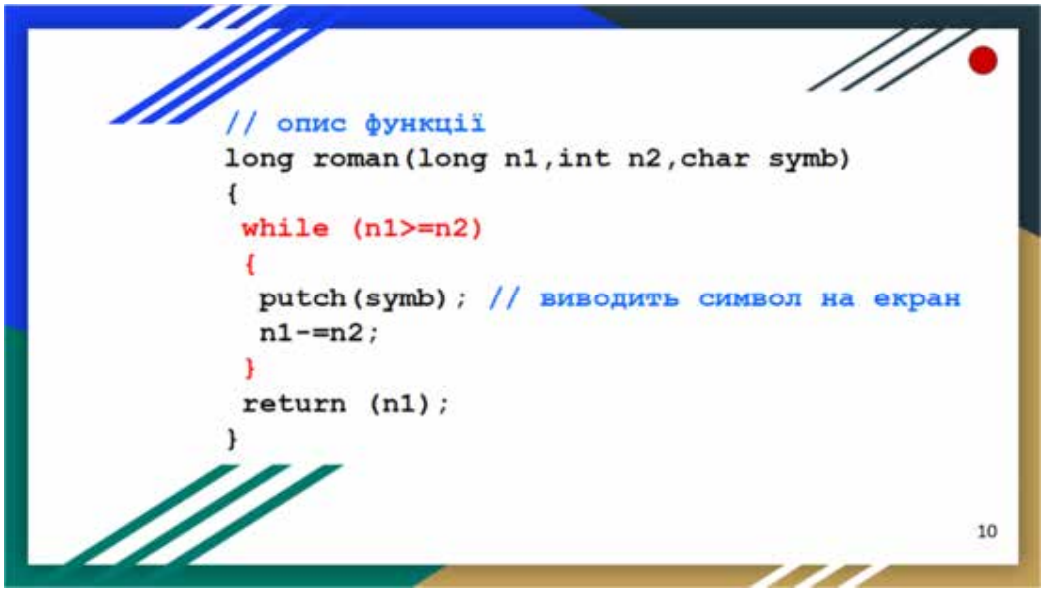
Приклад програми

```
/* арабське число пишеться римськими літерами */
//розділ препроцесорних директив
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
//розділ опису глобальних змінних та функцій
long roman(long,int,char); //заголовок функції
void main(void) //опис основної функції
{ long num;
  int rnum[7]={1000,500,100,50,10,5,1};
  char symb[7]={'M','D','C','L','X','V','I'};
  do
  {clrscr(); //очищення екрана
   printf("1000-M 500-D 100-C 50-L 10-X 5-X 1-I\n");
```

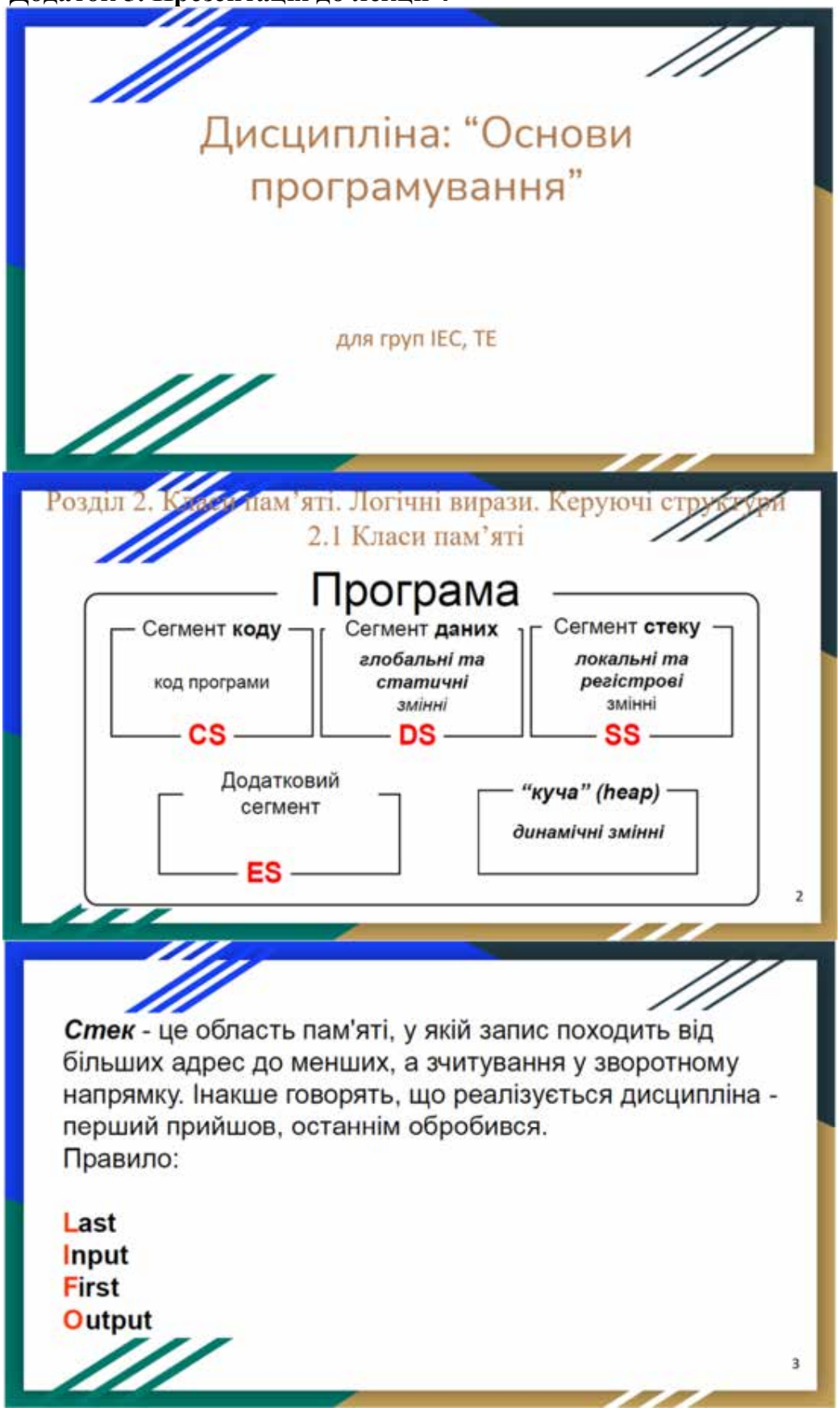
8

```
   print(" Введіть натуральне число (арабське)-> ");
    if(!(scanf("%ld",&num))||(num<0))
// перевірка правильності введення
    { printf("\n Помилка вводу ");
      printf("\n Для продовження натисніть будь-яку
клавішу ");
      getch();
      num=0;
    }
  } while (!num);
  printf(" Римське позначення числа -> ");
  // цикл визначення римских позначень
  for (int i=0;i<=6;i++)
  {   num=roman(num,rnum[i],rsymb[i]);  }
  getch();
} // завершення основної функції
```

9


// опис функції
long roman(long n1,int n2,char symb)
{
while (n1>=n2)
{
putch(symb); // виводить символ на екран
n1-=n2;
}
return (n1);
}
10

Додаток 5. Презентація до лекції 4



Клас пам'яті визначає місце, де розташовано об'єкт (внутрішні регістри процесора, сегмент даних, сегмент стека) і одночасно час існування.

У мові С існують 4 класу пам'яті:

1. Автоматичний (auto);
2. Реєстровий (register);
3. Статичний (static);
4. Зовнішній (extern).

4

Автоматичні змінні

Описуються в блоці таким чином

```
{ auto int a=123;
  auto char b;
  auto float c=45.28;
}
```

Зазвичай слово **auto** пропускається. Область дії автоматичних змінних - у межах блоку, після виходу із блоку їх значення стають невизначеними (усі що завгодно).

Якщо в деякому блоці є ***внутрішні блоки***, то область дії автоматичних змінних, описаних у зовнішньому блоці, поширюється й на внутрішні. Тобто у внутрішньому блоці їх можна не описувати.

5

```
main()
// початок зовнішнього блоку
{ int x=1; char z='w'; // (int) z=119
    if (x=1)
    { // внутрішній блок
          int y=2;
          float z=-345.5674;
          cout<<"y= "<<y<<"z= "<<z<<endl;
    }
printf (" x=%d zd=%d ze=%e zc=%c\n",
x,z,z,z);
  getch();
}
```

6

Реєстрові змінні

Як відомо, дані можуть розміщатися як в оперативній пам'яті, так і в реєстрах. Використання реєстрів зменшує кількість пересилань і прискорює виконання програми. Тому можливе використання реєстрових змінних, опис яких має вигляд

```
{ register int x=2;
}
```

Однак оскільки мовам високого рівня реєстри недоступні, про їхній стан відомо тільки компілятору, то використання реєстрів буде можливо тільки тоді, коли є вільний реєстр, і якщо реєстр може вмістити відзначену змінну.

Ефективні тільки трохи перших таких описів. Крім того, у реєстрах можуть зберігатися тільки змінні певних типів; найчастіше - це int, char або покажчик. Існує й інше обмеження на використання реєстрових змінних: до них не можна застосовувати операцію узяття адреси **&**.

7

Статичні змінні Задаються описом

```
{ static int a;
  static float b=3.15;
}
```

Після виходу із блоку їх значення зберігається. Воно буде початковим при наступному вході в цей блок.

```
void lvars ()
{ int z=0;
  z++;
  printf (" %d",z);
}
main ()
{ lvars();
  lvars ();
  lvars ();
}
```

На екрані: 1 1 1

Інший варіант(static int z)

```
void statvars ()
{ static int z;
  z++;
  printf (" %d",z);
}
main()
{ statvars ();
  statvars ();
  statvars ();
}
```

на екрані: 1 2 3

Якщо початкове значення не відмічено, за замовченням воно **дорівнює нулю**.

8

Глобальні змінні

Глобальні змінні є глобальними й доступні будь-яким функціям. Вони визначаються за межами функцій.

```
// vars.h
extern char* sms;
extern void global();
```

```
// main.cpp - головна програма
#include "conio.h"
#include "gl.h"
char *sms = "тест глобальних
змінних";
int main()
{  global(); }
```

а файл gl.h друкує рядок:

```
// gl.h
#include <iostream.h>
#include " vars.h"
void global( )
{ printf("%s "\n", sms);  }
```

```
// main.cpp -  головна програма
#include "conio.h"
#include <iostream.h>
// --- #include "gl.h"
// #include " vars.h"
extern char* sms;
extern void global();
void global( )
{printf("%s "\n", sms); }
char *sms = "тест глобальних
змінних";
int main()
{  global(); }
```

на екрані: "тест глобальних змінних"

9

Опис глобальних змінних діє на всі функції, які розташовані нижче.Статичні й глобальні змінні розміщуються в сегменті даних.

Зовнішні змінні діють у межах декількох файлів, але їх опису повинні бути продубльовані.

При виникненні ситуації одночасного звернення до глобальної та локальної змінної з однаковими іменами, С++ надає пріоритет локальній змінній.

Коли вам необхідно звернутися до глобальної змінної, чиє ім'я конфліктує з іменем локальної змінної, необхідно використовувати глобальний оператор дозволу С++ (::).

10

Приклад

```
number; // звернення до локальної змінної
::number; // звернення до глобальної змінної
```

Ця функція використовує оператор глобального дозволу для звернення до глобальної змінної:

```
#include <iostream.h>
 int number = 1001; // Глобальна змінна
 void show_numbers(int number)
 {  printf(" Локальна змінна number містить %d ",number);
    printf(" Глобальна змінна number містить %d ", ::number);
 }

void main(void)
{
  int some_value = 2002;
  show_numbers(some_value) ;
}
```

на екрані: Локальна змінна number містить 2002
на екрані: Глобальна змінна number містить 1001

11

Додаток 6. Презентація до лекції 5

Дисципліна: “Основи програмування”

для груп ІЕС, ТЕ

2.2. Логічні вирази

Відношення, логічні операції, умовні вирази

Існують ***чотири операції відносин*** **>, >=, <, <=**
і ***дві операції рівності*** **= =** та **!=**.

Чотири перші операції між собою рівноправні, операції рівності мають менший пріоритет.

У логічних виразах використовуються 3 логічних операції:

1. **!** - “заперечення”
2. **&&** - “І”, логічне множення
3. **||** - “АБО” з врахуванням ієрархії (**!, &&, ||**).

Логічні операції фактично виконуються над цілими, дійсними та покажчиками.

При цьому ***FALSE = 0, TRUE ≠ 0***.

2

Для роботи з окремими бітами поруч зі звичайними логічними операціями передбачено **6 порозрядних** логічних операцій:

1. **&** - порозрядне І: множення;
2. **|** - порозрядне АБО: додавання;
3. **^** - порозрядне виключаюче АБО;
4. **~** - доповнення;
5. **<<** - зсув вліво;
6. **>>** - зсув вправо.

x	y	x&y	x|y	x^y	x~	y~	y<<1	y>>2
0	0	0	0	0	1	1	*	*
0	1	0	1	1	1	0	*	*
1	0	0	1	1	0	1	*	*
1	1	1	1	0	0	0	*	*

Доповнення є унарною операцією, усі інші - бінарні НЕ можуть виконуватися над операціями типу ***float*** або ***double***.

3

Операції &, | можна використовувати для керування окремими розрядами (бітами).

Наприклад:
Роздивимося операцію визначення стану принтера в DOS. Кожен принтер (LPT1, LPT2, LPT3) має по три порта: порт виводу (базовий порт), порт стану і порт керування.
Адреси портів строго не фіксовані. Припустимо, в області даних BIOS по адресам ***0040:0008 (9), 0040:000A(B), 0040:000C(D)*** зберігаються адреси базових портів для LPT1, LPT2, LPT3.

№ біта	Опис
0-2	Не використовуються, зазвичай встановлено в 1
3	ERROR - помилка принтера – нема / є (0/1);
4	SLCT – принтер приєднано / не приєднано (1/0);
5	PE - бумага є / нема (0/1);
6	ACK - 1 – принтер виводить наступний символ / ні (0/1)
7	BUSY – принтер зайнятий / вільний (0/1)

4

Приклад програми зчитування стану принтеру

```
#include <dos.h>
main()
{ int statusport;   // Номер порту
  char stat; // Байт статуса
  int i; // Лічильник
  // Визначення адрес портів принтера - peek (сегмент, зміщення)
  statusport =peek(0x40, 0x0A); // взяти адресу порту статусу
  printf("Порт LPT = %03X\n", statusport);

 // перевірка станів
  printf("\n Вимкніть принтер\n ");
  printf(" Натисніть будь-яку клавішу \n");  getch();

  stat=inportb(statusport);
  printf(" Стан принтера - ");
  for (i=7; i>=0; i--)    if ( (stat>>i) & 1 ) printf("1");  else    printf("0");
  printf("\n Epson стан "принтер вимкнено" - 11110111\n");
  // помилка принтера (1) - немає (0)
}
```

5

Скажімо є такий запис стану: **11110111**

```
for (i=7; i>=0; i--)
  if ( (stat>>i) & 1 ) printf("1");
  else    printf("0");
```

i=7 1_{10}	i=6 3_{10}	i=5 7_{10}	i=4 15_{10}	i=3 30_{10}
10110111	11110111	11110111	11110111	11110111
* 00000001	* 00000011	* 00000111	* 00001111	* 00011110
00000001	00000001	00000001	00000001	00000001
00000001	00000001	00000001	00000001	00000000
На екрані: 1	11	111	1111	11110

i=2 61_{10}	i=1 123_{10}	i=0 247_{10}
11110111	11110111	11110111
* 00111101	* 01111011	* 11110111
00000001	00000001	00000001
00000001	00000001	00000001
111101	1111011	11110111

6

Є відмінність в виконанні звичайних логічних операцій і порозрядних.

(1&&2 == 1) (0&&2 == 0) (1&&0 == 0) (1&&3 == 1)

В цьому випадку операція логічного **І** (**&&**) виконує значення **1**, якщо обидва операнда мають ***НЕ*** нульові значення.

Якщо один із операндів рівен **0**, то результат також дорівнює **0.** Якщо значення першого операнда = **0**, то другий операнд **НЕ** вираховується.

(1&2 == 0) (0&2 == 0) (1&0 == 0) (1&3 == 1)

7

Операція доповнення **~** в кожному біті змінює $1 \Rightarrow 0$ і $0 \Rightarrow 1$.

- Операція зсуву **<<** і **>>** виконується на таку кількість розрядів, яка відмічена в операнді справа.

Наприклад: **x<<3** $\Leftrightarrow$ **x*2³ або y>>4** $\Leftrightarrow \frac{y}{2^4}$

Зсув вліво(**<<**) відповідає множенню першого операнда на степінь числа 2, рівну другому операнду.

Зсув вправо(**>>**) відповідає діленню першого операнда на степінь числа 2, рівну другому операнду.

Операція зсуву вправо (**>>**) виконується в різних машинах по різному:

1. З фіксацією знакового розряду (як арифметичний зсув);
2. Без фіксації (логічний зсув).

8

При зсуві вправо метод заповнення вільних лівих бітів залежить від типу першого операнда.

Якщо тип unsigned, то вільні ліві біти стають в нуль. В іншому випадку вони заповнюються копією знакового біта. Результат операції зсуву не визначено, якщо другий операнд від'ємний.

```
int_t=15 ->  0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
int_t >> 3
int_t=1 ->  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

int_tt1=-10 ->  1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0
int_t1 >> 3
int_t1=-2 ->  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0

unsigned_t=12

unsigned_t=12 ->  0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
unsigned_t >> 3
unsigned_t=1 ->  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

unsigned_t=-8

unsigned_t=65528 ->  1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
unsigned_t >> 3
unsigned_t=8191 ->  0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
```

9

Умовні вирази

В деяких випадках умовні оператори можна замінити умовним виразом, що має вид:

e1 ? e2 : e3,

де **e1**, **e2**, **e3** - вирази. Якщо значення виразу **e1** відмінне від 0 (ІСТИНА), то виконується вираз **e2**.

Інакше - вираз **e3**.

Аналог:

```
If (e1) e2;
else e3
```

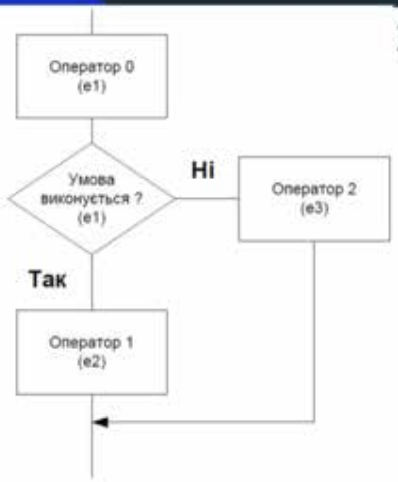
Наприклад: вибір максимального із двох чисел

```
z = a>b ? a : b;      // z = max (a, b)
```

10

e1 ? e2 : e3

Спочатку обчислюється вираз **e1**.




```
Z = (A > B) ? A : B;   // Z = MAX(A,B)
```

11

Дано:
масив n=40, необхідно реалізувати вивід елементів рядками по k=5 в кожному рядку і пробілом між елементами

```
int i, n=40, k=5, a[40];

for (i = 0; i < n; i++)
 if (i%k==k-1 || i==n-1) printf ("%6d%c",a[i],'\n' );
 else printf ("%6d%c",a[i],' ');

int i, n=40, k=5, a[40];

for (i = 0; i < n; i++)
  printf("%6d%c",a[i],(i%k==k-1 || i==k-1) ? '\n' : ' ');
```

12

Генерація та вивід масиву псевдодійсних чисел

При генеруванні дійсного числа варто враховувати
3 складові: знак ціла складова.дійсна складова

```
printf ("\nМасив дійсних чисел \n");
 randomize(); // ініціалізує генератор іншої випадкової величи
 int range=1000, precis=1000, N=4, j,i; // N=4 – кількість елементів в рядку
 float num;

for ( j = 0; pow(10,j) < precis; j++);//
//Функція random(N) генерує цілі значення в диапазоні від 0 до N-1
for ( i = 0; i < n; i++)
{
  num=pow(-1,random(2))*random(range+1)+random(precis+1)/(float)precis;

//вивід на екран значень  таблицею
printf ("%10.*f%c",j,num,(i%N==N-1 || i==n-1)?'\n':'|');
 }
getch();
```

13

Псевдогенератор дійсних чисел

```
int  random( int range_min, int range_max )
{
   return (range_min+rand()%(range_max-range_min+1));
//rand() - функція повертає випадкове ціле число в діапазоні від нуля до
RAND_MAX.
}

int main()
{
.....

srand( time( NULL ) );
// srand(time(NULL)) встановлює в якості бази для генератора поточний
час

}
```

14

Додаток 7. Презентація до лекції 6

Дисципліна: “Основи програмування”

для груп IEC, TE

2.3. Керуючі структури

Визначають, які операнды необхідно виконувати й у якому порядку або визначають потік керування в програмі. Будь-який алгоритм може бути реалізовано за допомогою трьох керуючих структур:

1.Послідовне виконання

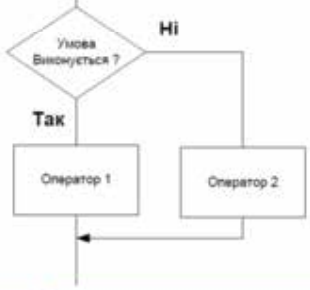
2. Умовне виконання

3. Цикл

2

Умовні оператори. Умовний оператор if

if (вираз) оператор 1;
[else оператор2;]
Якщо ***вираз*** ≠ 0 ⇒ Виконується ***оператор 1***
Якщо ***вираз*** = 0 ⇒ Виконується ***оператор 2***



3

Приклад:

```
// чи можна побудувати трикутник ?
main ()
{ float a, b, c;
  unsigned i;
  printf (“Введіть сторони a, b, c\n”);
  scanf (“ %f %f %f”, &a, &b, &c);

  i = (a+b>c)&&(b+c>a)&&(a+c>b);

  if (i)  printf (“Трикутник можна побудувати \n”);
  else  printf (“Трикутник побудувати неможливо !!! \n”);
}
```

4

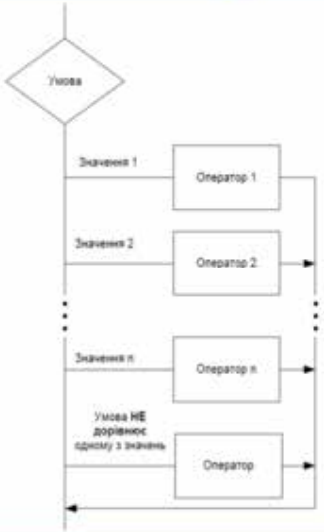
Перемикач switch

У тих випадках, коли виникає вибір з багатьма можливими результатами, доцільно використовувати перемикач.

Структура:

```
switch (вираз цілого типу)
{
 case константа 1: [оператор 1; break;]  // вираз = константа 1

  case константа 2: [оператор 2; break;]  // вираз = константа 2

. . . . . .

  case константа n: [оператор n; break;]  // вираз = константа n

  [default :оператор n+1;]  // вираз ≠ НІ одної з констант

 }
```

5



6

Приклад: ввести номер місяця та вивести його назву:

```
main ()
{ int month;
   printf ("Введіть номер місяця \n");
   scanf (" %2d",&month);
   switch (month)
   { case 1: printf ("Це січень. \n");
     case 2: printf ("Це лютий. \n");
      .....
     case 12: printf ("Це грудень. \n");
     default : printf ("Такого місяця немає. \n");
   }
}
```

Якщо ввести 2, то на екран буде виведено:

Це лютий.
.....
Це грудень.
Такого місяця немає.

7

Приклад: ввести номер місяця та вивести його назву

```
main ()
{ int month;
   printf ("Введіть номер місяця \n");
   scanf (" %2d",&month);
   switch (month)
   { case 1: printf ("Це січень. \n"); break;
     case 2: printf ("Це лютий. \n"); break;
      .....
     case 12: printf ("Це грудень. \n"); break;
     default : printf ("Такого місяця немає. \n");
   }
}
```

Якщо ввести 2, то на екран буде виведено:

Це лютий.

8

Константа вибору може бути й символом

```
main ()
{ char c;
c=getch(); // c=getche() – введення символу з друком на екрані
switch (c)
{   case 'a': printf ("Вибір зроблено на користь символу 'a'. \n"); break;
     .....
    case 'z': printf ("Вибір зроблено на користь символу 'z'. \n"); break;
    default : printf ("Вибір НЕ зроблено ☹ \n"); break;
  }
}
```

Один вибір за декількома параметрами вибору:

```
switch (month)
{ case 1:
  case 2:
  case 12: printf ("Це зима. \n"); break;
  case 3:
  case 4:
  case 5: printf ("Це весна. \n");
 }
```

9

Оператори циклу. Цикл с передумовою while

Оператор має структуру:


```
while (умова)
[{ ]
   оператор;
   [вираз]
[ }]
```

10

Завдання: Написати програму, яка визначає, чи є число n простим, тобто ділиться лише саме на себе і на 1. При створенні використати цикл while.

```
main ()
{ int n, i=1;
  printf ("Введіть ціле число (макс. 99999) \n");
  scanf (" %5d", &n);
   while (++i<n)
    if (n%i==0)        // остача від ділення
    { printf ("число НЕ просте \n");
      break;
     }
    if (i==n) printf ("число просте \n");
}
```

11

Можна використати оператор **continue** – повернутися на початок циклу.

Завдання: Написати програму, яка для чисел від 1 до n виводить всі числа, що НЕ кратні 5.

```
main ()
{ int n, i=0;
  printf ("Введіть ціле число \n");
  scanf ("%5d",&n);
   while (++i<n)
  {    if (i%5==0)   continue;
       printf ("%5d \n",i);
  }
}
```

12

Цикл з післяумовою do-while

Коли ж оператори циклу повинні виконуватися принаймні один раз, то можна використовувати інший варіант: **do** оператор **while** (умова).


```
do
{ оператор
  [вираз]
} while (умова)
```

13

Приклад: Перевірка на коректне ведення числа (перший символ-число).

```
main ()
{ int n, err;
   do
   { err=0;
     printf ("Enter number (max 99999) \n");
// не більше 5-ти символів
    if (!scanf ("%5d",&n))
    { printf ("Error input number \n Try again\n");
      err=1; // тригер помилки
      fflush(stdin); //очищення буферу вводу
    }
   }  while (err);
   printf ("Number is: %d\n",n);
}
```

14

Приклад: Перевірка на коректне введення числа (перший символ-число).

```
Enter number(max 99999)
s9999
Error input number
 Try again
Enter number(max 99999)
9s999
Number is: 9
```

```
Enter number(max 99999)
1234567
Number is: 12345
```

15

Оператор циклу for має вигляд:

for** (вираз 1; вираз 2; вираз 3) **оператор;
або
***for** (вираз 1; вираз 2; вираз 3)*
{ оператори;
}
І виконується наступним чином:
вираз 1;
***while** (вираз 2)*
{оператор; вираз 3;}

вираз **1** виконується лише ***один*** раз.
вираз **2** еквівалентно ***умові*** кінця циклу, *вираз* **3** виконується ***кожен раз*** в циклі.

16

Приклад: Перевірка на коректне ведення числа (перший символ-число).

17

Приклад: знаходження n!

```
void nfactor ()
{ int i, n, nfact=1;
  printf (“Введіть ціле число (макс. 99) \n”);
  scanf (“%2d”,&n);
  for (i=1; i<=n; i++) nfact*=i;
  printf (“ Факторіал %2d = %6d \n”, n, nfact);
}
```

18

Іноді в циклі for можна одразу використовувати декілька індексів: **вираз 1, вираз 2.**

Кома розділяє два вирази, які виконуються зліва направо.

Наприклад: Створити програму, яка переставить **символи в рядку**

```
reverse (char s[ ])
{ int c, i, j;
  printf (“Введіть рядок \n”);
  scanf (“ %s”,s);
  for (i=0, j=strlen (s)-1; i<j; i++, j--)
  {  c=s[i];   s[i]=s[j]; s[j]=c;  }
}
```

19

Варіанти обміну двух символів (цілі числа)

```
c=a;
a=b;
b=c;
```

```
a=a+b;
b=a-b;
a=a-b;
```

```
a^=b^=a^=b
```

```
reverse (char s[ ])
{ int c, i, j;
  printf (“Введіть рядок \n”);
  scanf (“ %s”,s);
  for (i=0, j=strlen (s)-1; i<j; i++, j--)
  {  c=s[i];   s[i]=s[j]; s[j]=c;  }
}
```

```
reverse (char s[ ])
{ int i, j;
  printf (“Введіть рядок \n”);
  gets(s);
  for (i=0, j=strlen (s)-1; i<j; i++, j--)
    str[i]^=str[j]^=str[i]^=str[j];
}
```

20

Варіанти обміну двух символів (цілі числа)

```
a=a+b;
b=a-b;
a=a-b;
```

```
a=5 ; b=7;
a=5+7=12;
b=12-7=5;
a=12-5=7;
a=7; b=5;
```

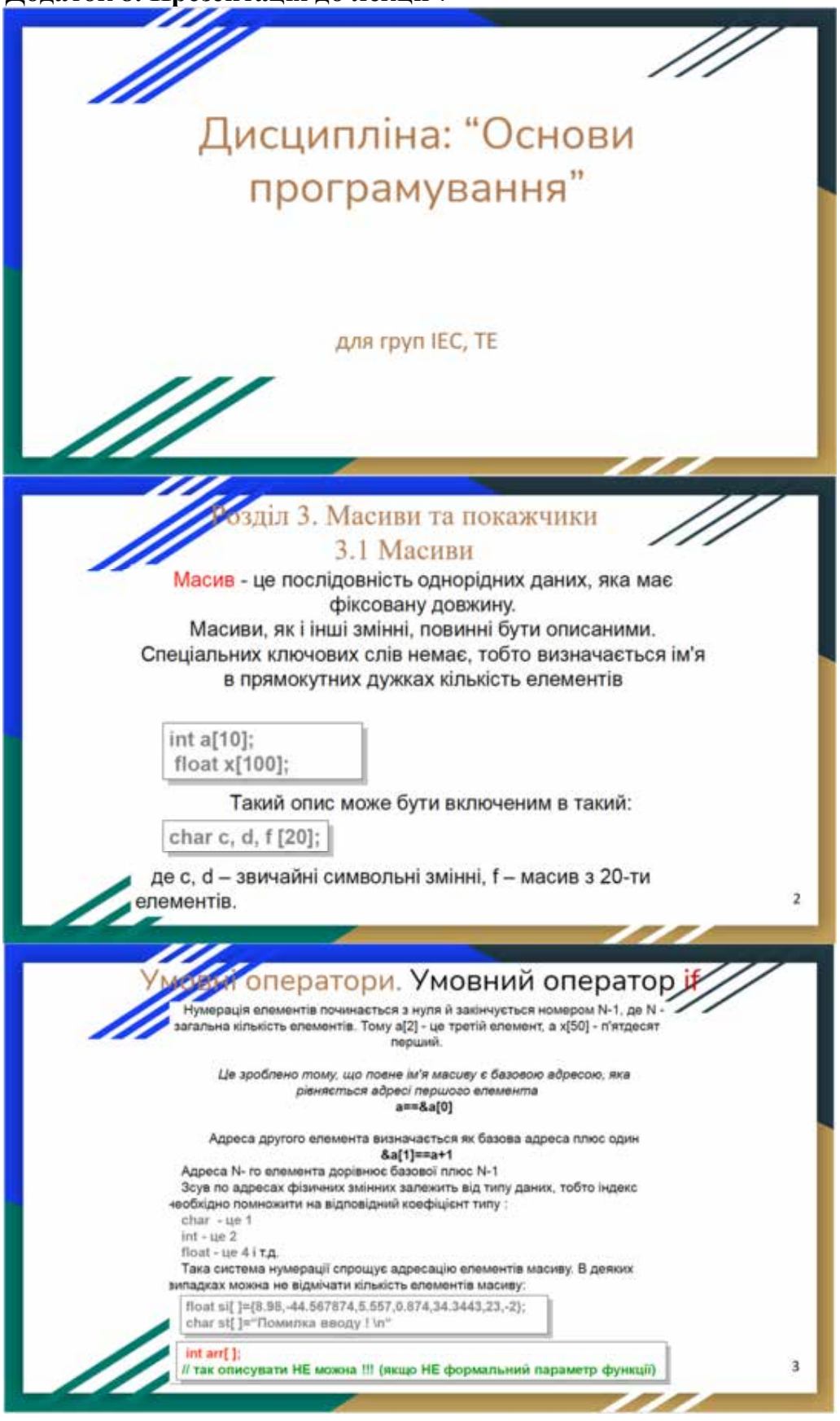
```
a^=b^=a^=b
```

```
a=5=00000101; b=7=00000111;
a^=b (a = a^b)
b^=a (b = b^a)
a^=b (a = a^b)

a=7=00000111; b=5= 00000101;
```

```
00000101-a (5)
00000111-b (7)
00000010- a = a^b (2)

00000111-b (7)
00000010-a (12)
00000101-b = b^a (5)

00000010-a (12)
00000101-b (5)
00000111-a =a^b (7)
```

21

Додаток 8. Презентація до лекції 7



Дисципліна: “Основи програмування”

для груп ІЕС, ТЕ

Розділ 3. Масиви та покажчики

3.1 Масиви

Масив - це послідовність однорідних даних, яка має фіксовану довжину.

Масиви, як і інші змінні, повинні бути описаними.

Спеціальних ключових слів немає, тобто визначається ім'я в прямокутних дужках кількість елементів

```
int a[10];
float x[100];
```

Такий опис може бути включеним в такий:

```
char c, d, f [20];
```

де c, d – звичайні символьні змінні, f – масив з 20-ти елементів.

2

Умовний оператор. Умовний оператор if

Нумерація елементів починається з нуля й закінчується номером N-1, де N - загальна кількість елементів. Тому a[2] - це третій елемент, а x[50] - п'ятдесят перший.

Це зроблено тому, що повне ім'я масиву є базовою адресою, яка рівняється адресі першого елемента

a==&a[0]

Адреса другого елемента визначається як базова адреса плюс один

&a[1]==a+1

Адреса N- го елемента дорівнює базової плюс N-1

Зсув по адресах фізичних змінних залежить від типу даних, тобто індекс необхідно помножити на відповідний коефіцієнт типу :

char - це 1

int - це 2

float - це 4 і т.д.

Така система нумерації спрощує адресацію елементів масиву. В деяких випадках можна не відмічати кількість елементів масиву:

```
float si[ ]={8.98,-44.567874,5.557,0.874,34.3443,23,-2};
char st[ ]="Помилка вводу ! \n"
```

```
int arr[ ];
// так описувати НЕ можна !!! (якщо НЕ формальний параметр функції)
```

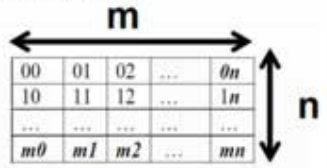
3

Багатовимірні масиви

Масив може містити не тільки дані як простих, так і похідних типів. Можна розглядати масив масивів.

Двомірний масив `int a[10][10]`
Окремий елемент a[2][3].
Перший індекс - рядки, другий стовпчики. Тому це буде елемент із третього рядка й четвертого стовпчика.

float matr[m][n]



Початкові значення багатомірних масивів задаються так:

```
int z[3][2]={{1, 2},{4, 5},{7, 8}};
```

4

Приклад виведення для розробника

```
#include <stdio.h>
int main()
{
    int i,j,z[3][2]={{1, 2},{4, 5},{7, 8}};
 for (i=0;i<3;i++)
 {printf("\n");
  for (j=0;j<2; j++) printf("z[%d][%d]=%d  ",i,j,z[i][j]);
 }
    return 0;
}
```

```
z[0][0]=1  z[0][1]=2
z[1][0]=4  z[1][1]=5
z[2][0]=7  z[2][1]=8
```

5

Приклад виведення для користувача

```
#include <stdio.h>
int main()
{
    int i,j,z[3][2]={{1, 2},{4, 5},{7, 8}};
 for (i=0;i<3;i++)
 {printf("\n");
  for (j=0;j<2; j++) printf("z[%d][%d]=%d  ",i+1,j+1,z[i][j]);
 }
    return 0;
}
```

```
z[1][1]=1  z[1][2]=2
z[2][1]=4  z[2][2]=5
z[3][1]=7  z[3][2]=8
```

6

Оператори керування ходом циклу

Оператор ***break*** – припинення циклу;
Оператор ***continue*** – обійти деякі оператори не виходячі із циклу (перейти на початок циклу);
Оператор ***goto*** – перейти на певну мітку, також ним реалізується вихід із певної кількості вкладених циклів. Мітка спеціально не описується.

Приклад: в матриці цілих чисел знайти перше від'ємне числоі вивести його координати.

```
main ()
{ int a[20][20], i, j, size1=20, size2=20;
……..
  for (i=0; i<size1; i++)
    for(j=0; j<size2; j++)  if (a[i][j]<0) goto label;
…….
  label: printf (" [ i=%2d ][ j=%2d ] \n", i, j);
}
```

7

Строкові масиви

Строка – це одномірний масив символів, котрий закінчується нулевим символом.

char str[ ]="Скоро атестація ☹"

Такий масив можна вводити та виводити з використанням символу перетворення ***s***.

При цьому вивід продовжується до появлення символа кінця рядку – '\0'.

printf("%s",st); ⟺ **printf(st);**

8

Ніяких стандартних дій над такими масивами НЕ передбачено

Для цього використовуються функції з бібліотек (бібліотеки ***string.h, stdlib.h***):

1. ***strcat (str1, str2)*** – прикріплення рядка(конкатенація);
2. ***strcmp (str1, str2)*** – порівняння двох рядків;
3. ***strcmpi (str1, str2); stricmp (str1, str2)*** – порівняння двох рядків (без врахування реєстра символів);
4. ***strcpy (str1, str2)*** – копіювання рядка;
5. ***strlen (str1)*** – визначення кількості символів в рядку;
6. ***atof(str1)*** – перетворення рядка ***str*** в число;
7. ***atoi (str)*** – перетворення рядка ***str*** в десяткове ціле;
8. ***strtod*** (***str1 ,&str2***) – перетворення рядка ***str1*** в число подвійної точності;
9. ***strtol*** (***str1, &str2, base***) – перетворення рядка ***str1*** в довге ціле з покажчиком **base**;

9

1. ***strcat (str1, str2)*** – прикріплення рядка (конкатенація)

```
char str1[]="Просто ";
char str2[]="тест";
printf("\n str1=%s, str2=%s ",str1,str2);
strcat (str1, str2);
printf(", strcat(str1, str2)= %s",str1);
```

```
str1=Просто , str2=тест , strcat(str1, str2)= Просто тест
```

2. ***strcmp (str1, str2)*** – порівняння двох рядків;

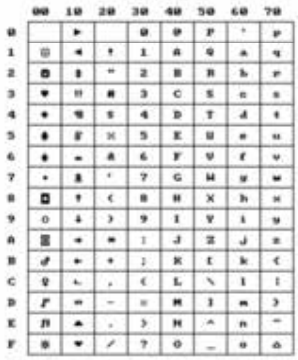
$$\text{strcmp}(str1, str2) = str1 - str2 = \begin{cases} >0, \text{ якщо ASCI}(str1[i]) > \text{ASCI}(str2[i]) \\ 0, \text{ якщо ASCI}(str1[i]) = \text{ASCI}(str2[i]) \\ <0, \text{ якщо ASCI}(str1[i]) < \text{ASCI}(str2[i]) \end{cases}$$

```
str1=Просто тест, str2=тест,  strcmp(str1, str2)= -83
```

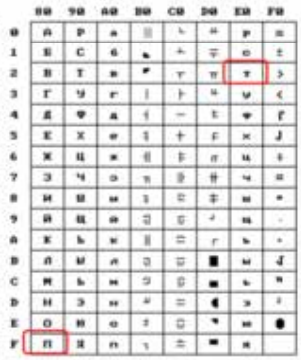
П – 143 т - 226

10

Основна таблиця ASCII



Розширена таблиця ASCII (cp866)



48-57 ----- 0..9 65-90 ----- A..Z

97-122 ----- a..z 128-159 ----- А..Я

160-175 ----- а..п 224-239 ----- р..я

11

3. ***strcmpi (str1, str2); stricmp (str1, str2)*** – порівняння двох рядків (вважає малі і великі літери однаковими **тільки для латинських літер**);

strcmpi (str1, str2)=str1-str2

stricmp (str1, str2)=str1-str2

```
str1=сест, str2=тест,  strcmpi(str1, str2)= -1
str1=Тест, str2=тест,  stricmp(str1, str2)= -80
str1=Test, str2=test,  stricmp(str1, str2)= 0
```

12

4. ***strcpy (str1, str2)*** – копіювання рядка

```
str1=Test, str2=test,  strcpy (str1, str2) = test
```

5. ***strlen (str1)*** - визначення кількості символів в рядку без врахування кінцевого нуля:

```
strlen ("abcde") == 5
```

6. ***atof(str1)*** – перетворення рядка ***str*** в число.
Розпізнає символьне представлення числа з плаваючою крапкою, якщо символи відповідають формату: **[пробіли] [знак] [ddd] [.] [ddd] [e|E[знак]ddd]**, де
[ddd] – число;
[e|E] – вказівник покажчика степеня при експоненціальній формі;
Припиняє перетворення на першому нерозпінаному (не цифра) символі.
У випадку переповнення ***atof*** повертає +(-) HUGE_VAL, глобальна змінна ***errno*** встановлюється в ERANGE.

```
char str3[ ]="-345.575784876";
printf("\n\n str3=%s, atof(str1) %20.18f",str3,atof(str3));
strcpy (str3, "-345.54563782rt75");
printf("\n\n str3=%s, atof(str1) %20.18f",str3,atof(str3));
```

```
str3=-345.575784876, atof(str1) -345.575784876000000000

str3=-345.54563782rt75, atof(str1) -345.545637820000024000
```

13

7. ***atoi (str1)*** – перетворення рядка ***str*** в десяткове ціле.
Розпізнає символьне представлення десяткового числа, якщо символи відповідають формату: **[пробіли] [знак] [ddd]**,
де [ddd] – число;
Припиняє перетворення на першому нерозпізнаному символі.
У випадку переповнення ***atoi*** повертає+(-)HUGE_VAL, глобальна змінна ***errno*** встановлюється в ERANGE.

```
char str5[]="-31534";
printf("str5=%s, atoi (str)= %d", str5, atoi (str5));
strcpy (str5,"-315yru34");
printf("str5=%s, atoi (str5)= %d", str5, atoi (str5));
```

```
str5=-31534, atoi (str)= -31534
str5=-315yru34, atoi (str5)= -315
str5=1234567891, atoi (str5)= 1234567891
str5=12345678912, atoi (str5)= -539222976
```

14

8. *double* ***strtod*** *(const char* ****str1,*** *char* ***** error*** *)* - перетворює рядки ***str1*** в число подвійної точності без втрати значення.

Функція повертає перетворене значення, якщо воно існує.
Якщо ***str1*** ≠ **NULL**, то вказівник на символ, наступний після останнього обробленого символа, зберігається в ***error***.
Якщо НЕ відбувається ніяких перетворень, то повертається нуль, а значення ***error*** зберігається в тій позиції, на яку посилається ***str1***.

```
char *error;
 char str4[ ]="-34536.45637824";
 printf("str4=%s, strtod(str4,&error)= %15.10lf, error=%s",str4,strtod(str4, &error),error);
strcpy (str4,"-34536.45ad64");
 printf("str4=%s, strtod(str4,&error)= %15.10lf,error=%s",str4,strtod(str4, &error),error);
```

```
str4=-34536.45637824, strtod(str4,&error)= -34536.4563782400 error=

str4=-34536.45ad64, strtod(str4,&error)= -34536.4500000000 error=ad64
```

15

9. *long int* ***strtol*** *(const char* ****str1,*** *char* *****error,*** *int* ***base)***
- перетворює рядки ***str1*** в довге ціле у відповідності з вказаним **base**, яке повинне знаходитися в діапазоні: 2 - 36 включно або бути рівним нулю.

Якщо **base** лежить між **2** і **36**, це значення використовується як основа системи числення данного числа.
Якщо **base** = **0**, початкові символи рядка, на які вказує ***str1***, використовуются для визначення основи.

- Якщо перший символ - **0** і другий - одна з цифр від **1** до **7**, то рядок інтерполюється як *вісімкове ціле*;
- Якщо перший символ **0**, а другий **x** або**X**, тоді рядок визначається як *шістнадцяткове ціле*;
- Якщо перший символ належить послідовності **1-9**, то рядок визначається як *десяткове ціле*.

16

Приклад використання функції **strtol**

```
char str6[]="-3445637";
 int base=0;
 printf("str6=%s, strtol(str6,&error,base)= %ld ,error=%s,
base=%d", str6, strtol(str6,&error, base),error, base);

 strcpy (str6,"-34456g37");
 printf("str6=%s, strtol(str6,&error,base)= %ld ,error=%s,
base=%d", str6, strtol(str6,&error, base),error, base);
```

```
str6=-3445637, strtol(str6,&error,base)= -3445637  error=
str6=-34456g37, strtol(str6,&error,base)= -34456  error=g37
```

17

Приклад використання функції **strtol** з різною основою системи числення

```
char *str, *error; long l; int bs;
str="3.1415926Stop";
l=strtod (str, &error);
printf ("string=%s\nstrtod=%ld  (base %d)\n error = %s\n\n\n",str,l, bs,error);
str="10110134932";
printf (" string=%s\n", str);

bs=1; l=strtol(str, &error, bs);
printf (" strtol=%ld (base %d)    error = %s\n",l,bs,error);
for (bs=2; bs<=36; bs*=2)
{  l=strtol(str, &error, bs);
   printf (" strtol=%ld (base %d)    error = %s\n",l,bs,error);
}
```

```
string=3.1415926Stop
strtod=3  (base 1896)
 error = Stop

 string=10110134932
 strtol=0 (base 1)    error = 10110134932
 strtol=45 (base 2)    error = 34932
 strtol=4423 (base 4)    error = 4932
 strtol=2134108 (base 8)    error = 932
 strtol=2147483647 (base 16)    error = 32
 strtol=2147483647 (base 32)    error = 932
```

18

Додаток 9. Презентація до лекції 8

Дисципліна: “Основи програмування”

для груп ІЕС, ТЕ

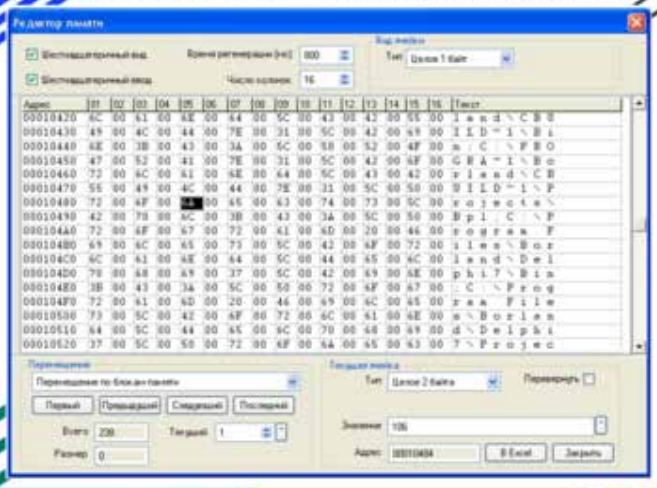
3.2 Покажчики

Під час виконання програми використовуються значення, які зберігаються по певних адресах.
Наприклад, в операторі присвоєння **var1=var2**; значення змінної **var2** копіюється в змінну **var1**.
У мові С передбачаються й такі змінні, які зберігають адреси змінних або так званні, покажчики.
Покажчик - це змінна, що містить адресу інший змінної
Наприклад:
int x - змінна, цілого типу;
int *px - покажчик.

Унарна операція ***&*** видає адресу об’єкта, так що оператор ***px=&x*** передає адресу ***x*** змінній ***px***; кажуть, що ***px*** «вказує» на ***x***.

2

Умовні оператори. Умовний оператор if

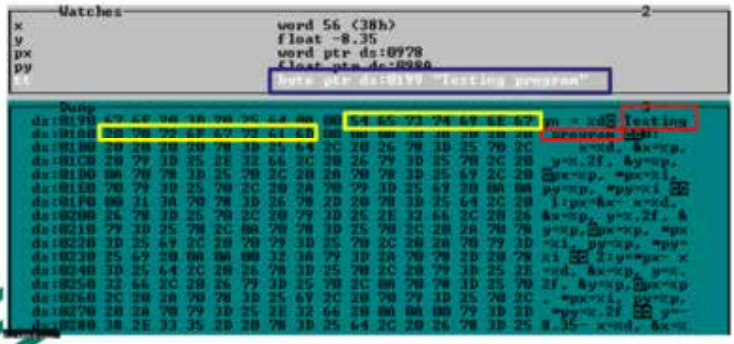


3

Приклад

Визначення рядку символів за допомогою покажчика на масив символів

```
char *tt="Testing program";
```



4

Операції над покажчиками

Якщо визначені змінні x, i

```
int x, i;
Int *px;
```

то можна оперувати й з їхніми адресами

```
px=&x;  // & (взяти адресу)
```

Справа від **&** може стояти тільки ***ім'я змінної*** або ***змінної з індексом***, а НЕ вираз загального виду.

Унарна операція "*" - по цій адресі витягти вміст.

```
int x=10, y,*px; // x=10
px = &x;         // px – адреса змінної x
y = *px;         // y=x
```

5

Операція **&** застосовується тільки до змінних і елементів масиву, конструкцій виду:

```
int* p, y;      // int* p; int y; але НЕ int* y;
int x, *p;      // int x; int* p;
int v[10], *p;  // int v[10]; int* p;
.....
p=&y;
y=*p;
```

&(x-1) і &3

НЕкоректні

Не можна також одержати адресу **реєстрової** змінної:

```
register x;
int* p;
.....
p=&x; // помилка!
```

6

Відзначимо, що в описі покажчика обов'язково визначається ***тип об'єкта***, з яким визначено покажчик.

Цей опис використовується при обчисленні виразів із покажчиками:

```
char c1 = 'a';
 char *p = &c1;    // в p зберігається адреса c1
 char c2 = *p;     // c2 = 'a'
```

7

```
char *tt="Testing program";
int x=56;
float y=-8.35;
int *px;
float *py;
printf(tt);
printf("\n\n0:      x=%d, &x=%p, y=%.2f, &y=%p, \npx=%p, *px=%i, py=%p, *py=%i
\n\n",x,&x,y,&y,px,*px,py,*py);
px=&x;
printf("1:px=&x- x=%d, &x=%p, y=%.2f, &y=%p,\npx=%p, *px=%i, py=%p, *py=%i
\n\n",x,&x,y,&y,px,*px,py,*py);
y=*px;
printf("2:y=*px- x=%d, &x=%p, y=%.2f, &y=%p,\npx=%p, *px=%i, py=%p, *py=%.2f
\n\n",x,&x,y,&y,px,*px,py,*py);
y=-8.35;
printf("y=-8.35- x=%d, &x=%p, y=%.2f, &y=%p,\npx=%p, *px=%i, py=%p, *py=%.2f
\n\n",x,&x,y,&y,px,*px,py,*py);
py=&y;
printf("3:py=&y- x=%d, &x=%p, y=%.2f, &y=%p,\npx=%p, *px=%i, py=%p, *py=%.2f
\n\n",x,&x,y,&y,px,*px,py,*py);
x=*py;
printf("4:x=*py- x=%d, &x=%p, y=%.2f, &y=%p,\npx=%p, *px=%i, py=%p, *py=%.2f
\n\n",x,&x,y,&y,px,*px,py,*py);
```

значення — адреса

8

значення

адреса

```
0:        x=56, &x=FFE2, y=-8.35, &y=FFDE,
px=0968, *px=4351, py=097A, *py=0

1:px=&x-  x=56, &x=FFE2, y=-8.35, &y=FFDE,
px=FFE2, *px=56, py=097A, *py=0

2:y=*px-  x=56, &x=FFE2, y=56.00, &y=FFDE,
px=FFE2, *px=56, py=097A, *py=0.00

y=-8.35-  x=56, &x=FFE2, y=-8.35, &y=FFDE,
px=FFE2, *px=56, py=097A, *py=0.00

3:py=&y-  x=56, &x=FFE2, y=-8.35, &y=FFDE,
px=FFE2, *px=56, py=FFDE, *py=-8.35

4:x=*py-  x=-8, &x=FFE2, y=-8.35, &y=FFDE,
px=FFE2, *px=-8, py=FFDE, *py=-8.35
```

9

Покажчики можна використовувати:
в операторах ***присвоєння***

int ****px, *py;*** *// **px*** і **py** відповідають однаковим типам даних

px=py

Єдиною константою, яку можна привласнити покажчику, є нуль, або ***NULL.***

10

До покажчика можна **додавати** або **віднімати** ціле число:

```
double *px, *py, *pz;
py=px+2; // 2*sizeof(double)
pz=py-4;
```

Це значить, що покажчик **px** необхідно збільшити на дві одиниці даних.
Коли **px** і **py** є покажчиками символів - **2 байта**.
Коли **px** і **py** є покажчиками плаваючих чисел з подвійною точністю - **16 байт.**
ЗАЛЕЖНО від типу даних у виразах з **покажчиками** здійснюється відповідне масштабування.

11

Для покажчиків визначена й операція обчислення:

pz=px-py *//* різниця адрес

Операція додавання покажчиків НЕ має ніякого змісту.

Визначені й операції збільшення й зменшення:
px++;
py--;
px+=i;

12

Приклад: функція, що підраховує число символів у рядку (не враховуючи завершального кінця рядку):

```
int strlen(char* p)
{   int i = 0;
    while (*p++) i++;
    return i;
}
```

Інший спосіб: відняти адресу початку рядка від адреси її кінця:

```
int strlen(char* p)
 {  char* q = p;
    while (*q++) ;
    return q-p-1;
 }
```

13

Використання постфіксної операції збільшення робить наступні цикли while ідентичними:

```
char *string="Testing";
.....
while (*string)
{  printf (string);
   string++;
}
```

```
Testingestingstingtingingngg
Testingestingstingtingingngg
```

```
while (*string)
{   printf (string++);}
```

14

Крім того, визначені оперції відносин:

== ! = > <

В арифметичних виразах замість імені змінної можна застосовувати операцію доступу за покажчиком.
Приклад:
Якщо адреса **px** в дорівнює адресі в **x**, то збільшити значення **x**, та записати в **y**, в іншому випадку, вивести значення в **px** на екран.

```
int x,y,*px;
.....

if (px==&x) y=(*px)+1; // y=x+1
else printf ("%d \n", *px);
```

15

Унарні операції * та & виконуються раніше арифметичних
Тому:

```
int x=9,y,*px=&x;
// якщо треба y=x+1, то;

y=*px+1; // а НЕ y=*(px+1);
```

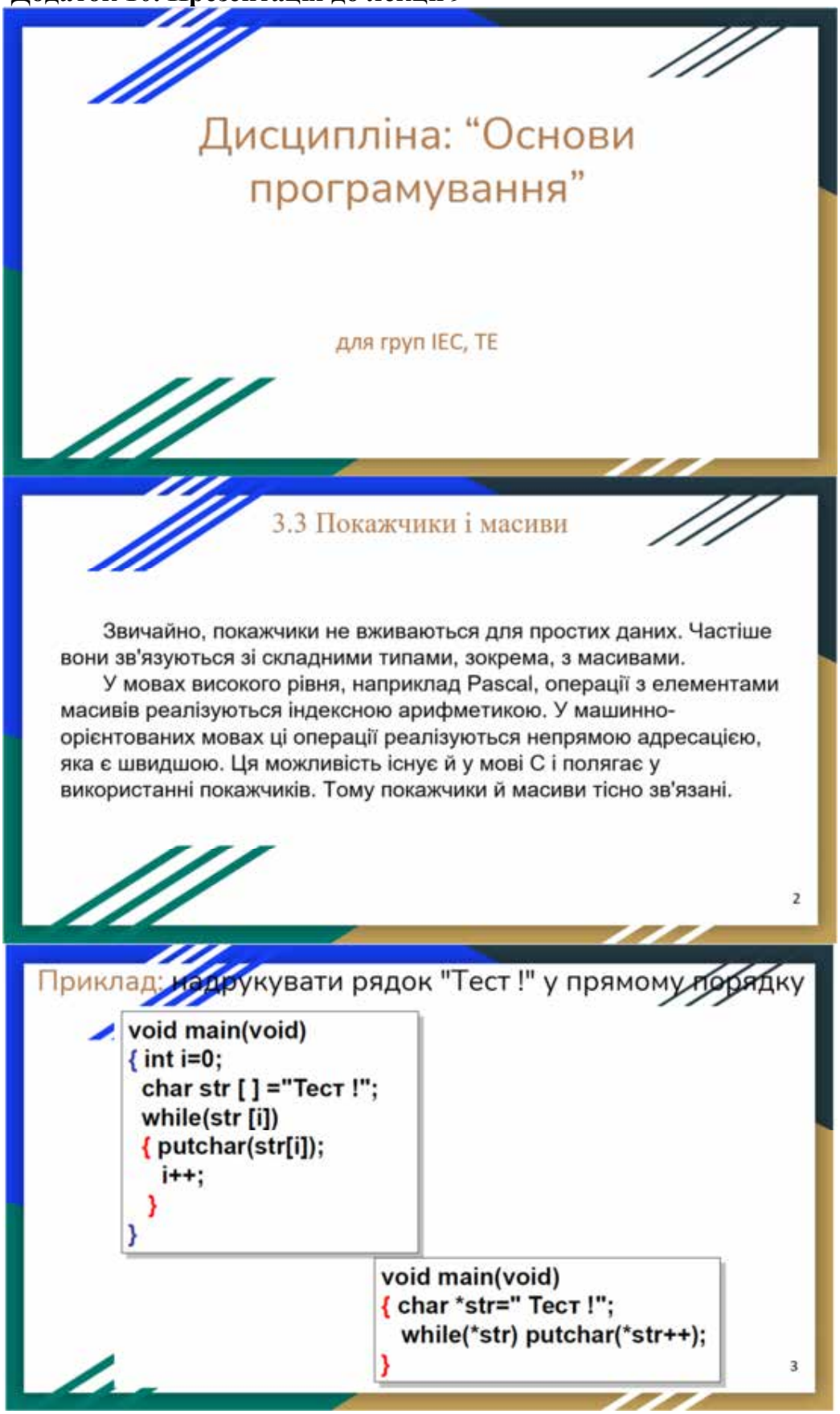
Між собою унарні операції рівноправні та виконуються справа наліво:

```
*px++; // вміст комірки px+1
(*px)++;  //  px+1 теж саме ++*px;
//збільшити адресу в px на 1
```

Для покажчиків можна використовувати операцію взяти адресу, де знаходиться сам покажчик **&px**.

16

Додаток 10. Презентація до лекції 9



Приклад: надрукувати рядок "Тест !" у зворотньому порядку

```
void main(void)
{  char str [ ] ="Тест !";
   int i=strlen(str);
   while(str [i])
   { putchar(str[i]);
     i--;
   }
}
```

```
void swapstr(void)
{ int i;
  char * st ="Тест !";
  for( i=strlen(st),st+=i-1;i;i--,putchar(*st--));
}
```

4

Ім'я масиву містить базову адресу цієї ділянки.

Тобто на увесь час виконання програми за ділянкою пам'яті закріплюється ***ім'я*** масиву. Тому воно є константою і його ***не можна змінювати під час виконання програми*** за допомогою оператора присвоєння.

Наприклад:

```
char st [ ]="Весна";
 st = "Літо"; // Помилка
```

Зміст масивів можна змінювати, наприклад, за допомогою функції копіювання або введення:

```
strcpy(st, "Літо");
```

5

Якщо задати рядок покажчиком
char *pr="тest!"; // DS:0291
Під час трансляції рядковій константі буде виділена пам'ять і початкова адреса цього рядка записана у покажчику **pr.** Спробуємо замінити текст:

```
pr="spring";
```

Але це не означає, що замість рядка **"тест!"** на його місце записується ***"spring".*** Просто в пам'яті виділяється нове місце під рядок ***"spring"*** і його базова адреса записується в покажчику **pr**.

```
pr="spring"; // DS:0341
```

Потрібно відзначити, що при роботі з масивами за допомогою покажчиків розмір масивів НЕ контролюється.

6

Якщо покажчик не буде ініціалізовано, то його значення невизначене (кожне). Тому ***можливі помилки*** при використанні покажчиків. Наприклад:

```
int *a;
 *a=25; // НЕ *a+=25;
 printf ("*a=%d\n", *a);
```

Необхідно спочатку виділити певну ділянку динамічної пам'яті:

(тип *) *malloc* (кіл-ть байт) - повертає покажчик на адресу

```
char *ptr;
/ / Виділяється 100 байт
// початкова адреса записується в ptr
ptr=(char*) malloc (100);
```

7

Тому правильно попередній фрагмент можна було б подати так:

```
int *a;
 a=(int *) malloc ( sizeof(int) );
 *a=25;
 printf ("*a=%d\n",*a);
```

Функція **malloc** поверне ***a = NULL***, якщо пам'яті НЕ виділено

```
#include <malloc.h>
 int *intarray;
 float *farray;
// резервує простір для 20 змінних цілого типу
  intarray=(int*)malloc(20*sizeof(int));

// резервує простір для 30 змінних типу float
farray=(float*)malloc(30*sizeof(float));
```

8

(тип *) *calloc* (кількість елементів n, розмір елемента в байтах) - виділяє простір для зберігання масиву з n елементів, кожний довжиною за вказаним розміром байт:
1. При виділенні, кожний елемент рівний 0.
2. Повертає ***NULL,*** якщо недостатньо пам'яті.

free (ім'я покажчика ***)*** - звільняє виділену пам'ять

```
int sizeM;
 char *pointer;
 printf (" Введіть розмір масиву, до 30000: ");
 scanf("%d", &sizeM);
 if (size > 0 && size <= 30000)
 {  pointer = (char*) calloc ( sizeM, sizeof(char));
    if (pointer != NULL) printf (" Пам'ять виділено " );
    else printf (" Неможливо виділити пам'ять " );
 }
.....
free(pointer);
```

9

(тип *) *realloc* (покажчик на раніше виділений блок пам'яті, новий розмір у байтах) - змінює розмір раніше виділеного блоку пам'яті, при цьому вміст блоку не змінюється.

Функція realloc повертає покажчик на перезахоплений блок пам'яті.

```
char *t1;
 t1=(char*) malloc(50*sizeof(char));

//перезахоплює  простір в 100 символів

 if (t1 != NULL)   t1=realloc( t1,100*sizeof(char) );
   if (t1 != NULL) printf (" Пам'ять перевиділено " );
   else printf (" Неможливо перевиділити пам'ять " );
```

if (t1) t1=realloc(t1,100*sizeof(char)); // некорректно

10

Покажчики і багатовимірні масиви

Існує двомірний масив:

```
int matr[3][2];
int *pr;
```

Ім'я **matr** дорівнює адресі базового елемента

Тому: **pr=matr**

```
matr==&matr[0][0];
```

```
pr+1==matr[0][1];
pr+2==matr[1][0];
```

і т.д.

За допомогою покажчика й відповідного зсуву можна добратися до будь-якого елемента масиву.

Елементи двовимірного масиву розташовуються в пам'яті рядками послідовно:

```
matr[0][0] matr[0][1]
matr[1][0] matr[1][1]
matr[2][0] matr[2][1]
```

11

Тому:

```
matr[0]=&matr[0][0]=matr        matr[1]=&matr[1][0]
```

і т.д.

Звідси:

matr[0]	matr[0][0]	matr[0][1]
matr[1]	matr[1][0]	matr[1][1]
matr[2]	matr[2][0]	matr[2][1]

А після того, як: **pr=matr;**

pr[0]	pr [0][0]	pr [0][1]
pr[1]	pr [1][0]	pr [1][1]
pr[2]	pr [2][0]	pr [2][1]

12

Для того, щоб скопіювати рядок, зовсім не потрібно його фізично копіювати, а досить лише скопіювати відповідну адресу в покажчику.

Також при сортуванні можна переставляти не рядки, а їх покажчики. Розглянемо програму введення, ***сортування та виводу рядків***. Використовуємо два масиви:

```
char **str;
char *tstr;
int nums; // максимальна кількість рядків
int numc; // максимальна кількість символів в рядку;
```

Сортування будемо проводити перестановкою покажчиків у масиві ***str.*** Сам же масив ***str*** буде залишатися незмінним.

Для порівняння рядків будемо використовувати функцію ***strcmp (str1, str2).***

13

```
#include <conio.h>
#include <string.h>
#include <stdlib.h>
char **str,*tstr;
void main(void)
{ int a,b,nums,numc;

 printf(" Кількість рядків = "); scanf("%d",&nums);
 printf(" Кількість символів в рядку = "); scanf("%d",&numc);

// виділення пам'яті для масиву рядків
 str=(char**)calloc(nums,sizeof(char*));

// для самих рядків
 for(a=0;a<nums;a++) str[a]=(char*)calloc(numc+1,sizeof(char));
 randomize();
 for(a=0;a<nums;a++)
 { b=0;
```

14

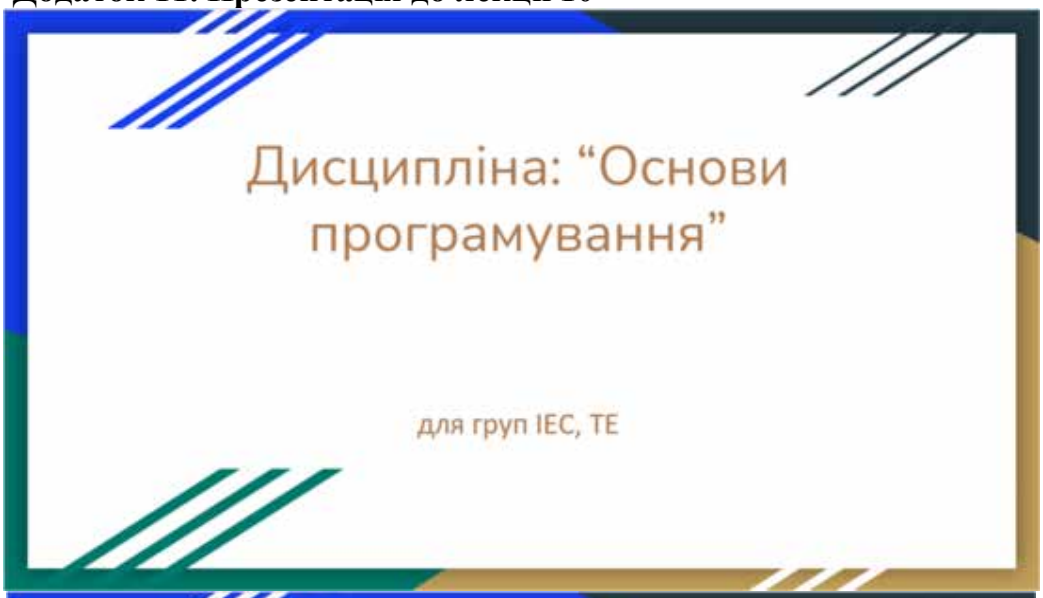
```
// генерація символів рядка
  do{
   int ch=random(257);
   if ((ch>64 && ch<91) || (ch>96 && ch<123)) {
    str[a][b]= ch;
    b++;
   }
   while (b<numc);
  str[a][b]= '\0';
 }
 printf("\nСгенерований масив з [%d] рядків ", nums);
 for (a=0;a<nums;++a) printf(" \n[%2d] рядок -> %s", a+1, str[a]);
// сортування за зростанням
  for(a=1;a<nums;++a) // перебір по рядкам
   for(b=nums-1;b>=a;--b)
   { // кількість переборів
    if(strcmp(str[b-1],str[b])>0)
    { // порівняння рядків
     tstr=str[b-1];
     str[b-1]=str[b];
     str[b]=tstr;
    }
   }
//
```

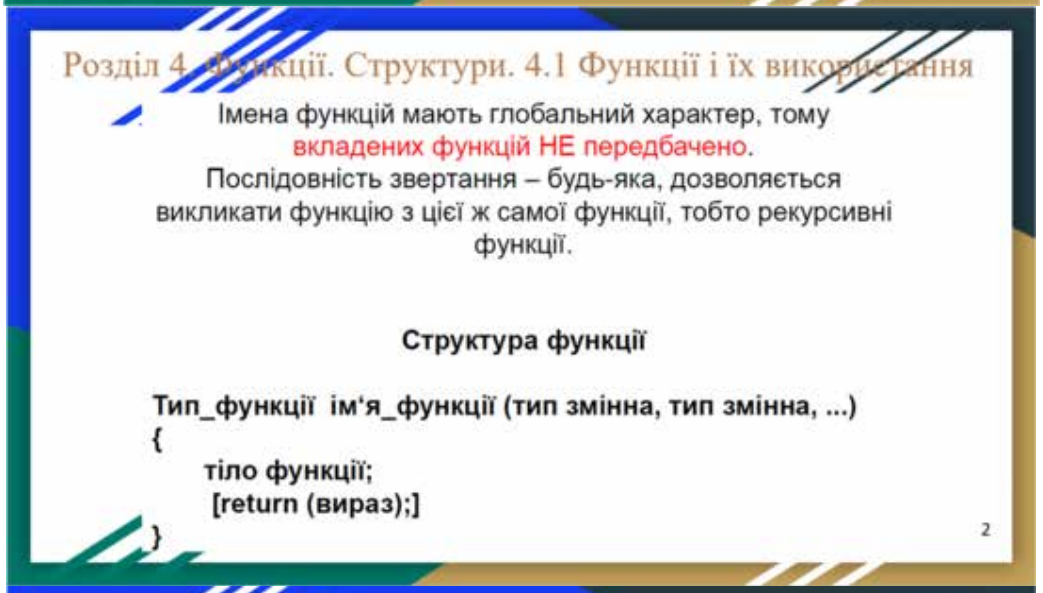
15

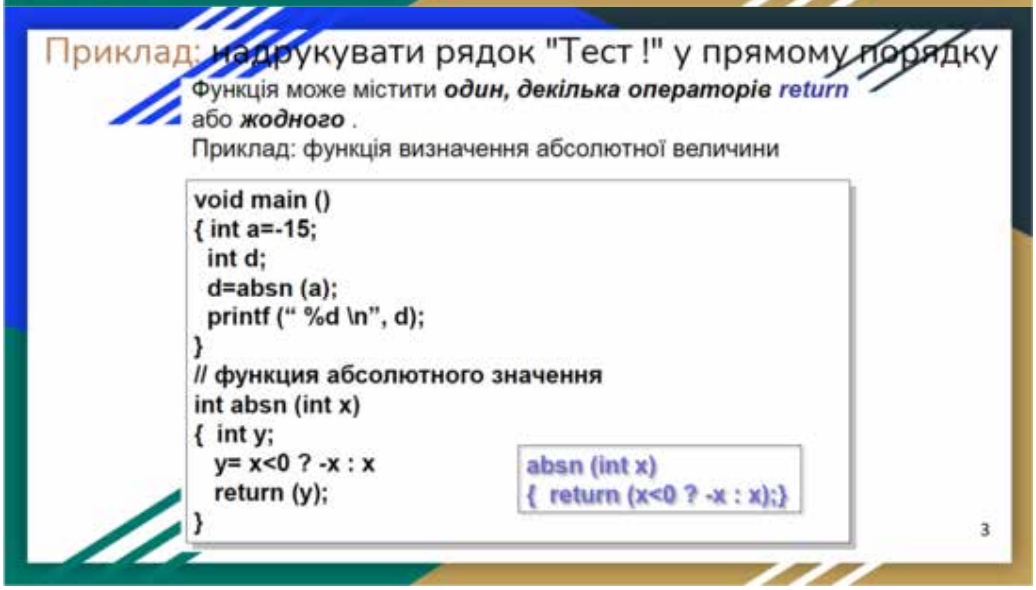
```
//
   getch();
   printf("\n\n Відсортований масив масив з [%d] рядків “, nums);
   for (a=0;a<nums;a++) printf(“ \n[%2d] рядок -> %s“, a+1, str[a]);

// звільнення пам'яті
   for (a=0;a<nums;a++) free(str[a]); // пам'ять під символи
   free(str); // пам'ять під рядки
   getch();
}
```

16

Додаток 11. Презентація до лекції 10







Фактичні й формальні параметри

Визначаючи функцію, визначають *формальні* параметри.
Викликаючи функцію, визначають *фактичні* параметри.

Приклад: перетворення **маленьких** латинських букв у **ПРОПИСНІ**.
У кодах ASCII прописні букви кодуються від A-65_{10} до Z-90_{10}.
маленькі від a-97_{10} до z-122_{10}. Перехід: код маленьких мінус 32_{10}.

```
char test(char cf)
{  if ((cf<'a') || (cf>'z')) return(cf);
   else return(cf+'A'-'a'); // 65-97=-32
}
main ()
{  char *st="test PrograM";
   for(a=0;a< strlen(st);a++) printf ("%c",test(st[a]));
}
```

На екрані: **TEST PROGRAM**

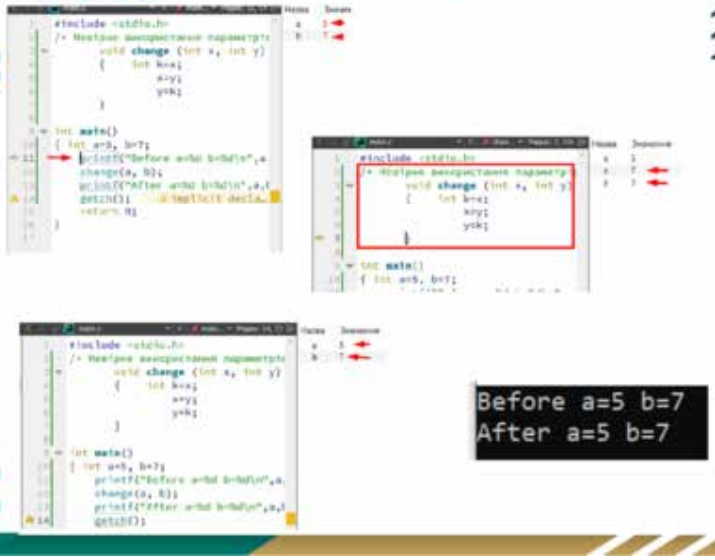
4

Оскільки передача параметрів відбувається за значенням, то у тілі функції не можна поміняти *значення* змінних, що є *фактичними параметрами*.

Приклад: Створити функцію, яка міняє значення між двома змінними.

```
// Невірне використання параметрів
    void change (int x, int y)
    {   int k=x;
            x=y;
            y=k;
    }
change (a,b); // a=5; b=8;
```

5



6

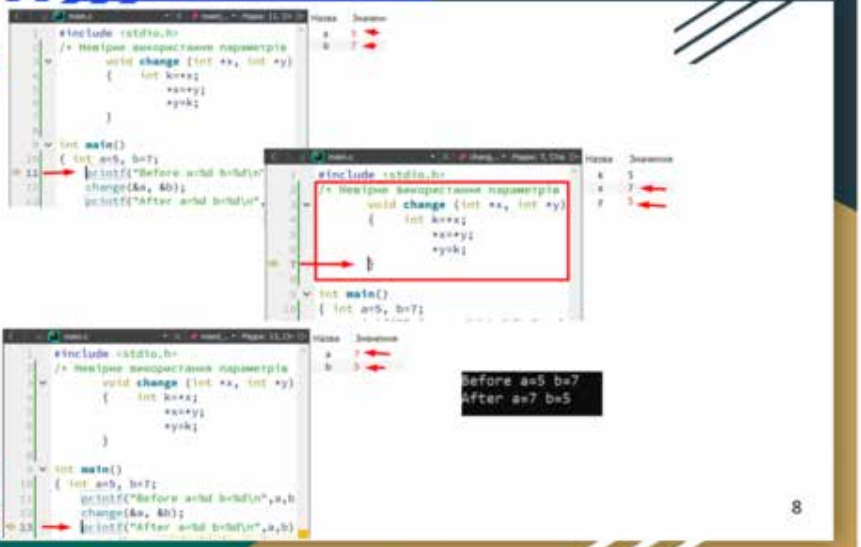
Однак, якщо в якості параметра передати покажчик на деяку змінну, використовуючи операцію ***переадресації*** , то можна змінити значення цієї змінної.
Приклад:

```
// Вірне використання параметрів
    void change (int *x, int *y)
    { int k=*x;
          *x=*y;
          *y=k;
    }
```

При виклику такої функції в якості фактичних параметрів повинні бути використані не значення змінних, а їх адреси:

```
change (&a,&b);
```

7



8

Масиви як параметри функцій

Одним з можливих варіантів використання масивів є опис їх у функції й передача їм значень.

```
void arrv(int ar[ ], int size)
{  for(int i=0;i<size;a++) printf("%d ", ar[i]); }

void main(void)
{ int a, arr[30];
  for (a=0;a<10;a++) arr[a]=a; // заповнення масиву
  arrv(arr,10); // виведення масиву на екран
  getch();
}
```

9

Рекурсивні функції

У мові С передбачені посилання функції саму на себе, тобто ***рекурсивні функції***.

Наприклад: обчислення **n!**

```
long factor (int n) // обчислення n-факторіала
{ if (n==1) return (1); // if (n) return (1); - помилка
  else return (n*factor (n-1));
}
void main (void)
{printf ( "%d", factor(5) ); // на екрані: 120
}
```

10

```
long factor (5)
{ if (5==1) return (1);
else
     return (5*factor (4)); // factor=5*24=120
}
```

```
long factor (4)
{ if (4==1) return (1);
else
     return (4*factor (3)); // factor=4*6=24
}
```

```
long factor (3) // обчислення n-факторіала
{ if (3==1) return (1);
else
     return (3*factor (2)); // factor=3*2=6
}
```

```
long factor (2)
{ if (2==1) return (1);
else
     return (2*factor (1)); // factor=2*1=2
}
```

```
long factor (1)
{ if (1==1) return (1);
else
     return (4*factor (3));
}
```

11

Крім рекурсивних функцій можна використовувати звертання до функцій як до ***фактичного параметра***.
Наприклад:

```
printf (" %d \n", scanf (" %f %e %s", &a, &b, str));
```

Тут параметром є звертання до функції **scanf.**

Крім вводу, функція **scanf** повертає ціле число, яке дорівнює кількості **коректно (по специфікатору) виконаних уведень**.

12

Особливості побудови програм з використанням функцій

Один з варіантів - розмістити всі функції в одному файлі.
Але для великих програм краще створити кілька файлів:
fil1.h, fil2.h.

Тоді для їхнього об'єднання можна використовувати препроцесорні директиви:

```
#include “fil1.h”;  // #include <fil1.h>
#include “fil2.h”; // #include <fil2.h>
```

13

Типи функцій

Правила визначення типів функції ідентичні правилам опису змінних.

Для визначення типу даних діють такі правила пріоритетів модифікаторів:

1. Чим ближче модифікатор до імені, тем вище його пріоритет;
2. Для модифікаторів одного рівня [], () мають вищий пріоритет за *;
3. Круглі дужки змінюють послідовність і мають вищий пріоритет за [].

() ⟹ [] ⟹ *

Тому **int *st[3][5]** буде означати:

масив покажчиків *st[3] з трьох елементів на массив из п'яти елементів цілочисельного типу

14

За правилами можна визначити й покажчик на функцію:
int (*func) (int, int);
АЛЕ **int *func (int, int)** означає функцію, яка передає покажчик на ціле.
Наприклад, є таке оголошення функції:

```
double (*fun_ptr) (int, int);
double proc (int, int);
```

Щоб зв'язати покажчик **fun_ptr** з певною функцією, потрібно привласнити йому ім'я функції без списку параметрів : **fun_ptr=proc;**

Після цього звертання до функції через покажчик буде мати вигляд:

```
(*fun_ptr) (2, 5);
```

15

Приклад:

```
int difference (int a, int b) { return (a-b);} // функція різниці
int sum (int a, int b) { return (a+b);} // функція суми

int (*fun_ptr) (int, int); // визначення покажчика на функцію

void main (void)
{int v1=15, v2=6,par;
  fun_ptr=difference;
  par=(*fun_ptr) (v1, v2);
  printf ("par_difference=%d \n", par);
  fun_ptr=sum;
  par=(*fun_ptr) (v1, v2);
  printf ("par_sum=%d \n", par);
}
```

```
par_difference=9
par_sum=21
```

16

Як і звичайні змінні, покажчики на функції можуть поєднуватися в масиви:

```
int  difference (int a, int b) { return (a-b);} // функція різниці

int  sum (int a, int b) { return (a+b);} // функція суми

// визначення масиву покажчиків на функції
int (*fun_ptr[2]) (int, int)={difference, sum};

void main (void)
{int v1=15, v2=6,par;
 par=(*fun_ptr[0]) (v1, v2);
 printf ("par_difference=%d \n", par);
 par=(*fun_ptr[1]) (v1, v2);
 printf ("par_sum=%d \n", par);
 }
```

17

main.c <Select Symbol>

```
#include <stdio.h>

int difference (int a, int b) { return (a-b);} // функція різниці

int sum (int a, int b) { return (a+b);} // функція суми

// визначення масиву покажчиків на функції
int (*fun_ptr[2]) (int, int)={difference, sum};

void main (void)
{int v1=15, v2=6,par;
 par=(*fun_ptr[0]) (v1, v2);
 printf ("par_difference=%d \n", par);
 par=(*fun_ptr[1]) (v1, v2);
 printf ("par_sum=%d \n", par);
}
```

C:\Qt\Qt5.12.0\Tools\QtCreator\bin\qtcreator_process_stub.exe

```
par_difference=9
par_sum=21
```

18

Додаток 12. Презентація до лекції 11

Дисципліна: “Основи програмування”

для груп IEC, TE

4.2. Структури або записи

Виникає необхідність використання комбінованих типів, елементами яких можуть бути змінні різних типів.

Опис структур має такий вигляд:

struct <ім'я>
{
опис елементів
} [Ім'я змінної, ...];

Наприклад:

```
struct person
{ char fio[20];
  int year;
  char sex;
};
```

2

Ім'я структури інакше називається тегом

```
struct person
{ char name[20]; // прізвище
  int year; // рік народження
  char gend; // стать
} stud1, stud2, stud3;
```

```
struct person stud1, stud2, stud3;
//person stud1, stud2, stud3;
```

stud1, stud2, stud3; є змінними структурного типу, тобто кожна складається із трьох елементів: прізвище, рік народження, стать.

3

Оскільки структури належать до складних типів, то для них не визначено ні однієї операції.
Ім'я елемента складається із двох частин

< ім'я структури>.< ім'я елементу>

Наприклад: замість року народження впровадити дату: рік, місяць, число (дата сама є структурою):

```
struct birth
{ int year;
  char month[10];
  int day;
};
```

```
struct person
{ char name[20];
   birth date;
   char gend; } stud1, stud2, stud3;

та ім'я елементу

stud1.date.day=25;
```

4

Як і для масивів, початкові значення можна робити і для структур:

```
struct persons stud1 = {"Petroff", 1972, 'f'};
```

самі структури можуть становити інші складні типи даних, наприклад, масиви структур:

```
struct persons
{ char name[20];
   int year;
  char gend;
} stud1,student[30];

Тоді:
student[0].name="koval";
```

5

Структурні змінні й покажчики

На відміну від масиву, *ім'я структурної змінної* не є адресою, це є вказівник на тип даних.

Наприклад: надрукувати масив даних про студентів,використовуючи масив та покажчик на структуру

```
void main (void)
{
 struct person
 {  char name[20];
    int year;
    char gend;
 } stud[30], *stpr; // stpr - покажчик на структуру

// struct person stud[30], *stpr;      // stpr - покажчик на структуру
 ..........
// виведення значень всіх елементів
 printf ("прізвище рік стать \ n");
 for (stpr=stud; stpr<stud+30; ++stpr) // ініціалізація покажчика
   printf (" %s %d %c \n", (*stpr).name, (*stpr).year, (*stpr). gend);
}
```

6

(*stpr).name, (*stpr).year, (*stpr).sex

Круглі дужки тут потрібні тому, що операція "." має більший пріоритет, ніж "*". І якщо дужок не буде, то вийде нісенітниця, тому що **stpr** - це покажчик, а не повне ім'я структури.

Допускається й більш просте звертання до елементів структури через покажчик за допомогою спеціальної операції "**->**" ***(мінус, більше)***:

stpr->name **еквівалентно** ***(*stpr).name***

Тому :

```
for (stpr=stud; stpr<stud+30; ++stpr) // ініціалізація покажчика
 printf (" %s %d %c \n", (*stpr).name, (*stpr).year, (*stpr). gend);
}
```

```
for (stpr=stud; stpr<stud+30; ++stpr) // ініціалізація покажчика
 printf (" %s %d %c \n", stpr->name, stpr->year, stpr-> gend);
}
```

7

Дозволено передавати ***структури*** як ***параметри функцій*** і передавати ***структури*** через ***ім'я функції***.

Арифметичні операції й відносини над структурами НЕ виконуються, тому що в цьому нема сенсу.

Приклад: дії над крапками на площині. Кожна крапка задається своїми координатами (нехай будуть цілими). Необхідно впровадити операцію додавання над двома крапками:

```
struct point
{ int x;
  int y;
} pt1,pt2;
int main()
{pt1.x=5; pt1.x=7; pt2.x=25; pt1.x=18;
  pt1= addpoint(pt1, pt2);
}
/ / Додавання координат двох точок
struct point addpoint (struct point p1, struct point p2)
{ p1.x+=p2.x
   p1.y+=p2.y
   return p1;
}
```

8

Можна створювати динамічні структури даних, різні списки, дерева і т.п.

Приклад:

```
/* Програма виведення даних, використовуючи масив, покажчик і масив покажчиків
на структуру */
#include <conio.h>
#include <stdlib.h>
#include <locale.h>
#define nf 15 // довжина прізвища
const n = 10; // розмір масиву
const nfio = nf; // довжина прізвища
// Створення структури
struct mybas
{ char fio[nf+1];
  unsigned year;
  char gend;
} ;

void main()
{ struct mybas stud3[n]; // Масив змінних типу структури
  struct mybas *stud5, *studuk; // покажчики на структуру
  unsigned a,i;
  SetConsoleCP(1251);
  SetConsoleOutputCP(1251);
```

9

```
srand(time (NULL)/2); // запуск генератора
// Через Масив змінних типу структури ----------
printf (" Масив змінних типу структури\n");
for(a=0;a<n;a++) { // генерація полів
 for(i=0;i<nfio;i++) stud3[a].fio[i]=(i==0)?(rand()%26)+97-32:(rand()%26)+97; // прізвище
 stud3[a].fio[i]='\0'; // визначаємо кінець рядка
 stud3[a].year=1950+(rand()%70); // рік
 stud3[a].gend=(rand()%2)?'m':'f'; // стать
}

// виведення даних з масиву
printf (" № Прізвище рік стать \n");
for(a=0;a<n;a++)
 printf("%3d -> %s %d %c\n", a+1, stud3[a].fio, stud3[a].year, stud3[a].gend ) ;
getch();

// Через покажчик на структуру ----------
printf("\n Покажчик на структуру\n");
stud5 =(struct mybas*) malloc (sizeof(struct mybas)); // виділення пам'яті
for (i = 0; i <nfio;i++)
 stud5-> fio [i] = (i==0)?(rand()%26)+97-32:(rand()%26)+97; // прізвище
stud5->fio[i]='\0'; // визначаємо кінець рядка
stud5->year=1950+(rand()%70); // рік
stud5->gend=(rand()%2)?'m':'f'; // стать // виведення даних
```

10

```
printf (" № Прізвище рік стать \n");
printf("%3d -> %s %d %c\n", a+1, stud5->fio, stud5->year, stud5->gend ) ;
getch();
free(stud5); // Звільняємо пам'ять
// через масив покажчиків на структуру ----------
printf("\n Масив покажчиків на структуру\n");
studuk =(struct mybas*) calloc (n, sizeof(struct mybas)); // виділення пам'яті
for(a=0;a<n;a++) {// генерація поля
 for(i=0;i<nfio;i++)
 studuk[a].fio[i]=(i==0)?(rand()%26)+97-32:(rand()%26)+97; // прізвище
 studuk[a].fio[i]='\0'; // визначаємо кінець рядка
 studuk[a].year=1950+(rand()%70); // рік
 studuk[a].gend=(rand()%2)?'m':'f'; // стать
}
printf (" № Прізвище рік стать \n");
for(a=0;a<n;a++)
 printf("%3d -> %s %d %c\n", a+1, studuk[a].fio, studuk[a].year, studuk[a].gend ) ;
getch();
free(studuk); // Звільняємо пам'ять
}
```

11

Результат виведення на екран

```
Масив змінних типу структури
  №    Прізвище        рік стать
 1 -> Qylqklmvvkcnvhl 1968  f
 2 -> Zgufrndvememlqu 1979  m
 3 -> Frxcxsnwndcouov 1961  f
 4 -> Ubktgpevilngujs 1992  m
 5 -> Cpgpqsyjsvurhua 1962  f
 6 -> Kdvknndnmpyvpku 1998  m
 7 -> Wkufuxorvjfjbjq 2007  f
 8 -> Jtthsiwmxbbmilg 1970  m
 9 -> Uyfthihttwjsgbi 1950  f
10 -> Sirruoktxzizssh 1980  f

Покажчик на структуру
  №    Прізвище        рік стать
11 -> Brthqhcrtjleofv 1984  m

 Масив покажчиків на структуру
  №    Прізвище        рік стать
 1 -> Fduyoonrjyzabsg 2005  f
 2 -> Boiqjrxdjmdczdm 2009  f
 3 -> Xhxdssoxvrnlmlm 1950  f
 4 -> Rikjfgdvnkverkf 1953  f
 5 -> Gfmsgurbbfbfumr 2019  m
 6 -> Cjukqryzfbenhcf 1979  f
 7 -> Fbjbkkietyiyltv 1983  f
 8 -> Quglzwmvfdceegs 1960  f
 9 -> Pceplqwzrvhevuo 2017  m
10 -> Hvmalnpdntrozvi 2003  f
```

12

Поля

Елементом структури може бути бітове поле, що забезпечує доступ до окремих бітів пам'яті.

НЕ МОЖНА:

Оголошувати бітові поля поза структурами;
Організовувати масиви бітових полів;
Застосовувати до полів операцію визначення адреси.

У загальному випадку тип структури з бітовим полем задається в наступному виді:

```
struct
{ unsigned < ідентифікатор 1> :  < довжина-поля 1>;
  unsigned < ідентифікатор 2> :  < довжина-поля 2>;
   . . . . . .
} ім'я змінної;
```

13

Приклад:

```
struct {
        unsigned keyword : 1;
        unsigned extern : 1;
        unsigned stat : 1;
        unsigned auto : 1;
        } flag;
```

Після цього кожне поле має своє ім'я й з ним можна оперувати як зі звичайної змінної
flag.stat=1;
Та **if (flag.keyword==0 && flag.auto==1) оператор;**
Кількість біт виділених під поле може бути різним:

```
struct {  unsigned cod1 : 2;
          unsigned cod2 : 4;
          unsigned cod3 : 8;
} pcode;
```

14

Між окремими полями можна ***робити пропуски:***

```
struct { unsigned cod1   : 2;
         unsigned cod2   : 4;
                         : 2;
         unsigned cod3   : 8;
} pncode;
```

1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32767
0	0	0	0	0	0			0	0	0	0	0	0	0	0
cod1		cod2						cod3							

Тут між другим і третім полем пропущено 2 біта. Якщо всі поля не вміщаються в одному байті, то переносяться в наступний.
Але так, що поле не розривається, а вирівнюється по цілому.

15

*Пусте поле з цифрою **0** означає перехід до наступного слова*

```
struct { unsigned cod1 : 2;
         unsigned cod2 : 4;
                       : 0;
         unsigned cod3 : 8;
} prcode;
```

Виходить, що третє поле буде записано з початку третього байта

1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	2	4	8	16	32	64	128		1	2	4	8	16	32	64	128
0	0	0	0	0	0				0	0	0	0	0	0	0	0
cod1	cod2								cod3							
Перше слово (перший байт)									Друге слово (третій байт)							

16

Об'єднання

Іноді необхідно зберігати дані різних типів у тому самому місці пам'яті. Наприклад, створити таблицю, елементами якої будуть цілі, дійсні числа та символи. Для таких випадків і передбачений тип об'єднання, опис подібний структур і має вигляд:

```
union < ім'я >
      { <тип> < змінна >;
        .....;
        } < змінна >;
```

Об'єднання можна розглядати як структуру, елементи якої мають **0- ий** зсув у пам'яті.
Звертання до елементів об'єднання таке ж, як і до структур.

17

Поточне значення елемента об'єднання губиться після того, як іншому елементу буде присвоєне значення:

```
union uname
{ int digit;
  double digf;
  char letter;
} fit;
fit.digit=12;
fit.digf=2.56;
fit.letter='a';
```

Транслятор НЕ контролює, якого типу було останнє значення. Це повинен робити сам розроблювач програми.

Байти							
1	2	3	4	5	6	7	8
digit							
digf							
letter							

18

З об'єднаннями можна працювати за допомогою покажчиків:

```
union uname *prf;
prf->digf=5.61;
```

Об'єднання можуть входити в структури, масивів і навпаки.

Наприклад:

```
struct
{ char *name;
  int flag;
  union
  { int digit;
    double digf;
    char *letter;
  }fit
}symt[N];
```

Це масив структур, кожна з яких складається із трьох елементів. Третій елемент є об'єднанням:

```
symt[i].fit.digf=3.14;
```

19

Визначення типу

У ANSI C є можливість створювати нові типи даних, які можуть спростити програму.

Для цього можна використовують 2 визначення типу:

1: **typedef <тип> <нова назва>**

2: **#define ім'я_макроса послідовність_символів**

Наприклад, якщо прагнемо в програмі замінити тип **float** на ім'я **REAL**, то це можна зробити так:

```
typedef float REAL; // замінити тип float іменем REAL
```

Далі в програмі можна писати: **REAL a, b, c;**

Фактично нових типів тут не створюється, а надається нова назва існуючим типам.

С допомогою директиви препроцесора:

```
#define REAL float; // REAL скрізь буде замінено на float
```

20

Однак існують і певні відмінності:

Визначення ***typedef*** виконується компілятором, а не препроцесором;

Функція ***typedef*** надає назву тільки типу даних і не може надавати ім'я константі;

Функція ***typedef*** має більш широкі можливості ніж препроцесор.

Наприклад:

```
typedef char* STRING;
#define STR char *;

STRING m, mp[20];   що еквівалентно char * m, * mp [20];
STR m1, mp1[20];    що еквівалентно char * m1, mp1
```

21

typedef можна використовувати і для структур:

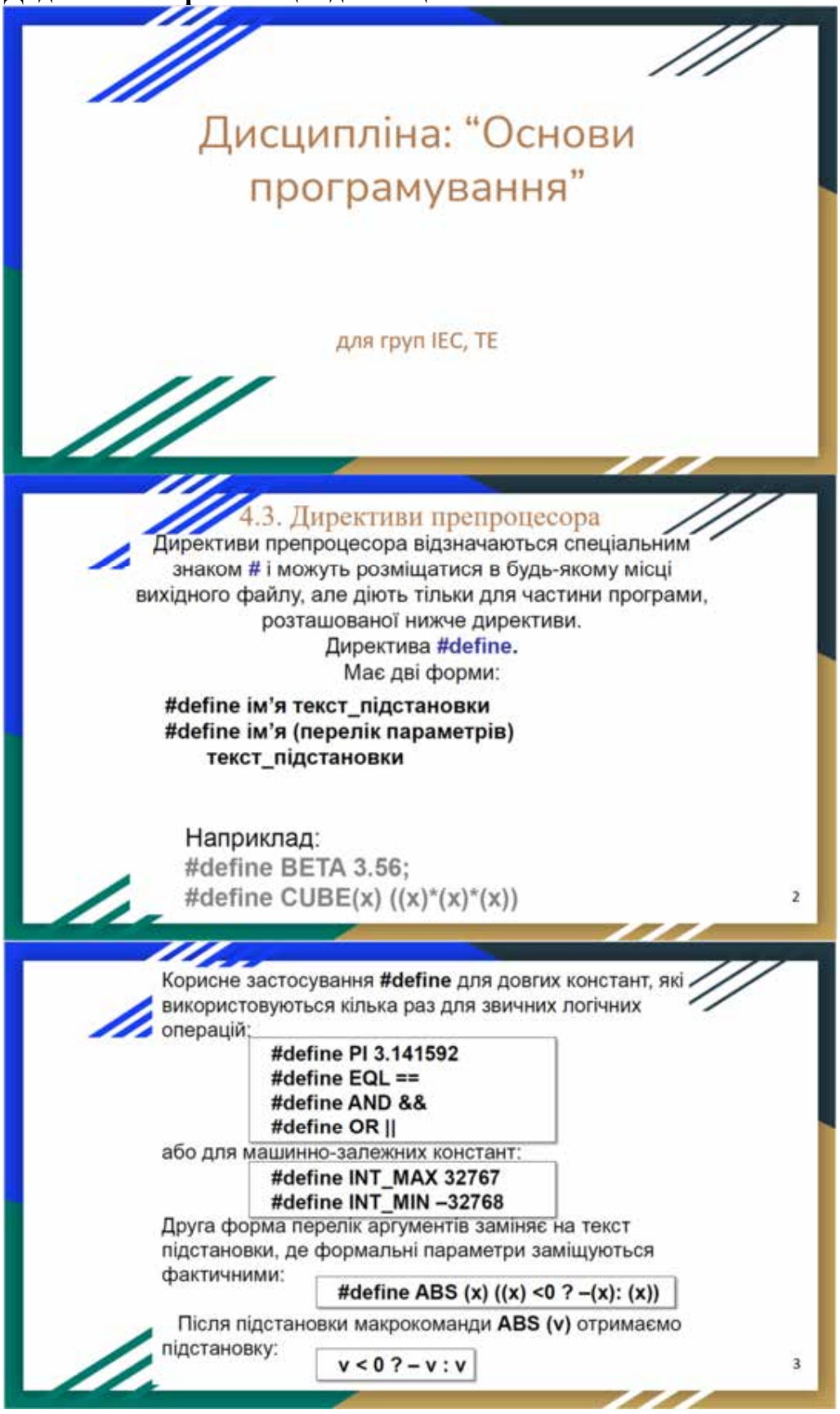
```
typedef struct COMPLEX
{ float re;
  float im;
};
```

Тоді **COMPLEX c, d, e;** визначить структуру типу **COMPLEX** без повторень слова **struct.**

```
typedef struct mybas
{ char fio[nfio+1];
  int year;
  char sex;
};
mybas stud3[n]; // Масив змінних типу структури
mybas * stud5,
```

22

Додаток 13. Презентація до лекції 12



Коли текст підстановки не вміщується в один рядок, для переносу можна використовувати зворотну дробову риску "\"

```
#define ZERO_ARRAY (array, size)
  { int i=0;  \
    while (i<size)  \
     array[i++]=0;  \
  }
```

Звертання в програмі:

```
#define MAX 50
void main ()
{ int values[MAX];
  ZERO_ARRAY (values, MAX); // – макрокоманда
}
```

Враховуючи особливості такої підстановки, для правильної реалізації потрібно використовувати круглі дужки.

Наприклад:

```
#define sqr (x)    (x)*(x)
```

4

Для директиви ***#define*** з параметрами можна використовувати дві препроцесорні операції:

1. ***Створення рядка*** - визначається одним знаком #
2. ***Об'єднання імен*** - визначається ##

5

Якщо перед формальним параметром розміщено символ #, то в результаті підстановки на тому місці буде розміщено аргумент у **подвійних лапках**.

Наприклад:

```
#define print (expr)    printf (# expr,"=%f \n", expr)
```

При звертанні:

print (y/z); буде текст **printf ("y/z","=%f \n", y/z);**

Якщо у визначенні вжити **arg1 ## arg2**, то в тексті - n1, n2 буде одне ім'я **n1n2**.

Наприклад:

```
#define unite (arg1, arg2)  arg1 ## arg2;
```

При звертанні:

unite (name, 2) створить ім'я **name2**

6

Визначення директиви **#define** може бути скасоване директивою:

#undef <имя>

Це дає можливість у наступному тексті програми використовувати певне ім'я в іншому розумінні. Наприклад:

```
#define NIL (char *) 0
#undef NIL
#define NIL (float *) 0
```

Тут спочатку ім'я **NIL** заміняється нульовим покажчиком на символ, потім визначення відміняється й далі це ім'я означає нульовий покажчик на дійсне.

7

Директиви умовної компіляції

Це директиви, які дозволяють не компілювати окремі частини вихідного файлу після перевірки константних виразів.

```
# ifdef  | # ifndef  <константний вираз>
        [текст програми]
[#else
        [текст програми]
#endif
```

 # ifdef повертає 1 (true) - якщо < *константний вираз* > було визначено директивою DEFINE

#ifndef повертає 1 (true) - якщо< *константний вираз* > було НЕ невизначено директивою DEFINE

8

Приклад:

```
#define MYDEFINE 1

#ifdef MYDEFINE
        WAIT WINDOW "MYDEFINE визначено"
#else
        WAIT WINDOW "MYDEFINE НЕ визначено"
#endif
```

```
#define MYDEFINE 1

#ifndef  MYDEFINE
        WAIT WINDOW NOT "MYDEFINE НЕ визначено "
#else
        WAIT WINDOW "MYDEFINE визначено "
#endif
```

9

Приклад: ініціалізація змінної x

```
#include <stdio.h>

#define SIZE 11
main()
{
#ifndef SIZE ; НЕ визначено
  int x=123;
  printf("x=%d\n",x);

#else

  static char x[SIZE]=«Інформатика»;
  printf("x=%s\n",x);
#endif
 }
```

10

Директива #if

***#if** константний вираз 1*

[текст програми 1]

***#elif** константний вираз 2*

[текст програми 2]

.....

#else

[текст програми 3]

#endif

11

Умовно текст,програми, що компілюється, починається директивою ***#if*** і закінчується обов'язково директивою ***#endif.***
Між цими директивами можуть бути розміщено кілька директив: **#elif** (це **else - if**) і одна директива **#else.**

Константні вирази НЕ можуть містити операцій **sizeof**, **перетворення типу** й констант **перерахування**.

Умовну компіляцію можна застосувати для програм, призначених для використання на різних **ЕОМ** та в **ОС.**

12

Приклад:

```
#include <stdio.h>

main()
{
        #if WIN32
        #pragma message("Compiling Win32")
        #endif

        #if DEBUG
        #pragma message("Compiling Debug")
        #endif

        return 0;
}
```

13

Директиви компілятора або прагми

Це інструкції компіляторові, які розміщаються в певних місцях програми й використовуються для керування діями компілятора, не впливаючи на програму в цілому.

Формат директиви:

#pragma <ім ' я> <послідовність символів>

14

Основні директиви:

#pragma argsused

#pragma exit

#pragma startup

#pragma inline

#pragma option

#pragma saveregs

#pragma warn

15

#pragma argsused - припустима тільки між визначеннями функцій і діє тільки на наступну функцію. Вона скасовує повідомлення рівня попередження.

#pragma exit – дозволяє програмі задати функцію (функції), які повинні викликатися при виході із програми (безпосередньо перед виходом із програми через **_exit**)

#pragma startup - дозволя програмі задати функцію (функції), які повинні викликатися при завантаженні програми (перед викликом main)

Синтаксис цих директив наступний:

#pragma exit ім'я-функції <пріоритет>

#pragma startup ім'я-функції <пріоритет>

16

#pragma inline - повідомляє компілятору, що програма містить вбудовані ассемблерні коди

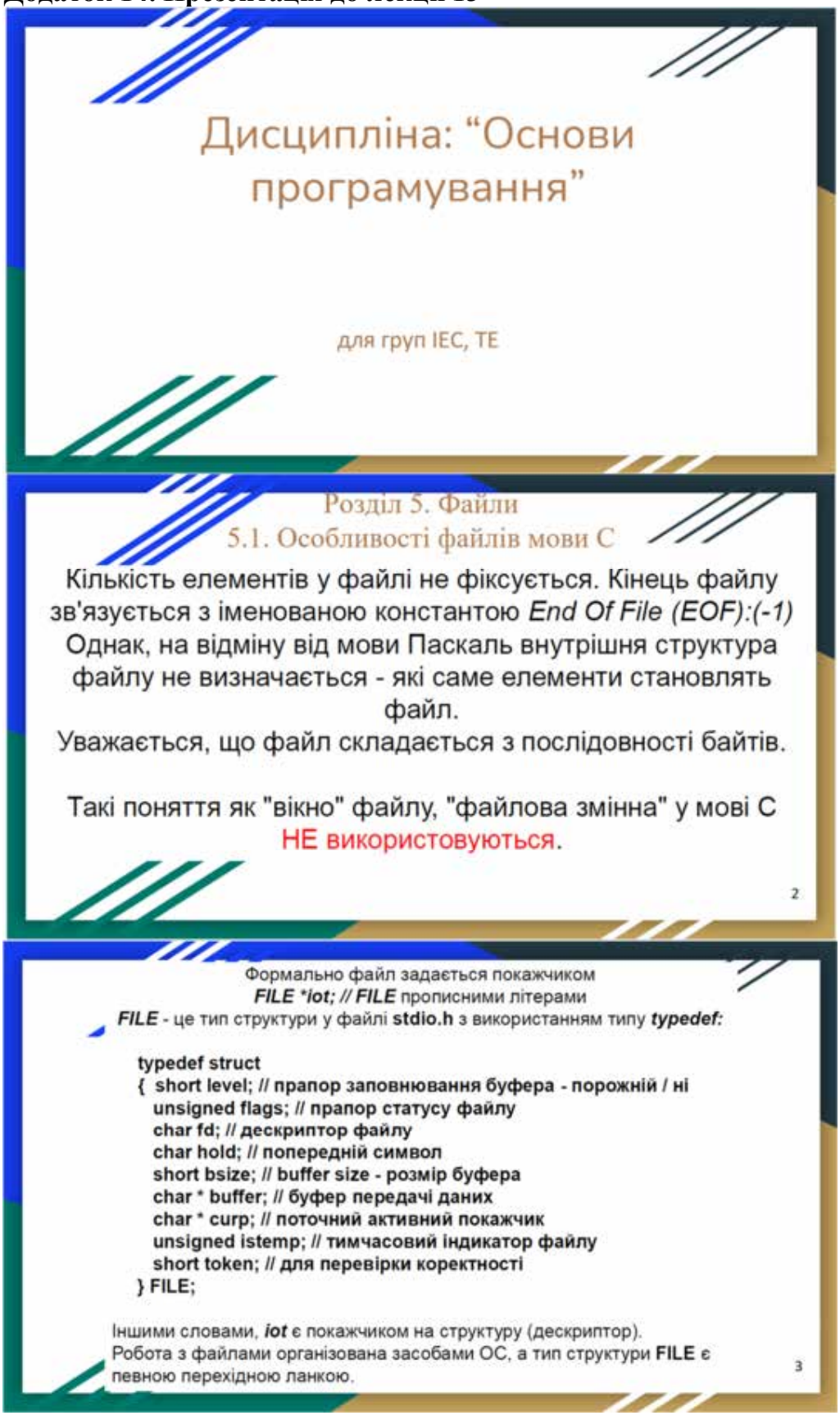
#pragma option - використовується для включення опцій компілятора командного рядка в код вашої програми

#pragma saveregs - гарантує, що при вході у функцію huge (куча) значення всіх регістрів залишаться без змін. Дана директива іноді буває потрібна для інтерфейсу з кодами мовою асемблера

#pragma warn - дозволяє перевизначати конкретні опції командного рядка **-wxxx** (або управляти опцією Display Warnings у діалоговім полі **Options | Compiler | Messages)**

17

Додаток 14. Презентація до лекції 13



У мові С немає власних засобів роботи з файлами, а всі дії реалізуються за допомогою функцій з бібліотек мови.

Ці функції розділяються на три класи:

1. **Верхнього рівня** (з використанням "потоку");

2. Для ***консольного*** термінала

3. **Нижнього рівня** (з використанням «дескриптора»)

4

1. Верхнього рівня (з використанням "потоку" C++);

```
#include <iostream>
 #include <fstream>
 using namespace std;
int main()
{ char one[64], two[32], three[16];
ofstream book_file("BOOKINFO.txt"); //запис в файл
book_file << "We_learn_to_write_programs_in_the_C++_language" << endl;
book_file << "Jamsa_Press" << endl;
book_file << "22.95" << endl;
ifstream input_file("BOOKINFO.txt") ; // читання з файлу
input_file >> one;
nput_file >> two; i
nput_file >> three;
cout << "one=" << one << "\ntwo=" <<two<< "\nthree=" << three<<endl;
return 0;
 }
```

```
one=We_learn_to_write_programs_in_the_C++_language
two=Jamsa_Press
three=22.95
```

5

2. Для ***консольного*** термінала й порту з безпосереднім звертанням до них

```
void main(char argc, char *argv[ ] )
```

Наприклад:
на екран треба вивести введене ім'я і рядок «скоро сесія !!! ».
Ім'я треба вказати у вигляді аргументу командного рядка.

6

```
// attention.exe
# include <stdio.h>
# include <stdlib.h> i
int main (int argc, char *argv [])
 {  SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
     if (argc!= 2) // argv [0] - ім'я програми
    {  printf ("Ви забули ввести своє ім'я. \n");
       return 0;
     }
    printf ("% s, скоро сесія !!! ", argv [1]);
    getch();
    return 0;
}
```

```
C:\>attention
Ви забули ввести своє ім'я.

C:\>attention Юрко
Юрко, скоро сесія !!!
```

7

3. Нижнього рівня (з використанням «дескриптора»)

Кількість дескрипторів залежить від конфігурації системи. Коли програма запускається, у неї зазвичай вже відкриті три дескриптора. До них належать такі:

0 - стандартний потік вводу (stdin);
1 - стандартний потік виводу (stdout);
2 - стандартний потік помилок (stderr).

Для читання й запису інформації призначено дві функції **read** і **write**.

read (fd, pointer, size);
write (fd, pointer, size);

fd – дескриптор файлу, стандартні потоки (0-2);
pointer – покажчик на буфер даних, до якого записується (зчитується) інформація;
size – довжина цього буфера.

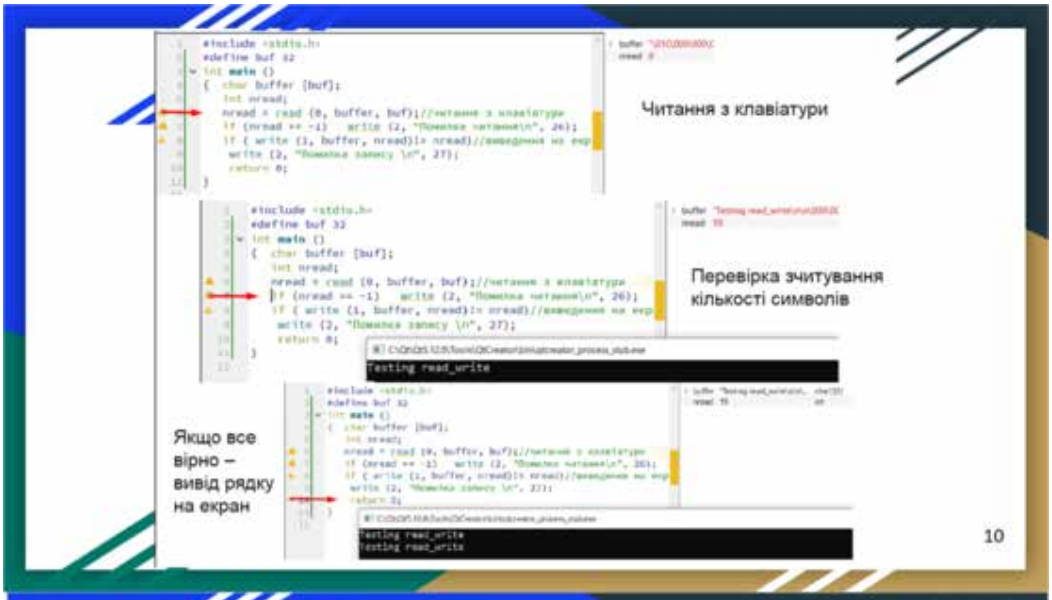
8

Приклад:

програма копіює перші 32 байти стандартного потоку введення (0) в стандартний потік виведення (1).

```
#include <stdio.h>
#define buf 32 // розмір буферу
 int main ()
{  char buffer [buf];
   int nread;
   nread = read (0, buffer, buf);//читання з клавіатури
   if (nread == -1)   write (2, "Помилка читання\n", 26);
   if ( write (1, buffer, nread)!= nread) //виведення на екран
    write (2, "Помилка запису \n", 27);
   return 0;
}
```

9



Потоковий обмін даними

Усі дані можна розглядати як послідовність окремих байт або потік.

Для користувача потік - це або файл на диску, або фізичне обладнання (дисплей або принтер).

Хоча потік є послідовністю окремих байт, функції обміну для потоку дозволяють обробляти дані різного розміру й формату (від одного символу до великих структур).

При цьому здійснюється буферизований форматний або безформатний обмін.

11

Відкриття потоків

Для виконання операцій з файлом його попередньо необхідно відкрити. Для цього використовується функція ***fopen ().*** Її параметрами є два рядки

fopen (char * ***name,*** char * ***mode)***

name; // це ім'я файлу, наприклад «D:\ prog.txt»
mode; // режими використання файлу

Існує три режими доступу до файлу:

1. Читання (read):

"r"- відкрити файл для читання, файл повинен існувати;
"r+" - відкрити файл одночасно для читання й запису, файл повинен існувати;

12

Відкриття потоків

2. Запис (write):

"w" - відкрити порожній файл для запису, якщо файл існує, він видаляється;

"w+" - відкрити порожній файл для читання й запису, якщо файл існує, він видаляється;

13

Відкриття потоків

3. Доповнення (append):

"**a**"- відкрити файл для читання й додавання починаючи з кінця, якщо файлу немає він створюється для читання або запису; // курсор наприкінці файлу

Відзначимо, що коли файл відкривається для запису, то увесь його зміст стає недоступним, тобто в логічному розумінні файл втрачається.

При доповненні зміст файлу зберігається й покажчик установлюється на кінець файлу.

14

Оскільки параметри **name** і **mode** є рядками, то вони беруться в лапки:

```
FILE *iot;
iot=fopen ("data.txt", "r+");
```

Функція **fopen** повертає покажчик на структуру **FILE,** яка описує даний відкритий файл.

Однак іноді, наприклад, при спробі відкрити неіснуючий файл для читання, функція *цього зробити НЕ зможе!*

15

Якщо файл не вдалося відкрити, функція **fopen** повертає значення **NULL**.

В stdio.h значення **NULL** - визначається як **0**, крім того, у глобальну змінну **errno** буде записано код помилки.

В програмах роботи з файлами бажано перевіряти умову відкриття файлу.

Наприклад:

```
if ( (iot=fopen (“data.txt”, “r+”) ) ! =NULL )
```

16

Приклад:

```
// Програма створює копію файлу «test.txt»
# include <stdio.h>
int main ()
{ FILE * in, * out;
  SetConsoleCP(1251);
  SetConsoleOutputCP(1251);
  if ((in = fopen ("C:\\test.txt ", "r")) == NULL)
  { fprintf (stderr, "Не можу відкрити файл \n");
    return (1);}
  if ((out = fopen (" D:\\test.bak", "w")) == NULL)
  { fprintf (stderr, "Не можу створити вих. файл \n");
    return (1);} // Посимвольне читання з файлу in і запис у файл out
  while (! feof (in)) fputc (fgetc (in), out);
  printf ("Файл створено, дані скопійовано \n");
  fclose (in);
  fclose (out);
  return (0);
}
```

17

```
Не можу відкрити файл
```

Файл відсутній

```
Не можу створити вих. файл
```

Місце створення закрито для користувача

```
Файл створено, дані скопійовано
```

18

Існує два типи файлів:

t - відкрити в текстовому режимі, тобто при введенні комбінації "Повернення каретки - переклад рядка" (ПК-ПР), вона перетворюється до єдиного символу "переведення рядка". При виведенні символ переведення рядка перетворюється в комбінацію ПК- ПР.

b - відкрити у двійковому (неперетворюючому) режимі;

Якщо **t** або **b** у рядку **type** НЕ задано, режим перетворення визначається змінної ***_fmode*** і режимом, установленим за замовчуванням.

19

Стандартні покажчики потоків

З кожним завданням автоматично зв'язується 5 потоків:

1. Стандартного введення (stdin);
2. Стандартного виводу (stdout);
3. Стандартного виводу повідомлень про помилки (stderr);

потоки зв'язані з консоллю користувача

4. Додаткового потоку (stdaux);
5. Стандартного потоку друку (stdprn).

Додатковий потік відносять до додаткового порту, до якого можна підключити допоміжну консоль, а стандартний друк - до принтеру.

Покажчики ***stdin, stdout, stderr, stdaux, stdprn*** є константами. Тому їм не можна привласнювати інші значення покажчиків потоків.
Однак кожний із цих потоків можна переадресувати на інше обладнання або інший потік за допомогою функції **freopen ("file.dat", "w", stdout)**
Потік **stdout** закріплюється за файлом ***file.dat*** у режимі запису.

Після цього виведення в ***stdout*** означає виведення у файл ***file.dat.***

20

Закриття потоків

Закриття потоків здійснюється для:

- окремого потоку функцією **fclose (iot)**;
- всіх потоків - функцією **fcloseall() –** ця функція НЕ закриває стандартні потоки.

При закритті потоків звільняються ВСІ буфери, ліквідуються покажчики на файл.

Якщо потоки не закриваються явно, то вони закриваються автоматично при завершенні програми.

Оскільки кількість відкритих потоків обмежено, то краще потоки в програмі закривати явно.

21

Додаток 15. Презентація до лекції 14

Дисципліна: “Основи програмування”

для груп IEC, TE

Функції обміну з потоками

У загальному випадку у файлах може зберігатися символьна інформація (форматовані файли) і двійкова (неформатовані файли).

Тому існують потоки текстові, які складаються із символів, розділених на рядки.

Двійкові потоки - це послідовність значень типу **char**.

Вважається, що стандартні потоки:
stdin, **stdout**, **stderr** є текстовими,
а **stdaux**, **stdprn** - ні.

2

Функції обміну з

Об’єкт операції	Операції обміну					
	Зчитування			Запис		
	Із потоку ***stdin***	Із будь-якого потоку	Із рядку C++	В потік ***stdout***	В будь-який потік	В рядок C++
Послідов. байт		fread			fwrite	
Окремий символ	fgetchar getchar getch	fgets getchar		fputchar putchar ungets	fputc putc	
Ціле значення (тип int)		getw			fput	
Рядок	gets	fgets		puts	fputs	
Форматовані дані	scanf	fscanf	sscanf	printf vprintf	fprintf vfprintf	sprintf vsprintf

Функція **fgets** і **fputs** діють аналогічно функціям **gets**, **puts**, але реалізовані інакше.

3

Обмін рядками

char * fgets (string, MAXLIN, point) - читання рядка з потоку покажчиком **point**
string - ім'я рядка, куди зчитуються символи рядка;
MAXLIN - максимальна кількість символів, яку потрібно прочитати.
Функція ***fgets*** припиняє роботу, якщо прочитано ***MAXLIN-1*** символів, або до зчитування символу закінчення рядка ('\n'). Повертає ***NULL***, якщо натрапили на кінець файлу.

```
Приклад: вибирається рядок символів з потоку inf.
char line[100], *result; FILE * inf;
result = fgets (line, 100, inf); // Покажчик, в який буде поміщена адреса масиву line
```

status=fputs (string, point) - запис рядка у ***потік***
Якщо відбулася помилка або кінець файлу, то функція повертає ***EOF.***

4

Форматний обмін

fscanf (point, "%d", &psi) – зчитування з потоку (***point***)
Функція передає кількість правильно прочитаних значень, а також **EOF**, коли файл вичерпано.

fprintf (point, "psi=%d \n", psi) - запис в потік (***point***)

У потік **point** записується ціле значення **psi**. Функція передає кількість правильно записаних значень.
Для обміну цілими даними передбачено дві функції:

getw (point);
putw (int, point);

```
int c;
FILE *point;
. . . . . .
c=getw (point);
putw (c, point);
```

5

Передбачено **дві** функції для обміну блоками байтів:

fread (buffer, size, count, point) - з потоку **point** зчитується **count** блоків, кожний з яких має розмір **size**, у буфер **buffer**.

fwrite (buffer, size, count, point) - з буфера **buffer** записується **count** блоків, розміром **size** у потік **point**.

Функції передають кількість записаних блоків.

6

Приклад:

```
struct mybas
{ char fio[25];
  int year;
  char sex;
} stud3 [5], stud5 ;

FILE *filw,*filr;

void main(void)
{ int i;
  . . . . . .
  fwrite(&stud3[i], sizeof(mybas), 1, filw);

  fread(&stud5, sizeof(stud5), 1, filr);
```

7

sscanf (string, format_string, list) - зчитування значення змінних з рядка **string** в **list**;
string - рядок;
format_string – формат перетворення значень;
list – куди зчитуються значення.

Функцію можна використовувати для перетворення символів у числа і навпаки.
Наприклад, рядок **"3.12"** перетвориться у дійсне число в змінній **value**.

```
char string[]="3.12";
float value;
sscanf (string, "%f", &value);
```

8

sprintf (string, format_string, list) - записує значення змінних з рядка **list** у рядок **string**
Функції можна використовувати для перетворення символів у внутрішню виставу чисел і навпаки.

```
char string[10];
float value=3.14159;
sprintf (string, "%10.5f", value);
```

У рядку **string** одержимо рядок **" 3.14159"** числа **value=3.14159** у формі з фіксованою крапкою.

9

Приклад:

зчитати інформацію з файлу даних цілого типу:
"DATA.TXT" , записати в змінну **temp** і вивести на екран.

```
void main (void)
{ int temp;
  FILE  *fp;
   if (( fp=fopen ("DATA.TXT", "r" )) ! =NULL)
    { while (fscanf (fp, "%d", & temp) ! =EOF)
       printf ("число t =%d \n", temp);
       fclose (fp);
   }
   else printf ("Помилка відкриття файлу DATA.TXT");
  getch ();
}
```

10

Довільний доступ до потоку

Існує п'ять функцій для позиціонування покажчика потоку:

1. ftell (point) - передає поточну позицію покажчика потоку:

```
long ipos;
ipos=ftelll (fp);
```

2. fgetpos(point, *long) – визначає поточну позицію покажчика потоку, передає 0, якщо визначення відбулося правильно й -1 інакше:

```
long *ipos;
fgetpos (fp, ipos);
```

3. fsetpos(point, *long) – установлює відповідну позицію покажчика потоку:

```
long *ipos;
fsetpos (fp, ipos)
```

11

4. fseek(fp, count, start), встановлення покажчика на будь-яку позицію покажчика потоку,
де **fp** - покажчик на файл;
count - кількість байт типу **long**, визначає абсолютну або відносну позицію у файлі;

start
- 0 – від початку файла; SEEK_SET
- 1 – від поточної позиції; SEEK_CUR
- 2 – від кінця файла. SEEK_END

```
fseek (fp, 0L, SEEK_SET); //на початок потоку
fseek (fp, n, SEEK_CUR); //на n байт від поточн. поз. вниз
fseek (fp, -n, SEEK_CUR); //на n байт від поточн. поз. вверх
fseek (fp, 0L, SEEK_END); //на кінець потоку
```

5. rewind (fp) перейти на початок файлу, аналогічно: ***fseek (fp, 0L, SEEK_SET)***.

12

```
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
#include <iomanip.h>
#include <stdio.h>
#define NUM 5

typedef struct mybas
{ char fio[25];
  int year;
  char sex;
} stud3[NUM];


FILE *filt,*filb;
FILE *filtp,*filbp,*fp;
```

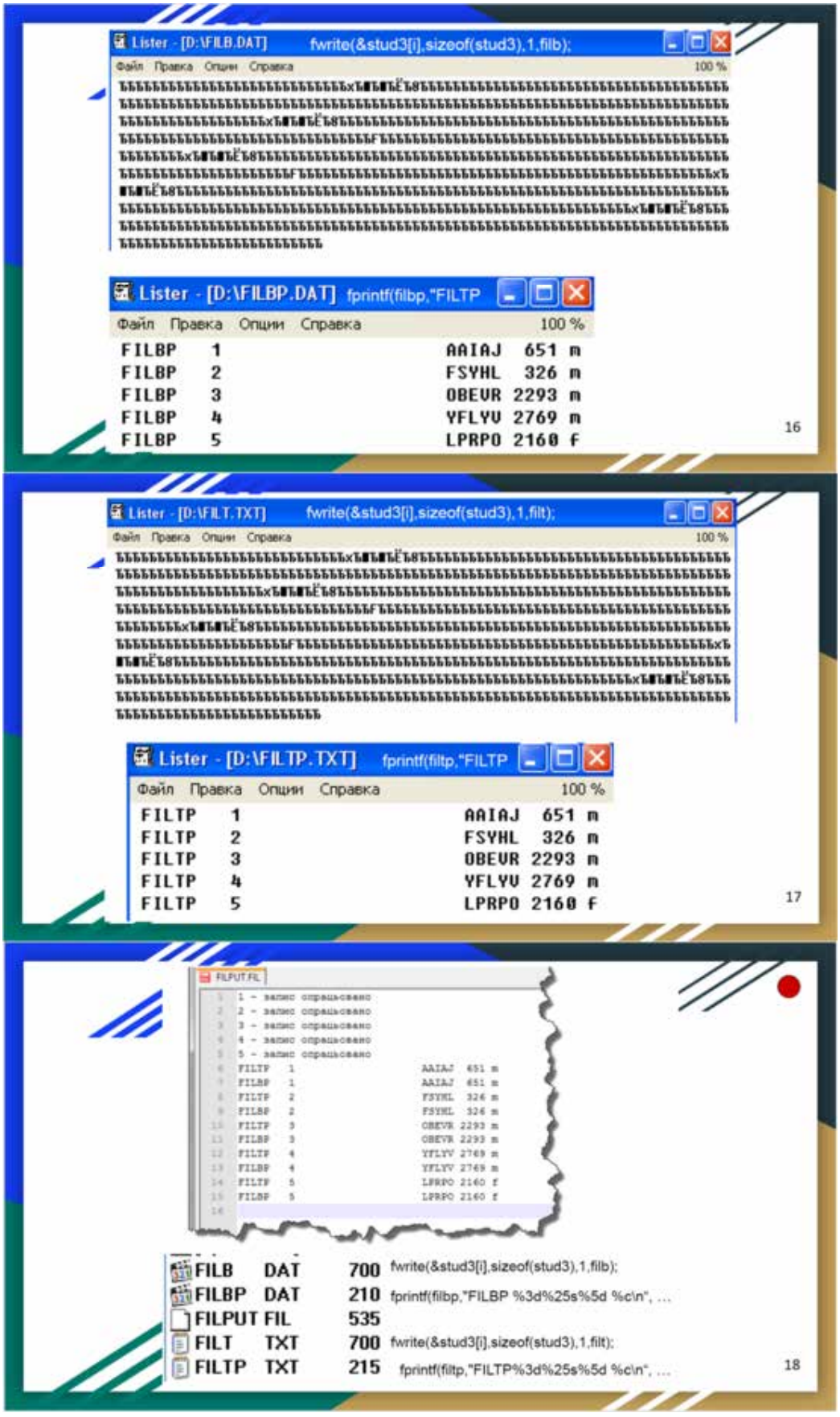
13

```
void main(void)
{ int a,i;
 clrscr();
 printf(«Відкриваємо файли і заповнюємо дані");
// Відкриття для запису текстового файлу
if (( filt=fopen("filt.txt","wt") ) == NULL)
 { printf("%d, Неможливо відкрити файл - filt.txt \n", stderr);
   getch();    exit(0); }

// Відкриття для запису двійкового файлу
 if (( filb=fopen("filb.dat","wb") ) == NULL)
 { printf("%d, Неможливо відкрити файл - filb.dat.\n", stderr);
   getch();    exit(0); }

// Відкриття для запису + читання текстового файлу
 if (( filtp=fopen("filtp.txt","w+t") ) == NULL)
 { printf("%d, Неможливо відкрити файл - filtp.txt.\n", stderr);
   getch();    exit(0); }

// Відкриття для запису + читання двійкового файлу
 if (( filbp=fopen("filbp.dat","w+b") ) == NULL)
 {printf("%d, Неможливо відкрити файл - filbp.dat.\n", stderr);
   getch();    exit(0); }
```

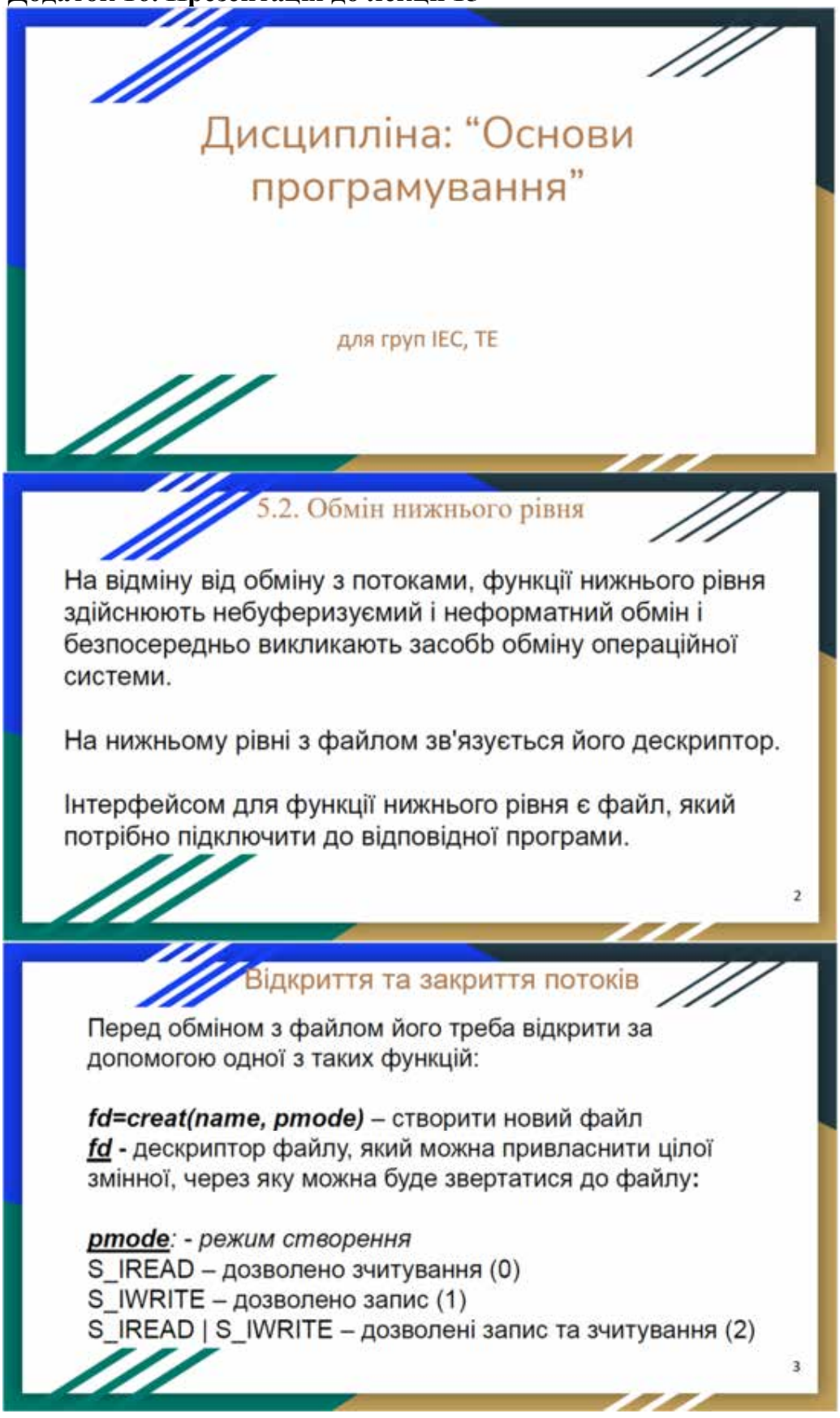
14

```
// Перевизначення стандартного потоку виводу
 if ( (fp=freopen("filPUT.FIL", "w+", stdout))== NULL)
   fprintf(stderr, " Помилка перенаправлення потоку stdout\n");
 for(a=0;a<NUM;a++) // генерація даних
 {
// генерація фамілії
   for(i=0;i<NUM;i++)   stud3[a].fio[i]=random(26)+65; stud3[a].fio[i]='\0';
   stud3[a].year=random(3000); // генерація року
   stud3[a].sex=random(2)?'m':'f'; // генерація статі
// передача даних у файл функцією – fwrite
    fwrite(&stud3[i],sizeof(mybas),1,filt);
    fwrite(&stud3[i],sizeof(mybas),1,filb);
// передача даних у файл функцією - fprintf -
    fprintf(filtp,"FILTP %3d%25s%5d %c\n",a+1,stud3[a].fio,stud3[a].year,stud3[a].sex);
    fprintf(filbp,"FILBP %3d%25s%5d %c\n",a+1,stud3[a].fio,stud3[a].year,stud3[a].sex);
// передача даних у потік stdout функцією - printf -
    printf("FILTP %3d%25s%5d %c\n",a+1,stud3[a].fio,stud3[a].year,stud3[a].sex);
    printf("FILBP %3d%25s%5d %c\n",a+1,stud3[a].fio,stud3[a].year,stud3[a].sex);
    printf(" %d - запис опрацьовано ", a+1);
 }
 fclose(stdout); // закрити потік перевизначення
 fcloseall();      // закрити всі відкриті файлові потоки
 getch();
}
```

15


Lister - [D:\FILB.DAT] fwrite(&stud3[i],sizeof(stud3),1,filb);
Файл Правка Опции Справка 100 %
Lister - [D:\FILBP.DAT] fprintf(filbp,"FILTP
Файл Правка Опции Справка 100 %
FILBP 1 AAIAJ 651 m
FILBP 2 FSYHL 326 m
FILBP 3 OBEUR 2293 m
FILBP 4 YFLYU 2769 m
FILBP 5 LPRPO 2160 f
16
Lister - [D:\FILT.TXT] fwrite(&stud3[i],sizeof(stud3),1,filt);
Файл Правка Опции Справка 100 %
Lister - [D:\FILTP.TXT] fprintf(filtp,"FILTP
Файл Правка Опции Справка 100 %
FILTP 1 AAIAJ 651 m
FILTP 2 FSYHL 326 m
FILTP 3 OBEUR 2293 m
FILTP 4 YFLYU 2769 m
FILTP 5 LPRPO 2160 f
17
FILPUT.FIL
1 - запис отрашьовано
2 - запис отрашьовано
3 - запис отрашьовано
4 - запис отрашьовано
5 - запис отрашьовано
FILTP 1 AAIAJ 651 m
FILBP 1 AAIAJ 651 m
FILTP 2 FSYHL 326 m
FILBP 2 FSYHL 326 m
FILTP 3 OBEVR 2293 m
FILBP 3 OBEVR 2293 m
FILTP 4 YFLYV 2769 m
FILBP 4 YFLYV 2769 m
FILTP 5 LPRPO 2160 f
FILBP 5 LPRPO 2160 f
FILB DAT 700 fwrite(&stud3[i],sizeof(stud3),1,filb);
FILBP DAT 210 fprintf(filbp,"FILBP %3d%25s%5d %c\n", ...
FILPUT FIL 535
FILT TXT 700 fwrite(&stud3[i],sizeof(stud3),1,filt);
FILTP TXT 215 fprintf(filtp,"FILTP%3d%25s%5d %c\n", ...
18

Додаток 16. Презентація до лекції 15



int fd=open (pathname, oflag [,pmode]) – відкрити файл

char *pathname – ім'я файлу

int oflag – дозволений тип операції

int rwmode – дозволений тип доступу

4

oflag: – дозволений тип операції
O_BINARY – в двійковому режимі;
O_TEXT – в текстовому режимі;
O_RDONLY – тільки для зчитування;
O_WRONLY – для запису.
O_RDWR – для зчитування і запису;
O_TRUNC – відкривається існуючий файл і весь зміст знищується;
O_APPEND – для запису в кінець файлу;
O_CREAT - створюється новий файл для запису;
O_EXCL- вертається значення помилки, якщо існує файл, обумовлений по **pathname** – ім'я файлу

```
handle =open ("data", O_CREAT | O_WRONLY);
```

5

```
# include <stdio.h>
# include <io.h>
# include <fcntl.h>

int handle;

if ((handle = open ("data", O_RDONLY)) == 1)

fprintf (stderr, "Помилка відкриття файлу: data \n");
```

6

int sopen (pathname, oflag, shflag [,pmode]) – відкриває файл для загального використання й підготовляє його до наступного розділеного читання або запису, що визначається значенням ***oflag*** або ***shflag***.

char *pathname – ім'я файла

int oflag - тип дозволених операцій

int shflag - дозволений тип розділення

int pmode дозволений тип доступу

7

shflag:

SH_COMPAT - встановлюється режим сумістності

SH_DENYRW – доступ читання й запису у файлі НЕ дозволено

SH_DENYWR – доступ запису у файлі НЕ дозволено

SH_DENYRD – доступ читання у файлі НЕ дозволено

SH_DENYNO – доступ читання й запису дозволено

8

int chmod(pathname,pmode) - змінює дозволений доступ для файлу, заданого ***pathname***

Якщо дозвіл на запис не задано, файл доступний тільки для читання.
В MS DOS усі файли доступні для читання, тому не можливе задання дозволу тільки на запис.

Тому режими:

S_IWRITE ⟺ S_IREAD | S_IWRITE

- є ***еквівалентними.***

9

Зчитування та запис даних

Для читання й запису інформації призначено дві функції **read** і **write**.

read (fd, pointer, size);

write (fd, pointer, size);

fd – дескриптор файлу, який був визначений функціями **create** і **open;**
pointer – покажчик на буфер даних, до якого записується (зчитується) інформація;
size – довжина цього буфера.

Якщо обмін успішний, то функції передають кількість фактично переданих байт. Коли помилка або кінець файлу, то функції передають значення EOF (-1).

10

Приклад: в один файл file2 дописати вміст file1

```
# include <stdio.h>
# include <io.h>
# include <fcntl.h>
# include <conio.h>
void main (void)
{ int hand1, hand2, size;
  char block [512], * file1, * file2;
  puts ("Ввести ім'я файлу (звідки дописувати) \ n");
  gets (file1);
    puts ("Ввести ім'я файлу (куди дописувати) \ n");
    gets (file2);
  printf (" %s %s  \n", file1, file2);
  if ((hand1=open (file1, O_RDONLY))==-1)
   { printf ("Помилка файлу %s \n", file1); return 1;  }
```

11

```
if ((hand2=open (file2, O_APPEND | O_CREAT))==-1)
    { printf ("Помилка файлу %s \n",file2);    return 1; }
  printf ("hand1=%d hand2=%d \n", hand1, hand2);
  while ((size=read (hand1, block, sizeof (block))) !=0)
  {   size=write (hand2, block, size);}
  close (hand1);
  close(hand2);
  getch();
}
```

12

Прямий доступ до файлів

Функції **read** і **write** здійснюють обмін у режимі послідовного доступу. При необхідності, можна реалізувати й прямий доступ до файлу. Для позиціонування файлу використовується функція
lseek (file_desc, no_bytes, start)
file_desc – дескриптор файла;
long no_bytes – задає абсолютну або відносну позицію байта у файлі;
start – ціле значення:
0 – від початку файла;
1 – від поточної позиции;
2 – від кінця файлу.

Як бачимо, формально функція **lseek** відрізняється від **fseek** - тільки першим аргументом є дескриптор файлу.

13

Також використовуються функції:

long tell (int handle) - повертає поточну позицію, на яку встановлено покажчик файлу.

filelenght (handle) - повертає кількість байтів у файлі.

unlink (file_specification) - видалення файлу

```
наприклад:
unlink ("C:\TEST.TXT");
```

Функція ***unlink*** повертає
0 - файл успішно видалено;
-1 - свідчить про помилку.
Функцію можна використовувати тільки тоді, коли файл НЕ відкрито.

14

Функції обміну з консоллю

Для небуферизованого обміну з консоллю призначені такі функції:
cgets - увести рядок з консолі;
cputs - вивести рядок на консоль;
getch - увести символ з консолі без відображення;
getche - те ж із зображенням;
putch - вивести символ на консоль;
cscanf (format-string[,argument...]) - форматироване уведення з консолі;
cprintf (format-string[,argument...]) - форматироване виведення на консоль;

15

kbhit - перевіряє, чи була натиснута клавіша й повертає ненульове значення, якщо клавіша була натиснута. Якщо ні, то вертає 0.
Прототипи відзначених функцій наведені в интерфейсному файлі **<conio.h>**.

kbhit можна використовувати для організації циклів, вихід з яких відбувається при натисканні клавіші:

while (! kbhit) оператор;

Просто затримка

while (! kbhit);

16

MS DOS функції обміну

Крім відзначених функцій обміну існують функції, які безпосередньо використовують системні виклики MS DOS:
_dos_open - відкрити файл;

_dos_create – створити новий файл;

_dos_close - закрити файл;

_dos_read - читати файл;

_dos_write - записати до файлу.

Прототипи цих функцій наведені в інтерфейсном файлі **dos.h.**

17

Функції для обміну з

void outport (int port, int word) – запис слова в порт з номером **port**.

Бібліотека:
DOS Win16
#include<dos.h>

void outportb (int port, char byte) – запис байта в порт з номером **port**.

Бібліотека:
DOS Win16
#include<dos.h>

18

Приклад передачі даних на COM

```
void send_to_com1 (char* val, int len)
 {
  init_com1(); // ініціювання COM порту
  int clocker=1;
  while (clocker<(len+1))
  {
    pause();
    asm
   { mov dx, 3f8h // адреса 3f8h - COM1
     mov al, val[clocker];
     out dx, al  // передати в COM1 байт
    }
   clocker++;
  }
 }
```

19

Приклад прийому даних з COM порта

```
char  receive_com1 (void)
 { char data='';
  init_com1();
  asm
 {   buf_clear:
    mov dx, 3fdh  // адреса 3fdh - готовність COM1
    in al, dx // прийняти з порту байт
    test al, 1 // байт нульовий ?
    jz buf_clear // якщо нульовий - перейти на початок
    mov data, al // передати байт в змінну data
  }
  return (data);
 }
```

20