ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра «Обчислювальна техніка та системи управління»

АЛГОРИТМІЧНІ МОВИ ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з дисципліни
«Комп'ютерна техніка та програмування»
для студентів факультету УПП

ЧАСТИНА 1

Харків 2012

Методичні вказівки розглянуто та рекомендовано до друку

на засіданні кафедри обчислювальної техніки та систем управління 29 березня 2011 р., протокол № 8.

Методичні вказівки призначені для студентів факультету УПП та відповідають робочій програмі дисципліни «Комп'ютерна техніка та програмування» для всіх форм і строків навчання.

Укладачі:

доценти С.Є. Бантюков,
Н.М. Завгородня,
старші викладачі С.О. Бантюкова,
О.Є. Пєнкіна

Рецензент

проф. Г.І. Загарій

АЛГОРИТМІЧНІ МОВИ ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з дисципліни
«Комп'ютерна техніка та програмування»
для студентів факультету УПП

ЧАСТИНА 1

Відповідальний за випуск Бантюкова С.О.

Редактор Ібрагимова Н.В.

Підписано до друку 29.09.11 р.

Формат паперу 60х84 1/16 . Папір писальний.

Умовн.-друк.арк. 1,5. Тираж 100. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту
61050, Харків - 50, майдан Фейєрбаха, 7
Свідоцтво суб'єкта видавничої справи ДК № 2874 від 12.06.2007 р.

УКРАЇНСЬКА ДЕРЖАВНА АКАДЕМІЯ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ
Кафедра „Обчислювальна техніка та системи управління"

АЛГОРИТМІЧНІ МОВИ ПРОГРАМУВАННЯ
МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни
«Комп'ютерна техніка та програмування»
для студентів факультету УПП
Частина 1

Харків 2012

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 29 березня 2011 р., протокол № 8.

Методичні вказівки призначені для студентів факультету УПП та відповідають робочій програмі дисципліни “Комп'ютерна техніка та програмування “ для всіх форм і строків навчання.

Укладачі :

доценти С.Є. Бантюков,
Н.М. Завгородня,
старші викладачі С.Є. Бантюкова,
О.Є. Пєнкіна

Рецензент

проф. Г.І. Загарій

ВСТУП

У наш час сфера розповсюдження обчислювальної техніки не має кордонів. Персональні ЕОМ дозволяють вирішувати важливі виробничі завдання, вести навчання фахівців, управляти складними процесами і т. п. Зараз вже очевидно, що навики роботи з такою технікою необхідні кожному інженерові.

Сьогодні ЕОМ виступає в ролі приємного співрозмовника, не стомлює, підказує, нагадує. Діалог між ЕОМ та людиною ведеться мовою, що допускає його однозначне тлумачення з обох сторін. Важко перерахувати всі мови та інструментальні системи програмування для ЕОМ, що реалізують їх. Найбільш популярними та широко використовуваними є Basic, C, Pascal.

Справжні методичні вказівки присвячені мові програмування C++. Вони складаються з семи робіт, у яких описані робота в середовищі програмування Borland C++, версії 3.1 (далі по тексту C++), елементарні конструкції мови C++, основні оператори та функції для програмування лінійних, розгалужених, циклічних обчислювальних процесів, обробки масивів.

C++ вважається мовою середнього рівня, тому що вона з'єднує в собі елементи мов високого рівня з функціональністю Асемблера.

Програма, написана мовою C++, дуже мобільна. Це означає, що програмне забезпечення, написане для комп'ютеру одного типу, можна легко адаптувати для комп'ютеру іншого типу.

Мову С++ часто називають структурованою мовою. Відмітною рисою структурованої мови є наявність можливості відокремлювати дані від програми та розбивати програму на окремі функції, кожна з яких є незалежною одиницею, що виконує певну задачу та допускає вилучення її з програми.

При використанні системи Borland C++ користувачі мають у своєму розпорядженні найшвидший та ефективний з можливостей компілятор та інтегроване середовище, що дозволяє максимально автоматизувати процес розроблення програм.

У методичних вказівках у записах форматів застосовують такі позначення:

<*значення*> – обов'язковий елемент, на місці якого повинна бути послідовність символів, що допущена в даному форматі;

[рядок_символів] – рядок символів, що укладено у квадратні дужки, вважається необов'язковим і може бути відсутньою.

Лабораторна робота 1
РОБОТА В ІНТЕГРОВАНОМУ СЕРЕДОВИЩІ ПРОГРАМУВАННЯ BORLAND C++ ВЕРСІЇ 3.1

Мета роботи: отримання навиків роботи в інтегрованому середовищі програмування Borland C++ версії 3.1, знайомство з командами основного меню, основними командами редактора.

Завдання та порядок виконання

1 Вивчити теоретичний матеріал.

2 Підготувати відповіді на контрольні запитання.

3 Запустити Borland C++. У вікні редактора набрати приклад навчальної програми. Випробувати команди редактора C++. Випробувати команди основного меню.

Контрольні запитання

1 Чому середовище для розроблення C++-програм називається інтегрованим?

2 З яких частин складається екран основного меню? Їх призначення.

3 Призначення опцій основного меню.

4 Команди редактора C++.

5 Команди основного меню.

Зміст звіту

1 Номер роботи, її назва, визначення мети.

2 Стислий зміст теоретичного матеріалу та відповіді на контрольні запитання.

3 Результати виконання завдання: схеми алгоритмів, програми, результати роботи програм.

4 Висновки до роботи.

Навчальний матеріал

1 Запуск C++. Для запуску C++ слід набрати в командному рядку *bc* і натиснути клавішу *Enter* або обрати опцію *«Borland C++»* (від англ. *option* - вибір) у меню користувача, що викликається натисканням функціональної клавіші *F2*. При цьому на екрані з'явиться картинка, що називається екраном основного меню і складається з 4 частин:

- основного меню;
- вікна редактора;
- вікна повідомлень;
- рядка підказки чи зазначення дій функціональних клавіш.

Для виходу з середовища C++ використовується комбінація клавіш *Alt+X* або команда *Quit* (Вихід) з опції *File* основного меню.

Розглянемо кожну з 4 частин.

Основне меню. Основне меню використовується або для зазначення дій, або для встановлення певних опцій середовища. Для переходу в основне меню використовують функціональну клавішу *F10*. Для вибору команд з основного меню користуються клавішами управління курсором, переміщаючи підсвічування на необхідну опцію та натискаючи клавішу введення. У таблиці 1.1 описуються дії кожної опції.

Вікно редагування. Під головним меню знаходиться вікно редагування. Програма редагується всередині цього вікна.

Вікно повідомлень. Вікно повідомлень знаходиться під вікном редагування та використовується для відображення повідомлень компілятора та укладача.

Таблиця 1.1 - Дія опцій основного меню

Опція	Дія
≡	Системні команди та програми перетворення
File	Команди управління файлами (завантаження, збереження і т. п.), виклик ДОС, вихід з С++
Edit	Містить команди редактора С++
Search	Виконує пошук, заміну символів у тексті, перехід до заданого рядка
Run	Компілює та виконує програму
Compile	Компілює програму, управляє багатофайловими об'єктами
Debug	Дозволяє влаштовувати різноманітні параметри налагодження
Project	Використовується для створення та супроводу великих багатофайлових програм
Options	Дозволяє влаштовувати різноманітні параметри середовища, компілятора та укладача
Window	Використовується при роботі з вікнами
Help	Допомога

Рядок підказки чи зазначення дій функціональних клавіш. При використанні основного меню в цьому рядку показується підказка про те, що виконує та чи інша команда. У режимі редагування в цьому рядку показуються функціональні клавіші інтерактивного середовища С++, що використовуються для активізації меню чи швидкого виконання стандартних функцій. Оскільки функціональні клавіші постійно активні, за їх допомогою можна викликати будь-яке меню, не зважаючи на поточну роботу. У рядку показуються функціональні клавіші, що найбільш придатні для роботи, що виконується в даний момент.

2 Використання редактора С++. Редактор С++ дозволяє маніпулювати цілими блоками тексту. С++-дії вимагають, щоб спочатку була виділена необхідна частина тексту, а потім виконана дія. Зробити виділення можна, помістивши курсор у початок блока, що виділяється, і натиснути комбінацію *Shift+клавіші управління курсором (стрілки)*. Курсор необхідно пересунути в кінець частини, що виділяється. При цьому С++ висвітить відмічену частину тексту.

Для переміщування блока необхідно спочатку забрати виділений блок у пам'ять - комбінація клавіш *Shift+Del* (при цьому виділений блок зникне на екрані), перемістити курсор у нове місце призначення та натиснути *Shift+Ins*. Блок з'явиться в новому місці. У пам'яті може зберігатися тільки один виділений блок. При наступному запам’ятовуванні блока, попередній блок буде знищено.

Для копіювання виділеного блока спочатку необхідно натиснути *Ctrl+Ins*, перемістити курсор у нове місце призначення та натиснути *Shift+Ins*. У новому місці з'явиться копія виділеного блока.

Для вилучення виділеного блока необхідно натиснути *Ctrl+Del*.

Для відміни останньої виконаної команди необхідно натиснути *Alt+Backspace*.

3 Використання команд основного меню. Нижче наведена таблиця деяких команд основного меню та відповідних їм клавіш для виконання основних дій при налагодженні та виконанні програм.

Таблиця 1.2 - Команди та відповідні клавіші основного меню

Клавіша	**Команда**
Shift+F1	Виклик допомоги
Ctrl+F1	Виклик контекстозалежної допомоги
F2	Збереження файлу
F3	Відкриття файлу
F9	Компілювання програми
Ctrl+F9	Виконання програми
F5	Збільшити вікно / зменшити вікно
F6	Наступне вікно
Alt+F3	Закрити вікно
Alt+F5	Екран користувача

Після набору тексту програми у вікні редактора програму необхідно виконати. Для цього використовується комбінація клавіш *Ctrl+F9*. Після натискання *Ctrl+F9* на екрані з'явиться вікно компілятора. У нижньому рядку вікна компілятора виводяться повідомлення про результат компіляції. Якщо програма не містить

синтаксичних помилок, у нижньому рядку з'являється повідомлення «Success : Press any key» і на екрані користувача відображається результат виконання програми. Для виклику екрана користувача необхідно натиснути комбінацію клавіш *Alt+F5*, при цьому екран основного меню зникне. Для повернення екрану основного меню необхідно натиснути будь-яку клавішу. Якщо в програмі містяться синтаксичні помилки, то в нижньому рядку з'являється повідомлення «Errors : Press any key».

Після натискання будь-якої клавіші вікно компілятора зникне і активним стане вікно повідомлень, у якому з'являться повідомлення про найбільш імовірні помилки. Якщо у вікні повідомлень містяться повідомлення про деякі помилки, то при пересуванні курсором по рядках повідомлень про помилки у вікні редактора курсор буде синхронно пересуватися по рядках, які містять зазначену помилку.

Для переходу у вікно редагування для виправлення помилок необхідно натиснути клавішу *F6*. Після виправлення помилок програму необхідно знову виконати, натиснувши комбінацію клавіш *Ctrl+F9*. Слід мати на увазі, що одна помилка може потягти за собою кілька інших помилок, усунувши яку, усуваються й інші.

Лабораторна робота 2
ВИВЧЕННЯ ЕЛЕМЕНТАРНИХ КОНСТРУКЦІЙ МОВИ С++

Мета роботи: вивчення правил запису елементарних конструкцій мови програмування С++, змінних, констант, операторів, виразів.

Завдання та порядок виконання

1 Вивчити теоретичний матеріал.
2 Підготувати відповіді на контрольні запитання.
3 Записати мовою С++ математичні вирази.

Контрольні запитання

1 Визначити поняття «ідентифікатор». Правила складання ідентифікаторів.
2 Змінні. Типи, діапазони їх значень, правила оголошення змінних.
3 Константи. Вигляди констант.
4 Основні класи операторів.
5 Визначити поняття «блок».
6 Визначити поняття «препроцесор мови С++». Директиви *#include, #define*.

Зміст звіту

1 Номер роботи, її назва, визначення мети.
2 Стислий зміст теоретичного матеріалу та відповіді на контрольні запитання.
3 Результати виконання завдання.
4 Висновки до роботи.

Навчальний матеріал

1 Ідентифікатори. Ідентифікатори - імена, що використовуються для звернення до змінних, функцій, міток та інших визначених користувачем об'єктів. Ідентифікатор може містити від одного до декількох символів. Перший символ обов'язково повинен бути літерою латинського алфавіту чи літерой підкреслення. Значущими символами ідентифікаторів С++ є тільки перші 32. Це означає, що якщо ідентифікатори двох змінних мають однакові перші 32 символи та розрізняються тільки з 33-го символу, то С++ ці змінні не розрізняє.

У мові С++ прописні та рядкові літери трактуються як різні. Наприклад, ідентифікатори *count*, *Count* і *Count* представляють три різні змінні. Ідентифікатор змінної не повинен співпадати з ключовим словом, з ім'ям бібліотечної чи функції користувача.

2 Змінні. Всі змінні перед використанням повинні бути явно оголошені.

У мові С++ використовуються дані 5 типів: символи, цілі числа, числа з рухомою точкою, числа з рухомою точкою подвійної точності та змінні без значення. Ключовими словами для оголошення змінних цих типів є *char, int, float, double* та *void* відповідно. Нижче наведено діапазони значень для змінних цих типів.

Тип змінної	**Діапазон значень**
char	від -128 до 127
int	від -32768 до 32767
float	від 3.4e-38 до 3.4e+38
double	від 1.7e-308 до 1.7e+308
void	значень нема

Оголошення змінних. Формат оператора оголошення змінних має вигляд

<тип> <список_змінних>;,

де *тип* - будь-який допустимий тип змінних;

список_змінних - один чи більше ідентифікаторів, розділених комами.

Наприклад: *int i, j, l;*
double less, roz;.

Тут змінні *i, j, l* оголошуються як змінні цілого типу, а змінні *less, roz* - як числа з рухомою точкою подвійної точності.

Залежно від місця оголошення змінних вони мають різні області використання. Змінні, що оголошені поза всіх функцій, включаючи функцію *main()*, називаються глобальними і можуть використовуватися всіма блоками програми. Змінні, що оголошені всередині функцій, називаються локальними і можуть використовуватися тільки всередині даної функції. Змінні також можуть оголошуватися в місці оголошення формальних параметрів функції. (Формальний параметр означає змінну у функції, що набуває значення аргументу, що міститься у виклику функції.) Формальні параметри працюють як звичайні локальні змінні.

3 Константи. Константами вважаються величини, що не змінюються у процесі виконання програми. Основні види констант:

Тип даного	**Вид константи**
char	*'a', '\n', '9'*
int	*1, 123, 21000*
float	*12.23, 4.34e-3*
double	*12345212, 1.0e100*

Рядкові константи. Крім зазначених видів констант, існує ще один вид - рядковий. Рядок - це набір символів, укладених у подвійні апострофи, наприклад *"це текст"*.

Не плутайте рядкові константи з символьними: перші укладаються у подвійні апострофи, другі - в одинарні.

Символьні константи зі зворотним слешем. Є символи, що не можна ввести з клавіатури, наприклад повернення каретки. Для таких символів передбачено спеціальні символьні константи зі зворотним слешем:

\f - встановлення на нову сторінку;
\n - встановлення на новий рядок;
\r - повернення каретки;
\t - горизонтальна табуляція;
\v - вертикальна табуляція.

Ініціалізація змінних. Змінним можна присвоїти значення під час їх оголошення. Загальний формат ініціалізації такий:

<тип> <ім'я_змінної> = <константа>;.

Наприклад: *char ch = 'a';*
int first = 0;.

4 Оператори. У мові С++ існують три основних класи операторів:

- арифметичні;
- логічні;
- оператори порівняння.

Арифметичні оператори

Оператор	**Дія**
+	Додавання
-	Віднімання та знак мінус
*	Множення
/	Ділення
%	Цілочисельне ділення (дає остачу від ділення цілих чисел)
++	Додатний приріст
--	Від'ємний приріст

Додатний та від'ємний приріст. Оператор «++» додає до свого операнда 1, а оператор «--» віднімає 1. Таким чином, оператори *x++;* та *x--;* еквівалентні операторам *x=x+1;* та *x=x-1;*.

Оператори «++» та «--» можуть стояти перед або після операндів, наприклад, оператор *x=x+1;* можна записати як *++x;* або як *x++;*.

Однак зазначені оператори працюють по-різному. Якщо оператори додатного чи від'ємного приросту знаходяться попереду операнда, то спочатку виконується операція приросту і після цього використовується значення операнда. Якщо оператор приросту знаходиться після операнда, то спочатку використовується поточне значення операнда і після цього виконується приріст. Наприклад, у випадку

x=10; *y=++x;*

C++ присвоює *y* значення *11*, оскільки спочатку виконується приріст *x* і після цього його значення присвоється *y*. Однак у випадку

x=10; *y=x++;*

y присвоється значення *10*, а після цього виконується збільшення значення *x*.

Приорітет арифметичних операцій такий:

найбільший ++, --
-
*, /, % ;
найнижчий +, - .

Послідовність операторів одного пріоритету виконується зліва направо. Для зміни цього порядку можна використовувати дужки. Вираз у дужках виконується в першу чергу.

Оператори порівняння та логічні оператори. Оператори порівняння дозволяють визначити співвідношення двох величин. Логичні оператори встановлюють взаємозв'язок цих співвідношень.

В основі роботи логічних операторів та операторів порівняння лежать поняття «істинно» та «хибно». Істінним вважається будь-який вираз, відмінний від нуля, нульове значення – «хибно».

Оператори порівняння	**Логічні оператори**

Оператор	Дія
>	більш ніж
>=	більше або дорівнює
<	менш ніж
<=	менше або дорівнює
==	дорівнює
!=	не дорівнює

Оператор	Дія
&&	AND
||	OR
!	NOT

Всі оператори порівняння та логічні оператори мають пріоритет нижче, ніж арифметичні оператори.

Пріоритети логічних операторів та операторів порівняння:

найбільший !
>, >=, <, <=
==, !=
&& ;
найнижчий || .

Як і в разі арифметичних виразів, порядок виконання цих операторів можна змінювати за допомогою дужок.

5 Вирази. Виразом у мові С++ вважається будь-яке допустиме поєднання операторів, констант і змінних.

Оскільки більшість виразів співпадає з алгебраїчними формулами, правила запису та виконання виразів звичайно приймають такими самими. Однак існують і свої особливості.

Перетворення типів у виразах. Якщо в арифметичному виразі зустрічаються операнди різних типів, то один з операндів підлягає перетворенню типів так, щоб він відповідав типу іншого операнда. Операнд для перетворення вибирається за таким правилом. У С++ основні типи мають визначений порядок старшинства, що визначає, який операнд перетворюється, а який ні. Якщо дивитися справа наліво, то порядок старшинства типів такий:

символьні < цілі < рухомі < подвійної точності.

Типи, які знаходяться праворуч, перевищують за старшинством

всі типи, що знаходяться ліворуч.

6 Основні математичні функції. Основні математичні функції наведено в таблиці 2.1.

Таблиця 2.1 - Основні математичні функції

Позначення функції в мові С++	**Пояснення**
sin(x)	Функція синуса. Обчислює синус аргументу
cos(x)	Функція косинуса. Обчислює косинус аргументу
tan(x)	Функція тангенса. Обчислює тангенса аргументу
log(x)	Функція натурального логарифма. Обчислює натуральний логарифм додатного аргументу
abs(x)	Функція - абсолютна величина. Обчислює абсолютне значення (модуль) цілого аргументу
fabs(x)	Функція - абсолютна величина. Обчислює абсолютне значення (модуль) аргументу з плаваючою точкою
exp(x)	Експоненціальна функція. Обчислює *e* в ступені *x*, де *e=2.71*
sqrt(x)	Функція квадратного кореня. Обчислює квадратний корінь додатного аргументу
pow(x,y)	Функція ступеню. Обчислює *x* в ступеню *y*

Для виклику функцій необхідно підключити бібліотеку <*math.h*> за допомогою директиви *#include* <*math.h*>. Про цю директиву буде сказано нижче.

Приклад запису функцій:

Математичний запис	Запис мовою С++
$z = sin(x)^2$	*z = pow(sin(x),2);*
$z = sin(x^2)$	*z = sin(pow(x,2));*.

7 Структура програми мовою С++. Мова С++ побудована на

концепції складених блоків, що називаються функціями. Програма складається з однієї чи більше функцій. При написанні програми пропонується спочатку створити функції, а після цього об'єднати їх.

Кожна функція являє собою програму, що містить один чи більше операторів і виконує одну чи декілька задач. У добре складеній програмі кожна функція виконує тільки одну задачу.

У загальному випадку функція має такий формат:

```
<тип_значення,_що_повертається> <ім'я_функції> (<параметри>)
{
  <оголошення змінних>;

  <тіло програми>;
}
```

Всі програми мовою С++ повинні мати функцію *main()*, оскільки саме з неї починається виконання програми. Програма може містити одну і тільки одну функцію з ім'ям *main()*.

8 Характерні особливості мови С++. Крапка з комою - обмежувач оператора. Таким чином, кожний окремий оператор закінчується цим символом.

С++ не розпізнає кінця рядка як обмежувач. Це означає, що для наочності можна групувати оператори на одному рядку, як показано нижче:

```
x = y;
y = y+1;
mul(x, y);
```

це те саме, що і

```
x = y;  y = y+1;  mul(x, y);
```

Для зручності читання програми можна розміщати пропуск в будь-яке її місце.

Блок - сукупність логічно зв'язаних операторів, укладених у

дужки. Введення додаткових дужок не викличе помилок і не сповільнить обчислення виразів. Додаткові дужки використовуються для того, щоб був ясний порядок обчислення виразів.

Розглянемо програму:

```
// Програма номер 1

#include <stdio.h>
main()
{
int  aud;
aud = 222;
printf("Номер аудиторії %d\n", aud);
}
```

Перший рядок. У С++ коментарі починаються з символів //. Все, що знаходиться після зазначених символів до кінця поточного рядка, компілятор С++ ігнорує. Для виділення блока коментарів використовують символи /* на початку блока та символи */ в кінці блоку коментарів.

Другий рядок - пустий рядок. Пусті рядки дозволені, вони не мають ніякого впливу на виконання програми.

Третій рядок - *#include <stdio.h>* - директива препроцесора, що підключає стандартну бібліотеку введення/виведення мови С++. Директива *#include* буде розглянута нижче.

П'ятий рядок *main()* визначає ім'я функції. Всі програми в С++ починають виконуватися з виклику основної функції *main()*. Виконання програми припиняється по досягненню кінця функції *main()*.

Наступний рядок складається з однієї фігурної дужки {, що означає початок основної функції *main()*.

Першим рядком програми всередині функції *main()* є *int aud;* - це є оголошенням змінної з ім'ям *aud*, що може набувати тільки цілих значень. Наступний рядок

aud = 222;

є оператором присвоювання. Цей оператор присвоює значення *222* змінній *aud*.

Наступний рядок виводить інформацію на екран:

printf ("Номер аудиторії %d\n ", aud);.

У С++ нема операторів введення/виведення. Замість цього існують функції введення/виведення, прототипи яких знаходяться у стандартній бібліотеці <*stdio.h*>. Бібліотека підключається за допомогою директиви препроцесора *#include* і викликається за необхідності. Виклик функції виконується просто: достатньо вказати ім'я функції і записати необхідні атрибути (аргументи).

Виклик функції *printf()* буде розглянуто при вивченні операторів введення/виведення.

В останньому рядку програми знаходиться фігурна дужка, що означає закривання, що говорить про завершення основної функції *main()*.

Так, як створюється функція *main()*, можна створювати й інші функції та викликати їх з інших частин програми. Наприклад, у наведеній нижче програмі використовується функція *hello()* для друкування на екрані слова *hello()*:

```
// Програма, що використовує дві функції

#include <stdio.h>
main()
{
void hello(void);        // прототип функції hello()
hello();                 // виклик функції hello()
}

void hello(void) // функція hello()
{
printf("hello");
}
```

Звичайно функція описується до того, як вона буде визначена. Опис інформує компілятор про існування функції, про тип значення, що повертається, а також про тип параметрів, що їй передаються. Опис функції часто називають прототипом функції. Опис функції має такий вигляд:

<тип_значення,_що_повертається> <ім'я_функції> (<параметри>);.

9 Препроцесор мови С++. У програмах мовою С++ широко використовуються можливості препроцесора. **Препроцесор** - це програма, що обробляє вхідний модуль до того, як він пройде через компілятор, заміняючи визначені імена еквівалентними їм рядками.

Рядки, що містять директиви препроцесора, розпочинаються з символу номеру *(#)*.

Директива *#include*. Директива *#include* здійснює підстановку замість себе тексту, зазначеного в директиві файлу. Вхідний файл, що буде читатися, повинен бути укладений у подвійні лапки або гострі дужки. Наприклад:

#include "stdio.h"
#include <stdio.h>

Якщо ім'я файлу укладене в лапки, то компілятор спочатку буде шукати ім'я файлу в поточному робочому каталозі. Якщо компілятор не знаходить файл, то він шукає ім'я файлу в будь-якому каталозі, який специфіковано в командному рядку.

Укладення ім'я файлу в гострі дужки означає, що пошук цього файлу буде здійснюватися у специфікованому каталозі С++. Якщо компілятор не знаходить файл, то пошук буде виконуватися у стандартному каталозі.

У мові С++ файли з поширенням *.h* називаються файлами-заголовками (Header File). Вони містять опис змінних, функцій, типів, що використовується багатьма програмами. У даному випадку у файлі *stdio.h* міститься опис, необхідний для використання стандартної бібліотеки введення/виведення мови С++. Ім'я файлу - це скорочення *Standard Input/Output*. Цей файл буде включатися перед всіма програмами, де є введення або виведення.

Директива *#define*. Директиву препроцесора *#define* застосовують для того, щоб визначити ідентифікатор та ланцюжок, який компілятор буде підставляти замість ідентифікатора кожний раз, коли він зустрічається у вхідному файлі. Ідентифікатор називається макроіменем, а процес підстановки називається макропідстановкою. Загальний формат цієї директиви має такий

вигляд:

#define <ідентифікатор> <ланцюг>.

Наприклад, для використання *True* у вигляді значення *1* та *False* в якості значення *0* необхідно оголосити два *#define* так, як показано нижче:

#define True 1
#define False 0.

Ці рядки змушують компілятор підставляти *1* або *0* кожний раз, коли він зустрічає у вхідному файлі *True* або *False*.

Наприклад, наведена нижче функція *printf()* буде друкувати на екрані *0 1 2*:

printf(" %d %d %d", False, True, True+1);.

Після того як макроім'я визначено, його можна використовувати у вигляді частини визначення інших макроімен. Наприклад:

#define One 1
#define Two One+One
#define Three One+Two.

Варіанти індивідуальних завдань до лабораторної роботи 2

Варіант	Функції для розрахунків
1	$q = \frac{2x + \ln^2 x + \sqrt{x}}{a + mc + t}.$
2	$t = \frac{2s + m^2}{a - x} + \sqrt{s + c^3} - x^2.$
3	$y = \frac{cx^2 + mb^3 + \ln x + 7}{b + m + \sqrt{c}}.$
4	$r = \frac{mc + \sqrt{2x} + f^2}{\sin^2 x + 3c}.$
5	$z = \frac{a + bx}{m} + \frac{c}{a + x} - q.$
6	$c = \frac{a + \sin 2x}{m - r} + \frac{\cos^2 x}{ax}.$
7	$f = \frac{3y}{m + x} + \sqrt{\frac{b}{m + 2x}} + \sin(m^2 + x).$
8	$r = \sqrt{t + 2q} + \frac{m + x}{\ln x} + \frac{c + 2{,}5d}{a + x} + \sqrt{x^2 + a}.$
9	$s = \frac{7}{m + 2t} + \frac{a}{b + cx} + m^2 - 4z + \sqrt{a + 2cx^2}.$
10	$z = \frac{2x}{b + y} + \frac{c}{b - y} + \frac{2bx}{b - xy} + 2\ln(\lvert x - a\rvert).$
11	$q = mc^2 + 2f^5 + \cos^2 x + \frac{a}{c} + \frac{b}{x} + \sqrt{f}.$
12	$z = \frac{m^2 cx}{\sqrt{a}} + bc^3 - \ln a + \sin 2x^2.$
13	$r = \frac{c + 2x^2}{m} + at^2 + \sqrt{x} + \frac{2 + x}{c}.$
14	$q = \frac{3m}{ax} + \sqrt{b + cx} + \lg(\lvert b - mx\rvert) + r^3 + \cos x.$
15	$q = \frac{at^2 + rc - ax}{b + \sqrt{c + x} - 5} + (mx + 2t + c)^3.$

Лабораторна робота 3
ПРОГРАМУВАННЯ ЛІНІЙНИХ ОБЧИСЛЮВАЛЬНИХ

ПРОЦЕСІВ

Мета роботи: вивчення оператора присвоювання, функцій введення/виведення. Придбання навичок складання програм лінійних обчислювальних процесів і виконання їх на ПЕОМ.

Завдання та порядок виконання

1 Вивчити теоретичний матеріал.
2 Підготувати відповіді на контрольні запитання.
3 Скласти програми мовою С++ згідно зі складеними алгоритмами.
4 Ввести підготовлені програми у ПЕОМ, виконати їх та отримати результати розрахунків.

Контрольні запитання

1 Структура програм мовою С++.
2 Формат запису оператора присвоювання.
3 Формат запису функції введення. Коди формату.
4 Формат запису функції виведення. Коди формату, модифікатори.

Зміст звіту

1 Номер роботи, її назва, визначення мети.
2 Стислий зміст теоретичного матеріалу та відповіді на контрольні запитання.
3 Результати виконання завдання: схеми алгоритмів, програми, результати роботи програм.
4 Висновки до роботи.

Навчальний матеріал

1 Оператор присвоювання. Оператор присвоювання зображується знаком рівності «=». Він присвоює значення, що отримано у правій частині, змінній у лівій частині виразу.

Формат оператора присвоювання:

<ім'я_змінної> = <значення>;.

Наприклад: aud = 222;

2 Функція *scanf()*. Універсальною функцією введення є функція *scanf()*. Вона може читати всі типи даних і автоматично перетворює числа в належний внутрішній формат.

Формат функції *scanf()*:

scanf ("<керівний_рядок>", <список_аргументів>);.

Список аргументів повинен містити стільки аргументів, скільки специфікацій формату знаходиться в керівному рядку.

У керівному рядку перед специфікаторами формату введення, які повідомляють функції *scanf()* тип даних, що будуть читатися, розташовується знак %.

Коди формату функції *scanf()*:

Код	Значення
%d	Читати десяткове ціле
%f	Читати число з рухомою точкою
%c	Читати окремий символ
%s	Читати рядок символів
%o	Читати вісімкове число
%x	Читати шістнадцяткове число
%p	Читати покажчик

Іменам змінних, що отримують значення шляхом введення з клавіатури, повинен передувати знак *&*. (Всі змінні, які використовуються для набування значень шляхом функції *scanf()*, повинні передаватися їх адресами. Це означає, що всі аргументи повинні бути покажчиками на змінні, які використовуються у вигляді аргументів.)

Наприклад, для того щоб зчитати ціле у змінну *count*, можна використати такий виклик функції *scanf()*:

scanf("%d", &count);

читання двох значень у дві змінні відповідно:

scanf("%d,%d", &i, &j);.

3 Функція *printf()*. Для форматного виведення використовується функція *printf()*. Її формат

printf("<керівний_рядок>", <список_аргументів>);.

Тут «керівний_рядок» містить або символи, що виводяться на екран, або специфікації формату, що визначають спосіб подання аргументів, що виводяться, або і те й інше. Нижче наводяться допустимі специфікації формату.

Код	Значення
%d	Виводити десяткове ціле
%f	Виводити число з рухомою точкою
%c	Виводити окремий символ
%s	Виводити рядок символів
%o	Виводити вісімкове число
%x	Виводити шістнадцяткове число
%p	Виводити покажчик

Специфікації формату можна включати в будь-яке місце керівного рядка. При викликанні функція *printf()* проглядає керівний рядок і запам'ятовує специфікації формату. Після цього вона виводить на екран символи в порядку їх появи, а при зустрічі аргументу виводить його відповідно до специфікації формату. При цьому функція ставить у взаємно однозначну відповідність зліва направо специфікації формату та аргументи. Число специфікацій формату визначає число очікуваних аргументів. Наприклад:

printf (" %s %d", "це рядок", 100);
виводить на екран: *це рядок 100*

printf ("це рядок %d", 100);
виводить на екран: *це рядок 100*

printf ("число %d - десяткове, %f - дробове", 10, 110.789);
виводить на екран: *число 10 - десяткове, 110.789 - дробове.*

Команди формату можуть мати модифікатори, які специфікують ширину поля, кількість десяткових розрядів і прапорець вирівнювання за першим лівим знаком чи розрядом. Ціле, що розташовується між знаком % і командою формату, діє як специфікатор мінімальної ширини поля. Цей специфікатор змушує ПЕОМ доповнювати виведення пропусками чи нулями, для того щоб забезпечити визначену мінімальну його довжину. Якщо ланцюжок символів чи число перевищує цей мінімум, то функція *printf()* буде друкувати їх повністю, навіть якщо вони виходять за межу цього мінімуму. За умовчанням, для доповнення виведення використовуються пропуски. Щоб доповнити виведення нулями, необхідно розташувати *0* перед специфікатором ширини поля. Наприклад, *%05d* буде доповнювати число, яке складається менш ніж з *5* цифр, нулями перед значенням числа, для того щоб загальна довжина була рівною п'яти.

Щоб специфікувати кількість десяткових розрядів, які необхідно надрукувати в разі числа з десятковою точкою, десяткова точка розміщується після специфікатора ширини поля та за нею - кількість десяткових розрядів. Наприклад, *%10.4f* буде виводити число, загальне поле виведення якого *10* позицій, з них *4* позиції - дробова частина та *1* позиція - десяткова крапка. Коли формат, подібний цьому, застосовується до ланцюжків символів чи цілих, число, що слідує за точкою, специфікує максимальну довжину поля. Наприклад, *%5.7s* буде виводити ланцюжок символів довжиною не менше п'яти символів та не більше, ніж сім символів. Якщо ланцюжок більший, ніж максимальна ширина поля, то символи, що залишилися, будуть відсічені.

За умовчанням усе виведення вирівнюється праворуч: якщо ширина поля більше, ніж дані, що друкуються, то такі дані

розташовуються на правому краю цього поля. Можна викликати вирівнювання інформації ліворуч, розташовуючи знак мінус безпосередньо після *%*. Наприклад, *%-10.2f* буде вирівнювати ліворуч число з рухомою точкою з двома десятковими розрядами у десятипозиційному полі.

Деякі приклади керівних рядків для виведення даних:

Функція printf()	Виведення
("%9.2f", 123.321)	123.32
("%-9.2f", 123.321)	123.32
("%5.7s", "123456789")	1234567
("%9d", 555)	555
("%09d", 555)	000000555

Приклад. Скласти схему алгоритму та програму для обчислення функції:

*s = 2*r, якщо r = a + 2*b + ab, b = 7*a + 6*a*d – z.*

```
#include <stdio.h>
#include <math.h>
main()
float a,b,d,r,s,z;
printf("Запровадьте значення a,d,z:");
scanf("%f %f %f",&a,&d,&z);
b=7*a+6*a*d-z;
r=a+2*b+sqrt(a*b);
s=2*pow(r,fabs(a));
printf("Значення s= %f,b= %f,r= %f",s,b,r);.
```

Варіанти індивідуальних завдань до лабораторної роботи 3

1 рівень

1 $y = ax^2 + bx + c, \quad a = 0.24, \quad c = 1.47.$

2

$$z = \frac{(a+b)(a+c)}{(a+d+c)}, \quad a = 0.24, \quad c = -4, \quad d = -3.56$$

.

3 $s = \sqrt{\left|a + sin(a+b) - (a+b)^2\right|}, \quad a = 0.23.$

4

$$z = (1 - y)/(1 + y) + \sqrt{(1+y)(1+y^2)} + 2fy, \quad y = 11$$

.

5

$$s = (a+b)\sqrt{(a^2+f^2)} + \sqrt{(a+b^3)+2(a+b)}, \quad a = 0.22, \quad b$$

.

2 рівень

6

$$z = \frac{1-y}{1+y} + \sqrt{(1+y^5)(1+y)} + 2fy, \quad y = 11, \quad f = 2y + sin$$

.

7

$$s = (a+b)\sqrt{a^2+f^2} + \sqrt{(a+b^3)+2(a+b)}, \quad f = 2a + b, \quad b =$$

.

8

$$z = \frac{(a+b)(a+c)}{(a+d+c)}, \quad a = -9, \quad d = (a+b)cos(|a|), \quad c = -$$

.

9 $m = 2 - \frac{3x}{3+b} - \frac{y}{0.9-b}, \quad b = 9, \quad x = 3y + 2b.$

10 $s = 1 + x^2 + ax^3 + bx^4 - \sqrt{ab}, \quad x = a(a + |b|).$

3 рівень

11

$s = 2r^{-|a|}, \quad r = a + 2b + \sqrt{ab}, \quad b = 7a = sin(a) - z.$

12

$s = 1 + ax^2 + bx^3 + cx^4 - \sqrt{ab}, \quad x = a(a + |b|), \quad a = b^2 + c^{0.2}$

.

13

$z = 2(a + b)(a + c)(a + b + d)^2, \; a = -6, \quad c = -4, \quad d = (a +$

.

14

$y = \frac{x(x-1)(x-2)}{(x-a)(x-b)(x-c)} + 2cos(abx), \quad x = a + b - c, \quad c =$

.

15

$s = ab + x^3 + bx^4 - \sqrt{ab}, \quad x = a(a + |b|), \quad a = 3b + b^3.$

Робота 4
ПРОГРАМУВАННЯ РОЗГАЛУЖЕНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Мета роботи: вивчення операторів передачі управління. Придбання навичок складання програм розгалужених обчислювальних процесів і виконання їх на ПЕОМ.

Завдання та порядок виконання

1 Вивчити теоретичний матеріал.

2 Підготувати відповіді на контрольні запитання.

3 Скласти програми мовою С++ згідно зі складеними алгоритмами.

4 Ввести підготовлені програми у ПЕОМ, виконати їх та отримати результати розрахунків.

3 Контрольні запитання

1 Формат та логіка дії оператора *if*.

2 Який вираз в мові С++ є істинним?

3 Чи можуть в операторі *goto* використовуватися такі мітки: *10, m1, мітка10, _20m, 20met*?

4 Зміст звіту

1 Номер роботи, її назва, визначення мети.

2 Стислий зміст теоретичного матеріалу та відповіді на контрольні запитання.

3 Результати виконання завдання: схеми алгоритмів, програми, результати роботи програм.

4 Висновки до роботи.

5 Навчальний матеріал

1 Оператор умовної передачі управління *if*. Формат оператора if має вигляд

if (<умова>) <оператор>;
[else <оператор>];.

Якщо умова виконується («істина» або будь-який знак, відмінний від нуля), то комп'ютер виконає оператор чи блок операторів, наступний за оператором if. В іншому випадку, якщо існує частина *else*, виконується оператор чи блок операторів,

наступний за *else*.

У вигляді умовного виразу оператора *if* можна використовувати будь-який дозволений у С++ вираз, а не тільки вирази порівняння та логічні вирази. Вираз в операторі *if* аналізується та розрізняється нуль або ненульове значення.

Приклад. Програма виводить на екран повідомлення «Правильно», якщо вгадано задумане число та «Неправильно», якщо число не вгадано.

```
#include <stdio.h>
main()
{
  int  magic;
  int  guess;
  magic=5;    // задумане число
  printf (" Ваше число: ");
  scanf ( "%d", &guess );
  if (guess==magic)  printf ("  **Правильно**\n ");
  else  printf ("  **Неправильно**\n ");
}
```

Якщо при виконанні умови необхідно виконати групу операторів, то використовують блок

```
if (x) {

група операторів

}
else {

група операторів

}
```

Укладені оператори *if*. Укладений оператор *if* може бути

наступним або після частини *if*, або після частини *else*. Складність у тому, що важко зразу сказати, якому *if* відповідає який *else*.

Розглянемо, наприклад, такий фрагмент:

```
if (x)
if (y)  printf (" 1 ");
else  printf (" 2 ");.
```

На який *if* посилається *else*? Є просте правило: *else* пов'язане з ближчим *if*, що не має парного *else*. І *if*, і *else* повинні знаходитися всередині одного блока. У даному випадку *else* пов'язане з оператором *if(y)*. Для того щоб пов'язати *else* з оператором *if(x)*, необхідно використати фігурні дужки:

```
if (x) {
if (y)  printf (" 1 ");
}  else printf (" 2 ");.
```

2 Оператор безумовної передачі управління *goto*. Формат оператора *goto* має вигляд

goto <метка>;.

Для роботи оператора *goto* необхідна присутність мітки. Мітка - це повноправний ідентифікатор, за яким стоїть двокрапка. Мітка повинна знаходитися у тій самій функції, що і оператор *goto*.

Наприклад, можна організувати цикл, що працює 100 раз, за допомогою операторів *if* та *goto*:

```
        x=1;
loop:
        x++;
        if(x<100)  goto loop;.
```

Приклад. Скласти схему алгоритму та програму обчислення значення функції:

$$\begin{cases} a\,x + b, & \text{при } -2 < x <= 2, \\ \\ \end{cases}$$

y = k + x - a/b, при 2 < x < 4,
x a/2 + b, при x = 4 або x = 4.5,
x + ab, в інших випадках.

```
#include <stdio.h>
#include <math.h>
main()
{
float a,b,x,k,y;
printf("Ввести a,b,x,k:");
scanf("%f%f%f%f",&a,&b,&x,&k);
y=k+x-a/b
if(x>-2) {
if (x<=2) y=pow(a,2)*x+b;}
else if (x>2) {
if (x<4) y=k+x-(a/b);}
else if (x==4)
y=pow(x,3)*a/2+b;
else if (x==4.5)
y=pow(x,3)*a/2+b;
else y=x+a*b;
printf("y=%f\n",y);
}
```

Варіанти індивідуальних завдань до лабораторної роботи 4

1 рівень

1 $z=\begin{cases} sin(x+y), & якщо\ x>1 \quad або \quad y>1; \\ |x+y|, & якщо\ x<-1 \quad і \quad x>y; \\ e^{x-y}, & якщо\ 0\le x\le 1, \\ cos(x+1) & в\ інших\ випадках. \end{cases}$

x,y - довільні значення

2 $y=\begin{cases} |x^3|, & якщо\ x<ab\ \ i\ \ x<-2; \\ sin(x^a+b), & якщо\ x>ab\ \ або\ \ x>2; \\ a+b+1, & якщо\ 0<x<1\ \ i\ \ x=ab; \\ x-(a+b)^2 & в\ інших\ випадках. \end{cases}$

x,a,b - довільні значення

3 $f=\begin{cases} c+x, & якщо\ 1\le x\le 3; \\ c-x, & якщо\ 3<x\le 10 \quad i \quad c>0; \\ |c|^x, & якщо\ x>10; \\ x & в\ інших\ випадках. \end{cases}$

c,x - довільні значення

4 $w=\begin{cases} e^v, & якщо\ v>x \quad або \quad v>y; \\ (x+y), & якщо\ \ v=10\ \ i\ \ y>x>15; \\ v^{x+y}, & якщо\ \ v=x; \\ 1 & в\ інших\ випадках. \end{cases}$

v,x,y - довільні значення

5 $t=\begin{cases} s+2z, & якщо\ \ s+z=2\ \ або\ \ s+z=1; \\ s-z^2, & якщо\ \ s+z>5\ \ і\ \ z>0; \\ |s+z|, & якщо\ \ 3<s+z\le 5; \\ a+s & в\ інших\ випадках. \end{cases}$

a,s,z - довільні значення

2 рівень

6 $y=\begin{cases} a+b+c, & якщо\ \ a+b>c; \\ (a+b)-c, & якщо\ \ a+b<c\ \ і\ \ c>5; \\ |sin(b)|, & якщо\ \ a+b=c; \\ a^{bc} & в\ інших\ випадках. \end{cases}$ z=

$\begin{cases} 1, & якщо\ \ y<0; \\ 2, & якщо\ \ 0\le y<5; \\ 3, & якщо\ \ 5\le y<10; \\ 4, & якщо\ \ y\ge 10. \end{cases}$

a,b,c - довільні значення

7 $t=\begin{cases} z+x-sin(y), & якщо\ \ z+x>sin(y); \\ x-z\,sin(y), & якщо\ \ z+x=sin(y); \\ ln|xyz|, & якщо\ \ z+x<sin(y). \end{cases}$ $s=\begin{cases} t-1, & якщо\ \ 0<zt<5; \\ z, & якщо\ \ zt>10\ \ або\ \ zt<-3; \\ t, & якщо\ \ zt=0; \\ 0, & в\ інших\ випадках. \end{cases}$

y,z,x - довільні значення

8 m=$\begin{cases} \sin^2(x)+\cos(x), & якщо\ x>0; \\ \dfrac{x+y}{a^2+2}+\sin\left(\dfrac{x+y}{a^2+2}\right), & якщо\ x<0\ і\ a>0; \\ x+|a-y| & в\ інших\ випадках. \end{cases}$

n=$\begin{cases} |m+c|, & якщо\ m>10\ або\ m=4; \\ m-c, & якщо\ m<3; \\ m, & якщо\ 5<m<8; \\ mc & в\ інших\ випадках. \end{cases}$

x,y,a,c - довільні значення

9 x=$\begin{cases} t+q, & якщо\ t>10\ і\ t=q; \\ t+1, & якщо\ t\le -10; \\ t, & якщо\ t=5; \\ 1+tq & в\ інших\ випадках. \end{cases}$ y=

$\begin{cases} \sin(x+t), & якщо\ x<t; \\ \ln|x+t|, & якщо\ x=t; \\ e^{xt}, & якщо\ x>t. \end{cases}$

t,q - довільні значення

10 k=$\begin{cases} x+y^2+2, & якщо\ x=y+2; \\ x+y+2, & якщо\ x>y+2\ і\ y=3; \\ \sin(x)+\sin(2), & якщо\ 0<x<y+2; \\ x+\ln|y+2y| & в\ інших\ випадках. \end{cases}$ z=

$$\begin{cases} 2k, & якщо\ \ k<2; \\ 3k, & якщо\ \ 2\le k\le 6; \\ 4k, & якщо\ \ 6<k<10; \\ 5k, & якщо\ \ k\ge 10. \end{cases}$$

x,y - довільні значення

3 рівень

11 a= $\begin{cases} x\sin(y), & якщо\ \ x>y; \\ y\sin(x), & якщо\ \ x\le y. \end{cases}$ l=

$$\begin{cases} ln(a^2+1), & якщо\ \ 15\le a<20; \\ 2a+7, & якщо\ \ 10\le a<15; \\ 1-ab, & якщо\ \ 8<a<10; \\ \frac{4a}{7(a+b)^2+5}, & якщо\ 0\le a<8\ \ і\ \ a>b; \\ a+b & в\ інших\ випадках. \end{cases}$$

p= $\begin{cases} t+a, & якщо\ \ t>0\ \ або\ \ a>0; \\ t-a, & якщо\ \ t<0\ \ і\ \ t<0; \\ |a| & в\ інших\ випадках, \end{cases}$ b=$ln|a+xy|$, $t=3l-1$.

x,y - довільні значення

12 $a=b\cos(b)$ z=

$$\begin{cases} 25y+a, & якщо\ \ y>a\ \ і\ \ a>10; \\ y+3a, & якщо\ \ a<y<20; \\ \cos(ya), & якщо\ \ y=a; \\ 1 & в\ інших\ випадках. \end{cases}$$

v= $\begin{cases} z^5+z+1, & якщо\ \ 1<z^5<3; \\ \cos(z^5+z+1), & якщо\ \ 3\le z^5<7; \\ ln(z^5+z+1), & якщо\ \ 7\le z^5<10; \\ \frac{1}{z^4+z^2+1}, & якщо\ \ z^5\ge 10\ \ або\ \ z^5\le 1. \end{cases}$ w= $\begin{cases} v+1, & якщо\ \ v+1\le 0; \\ \frac{1}{v^2+1}, & якщо\ \ v+1>0. \end{cases}$

y,b - довільні значення

13 $x=\begin{cases} t^3+5, & якщо\ t<-2; \\ t-1, & якщо\ -2\le t<0; \\ sin(t), & якщо\ 0\le t<5; \\ ln(t), & якщо\ t\ge 5. \end{cases}$ $z=\begin{cases} e^x, & якщо\ x<0; \\ x^3, & якщо\ x\ge 0. \end{cases}$

$y=\begin{cases} a+z, & якщо\ a+z>0\ \ i\ \ z<0; \\ az, & якщо\ z>5\ \ або\ \ a+z<-3; \\ a-x & в\ інших\ випадках. \end{cases}$

$f=\begin{cases} \dfrac{1}{(ay)^2+1}, & якщо\ y=-4; \\ 2xy, & якщо\ xy>0\ \ i\ \ x>0; \\ \dfrac{y}{x^4+1} & в\ інших\ випадках. \end{cases}$

a,t - довільні значення

14 $b=t^3+t-1$ $c=\begin{cases} 1, & якщо\ b<0; \\ 2, & якщо\ b\ge 0. \end{cases}$ $d=\begin{cases} 1, & якщо\ a\le 1 \\ 3, & якщо\ 1<a\le 3; \\ 5, & якщо\ >3 \end{cases}$.

$a=\begin{cases} b^3, & якщо\ b>0\ \ i\ \ c>0\ \ або\ \ b>10 \\ b+c, & якщо\ b<0\ \ i\ \ c=0; \\ \dfrac{1}{(bc)^2+5}, & якщо\ b=0\ \ або\ \ -5<c<-1; \\ \dfrac{1}{(b+1)^2+3} & в\ інших\ випадках. \end{cases}$

$f=\begin{cases} abd, & якщо\ ab<d+1; \\ \dfrac{ab}{d^4+1}, & якщо\ ab=d+1; \\ \dfrac{ad}{b^2+2}, & якщо\ ab>d+1. \end{cases}$

t - довільні значення

15 $x=\begin{cases} t+1, & якщо\ t<1; \\ t, & якщо\ t=1; \\ t-1, & якщо\ t>1. \end{cases}$ $y=\begin{cases} a^3+x, & якщо\ a<x<10\ \ i\ \ a<10; \\ a+sin(x), & якщо\ 10<x\le 15\ \ i\ \ x=a; \\ ln|xa|, & якщо\ x=20\ \ або\ \ 25<a\le 30; \\ e^{x-a} & в\ інших\ випадках. \end{cases}$.

$p=\begin{cases} 1+z, & якщо\ z=0\ \ i\ \ y=0; \\ z+y, & якщо\ zy<0\ \ ; \\ 25 & в\ інших\ випадках. \end{cases}$ $z=a^2+y^3-5$.

t,a - довільні значення.

Лабораторна робота 5
ПРОГРАМУВАННЯ ЦИКЛІЧНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Мета роботи: вивчення операторів організації циклічних обчислювальних процесів. Придбання навиків складання програм з циклічними обчислювальними процесами та виконання їх на ПЕОМ.

Завдання та порядок виконання

1 Вивчити теоретичний матеріал.

2 Підготувати відповіді на контрольні запитання.

3 Написати програми мовою С++ згідно зі складеними алгоритмами.

4 Ввести підготовлені програми у ПЕОМ, виконати та отримати результати розрахунків.

Контрольні запитання

1 Які бувають цикли? У чому їх різниця?
2 Які оператори призначені для організації циклів у мові С++?
3 Формат запису циклу *for*.
4 Засоби завдання приросту в циклі *for*.
5 Формат запису циклу *while*.
6 Формат запису циклу *do-while*.
7 З якою метою використовується оператор переривання циклу?
8 З якою метою використовується оператор продовження циклу?

Зміст звіту

1 Номер роботи, її назва, визначення мети.
2 Стислий зміст теоретичного матеріалу та відповіді на контрольні запитання.
3 Результати виконання завдання: схеми алгоритмів, програми, результати роботи програм.
4 Висновки до роботи.

Навчальний матеріал

Цикли дозволяють багаторазово повторювати деякий набір операцій до тих пір, поки не виконається деяка умова. У мові С++ розрізняють цикли *for, while* та *do-while*.

1 Цикл *for*. Формат циклу *for*:

for(<ініціалізація>; <умова>; <крок>) <оператор>;

де *ініціалізація* - це оператор присвоювання початкового значення керівній змінній циклу;
умова - вираз порівняння, що перевіряє досягнення керівною змінною циклу кінцевого значення;
крок - визначає приріст, на який зміниться керівна змінна при кожному проходженні циклу.

У вигляді прикладу розглянемо програму, яка виводить на екран цифри від 1 до 100:

```
#include <stdio.h>
main ()
{
 int  x;
 for(x=1; x<=100; x++) printf ("%d", x);
}
```

У циклі *for* можна вказати від'ємний приріст для керівної змінної:

```
#include <stdio.h>
main ()
{
int x;
for (x=100; x>0; x--) printf ("%d", x);
}
```

Використання оператора приросту керівної змінної не є обов'язковим. Замість нього може стояти будь-який допустимий оператор присвоювання. Наприклад, такий цикл буде виводити на екран числа від *0* до *100* з кроком *5*:

```
#include <stdio.h>
main ()
{
int  x;
for (x=0; x<=100; x=x+5)  printf ("%d", x);
}
```

Організувавши блок, можна за допомогою циклу *for* багаторазово виконати цілу групу операторів:

```
#include <stdio.h>
main ()
{
int i;
```

```
for (i=0; i<100; i++)  {
                          група операторів;
                       }
  }
```

Важливою особливістю циклу *for* є той факт, що на початку циклу завжди перевіряється умова. Якщо умова не виконується, то не виконуються і оператори всередині циклу.

Іншою особливістю є те, що цикл *for* не вимагає визначення всіх частин. У наступному прикладі цикл буде працювати до тих пір, поки не буде введено число 123:

```
for (x=0; x!=123;)  scanf ("%d", &x);.
```

При введенні з клавіатури числа 123 умовна частина дає «хибно» і цикл завершується.

Укладені цикли *for*. Укладений цикл *for* може знаходитися безпосередньо після умовою зовнішнього циклу:

```
for(i=0; i<10; i++)
    for(j=0; j<8; j++)
       {
        оператори
       },
```

або знаходитися всередині блока

```
for(i=0; i<10; i++)
     {
       оператори
       for(j=0; j<8; j++)
            {
             оператори
             }
     }
```

2 Цикл *while*. Формат циклу *while*:

while(<умова>) <оператор>;,

де *умова* - будь-який допустимий вираз;
оператор - один оператор чи група операторів.

Цикл виконується до тих пір, поки умова істинна. При хибності умови управління передається на оператор, наступний за оператором циклу.

У циклі *while* умова перевіряється на початку циклу. Тобто тіло циклу може взагалі не виконатися.

Приклад. Програма виводить на екран числа від 0 до 4.

```
#include <stdio.h>
main()
{
int i=0;
while(i < 5)  printf("%d\n", i++);
}
```

3 Цикл *do-while*. На відміну від циклів *for* та *while*, у циклі *do-while* перевірка умови виконується в кінці циклу. Тому цикл *do-while* завжди виконається хоча б один раз.

Загальний формат цього циклу має вигляд

```
do  {
      <оператор>;
} while(<умова>);
```

Приклад використання циклу *do-while* для читання з клавіатури різноманітних чисел до тих пір, поки одне з них не стане меншим за 100:

```
#include <stdio.h>
main()
{
int num;
do {
scanf ("%d", &num);
} while(num>=100);
```

```
}
```

4 Оператор переривання циклу *break*. Для вимушеного переривання циклу під час його нормального виконання використовують оператор break. При зустрічі оператора break всередині циклу комп'ютер припиняє виконання циклу, і управління передається на оператор, що сходитися після даного циклу.

Наприклад:

```
#include <stdio.h>
main()
{
int  t;
for(t=0; t<100; t++ )
        { printf("%d", t);
          if(t==10)  break;
        }
}
```

Break здійснює вихід тільки з внутрішнього циклу.

5 Оператор *continue*. За дією оператор *continue* подібний оператору *break*, однак він не завершує цикл, а дає команду на наступну ітерацію. При цьому оператори тіла циклу, що залишилися, не виконуються. Наприклад, така програма виводить на екран тільки парні числа:

```
#include <stdio.h>
main ()
{
int  x;
for (x=0; x<100; x++)
      {
       if (x%2)  continue;
       printf ("%d", x);
       }
}
```

Тут, при появі непарного числа виконується частина if, бо

остача від ділення непарного числа на 2 завжди дорівнює 1 або “істинна”.

Приклад. Скласти схему алгоритму та програму обчислення значень функції:

*z = cos(a*y + 1), при y = b + d – e^{-x} , a = 2, b = 2, x ∈ [-3; b], h = 0,8.*

```
#include <stdio.h>
#include <math.h>
#include <conio.h> /* Улеопюіжссся блбплт-
цжок, сжтбщлестм епя вкотфкхцасся шчсоълм clrscr() */
main()
{
float a,b,d,x,y,z;
clrscr();  // Шчсоъля тікєжссся жофасч
printf("Йауфтваеїцж d: ");
scanf("%f",&d);
a=2;   b=2;
       for (x=-3; x<=b; x=x+0.8)
       {  y=b+d-exp(-x);
        z=cos(a*y+1);
         printf("z=%f \n",z);
        }
}
```

Варіанти індивідуальних завдань до лабораторної роботи 5

1 рівень

1 $z = cos(x), \quad x = 24e^{t}, \quad t \in [3;9], \quad h_t = 1.5$

2 $m = 2.49x^3 - \frac{b}{x^2 + s^2}, \quad s = 3.63\sin(a + x), \quad x \in [-1; a/$

3

$$t = 2m - \sin^2(x) + \frac{3}{2x}; \quad m = 36x^2; \quad x \in [0.2; 1.6]; \quad h_x$$

4

$$p = 3b^2 - \frac{\ln|a - c|}{a + 2b}; \quad b = 0.75a^2 + c; \quad a \in [-3; 3]; \quad h_a = 0.5;$$

5

$$z = \cos(ay^2 + 1); \quad y = b + d^2 - e^{-x}; \quad x \in [-3; b]; \quad b = 2;$$

2 рівень

6 $$n = 3; \quad e = \sum_{i=1}^{n} i^2; \quad f = (e - n)!.$$

7 $$n = 5; \quad a = (10 - n)!; \quad t = \sum_{k=1}^{20} \sin(a + k).$$

8

$$k = n!; \quad n = 2; \quad z = \sum_{i=1}^{k} (a + b)^i; \quad p = \prod_{i=1}^{k} (a + b)^i; \quad a = 2;$$

9 $$c = \frac{m!}{n!(m - n)!}; \quad m = \sum_{i=1}^{5} i; \quad n = \prod_{j=1}^{3} j.$$

10

$$r = \cos\left(\sum_{i=1}^{n} \ln(i) + \prod_{j=1}^{m} \sin(j^3)\right); \quad f = 3; \quad n = f!; \quad m = (n - 2$$

3 рівень

11

$$y = b^2 \sin(\cos(x + m - n)); \quad x \in [100; 200]; \quad h_x = b^3 +$$

$$m = \prod_{k=p}^{r}(a-b)^k; \quad a=4; \quad b=2; \quad t1=3; \quad t2=5; \quad p=$$

12

$$z = ax + by + c; \qquad y = x^3 + x^2 + 1; \qquad x = 2; \qquad a = (x+5)!;$$

$$b = \sum_{i=1}^{n} i^3 + \frac{1}{y!}; \quad c = \prod_{j=3}^{5}(a+b)^j.$$

13

$$t = x! + y! + n; \qquad x = \sum_{i=1}^{4} i; \qquad y = \prod_{i=1}^{3} i; \qquad n = \sum_{k=1}^{5}(k-1).$$

14 $z = a\cos(a+b); \qquad a \in \lfloor a1; a2 \rfloor; \qquad h_a = 50;$

$$a1 = 5!; \quad a2 = (n+5)!; \quad b = \sum_{i=1}^{a2-a1} i; \quad n = 1.$$

15

$$f = \frac{\sum_{i=1}^{n}(a+b)^i + \prod_{j=j1}^{j2}(ab)^j}{(n+4)!} - \sum_{k=1}^{3}(a+b)^k \prod_{j=4}^{6} b^j; \qquad n = 4;$$

$$a = 0.3; \quad b = 1.7; \quad j1 = 3; \quad j2 = 6$$

Лабораторна робота 6
ПРОГРАМУВАННЯ ОБРОБКИ ОДНОВИМІРНИХ МАСИВІВ

Мета роботи: вивчення методики обробки одномірних масивів. Придбання навиків складання програм для обробки одномірних масивів та виконання їх на ПЕОМ.

Завдання та порядок виконання

1 Вивчити теоретичний матеріал.

2 Підготувати відповіді на контрольні запитання.

3 Написати програми мооюі С++ згідно зі складеними алгоритмами.

4 Ввести підготовлені програми у ПЕОМ, виконати та отримати результати розрахунків.

Контрольні запитання

1 Визначити поняття «розмір масиву».

2 Визначити поняття «розмірність масиву».

3 Формат оголошення одновимірного масиву.

4 Функція введення рядка з клавіатури. Особливості її використання.

5 Ініціалізація одновимірних масивів.

Зміст звіту

1 Номер роботи, її назва, визначення мети.

2 Стислий зміст теоретичного матеріалу та відповіді на контрольні запитання.

3 Результати виконання завдання: схеми алгоритмів, програми, результати роботи програм.

4 Висновки до роботи.

Навчальний матеріал

1. Формат оголошення одновимірного масиву:

<тип> <ім'я_масиву> [<розмір>];,

де *тип* - визначає базовий тип масиву. Базовий тип визначає тип кожного елемента масиву;

розмір – кількість елементів, що може містити масив.

Наприклад, оголошення масиву цілого типу з ім'ям *sample*, що містить 10 елементів:

int sample [10];.

Перший елемент масиву отримує індекс *0*. Таким чином, оголошено масив з десятьма елементами: від *sample[0]* до sample[9].

У наступному прикладі організується масив цілого типу, елементам якого присвоюються значення від *0* до *9*:

```
main()
{
int  x[10];
int  t;
for(t=0; t<10; t++)  x[t]=t;
}
```

У мові С++ не здійснюється перевірка меж масиву - ніщо не контролює індекс при виході за межі значення в масиві. Наприклад, якщо перевищення значення індексу масиву трапиться під час виконання оператора присвоювання, то зайві значення можуть присвоїтися іншим змінним, що розташовані в пам'яті ПЕОМ після поля елементів масиву.

2 Рядки. Найчастіше одновимірні масиви використовуються для створення символьних рядків. Рядок складається з масиву символів, що закінчуються символом *'\0'*. У зв'язку з цим символьний масив повинен містити на один елемент більше, ніж містить символів рядок, який записується в нього.

Незважаючи на те, що в мові С++ відсутній тип даних «символьний рядок», мова дозволяє записувати символьні константи. Згадаємо, що символьна константа - це набір символів, що укладено в подвійні лапки.

3 Зчитування рядка з клавіатури. Найкращим способом введення рядка з клавіатури є використання бібліотечної функції *gets()*. Формат її такий:

gets(<ім'я_масиву>);

Для зчитування рядка необхідно викликати функцію *gets()* з неіндексованим ім'ям масиву як аргумент. Функція *gets()* буде продовжувати зчитувати символи до тих пір, поки не буде натиснута клавіша *Enter*.

Приклад програми, що виводить на екран введенний з клавіатури рядок.

```
// Приклад запровадження та друку на екрані рядка

#include <stdio.h>
main()
{
char str[80];
gets (str);
printf ("%s", str);
}
```

У вигляді аргументу функції *printf()* використовується ім'я масиву *str*. Це ім'я не є індексованим. Неіндексоване ім'я масиву можна розміщати там же, де і строковую константу.

4 Виведення рядка на екран. Для виведення рядка на екран використовується бібліотечна функція *puts()*. Формат її такий:

puts(<ім'я_масива>);.

Для виведення рядка необхідно викликати функцію *puts()* з неіндексованим ім'ям масиву як аргумент точно так, як функцію *gets()*.

Звернення до функції *puts()* вимагає значно менше машинних витрат, ніж таке саме звернення до функції *printf()*, тому що функція *puts()* може тільки виводити ланцюжок символів, вона не може виводити числа чи виконувати перетворення форматів.

5 Ініціалізація одновимірних масивів. Загальний формат ініціалізації масиву:

<тип> <ім'я_масива> [<розмір>] = { <список_значень> };,

де *список_значень* - це список розділених комами констант, сумісних за типом з базовим типом масиву.

Оператор ініціалізації розмістить першу константу в перший елемент масиву, другу константу - у другий елемент і т. д. У наступному прикладі ініціалізується 10-ти елементний масив цілих чисел із значеннями від 1 до 10:

int i[10] = {1,2,3,4,5,6,7,8,9,10};.

Тут елемент *i[0]* буде мати значення *1*, елемент *i[9]* - значення *10*.

Символьні масиви, що містять рядки, допускають записувати скорочений формат ініціалізації у вигляді:

char <ім'я_масива> [<розмір>] = "<рядок>";.

Наприклад, у наступному фрагменті масив *str* ініціалізується рядком *"hello"*:

char str[6] = "hello";.

Цей запис еквівалентний запису:
char str[6] = {'h','e','l','l','o','\0'};.

Оскільки всі рядки у С++ повинні закінчуватися нулем, масив повинен містити місце і для нього.

6 Ініціалізація безрозмірних символьних масивів. Підраховувати вручну кількість символів у кожному повідомленні для завдання розміру масиву дуже трудомістко. Однак можна змусити С++ автоматично встановити розмір масива за допомогою ініціалізації безрозмірних масивів. Для цього в операторі ініціалізації не слід зазначати розмір масиву, і С++ автоматично створить масив, що зможе містити присутній ініціалізатор. Наприклад:

char e[] = "cannot open file\n";.

Приклад. Обчислити середнє арифметичне значення 10-ти елементів масиву $A=\{A_i\}$, $i=0,...,9$.

```
#include <stdio.h>
#include <conio.h>
main()
{
int a[10];
int i,summa;
float sr;
clrscr();
for(i=0; i<10; i++) {
printf("Запровадьте a[%d]: ",i);
scanf("%d",&a[i]);
}
summa=0;
for (i=0; i<10; i++) summa=summa+a[i];
sr=(float)summa/10;
```

```
printf("Середнє арифметичне = %f",sr);
}
```

Приклад. Обчислити середнє арифметичне значення парних елементів 1 < *Ai* <*15* масиву *A={Ai},* що містить 20 елементів.

```
#include <stdio.h>
#include <conio.h>
main()
{
int a[20];
int i,summa,kol;
float sr;
clrscr();
for(i=0; i<20; i++) {
        printf("Введіть a[%d]: ",i);
        scanf("%d",&a[i]);
        }
summa=0;  kol=0;
        for (i=0; i<20; i=i+2) {
        if(a[i]>1 && a[i]<15) {
        summa=summa+a[i];
        kol=kol+1; }
        }
sr=(float)summa/kol;
printf("Середнє арифметичне = %f",sr);
}
```

Варіанти індивідуальних завдань до лабораторної роботи 6

Скласти схему алгоритму обробки одновимірного масиву відповідно до варіанта завдання.

1 рівень

Масив М містить К елементів М(i), i = 1,К.

1 Визначити кількість від'ємних елементів і обчислити їхнє середнє арифметичне значення.

2 Обчислити суму елементів з непарними порядковими номерами.

3 Обчислити добуток елементів з парними порядковими номерами.

4 Обчислити середнє арифметичне елементів М(i) > 1 і М(i) < 15.

5 Обчислити загальну кількість від'ємних елементів і М(i) > 20.

2 рівень

Масив А містить 30 елементів А(i), i = 1,30. Скласти схему алгоритму формування масиву Т з елементами Т(i).

6 $$T(i) = A(i) + \frac{C}{D}, \qquad \text{де} \quad C = \sum_{i=1}^{30} A(i), \quad D = \prod_{i=1}^{10} A(i).$$

Визначити середнє арифметичне позитивних елементів масиву.

7 $$T(i) = \begin{cases} \sum_{i=1}^{30} A(i) + A(i)/2, & \text{якщо } A(i) < 0; \\ \prod_{i=1}^{15} A(i) - 2A(i), & \text{якщо } A(i) > 0; \\ 0 & \text{в інших випадках.} \end{cases}$$

Визначити кількість нульових елементів у масиві А.

8 $$T(i) = \begin{cases} A(i) + \sum_{i=1}^{13} A(i), & \text{якщо } 0 < A(i) < 5; \\ 2A(i) + \prod_{i=1}^{15} A(i), & \text{якщо } 5 < A(i) < 10; \\ 0 & \text{в інших випадках.} \end{cases}$$

Визначити суму від'ємних елементів з парними індексами.

9

$$T(i)=\begin{cases} A(i)+\sum_{i=2}^{39} A(i), & \text{якщо } 0<A(i)<10; \\ 2+\prod_{i=5}^{26} A(i), & \text{якщо } 10<A(i)<15; \\ 0 & \text{в інших випадках.} \end{cases}$$

Визначити добуток всіх елементів A(i) > 50.

10

$$T(i)=\begin{cases} A(i)-\sum_{i=1}^{39} A(i), & \text{якщо } -5<A(i)<5 \quad \text{і} \quad A(i)= \\ A(i)+\prod_{i=7}^{13} A(i), & \text{якщо } 10<A(i)<15 \quad \text{або} \quad A(i) \\ 0 & \text{в інших випадках.} \end{cases}$$

Визначити середнє арифметичне елементів з парними індексами.

3 рівень

Дано масиви K і M з елементами K(i) і M (i) відповідно. i= 1,20.
Скласти схему алгоритму формування масиву T з елементами T(i).

11 $$T(i)=K(i)+\frac{\sum_{i=1}^{15} K(i)+\sum_{i=4}^{12} M(i)}{\prod_{i=3}^{11} K(i)+\prod_{i=2}^{10} M(i)}.$$

Визначити різницю між добутком і середнім арифметичним додатних елементів.

12 $$T(i)=\frac{K(i)}{M(i)}+\frac{\sum_{i=5}^{12} K(i)+\prod_{i=1}^{8} M(i)}{\sum_{i=8}^{16} (K(i)*M(i))}$$

Визначити значення першого максимального елемента і його порядковий номер.

13

$$T(i)=\begin{cases} K(i)+\sum_{i=1}^{19} M(i), & \text{якщо } 0<K(i)<5 \quad \text{і} \quad M(i)< \\ K(i)+\prod_{i=1}^{11} M(i), & \text{якщо } K(i)>0 \quad \text{і} \quad M(i)>0; \\ 0 & \text{в інших випадках.} \end{cases}$$

Визначити кількість і суму всіх елементів K(i) > 0.

14

$$T(i) = \begin{cases} M(i) + \sum_{i=1}^{26} K(i), & \text{якщо } 0 < M(i) =< 10; \\ M(i) - \prod_{i=7}^{27} M(i), & \text{якщо } 10 < M(i) < 15 \text{ або } K(i \\ 0 & \text{в інших випадках.} \end{cases}$$

Визначити різницю між максимальним і мінімальним елементами.

15

$$T(i) = \begin{cases} M(i) + \sum_{i=5}^{33} (K(i) * M(i)), & \text{якщо } -5 < K(i) \\ M(i) + \prod_{i=3}^{12} (K(i) + M(i)), & \text{якщо } 5 < K(i) < 1 \\ 0 & \text{в інших випадках.} \end{cases}$$

Визначити різницю між номерами максимального і мінімального елементів.

Лабораторна робота 7
ПРОГРАМУВАННЯ ОБРОБКИ ДВОВИМІРНИХ МАСИВІВ

Мета роботи: вивчення методики обробки двовимірних масивів. Придбання навиків складання програм обробки двовимірних масивів і виконання їх на ПЕОМ.

Завдання та порядок виконання

1 Вивчити теоретичний матеріал.

2 Підготувати відповіді на контрольні запитання.

3 Написати програми мовою С++ згідно зі складеними алгоритмами.

4 Ввести підготовлені програми у ПЕОМ, виконати їх та отримати результати розрахунків.

Контрольні запитання

1 Як визначити кількість елементів двовимірного масиву?

2 Які циклічні обчислювальні процеси застосовуються при обробці двовимірних масивів?

3 Як у циклі *for* задати обробку елементів масиву, розміщених у столпчиках з непарними номерами?

4 Як виконується ініціалізація двовимірних масивів?

Зміст звіту

1 Номер роботи, її назва, визначення мети.

2 Стислий зміст теоретичного матеріалу та відповіді на контрольні запитання.

3 Результати виконання завдання: схеми алгоритмів, програми, результати роботи програм.

4 Висновки до роботи.

Навчальний матеріал

1 Мова С++ дозволяє оголошувати багатовимірні масиви. Найпростішим з багатовимірних масивів є двовимірний масив. Наприклад, для оголошення двовимірного масиву цілих чисел з ім'ям *dvmas* та розмірами *10*20* необхідно записати:

int dvmas[10][20];.

У наступній програмі організується двовимірний масив із значеннями елементів від *1* до *12*.

```
main()
{
int i, j, num[3][4];
for (i=0; i<3; i++)
        for (j=0; j<4; j++)
                num[i][j] = (i*4)+j+1;
}
```

Тут елемент *num[0][0]* буде мати значення *1*, елемент *num[0] [1]* - значення *2*, елемент *num[0][2]* - значення *3* і т. д.

2 Ініціалізація двовимірних масивів. Двовимірні масиви ініціалізуються так само, як і одновимірні. Наприклад, так можна проініціалізувати двовимірний масив *sqrs* числами від *1* до *10* та їх

квадратами:

```
int  sqrs[10][2] = {
                    1, 1,
                    2, 4,
                    3, 9,
                    4, 16,
                    5, 25,
                    6, 36,
                    7, 49,
                    8, 64,
                    9, 81,
                    10, 100
                    };
```

3 Ініціалізація безрозмірних числових масивів. Ініціалізація безрозмірних масивів не обмежується тільки одновимірними масивами. Однак для багатовимірних масивів необхідно зазначати всі індекси вимірів, крім крайнього лівого. У вигляді прикладу розглянемо оголошення двовимірного масиву *sqrs* як безрозмірного:

```
int  sqrs[][2] = {
                  1, 1,
                  2, 4,
                  3, 9,
                  4, 16,
                  5, 25,
                  6, 36,
                  7, 49,
                  8, 64,
                  9, 81,
                  10, 100
                  };
```

Перевагою є те, що можна подовжити або скоротити таблицю, не змінюючи індекс по першому виміру.

Приклад. Задано масив *A* з елементами a_{ij}, $i=0,...,4$, $j=0,...,5$. Визначити середнє арифметичне значення додатних елементів, розміщених у стовпцях з парними номерами.

```
#include <stdio.h>
#include <conio.h>

main()
{
int a[5][6];
int i,j,summa,kol;
float sr;
clrscr();
for(i=0; i<5; i++)
for(j=0; j<6; j++) {
                    printf("Ввести a[%d][%d]:",i,j);
                    scanf("%d",&a[i][j]);
                    }
summa=0;  kol=0;
for(i=0; i<5; i++)
for(j=0; j<6; j=j+2) {
                    if(a[i][j]>0) {
                          summa=summa+a[i][j];
                          kol=kol+1; }
                     }
sr=(float)summa/kol;
printf("Середнє арифметичне = %f",sr);
}
```

Приклад. Задано масив *A* з елементами a_{ij}, $i=0,...,3$, $j=0,...,4$. Визначити різницю між кількістю додатних і від'ємних елементів.

```
#include <stdio.h>
#include <conio.h>
main()
{
int a[4][5];
```

```
int i,j,kol_pol,kol_ot,raznost;
clrscr();
// Введення масиву
for(i=0; i<4; i++)
for(j=0; j<5; j++) {
                    printf("Запровадити a[%d][%d]:",i,j);
                    scanf("%d",&a[i][j]);
                   }
// Визначення кількості додатніх та від'ємних елементів
kol_pol=0;  kol_ot=0;
for(i=0; i<4; i++)
for(j=0; j<5; j++) {
                   if(a[i][j]>0)
                         kol_pol=kol_pol+1;
                   if(a[i][j]<0)
                         kol_ot=kol_ot+1;
                   }
raznost=kol_pol-kol_ot;
printf("raznost=%d",raznost);
}
```

Варіанти індивідуальних завдань до лабораторної роботи 7

1 рівень

Дано масив A з елементами A(i,j), i=1,5; j=1,10. Скласти схему алгоритму визначення:

1) середнього арифметичного додатних елементів;
2) добутку від´ємних елементів;
3) кількості елементів, що >1 і <15, тобто 1<A(I,J)<15;
4) різниці між добутком всіх елементів і сумою від´ємних елементів;
5) середнього арифметичного від´ємних елементів масиву.

2 рівень

Дано масив А з елементами А(i,j), i=1,10; j=1,15. Скласти схему алгоритму визначення:

1) суми від´ємних елементів у парних рядках масиву;
2) добутку додатних елементів у непарних стовпцях масиву;
3) різниці між кількістю від´ємних і додатних елементів у парних рядках масиву;
4) середнього арифметичного від´ємних елементів у непарних стовпцях;
5) різниці між сумою і добутком елементів >10 у парних рядках масиву.

3 рівень

Дано масив В з елементами В(i,j), i=1,N,j=1,M. Скласти схему алгоритму для визначення:

1) добутку від´ємних елементів у кожному непарному рядку масиву;
2) суми додатних елементів у кожному непарному стовпці масиву;
3) середнього арифметичного від´ємних елементів у кожному парному рядку масиву;
4) різниці між середніми арифметичними додатних і від´ємних елементів у кожному непарному стовпці масиву;
5) різниці між сумою елементів >10 у кожному парному рядку і сумою елементів <10 у кожному непарному стовпці масиву.