МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний автомобільно-дорожній університет

Симбірський Г.Д.

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни “Інформаційні технології”
за напрямом підготовки 6.050702 “Електромеханіка”
(Розділ “Мова програмування *C*++”)

Харьков – 2014

Зміст

Вступ до курсу

Алгоритмічна мова високого рівня С++ є однією з основних мов програмування та ще називається мовою об'єктно-орієнтованого програмування. Чому С++ одержав цю другу назву ми розглянемо у подальших лекціях. Ми будемо вивчати мову С++ у складі програмного пакету Microsoft Visual Studio 6.0 - Microsoft Visual C++ 6.0. Мова С++ дозволяє вирішувати безліч задач – від простих шкільних завдань до надскладних задач ядерної фізики та метеодосліджень. Також ця мова програмування застосовується при розробці величезних баз даних, наприклад, баз виборців чи громадян цілих країн.

Лекція 1

Алгоритмізація задач

Мета лекції. Вивчити основні конструкції схем алгоритмів

Основні питання лекції.

1. Основні поняття і визначення.
2. Прості дії та їх базові конструкції.
3. Складні дії та їх базові конструкції.
4. Види алгоритмів.

1. Основні поняття і визначення

Алгоритм розв'язання задачі – це послідовність дій, які необхідно виконати над даними, щоб одержати результат. Описати (вказати) дії, які необхідно виконати, можна по різному. Наприклад, для наступної задачі перелік дій і послідовність їх виконання можна сформулювати словами:

1. Ввести в ЕОМ початкове значення *t.*
2. Обчислити значення проміжної змінної $c = c = ln(1 + t)$.
3. Обчислити значення проміжної змінної $a = 1 + |Cos(t)|$.
4. Обчислити значення імовірності *Q* передачі команди у мережі.
5. Вивести розраховану імовірність *Q* на екран дисплею або на друкарський пристрій.

Цей алгоритм зрозуміє людина, яка володіє українською мовою. Для того, щоб алгоритм зрозуміла кожна людина його треба представити у якійсь іншій формі. Для цього є загальноприйняті стандартні форми опису (представлення) алгоритму. Однією з таких форм представлення алгоритму є зображення алгоритму у вигляді схеми (ГОСТ 19.002-90, 19.003-90).

Розглянемо, як у вигляді схеми можна представити алгоритм.

Кількість дій та порядок їх виконання залежить від характеру та складності задачі. Однак, при розробці алгоритму використовується невелика і обмежена кількість *типових дій*.

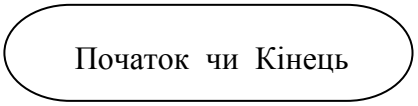
Типові дії в алгоритмі представляються у вигляді означених конструкцій, котрі будемо називати *базовими*.

Кожна базова конструкція має строго визначену структуру (зображення), один вхід і один вихід. У схемах алгоритмів базові конструкції з'єднуються друг з другом послідовно лініями потоку. У схемах алгоритмів лінії потоку, які направлені з верху вниз та зліва направо стрілками не помічаються. Лінії потоку, які направлені з низу в гору та справа уліво обов'язково повинні закінчуватися стрілками.

Базові конструкції розділяються на прості і складні.

2. Прості дії та їх базові конструкції

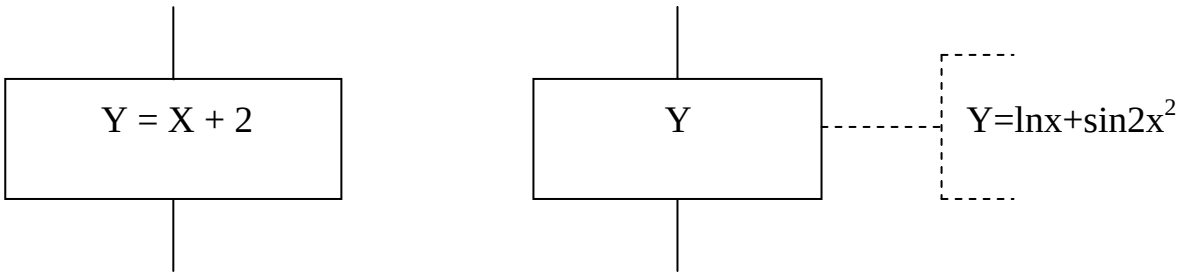
Початок (кінець) алгоритму зображується еліпсом:



Проста дія – це одна операція. До простих дій відносяться:

- присвоювання;
- введення;
- виведення.

Присвоювання – дія, в результаті якої змінна отримує початкове або нове значення. Функціональний блок присвоювання в схемах алгоритмів позначається символом *процес* (прямокутник).

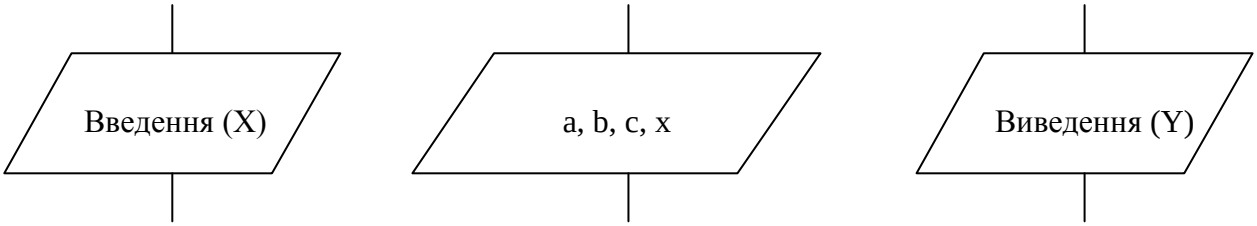


У середині символу записується команда або ім'я змінної. Якщо математичний вираз має складний вигляд, тоді функціональний блок має коментар.

Введення – присвоювання змінній початкового значення.

Виведення – це виведення даних для відображення або зберігання.

Функціональний блок введення чи виведення в схемах алгоритмів позначається символом *дані* (паралелограм). У середині символу записується слово Введення чи Виведення і в круглих дужках ім'я змінної або просто перелічуються імена змінних.



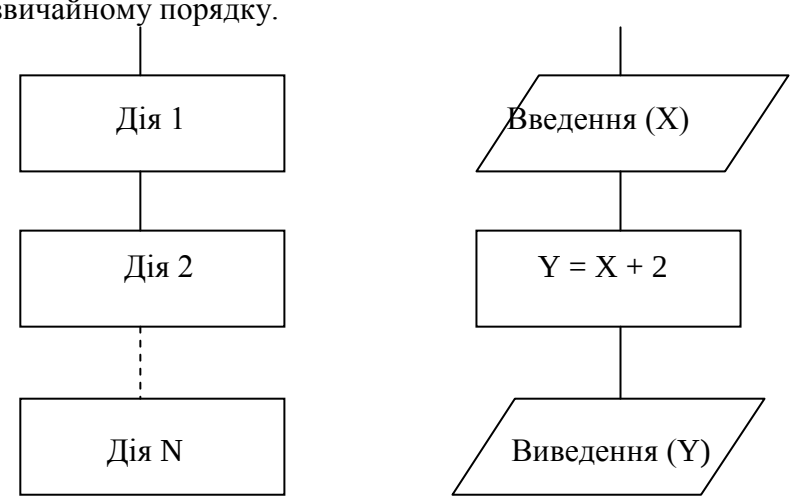
3. Складні дії та їх базові конструкції

Складні дії вміщують в собі прості дії та *умови* їх виконання. До складних дій відносяться:

- проходження;
- вибір або розгалуження;
- повторювання або цикл.

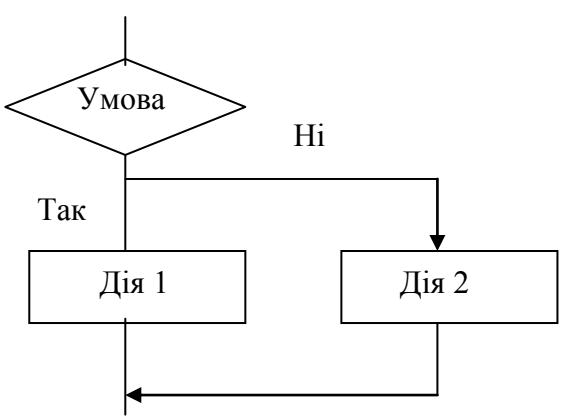
Проходження – проста послідовність дій, яка виконується одна за одній.

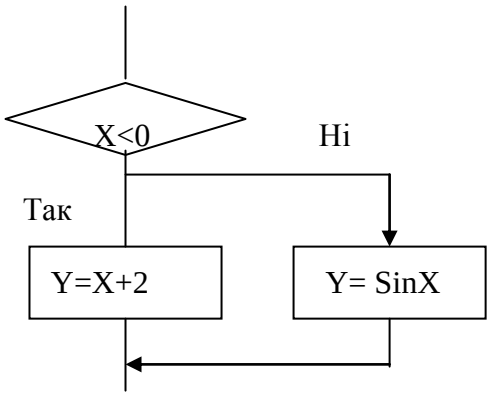
Конструкція проходження – це послідовність простих функціональних блоків, які виконуються у звичайному порядку.



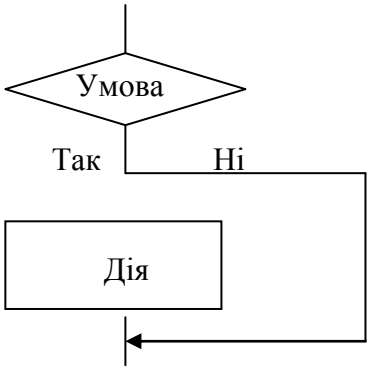
*Вибір (*або розгалуження) – проходження по одній із двох можливих гілок алгоритму в залежності від деякої умови.

Конструкція вибір має в собі *перевірку умови* та дві гілки. Перевірка умови в схемах алгоритмів зображається символом *Рішення* і позначається ромбом.

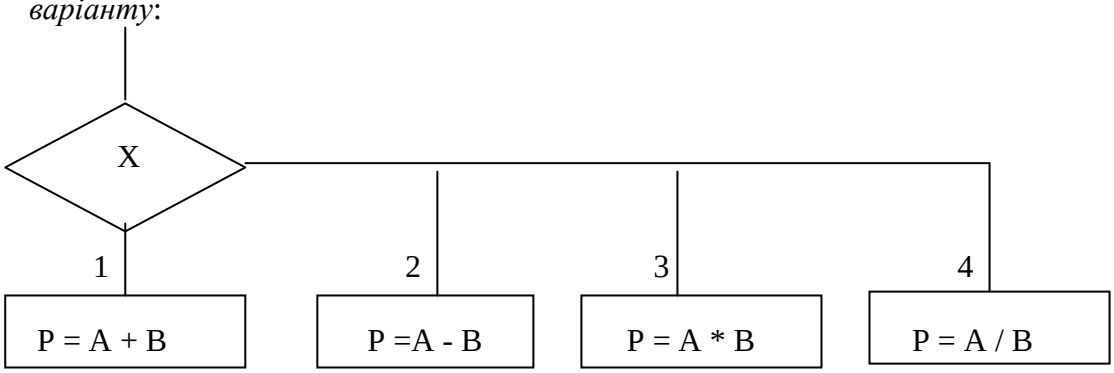




Інколи в одному з напрямів дія може не виконуватися. Тоді конструкція *Вибір* (розгалуження) буде мати вигляд:



Для того, щоб вибрати одну гілку із багатьох треба використовувати конструкцію *Вибір варіанту*:



В залежності від значення змінної X (X = 1, 2, 3, 4) обчислювання проходить по одній із гілок.

Повторення (*цикл)* – це багатократне виконання послідовності дій в залежності від деякої умови.

Для організації повторювання дій використовується змінна, яка носить назву *параметр циклу.* Послідовність дій, яка повторюється декілька разів, називають *тілом циклу***.** Умова може перевірятися до чи після тіла циклу.

Конструкція *цикл* вміщує в собі:

1) перевірку умови виходу з циклу;

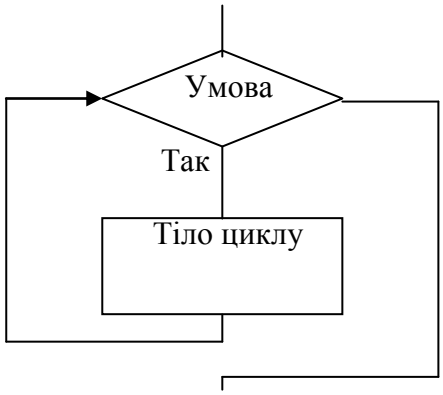
2) тіло циклу.

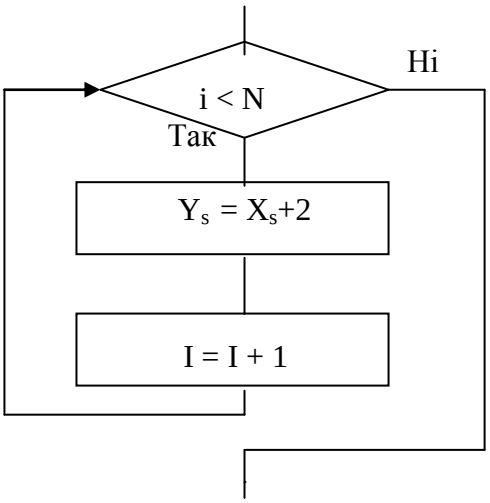
Кожному варіанту циклу відповідає своя конструкція *циклу.*

Умова повторювання дій у тілі циклу вказується в середині символу *рішення (*ромб) і перевіряється при кожному входженні в цикл. Цикл виконується до тих пір, поки додержується умова. У протилежному випадку (умова не додержується) – вихід із циклу.

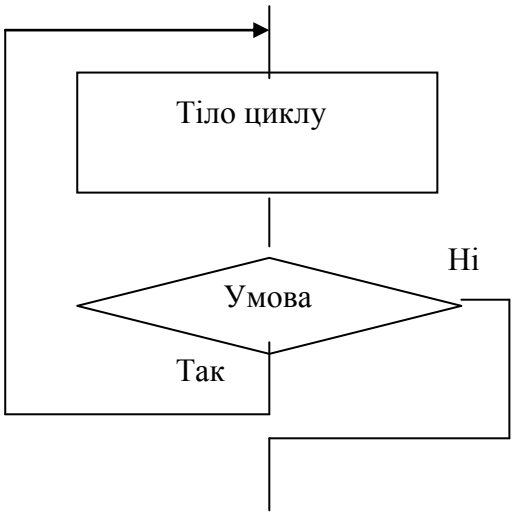
Початкове значення параметру циклу (i=1) присвоюється до першого входження в цикл.

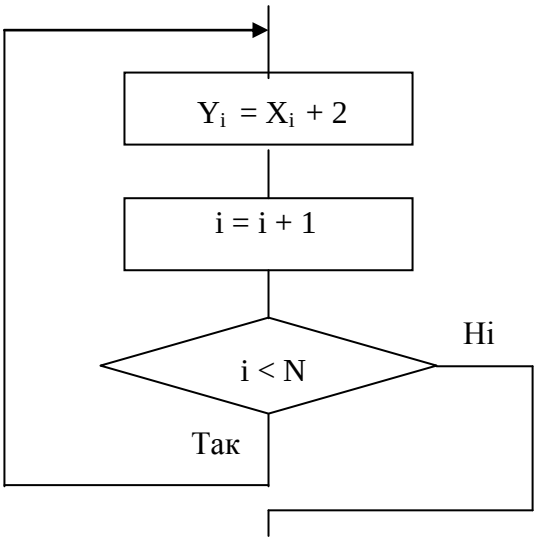
Конструкція *цикл* з передумовою має вигляд:



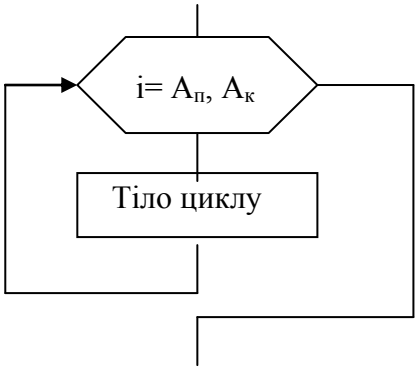


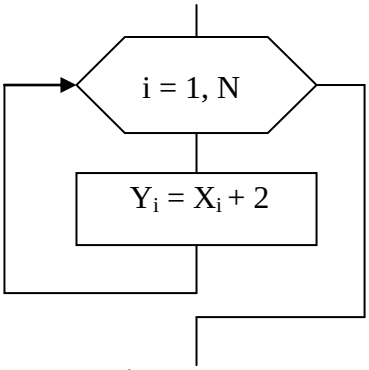
Конструкція *цикл* з після умовою має вигляд:





Коли кількість повторювань тіла циклу заздалегідь відома, доцільно використовувати конструкцію *Модифікація.* Умова повторювання тіла циклу вказується в середині символу *Модифікація.* Така конструкція в схемах алгоритмів має вигляд:





Де $A_п$ - початкове значення параметру циклу; $A_к$ - кінцеве значення параметру циклу.

Цикл виконується наступним чином. Параметру циклу *i* послідовно присвоюються значення $A_п$, $A_п$ +h, ($A_п$ +h) + h, … , $A_к$ (де h – крок зміни параметру циклу) і кожний раз виконується тіло циклу. Коли $i > A_к$ відбудеться вихід із циклу. Коли крок зміни параметру циклу h = 1, то його просто не вказують в символі *Модифікація*.

4. Види алгоритмів

Алгоритм розв'язання задачі – це послідовність дій, які необхідно виконати над початковими даними, щоб одержати результат. В залежності від порядку виконання дій алгоритми розподіляють на:

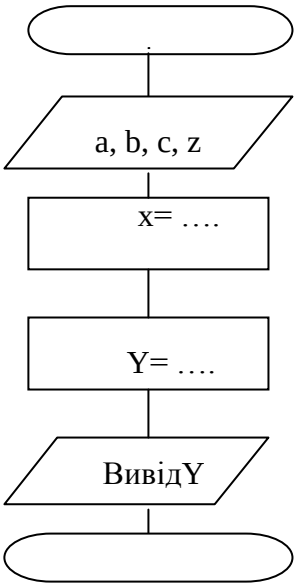
- лінійні;

- що розгалужуються;
- циклічні.

Лінійний алгоритм. У лінійному алгоритмі дії виконуються послідовно одна за другою. Розглянемо це на прикладі обчислення одного значення функції

$$Y = \frac{5\sin(az + bz + c)}{2 + (az + bz + c)}$$

при відповідних значеннях початкових даних a, b, c, z. Введемо проміжну змінну x = az + bz + c. Тоді Y = 5 Sin x (2 +x). Схема алгоритму буде мати наступний вигляд.



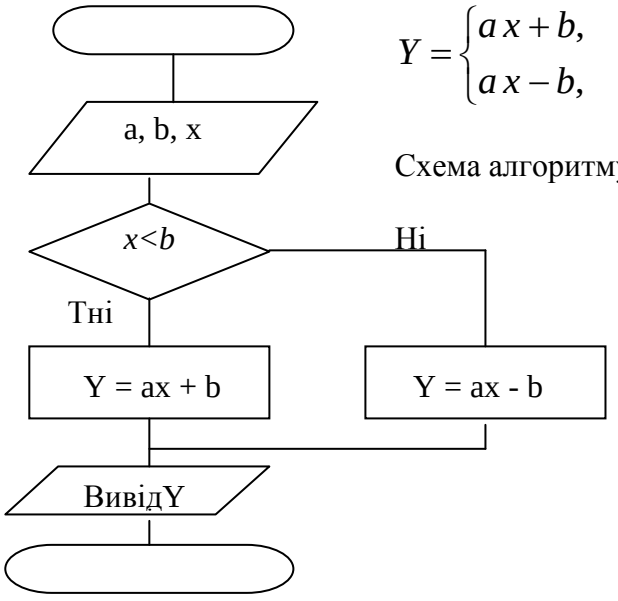
В алгоритмі виконуються наступні дії:

1. Введення початкових даних a, b, c, z.
2. Присвоювання змінній x значення виразу az + bz + c.
3. Присвоювання змінній Y значення виразу 5 Sin x (2 +x).
4. Виведення значення Y.

Алгоритм, що розгалужується передбачає вибір однієї з декількох послідовностей дій в залежності від умови. Розглянемо конкретний приклад:

$$Y = \begin{cases} a\,x + b, & \text{ÿêùî} \quad x < b; \\ a\,x - b, & \text{ÿêùî} \quad x \geq b. \end{cases}$$

Схема алгоритму буде мати наступний вигляд:

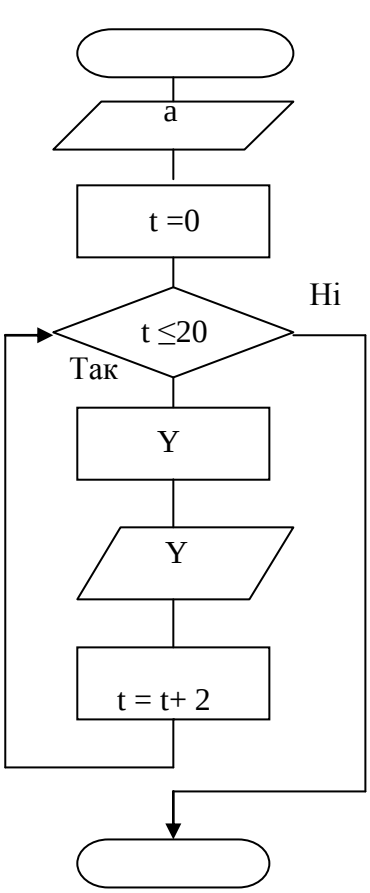


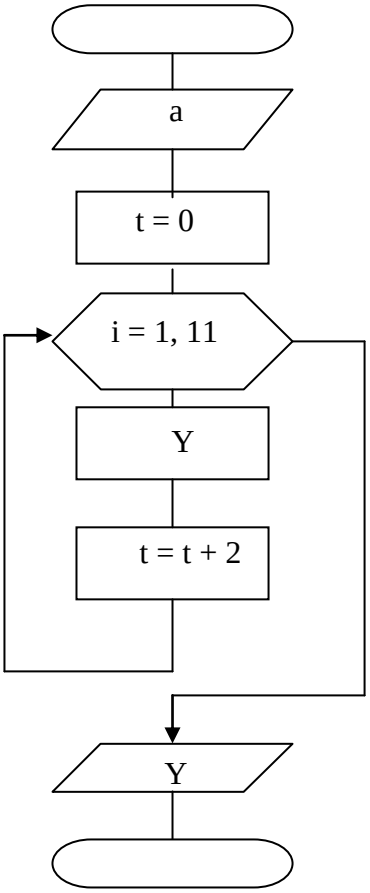
Циклічний алгоритм – це алгоритм, у якому повторюється одна і таж послідовність дій над різними даними.

Для організації циклу в алгоритмі необхідно передбачити наступні дії.

1. Завдання початкового значення параметру циклу (підготовка циклу).
2. Зміна значення параметру циклу пере кожним новим повторенням циклу.
3. Перевірку умови закінчення повторювань циклу за значенням параметру циклу.

Циклічний алгоритм розглянемо на прикладі обчислення значення функції Y = exp(-at) при зміні значення t від 0 до 20 з кроком 2.





Питання для самоконтролю.

Укажіть правильну відповідь:

1. Байт -- це:
   1. 7 двійкових розрядів
   2. 10 двійкових розрядів
   3. 8 двійкових розрядів
2. У ЕОМ виконується:
   1. Алгоритм
   2. Задача
   3. Програма
3. У двійковій системі числення записано 101. В десятковій системі числення -- це:
   1. 101
   2. 23
   3. 5
4. Алгоритм -- це:
   1. Перелік дій
   2. Перелік дій і довільне їх виконання
   3. Перелік дій і послідовність їх виконання
5. На схемі алгоритму зображений ромб. Це:
   1. Початок алгоритму
   2. Введення даних
   3. Умова розгалуження

6. Яка послідовність рішення задачі на ЕОМ:
    1. Задача -- програма -- алгоритм
    2. Програма -- задача -- алгоритм
    3. Задача -- алгоритм -- програма
7. На схемі алгоритму зображений прямокутник. Це:
    1. Початок алгоритму
    2. Обробка даних
    3. Введення даних
8. Початок алгоритму в схемах алгоритмів позначається:
    1. Ромбом
    2. Овалом
    3. Прямокутником
9. На схемі алгоритму зображений паралелограм. Це:
    1. Введення - виведення даних
    2. Обробка даних
    3. Кінець алгоритму
10. При розробці алгоритму на першому кроці використовується принцип:
    1. "Як зробити?"
    2. "Що зробити?"
    3. Детальний опис дій
11. Конструкція "Повторення" (Цикл) вміщує в собі:
    1. Перевірку умови і тіло циклу
    2. Тіло циклу і параметр циклу
    3. Перевірку умови і параметр циклу
12. Для зображення розгалуження в схемах алгоритмів використовується конструкція:
    1. Перевірка
    2. Повторення
    3. Вибір
13. Кінець алгоритму в схемах позначається:
    1. Овалом
    2. Паралелограмом
    3. Не позначається
14. Складові дії -- це:
    1. Цикл, розгалуження, присвоєння
    2. Присвоєння, введення, висновок
    3. Повторення, розгалуження, проходження
15. Цикл з відомим числом повторення в схемах алгоритмів зображається конструкцією:
    1. Вибір
    2. Проходження
    3. Модифікація
16. При виконанні циклу параметр циклу змінюється:
    1. Перед входом в цикл
    2. При виконанні тіла циклу
    3. Після виходу з циклу
17. При організації циклу установка початкових значень параметру циклу обов'язкова:
    1. Для всіх типів циклу
    2. Тільки для циклу з передумовою
    3. Тільки для циклу з відомим числом повторювань

Лекція 2
ЗАГАЛЬНІ ВІДОМОСТІ ПРО МОВУ С (С++)

Мета лекції. Вивчити основні конструкції мови С (С++): алфавіт мови, стандартні типи даних, структуру програму на мові С (С++) та оголошення змінних і констант у програмі.

Основні питання лекції.

1. Алфавіт мови.
2. Ключові слова, ідентифікатори, коментарі.
3. Стандартні типи даних.
4. Структура програми на мові С (С++).
5. Оголошення змінних і констант

Мова С була розроблена в 1972 році. Вона була створена як інструмент для написання операційної системи Unix і орієнтована на розробку системних програм. Мова С є мовою високого рівня. Вона має повний набір конструкцій структурного програмування, підтримує модульну і блокову структуру програм, забезпечує можливість роздільної компіляції модулів. У той же час мова С має набір низькорівневих засобів, що дозволяють мати доступ до апаратурних засобів комп'ютера.

На початку 80-х років в результаті доповнення і розширення мови С була створена по суті нова мова, яка була названа «Сі з класами». В 1983 році ця назва була замінена на С++.

Таким чином, мова С++ розглядається як надбудова до мови С, а це означає, що будь-яка коректна програма на С є також коректною програмою на С++ (за виключенням лише деяких несуттєвих розрізнень).

1. Алфавіт мови

Алфавіт мови С(С++) складається з латинських букв (прописних і рядкових), арабських цифр, спеціальних і управляючих символів.

1. *Букви* – A, B, C, …, Z, a, b, c, … , z.

2. *Цифри* – 0, 1, 2, …, 9.

3.*Спеціальні символи*. Ці символи використовуються з одного боку для організації процесу обчислень, а з іншого - для передачі компілятору певного набору інструкцій.

Спеціальні символи мови С++ .

Символ	Найменування	Символ	Найменування
,	кома	)	кругла дужка права
.	крапка	(	кругла дужка ліва
;	крапка з комою	}	фігурна дужка права
:	двокрапка	{	фігурна дужка ліва
?	знак питання	<	менше
'	апостроф	>	більше
!	знак оклику	[	квадратна дужка
|	вертикальна риска	]	квадратна дужка
/	дробова риска	#	номер
\	зворотна риска	%	відсоток
~	тильда	&	амперсанд
*	зірочка	^	логічне не
+	плюс	=	рівно
-	мінус	"	лапки

До спеціальних символів відносяться також розділові символи: пробіл, символи табуляції, переводу рядка, повернення каретки, нова сторінка і новий рядок. Ці символи відділяють один від одного об'єкти, визначені користувачем, до яких належать константи і ідентифікатори. Послідовність розділових символів розглядається компілятором як один символ (послідовність пропусків).

4. *Управляючи символи*. Окрім виділених груп символів у мові С++ широко використовуються, так звані, управляючі послідовності, тобто спеціальні символьні комбінації, що використовуються у функціях введення і виведення інформації. Управляюча послідовність будується на основі використання зворотної дробової риски (\) (обов'язковий перший символ) і комбінації латинських букв і цифр.

Управляючі послідовності. Таблиця 2

Управляюча послідовність	**Найменування**
\b	Повернення на крок
\n	Перехід на новий рядок
\r	Повернення каретки
\f	Нова сторінка
\"	Лапки
\'	Апостроф
\0	Нуль-символ
\t	Горизонтальна табуляція
\v	Вертикальна табуляція
\a	Звуковий сигнал
\ \	Зворотна дробова риска
\?	Знак питання

До цієї групи відносяться спеціальні управляючи символи для посилань на символи за їхніми кодами

\0dd – восьмеричний код символу;

\0xdd – шістнадцятирічний код символу.

Наприклад. \032 – це символ з восьмеричним кодом 32.

2. Ключові слова, ідентифікатори, коментарі

Із елементів алфавіту мови С(С++) – букв, цифр, спеціальних символів будуються складні конструкції – слова. До них відносяться:

- ключові (службові) слова;
- ідентифікатори.

Ключові слова - це зарезервовані слова, які наділені певним сенсом. Їх можна використовувати тільки відповідно до значення, відомого компілятору мови С++.

Наведемо список ключових слів:

auto double int struct break else long switch register tupedef char extern return void case float unsigned default for signed union do if sizeof volatile continue enum short while.

Ідентифікатори – це імена змінних, констант, функцій і типів даних, що

використовуються в програмі. Ідентифікатором може бути довільна послідовність латинських букв (прописних і рядкових), цифр і символу підкреслення, яка починається завжди з букви або символу підкреслення. Наприклад:

A1, ch, ALPHA, B_27.

Мова С (С++) розрізняє прописні і рядкові букви. У мові С (С++) Ff, FF, fF, ff – це чотири різних ідентифікатори.

Слід зазначити важливі особливості при виборі ідентифікатора.

По-перше, ідентифікатор не повинен співпадати з ключовими словами, із зарезервованими словами та іменами функцій бібліотеки компілятора мови С++.

По-друге, слід звернути особливу увагу на використання символу підкреслення (_) як першого символу ідентифікатора, оскільки ідентифікатори побудовані таким чином, можуть співпадати з іменами системних функцій і (або) змінних, а також, при використанні таких ідентифікаторів програми можуть виявитися непереносними, тобто їх не можна використовувати на комп'ютерах інших типів.

По-третє, на ідентифікатори, що використовуються для визначення зовнішніх змінних, повинні бути накладені обмеження, сформовані редактором зв'язків (відзначимо, що використання різних версій редактора зв'язків, або різних редакторів накладає різні вимоги на імена зовнішніх змінних).

Коментар. Часто буває корисно вставляти в програму пояснюючий текст, який призначається як коментар тільки для читаючої програму людини, і ігнорується компілятором у програмі. У С++ це можна зробити одним з двох способів.

Перший спосіб запозичений з С. Тут символи «/*» починають коментар, що закінчується символами «*/». Вся ця послідовність символів еквівалентна символу пропуску. Цей спосіб корисно використовувати для багаторядкових коментарів і вилучення частин програми при редагуванні:

```
#include  <iostream. h>   /* директива компілятору для включення в програму
                              додаткового файлу iostream. h */
```

У другому способі символи «//» починають коментар, який закінчується в кінці рядка, на якому вони з'явилися. Цей спосіб найбільш корисний для коротких коментарів.

```
int a=17;  // ініціалізація змінної a.
```

У програмах можна використовувати будь-який з цих двох видів коментарів.

3. Стандартні типи даних

Люба програма обробляє дані. Але дані можуть бути присутні безпосередньо в тексті програми. В цьому випадку вони представляються літералами. Літерали можуть бути числовими, символьними і рядковими.

Числові літерали (числа) можуть бути цілими і дійсними.

Цілий літерал – це ціле число, яке записане в тексті програми. Число може бути записано в десятковій, восьмирічній або шістнадцятирічній системах числення.

Для того, щоб відрізнити в якій системі числення записано число, у восьмирічній системі перед числом записують нуль, а в шістнадцятирічній – латинську букву «х». Наприклад:

123, 1999 - числа в десятковій системі числення;

011, 0177 - числа у восьмирічній системі числення;

0x91, 0x154 - числа в шістнадцятирічній системі числення.

Дійсні числа записуються у звичайній десятковій або в експоненціальній формі. Наприклад:

123.5, 3.14, 0.95, 0.00001.

Число 123.5 можна записати в експоненціальній формі наступним чином. Цей формат представляє числа як серію цифр, званих мантисою і порядком, який представляє ступінь 10-ти:

$123.5 * 10^0 = 12.35 * 10^1 = 1.235 * 10^2 = 1235 * 10^{-1}$.

У мові С (С++) замість цифри 10 записується буква Е:

123.5E0 = 12.35E1 = 1.235E2 = 1235E-1.

Символьний літерал (символьна константа) необхідна для представлення одного символу. Це любий символ, який заключний у одинарні лапки:

‘W’, ‘&’, ‘3’, ‘Ф’.

Рядковий літерал – це люба послідовність символів, яка заключна у парні лапки:

“ALPHA”, “ТАБЛИЦЯ”, “Це рядок \n”, “123/5”.

Однак дані можуть розміщувати не тільки в тексті програми, але і зберігатися в пам'яті машини. Числа, що зберігаються в оперативній пам'яті, займають по довжині різну кількість байтів. Кількість байтів, яке займає число в пам'яті, визначається його типом.

У мові С (С++) визначено *п'ять* базових (стандартних) типів даних.

char – символьний;

int – цілий;

float – дійсний;

double – дійсний з подвійною точністю;

bool – бульовий (логічний).

На основі цих типів будуються інші типи даних. Для цих цілей використовуються модифікатори типу, які ставляться (записуються) перед відповідним типом. До модифікаторів типу відносяться:

short – короткий;

long – довгий;

signed – знаковий;

unsigned - без знаковий.

Символьні типи. Символьні дані включають алфавітно-цифрові елементи, які визначають заголовні і рядкові букви, цифри, знаки пунктуації і спеціальні символи. Для зберігання символу у С++ використовується простий тип char.

Символьний тип у С++

Тип	Розмір	Діапазон числа
char	1 байта	0... 255=0... 2^8-1

Цілі типи. Цілі числа – це позитивні або негативні цілі числа, що складаються із знаку і послідовності цифр. Про цілі числа говорять як про знакові (signed) числа.

Цілими типами в мові С++ є: int, short int, long int.

Цілі типи даних

Тип	Розмір	Діапазон без знакового числа	Діапазон числа із знаком
short int	2 байти	0.... 65535=2^{16}-1	-2^{15}... 2^{15}-1= = -327678... 32767
int	4 байти	0.... 4294967295 = 2^{32}-1	-2^{31}... 2^{31}-1
long int	4 байти	0.... 4294967295 = 2^{32}-1	-2^{31}... 2^{31}-1

****Примітка***. Загальний тип int ідентифікує цілі, чия довжина у байтах залежить від комп'ютера і компілятора. В даній таблиці представлена розмірність типу int для 32-розрядної операційної системи.

Дійсні типи. Дійсні числа – це множина раціональних і ірраціональних чисел. Дійсними типами в мові С++ є: float – числа з плаваючою крапкою одинарної точності, double - числа з плаваючою крапкою подвійної точності, long double. Тип long double застосовується для обчислень, що вимагають високої точності.

Дійсні типи даних у С++

Тип	Розмір	Діапазон числа
float	4 байти	-2^{31}... 2^{31}-1
double	8 байтів	-2^{63}... 2^{63}-1
long double	8 байтів	-2^{63}... 2^{63}-1

- ***Примітка***. Типи double і long double введені для сумісності різних компіляторів С++.

Інші типи даних.

Тип даних **void** *створити неможна***.** Він застосовується у функціях, що не повертають ніякого значення.

Покажчики, на відміну від змінних інших типів, не містять даних у звичному розумінні цього слова. Натомість, покажчики містять адреси пам'яті, де зберігаються дані.

Масиви є прикладом наборів даних. Існують одновимірні і багатовимірні масиви. Одновимірний масив – це кінцевий, послідовний список елементів одного і того ж типу даних. Багатовимірний масив є структурованим типом даних, який

створюється шляхом вкладення одновимірних масивів.

Рядки – це особливі форми масивів, що містять символьні дані, які поводяться з символами як з одним об'єктом і надають операції для доступу до послідовностей символів у рядку.

Записи – це структури, які зв'язують елементи (записи) різних типів в один об'єкт.

Файли – це структури, представлені даними, що зберігаються, у сукупності з операціями передачі.

Константи в мові С (С++) представляють фіксовану (постійну) величину, яка не може бути змінена при виконанні програми. Константи можуть бути будь-якого базового типу даних.

У Мові С(С++) можна використовувати константи, записані у восьмеричній і шістнадцятирічний системах числення. Для запису шістнадцятирічної константи необхідно використовувати 16 символів:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, З, D, E, F.

Для запису восьмеричної константи використовуються перші 8 цифр десяткової системи числення:

0 1 2 3 4 5 6 7 .

Надалі нам необхідно буде представляти константи в різних системах числення, у тому числі і в двійковій. Запис чисел у різних системах числення представлений в таблиці

Десяткове число	8 – ричне число	16 – ричне число	Двійкове число
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	З	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111

4. Структура програми на мові С (С++)

Програма на мові С (С++) є сукупністю наступних елементів.

1. Директив препроцесора (вони визначають перетворення тексту програми перед компіляцією).
2. Вказівок компілятору.
3. Оголошень і визначень.
4. Одну або декілька функцій.

Функції описують сукупність дій, які повинна виконати програма.

Якщо функцій декілька то серед них виділяться одна – ***головна***, яка має ім'я *main().* З неї починається виконання програми. Якщо програма містить єдину функцію, то вона *обов'язково* повинна мати ім'я *main().*

Структура програми на мові С (С++)

директива препроцесора 1 # директива прелроцессора 2 ..... # директива прелроцессора N
вказівка компілятору 1 # вказівка компілятору 2 ..... # вказівка компілятору М
Оголошення і визначення змінних, констант функцій

void main ()
{
Тіло функції
}

rs

Початкова програма може містити довільне число директив, вказівок, оголошень і визначень. Вони можуть розташовуватися в будь-кому місті програми, але діяти в програмі вони починають тільки з того місця, де вони розташовані.

У початковий текст програми можна вставляти текст з іншого файлу за допомогою директив препроцесора # include (включати):

include < ім'я файла>

Якщо ім'я файлу розміщено в кутових дужках, то пошук потрібного файлу проводиться тільки в стандартних каталогах. Якщо ім'я файлу розміщено в лапках – пошук потрібного файлу проводиться в поточному каталозі. Наприклад, директива

include <stdio. h>

повідомляє препроцесор про необхідність включити в програму файл stdio.h. Цей файл містить прототипи всіх функцій стандартного введення-виведення даних в мові С (С++).

Вказівки компілятору здійснюються за допомогою директиви препроцесора # pragma. Ця директива служить для установки параметрів компілятора. Директива має вигляд:

pragma директива

Різні компілятори мають різні набори директив # pragma. Часто різні по виду директиви в різних компіляторах виконують схожі дії.

Наприклад, директива

pragma argsused

означає, що компілятору не потрібно видавати застережливих повідомлень, що параметри функції у середині неї ніяк не використовуються.

Після всіх директив і вказівок препроцесору в програмі розташовується визначення функції main (). Будь-яка програма на С (С++) містить цю функцію, яка є її вхідною крапкою.

Функція main () – це окремий випадок функції взагалі. В мові С (С++) функції є будівельними блоками програми, спроектовані для вирішення конкретних задач.

Структура функції. Будь-яка функція на С (С++) має наступну структуру.

Визначення функції складається з двох частин: першого рядка int main(), який називається *заголовком функції,* і решти частини, взятої у фігурні дужки, яка є *тілом функції.*

Заголовок функції у короткій формі визначає інтерфейс між функцією і рештою програми, а тіло функції містить інструкції для комп'ютера, тобто визначає те, що власне робить функція.

Заголовок функції має наступний вигляд.

Тип ІмяФункциі (список параметрів)

Тип результату визначає тип значення, яке повертає функція. Якщо тип не вказаний, то за умовчанням передбачається, що функція повертає порожнє (void) значення.

За ім'ям в круглих дужках поміщаються її аргументи (список параметрів). Функції можуть не мати аргументів (параметрів), але круглі дужки необхідні у будь-якому випадку. Якщо аргументів декілька, то вони відділяються один від одного комами.

Наприклад, заголовок функції main() записується таким чином

int main()

або

void main().

5. Оголошення змінних і констант

Тіло функції складається з визначення (опису) змінних і констант і операторної частини.

У мові С (С++) спеціального розділу в програмі (як наприклад, у мові Паскаль), в якому б оголошувалися (описувалися) змінні і константи, нема. Оголошувати змінні і константи можна в будь-якому місці програми. Проте оголошення змінної обов'язково повинне передувати *її використовуванню* в програмі.

Оголошення змінних будемо розміщувати на початку функції, відразу за заголовком і забезпечувати інструкцію оголошення коротким коментарем про призначення змінної.

Інструкція оголошення змінної виглядає таким чином:

```
Тип  Ім'яЗмінної;
float x;
```

Інструкцію оголошення змінної можна використовувати для ініціалізації змінної:

```
Тип  Ім'яЗмінної = ПочатковеЗначення;
int x = 12;
```

Інструкція оголошення константи виглядає таким чином:

```
const  Тип  Ім'яКонстанти  = Значення;
const float pi = 3.14159;
```

Коли змінна оголошується в програмі, дуже велике значення має те, в якому місці програми вона була оголошена. Правило, яке визначає, де змінна може бути використана, залежить від того, де змінна була оголошена, і називається правилом видимості змінної.

У мові С (С++) можуть бути *три місця*, де змінна може бути оголошена.

1. Поза якими-небудь функціями, у тому числі і main(). Така змінна називається *глобальною* і може використовуватися в будь-якому місці програми.
2. Змінна може бути оголошена у середині тіла функції (у середині блоку). Оголошена таким чином змінна, називається *локальною* і може використовуватися тільки у середині цієї функції (блоку). Поза цією функцією (блоку) така змінна невідома.
3. Змінна може бути оголошена як параметр функції.

Приклад оголошення змінних в різних місцях програми.

```
# include <iostream.h.>
char ch;    // оголошення глобальної змінної ch
void main()
```

```
{
int k;  // оголошення локальної змінної k
.....
}
func 1(int m) //  оголошення змінної  m як параметра функції
{
......
}
```

Висновок. Програма на мові С (С++) є сукупністю наступних елементів.

1. Директив препроцесора (вони визначають перетворення тексту програми перед компіляцією).
2. Вказівок компілятору.
3. Оголошень і визначень.
4. Одну або декілька функцій.

Функції описують сукупність дій, які повинна виконати програма. Якщо функцій декілька то серед них виділяться одна – ***головна***, яка має ім'я *main().* З неї починається виконання програми. Якщо програма містить єдину функцію, то вона *обов'язково* повинна мати ім'я *main().*

Питання для самоконтролю.

1. Укажіть правильний запис дійсного числа в мові С (С++):
   1. 0.5
   2. 0,5
   3. 0:5
2. У результаті рішення задачі на ЕОМ на екрані з'явилася відповідь: 2.300000E+01. Яке це число:
   1. 0.23
   2. 2.3
   3. 23
3. Запис символьної константи в мові С (С++) здійснюється так:
   1. M
   2. 'M'
   3. "M"
4. У мові С (С++) ідентифікатор (ім'я) записується так:
   1. 1A
   2. A.1
   3. A1
5. Виконується послідовність операторів

   X=2; Y=3; X=Y; Y=X;

   Укажіть правильний результат виконання цих операторів:
   1. X=2 і Y=3
   2. X=3 і Y=3
   3. X=3 і Y=2
6. Необхідно вивести на екран обчислене значення X. Який оператор виводу правильний:
   1. cout<<X=<< X;
   2. cout<<”X=”<<X;
   3. cout<<X=;
7. Виконується послідовність операторів:

   X=2; Y=3; cout<<X<<Y;

   Який результат буде виведений на екран:

1. 2 3
   2 23
   3. 3 2

8. Оператор cout<<X<<endl; використовується для:
   1. Переводу курсора виведення на новий рядок
   2. Пропуску одного символу в слові, що виводиться
   3. Закінчення процесу виведення даних
9. Які з простих типів даних можна виводити на екран використовуючи оператор виводу:
   1. Усі відомі типи даних
   2. Тільки дані, які несуть цифрову інформацію
   3. Тільки дані цілого, дійсного і символьного типу
10. Укажіть правильний опис константи:
   1. const int M: 25;
   2. const int M = 25;
   3. const int M := 25;
11. Який вигляд повинен мати файл для введення числових значень початкових даних:
   1. 2,0 1,7 4,3
   2. 2.0 1.7 4.3
   3. 2.0, 1.7, 4.3
12. У програмі на мові С (С++) зустрівся такий запис:
   X, Y, Z = A;
   Укажіть на можливість такого запису:
   1. Можливо
   2. Не можливо
   3. Можливо в окремих випадках
13. Виконується послідовність операторів:
   X=2; cout<<X+1; Що буде видано на екран дисплея:
   1. X+1
   2. 2
   3. 3
14. Чи можна використовувати такий оператор введення cin>>X+1;
   1. Можна
   2. Не можна
   3. У окремих випадках можна
15. Відомо, що змінні A,B,C мають тип BOOLEAN. При цьому
   A= TRUE; B= FALSE; C= FALSE;
   Виконуються операції: A && B; A || B; !C;
   Укажіть результат виконання операцій:

Лекція 3
ОПЕРАЦІЇ І ВИРАЗИ У МОВІ С (С++)

Мета лекції. Вивчити сукупність операцій мови С (С++), порядок і особливості їх виконання.

Основні питання лекції.

1. Вирази.
2. Операції мови С (С++).
3. Арифметичні операції
4. Операція присвоєння
5. Арифметичні операції з присвоєнням

6. Операції введення-виведення
7. Математичні функції

1. Вирази.

У математиці із змінних, констант та функцій складаються формули. Так само в мові С (С++) наступним рівнем представлення даних після одиночних змінних і констант є *вирази.*

Вирази в мові С (С++) – це деяка допустима комбінація змінних, констант, функцій і знаків операцій. Вирази в С (С++) відрізняються від математичних формул наступним:

- набір (зображення) операцій, що сполучають члени виразу, відрізняється від алгебри;
- вирази записуються в рядок.

Наприклад, формула

$$R = \frac{(A+B).C}{A+B+C}$$

у мові С (С++) запишеться таким чином:

$$R = (A + B)*C / (A + B + C).$$

2. Операції мови С (С++).

Кожна операція в мові С (С++) характеризується своїм пріоритетом. Пріоритет операції визначає порядок, в якому проводиться обчислення значення виразу. Нижче в таблиці приведені операції, які розташовуються в порядку убування пріоритету.

Крім того, кожна операція має свій порядок виконання – зліва на право або з права наліво.

Знак операції – це символ або комбінація символів, які повідомляють компілятор про необхідність провести певні арифметичні, логічні або інші дії.

Для кожної операції мови С (С++) визначено кількість операндів:

- один операнд – унарна операція; наприклад, -X – змінює знак операнда;
- два операнди – бінарна операція; наприклад, операція складання двох чисел A + B;
- три операнди – операція умова ?: . Така операція тільки одна. Мова про неї піде нижчим.

Повний список операцій приведений в таблиці.

Операції мови С (С++)

Операція	Опис	Пріоритет	Порядок виконання
Первинні і постфіксні операції			
[ ]	Індексація масиву	16	Зліва на право
( )	Виклик функції	16	Зліва на право
.	Елемент структури	16	Зліва на право
->	Елемент покажчика	16	Зліва на право

+ +	Постфіксний інкремент	15	Зліва на право
- -	Постфіксний декремент	15	Зліва на право
	Одномісні (унарні) операції		
+ +	Префіксний інкремент	14	Справа наліво
- -	Префіксний декремент	14	Справа наліво
sizeof	Розмір в байтах	14	Справа наліво
(тип)	Приведення типу	14	Справа наліво
~	Порозрядне NOT		
!	Логічне Не	14	Справа наліво
-	Унарний мінус	14	Справа наліво
&	Узяття адреси	14	Справа наліво
*	Розименування покажчика	14	Справа наліво
	Мультиплікативні операції		
*	Множення	13	Зліва на право
/	Розподіл	13	Зліва на право
%	Узяття по модулю	13	Зліва на право
	Адитивні операції		
+	Складання	12	Зліва на право
-	Віднімання	12	Зліва на право
	Операції порозрядного зсуву		
<<	Зсув вліво	11	Зліва на право
>>	Зсув управо	11	Зліва на право
	Операції відношення (порівняння)		
<	Менше	10	Зліва на право
<=	Менше або рівно	10	Зліва на право
>	Більше	10	Зліва на право
>=	Більше або рівно	10	Зліва на право
=	Рівно	9	Зліва на право
!=	Не рівно	9	Зліва на право
	Порозрядні операції		
&	Порозрядне AND	8	Зліва на право
^	Порозрядне XOR	7	Зліва на право
!	Порозрядне OR	6	Зліва на право
	Логічні операції		
&&	Логічне AND	5	Зліва на право
!!	Логічне OR	4	Зліва на право
	Умовна операція		
?:	Умовна операція	3	Справа наліво
	Операції привласнення		
=	Привласнення	2	Справа наліво
*=	Привласнення множення	2	Справа наліво
/=	Привласнення ділення	2	Справа наліво
%=	Привласнення модуля	2	Справа наліво
+=	Привласнення суми	2	Справа наліво
-=	Привласнення різниці	2	Справа наліво

<<=	Привласнення лівого зсуву	2	Справа наліво
>>=	Привласнення правого зсуву	2	Справа наліво
&=	Привласнення AND	2	Справа наліво
^=	Привласнення XOR	2	Справа наліво
!=	Привласнення OR	2	Справа наліво
,	Кома	1	Справа наліво

Розглянемо особливості виконання операцій в С (С++).

3. Арифметичні операції

У мові С++ використовуються чотири основні математичні операції: складання, віднімання, множення і ділення, що позначаються:

+ - складання;
- - віднімання та унарний мінус;
* - множення;
/ - ділення;
% - узяття по модулю (ділення по модулю).

Ці операції застосовуються як до цілих типів даних, так і до дійсних.

Проте існують також і інші арифметичні операції, використання яких не таке очевидне.

Операції *складання, віднімання, множення і ділення* виконуються зліва на право. Якщо операнди мають один тип, то результат арифметичної операції матиме той же тип.

Всі операції можуть застосовуватися до всіх типів вбудованих (стандартних) типам даних. Якщо їх застосовувати до дійсних чисел, то для таких операндів все зрозуміло і законно – це звичні арифметичні операції.

Якщо операція *ділення* застосовується до цілочисельних операндів, то результат виконання операції буде так само *цілим числом,* тобто залишок від ділення відкидається. Наприклад, 11/3 буде рівно 3, а вираз 1/2 буде рівний нулю.

Операція *ділення по модулю (узяття по модулю)* використовується тільки з цілочисельними операндами і дає залишок від ділення першого операнда на другий. Так, результат виконання операції 11%3 буде рівний 2. Результатом операції 5%5 буде нуль.

Операція унарний мінус дозволяє змінити знак числа на протилежний.

4. Операція присвоєння

Операція присвоєння = не представляє особливих труднощів Операція присвоєння має наступний вигляд:

A = V;

де
A – змінна будь-якого базового типу даних мови С (С++)*;*
V – вираз.

Операція виконується таким чином. Обчислюється значення виразу *V*, що стоїть в лівій частині. Потім обчислене значення *V* присвоюється змінній *A*. Наприклад:

```
# include <iostream.h>
void main()
{
float x;
int a = 3;
int b = 4;
float c = 1.5;
x = a + b - c;
.....
}
```

У результаті виконання цієї операції змінній *x* буде присвоєно значення 5.5.

Особливістю операції присвоєння в мові С (С++) є можливість записати такий оператор:

```
a = b = c = x * y;
```

Таке багатократне присвоєння при написанні програм на С (С++) зустрічається досить часто. Присвоєння виконується справа наліво. Спочатку обчислюється значення x * y, а потім це значення присвоюється c, потім *b* і лише потім *a*.

5. Арифметичні операції з присвоєнням

Мова С(С++) має у своєму розпорядженні засоби для того, щоб скоротити розмір коду і зробити його наочним. Одним з таких засобів є арифметичні операції з присвоєнням. Вони допомагають надати характерного вигляду програмному коду у стилі С++.

У більшості мов програмування типовим є оператор, подібний до наступного:

```
total = total + item;          // складання total з item
```

У даному випадку ви виконуєте складання із заміщенням вже існуючого значення одного з доданків. Але така форма оператора не відрізняється стислістю, оскільки нам доводиться двічі використовувати в ньому ім'я total. У С (С++) існує спосіб скоротити подібні оператори, застосовуючи арифметичні операції з присвоєнням. Такі операції комбінують арифметичну операцію і операцію присвоєння, тим самим виключаючи необхідність використання імені змінної двічі. Попередній оператор можна записати за допомогою складання з присвоєнням таким чином:

```
total += item;          // складання total з item
```

З присвоєнням комбінується не тільки операція складання, але й інші арифметичні операції: -=, *=, /=, %= і т.д.

6. Математичні функції

При обчисленнях досить часто доводиться витягувати квадратне коріння, підносити до ступеня, використовувати яку-небудь тригонометричну функцію. В язиці З (С++) є математичні функції, які можна вставляти в арифметичні вирази. Такі функції розташовуються в заголовному файлі <math.h>. Перелік математичних функцій приведений в таблиці.

Математичні функції – файл math.h

Функція	Прототип і скорочений опис дій
abs(x)	int abs(int x); повертає абсолютне значення цілого аргументу x.
fabs(x)	double fabs(double x); повертає абсолютне значення аргументу float x подвійної точності.
labs(x)	long labs(long x); повертає абсолютне значення цілого аргументу long x.
cabs(znum)	double cabs(struct complex znum); обчислює абсолютне значення комплексного числа znum. Визначення структури (типу) complex – у файлі math.h.
exp(x)	double exp(double x); Обчислює e^x.
log(x)	double log(double x); обчислює значення натурального логарифму ln x.
log10(x)	double log10(double x); обчислює значення log x.
pow(x,y)	double pow(double x, double y); обчислює значення $x^{y.}$
pow10(p)	double pow10(int p); обчислює значення 10^p.
sqrt(x)	double sqrt(double x); обчислює значення $\sqrt{x}$.
floor(x)	double floor(double x); знаходить найбільше ціле, яке не перевершує значення x.
fmod(x,y)	double fmod(double x, double y); визначає остачу від ділення x/y.
ldexp(v,x)	double ldexp(double v, int x); обчислює значення $v*2^x$.
hypot(x,y)	double hypot(double x, double y); обчислює значення гіпотенузи z по значенням катетів x та y.
poly(x,n,c[ ])	double poly(double x, int n, double c[ ]); обчислює значення поліному ступеня n.
sin(x)	Функція синуса. Кут (аргумент) задається в радіанах.
asin(x)	Функція арксинуса. Значення аргументу повинно находитись у діапазоні від –1 до +1.
sinh(x)	Визначає значення гіперболічного синуса для x.
cos(x)	Функція косинуса. Кут (аргумент) задається в радіанах.
acos(x)	Функція арккосинуса. Значення аргументу повинно находитись у діапазоні від –1 до +1.
cosh(x)	Визначає значення гіперболічного косинуса для x.
tan(x)	Функція тангенса. Кут (аргумент) задається в радіанах.
atan(x)	Функція арктангенса. Кут (аргумент) задається в радіанах.
atan2(y,x)	double atan2(double y, double x); функція арктангенса від значення y/x
tanh(x)	Визначає значення гіперболічного тангенса для x.

При зверненні до цих функцій необхідно дотримуватись наступних вимог:
- *x* і *y* повинні мати тип double;
- *n* повинне мати тип int;
- кути (аргументи) в тригонометричних функціях задаються в радіанах.

Всі функції повертають значення типу double.

При зверненні до функцій можуть виникати помилки області і помилки діапазону. Помилка області виникає, якщо аргумент функції виходить за область значень, для якої визначена функція. Помилка діапазону виникає у тому випадку, коли результат функції не може бути представлений у вигляді double.

Розглянемо приклад використання математичних функцій при записі арифметичних виразів.

Математичний вираз:

$$y = \frac{e^a x^3 + Cosb^2}{\sqrt{\ln x + tgx - SinQ}}.$$

На мові С (С++) цей вираз матиме наступний вигляд:

```
y= (exp(a)*pow(x,3)+cos(pow(b,2)))/sqrt(log(x)+tan(x)-sin(Q));
```

7. Операції введення-виведення

У мові С (С++) відсутні багато вбудованих в транслятор засобів, властивих розвиненим мовам програмування. В мові немає вбудованих методів доступу до файлів, немає операцій введення-виведення даних в їх звичайному розумінні. Виконання таких операцій забезпечується стандартною бібліотекою підпрограм.

Стандартні функції введення-виведення зберігаються в доступному бібліотечному файлі (заголовному файлі). Це файл – iostream.h. Щоб зв'язати програму користувача зі стандартною бібліотекою введення-виведення, необхідно на початку програми передбачити директиву препроцесора

```
# include <iostream.h>
```

У програмі на С (С++) автоматично відкриваються три файли для введення-виведення:

- стандартний файл введення (з клавіатури);
- стандартний файл виведення (на екран дисплея);
- стандартний файл повідомлень про помилки.

У мові С++ використовуються оператори введення-виведення.

Операція *введення* – це витяг даних з вхідного потоку і присвоєння значень змінним. Структура оператора введення наступна:

```
cin >> ім'я змінної.
```

Наприклад,
```
cin >>a;
cin >>xn>>xk>>dx;
```

Ключове слово cin пов'язано із стандартним потоком введення даних з клавіатури. Операція >> в мові С (С++) має двояке значення. Залежно від контексту – це порозрядна операція зсуву вправо або операція введення. Використовуючи cin і >> можна читати з клавіатури значення змінних основних типів даних (включаючи і рядки символів).

Операція *виводу* – це розміщення значень змінних у вихідний потік і відображення їх на екрані дисплея. Оператор виводу має наступний вигляд:

```
cout << x;
cout << x <<endl;
```

В результаті виконання першого оператора на екран дисплея буде виведено значення змінної x. При виконанні другого оператора на екран дисплея буде виведено значення змінної x і курсор переміщається на наступний рядок. Тут endl (end line) – ознака переміщення курсору на наступний рядок. Ключове слово cout – це потік стандартного виводу, за умовчанням зв'язаний з екраном дисплея. Використовуючи cout і << на екран дисплея можна виводити значення даних будь-якого вбудованого (стандартного) типу, включаючи і символьні рядки.

Наприклад, оператор

cout << “Введіть початкові дані a, b, c :”<<endl;

виводить на екран дисплея запрошення :

Введіть початкові дані a, b, c :

і курсор переміщається на наступний рядок. За цим оператором повинен слідувати оператор введення:

cin >> a >> b >> c;

Операцію >> і << можна використовувати підряд в одному операторі стільки раз, скільки це необхідне для введення або виведення даних.

Операції порозрядного зсуву >> і << мають певний пріоритет. Цей пріоритет не змінюється в конструкціях cin >> і cout <<. Так, при виконанні оператора

cout <<2 + 3 + 4 ;

на екран дисплея буде виведено 9, оскільки ранг арифметичних операцій вище за ранг операції <<.

Програма введення двох чисел, обчислення їх суми і видачі результату на екран дисплея на мові C (C++) матиме вигляд.

```
# include <iostream.h>
 void main()
  {
    int a, b, c;
    cout << “Введіть значення a і b :”<<endl;
    cin >> a >> b;
    c = a + b ;
    cout << “ СУМА = “ << c << endl;
  }
```

Висновок. Кожна операція в мові C (C++) характеризується своїм пріоритетом. Пріоритет операції визначає порядок, в якому проводиться обчислення значення виразу.

Крім того, кожна операція має свій порядок виконання – зліва на право або з права наліво.

Знак операції – це символ або комбінація символів, які повідомляють компілятор про необхідність провести певні арифметичні, логічні або інші дії.

Для кожної операції мови C (C++) визначено кількість операндів:

- один операнд – унарна операція; наприклад, -X – змінює знак операнда;
- два операнди – бінарна операція; наприклад, операція складання двох чисел A + B;
- три операнди – операція умова ?: . Така операція тільки одна. Мова про неї піде нижчим.

Питання для самоконтролю

1. Виконується фрагмент програми
```
int a=7, b=4, x;
x=a / b;
```
Укажіть результат виконання операції (чому дорівнює x).
2. Виконується фрагмент програми
```
int a=7, b=3, x;
x=a & b;
```
Укажіть результат виконання операції (чому дорівнює x).
3. Виконується фрагмент програми
```
int m = 5;
m - = 3;
```
Укажіть результат виконання операції (чому дорівнює m).
4. Виконується фрагмент програми:
```
int  a = 5, b =3, x ;
  x = a % b;
```
Укажіть результат виконання операції:
5. У мові С(С++) для обчислення тригонометричної функції tgX використовується функція:
    1. sin (x) / cos (x)
    2. tg (x)
    3. tan (x)
6. У мові С(С++) використовується стандартна функція pow (x,y). Це:
    1. Функція для обчислення залишку від розподілу x на y
    2. Функція для зведення x у ступінь y
    3. Такої функції в С немає

Лекція 4

ОПЕРАЦІЇ І ВИРАЗИ У МОВІ С (С++)

Основні питання лекції.

6. Операція приведення типу.
7. Операції інкремент і декремент
8. Операції відношення (порівняння).
9. Логічні операції.
10. Порозрядні (побітові) операції.
11. Операція умова ?: (Умовна операція)
12. Операція sizeof.

1. Операція приведення типу.

Присвоєння – єдина операція, що міняє значення (вміст) одного з своїх операндів (не рахуючи операцій інкремента і декремента, які будуть розглянуті нижче).

У арифметичному виразі можуть бути присутні операнди різних типів – як цілі так і дійсні. Крім того, і ті і інші можуть мати різну довжину (chort, long). В той же час *обидва операнди* будь-якої арифметичної операції повинні мати *один і той же тип*.

У мові С (С++) при обчисленні значення такого виразу відбувається автоматичне приведення типів. Суть приведення типів наступна.

На кожному кроці обчислення значення виразу виконується *одна операція* над *однією парою операндів.* Якщо їх тип різний, то операнд меншого рангу приводиться до типу більш високого *рангу*. Під *рангом типу* розумітимемо діапазон значень, який підтримується даним типом. За збільшенням рангу типи слідують в наступному порядку.

char
short
int, long
float
double
long double

Правила, які використовуються для автоматичного приведення типів при обчисленні значення арифметичного виразу, наступні.

1. Всі змінні типу char, short int перетворяться в int.

2. Для будь-якої пари операндів вони перетворюються до одного типу. Наприклад, якщо один з операндів double, то і інший перетвориться в double.

3. У операторі присвоєння кінцевий результат наводиться до типу змінної в лівій частині оператора. При цьому ранг типу може як підвищуватися, так і знижуватися.

Можливе і *примусове* приведення типу, яке виконується за допомогою операції приведення типу, яка має наступну структуру:

(*тип*) вираз;

Тут *тип* – один із стандартних (вбудованих) типів даних мови С (С++).
Операція може застосовуватися до будь-якого операнду у виразі.

Наприклад, якщо ви хочете, щоб результат ділення змінної X типу int на 2 мав тип float, запишіть у програмі

(float) X/2;

Щоб уникнути помилок в обчисленнях потрібно точно собі уявляти, що при цьому відбувається, і при необхідності застосовувати операцію приведення типу, яка явно перетворить операнд в той або інший тип.

Іноді вважають, що у виразі всі операнди наперед (при трансляції програми) приводяться до типу, що має найбільший ранг, а вже потім проводиться обчислення виразу. Це не так! Приведення типів виконується послідовно для кожної поточної пари операндів. Розглянемо приклад.

Необхідно обчислити об'єм кулі за формулою:

$$V = 4/3*Pi*R^3;$$

де R – радіус кулі.
Недовго думаючи записуємо цю формулу на мові С (С++).

```
# include < iostream. h>
```

```
 void main()
  {
      float V;
      float R = 1.0;
      const float Pi=3.1415;
      V = 4 / 3 * Pi * R * R * R;
      .....
}
```

Усі операції в правій частині виразу мають однаковий пріоритет. Отже, обчислення значення виразу проводиться послідовно зліва направо. На першому кроці проводиться ділення 4/3. Результат ділення – ціле число, тобто 1. Далі ця одиниця перетвориться в дійсне число – 1.0. А далі все піде за правилом. В результаті одержимо:

V = 1.0 *3.1415* 1.0* 1.0 *1.0 = 3.1415.

Насправді повинно бути наступне (відповідно до правил математики):

V = 1.333 *3.1415* 1.0* 1.0 *1.0 = 4.185.

Якщо в програмі останній рядок записати таким чином

V = (float)4 / 3 * Pi * R * R * R;

то результат обчислення буде вірним, тобто 4.185.

2. Операції інкремент і декремент

Ці операції властиві тільки мові С (С++). Одномісні (унарні) операції інкременту і декременту – відповідно збільшують *або* зменшують *свій* операнд (обов'язково змінну) на одиницю. Вони змінюють значення самої змінної, тобто є прихованою формою операції присвоєння. Іноді ці операції застосовують як самостійний оператор:

i ++; i--;

що еквівалентне запису

i = i + 1; i = i – 1;

Але ці операції можуть використовуватися і у виразах

sum = sum + x * ++i;

Інкремент і декремент реалізуються у двох формах: префіксної ++i і постфіксної i++. Особливості виконання операцій різних форм:

++i (--i) – змінна i спочатку збільшується (зменшується) на одиницю, а потім її значення використовується у виразі.

i++ (i--) – значення змінної i спочатку використовується у виразі і тільки після обчислення виразу змінна збільшується (зменшується) на одиницю.

Приклад 1.

```
void main()
{
  int a, i = 0;
a = ++i;
…..
}
```

У результаті виконання операції значення a буде дорівнюватися 1.

Приклад 2.

```
Void main()
{
  int a, i = 0;
a = i++;
…..
}
```

У результаті виконання операції значення *a* буде дорівнюватися 0.

2. Операції відношення

Операція відношення порівнює між собою два значення. Значення можуть бути як стандартних типів мови С++, наприклад char, int або float, так і типів, визначених користувачем. Порівняння встановлює одне з трьох можливих відносин між змінними: рівність, більше або менше. Результатом порівняння є значення істина (true) або неправда (false).

Операція	**Назва**
>	більше
<	менше
= =	рівно
!=	не рівно
>=	більше або рівно
<=	менше або рівно

Операції відношення мають нижчий пріоритет ніж арифметичні операції. Це визначає, що вираз 12>10+1 розглядається як вираз 12>(10+1).

4. Логічні операції.

Мова С++ надає три логічні операції для об'єднання або зміни існуючих виразів. Цими операціями є:

&& - логічна операція І (AND);

|| - логічна операція АБО (OR);

! - логічна операція НІ (NOT).

Логічні операції в С (С++) відповідають класичним логічним операціям в обчислювальній техніці. Результат виконання цих операцій відповідає таблиці істинності.

X	Y	X && Y	X || Y	!X	X xor Y
1	1	1	1	0	0
1	0	0	1	0	1
0	1	0	1	1	1
0	0	0	0	1	0

Операція xor називається операцією що «виключає АБО». У мові С (С++) немає знаку логічної операції xor. Вона може бути реалізована за допомогою інших

логічних операцій (AND, OR, NOT). Проте ця операція присутня в порозрядних операціях, які будуть розглянуті нижче.

Пріоритет виконання логічних операцій (в порядку його убування):

! (NOT), && (AND), || (OR).

Операції виконуються зліва направо.

Розглянемо приклад.

```
void main ()
{
  bool a = true;
  bool b = true;
  bool c = false;
  bool y;
    y= a && b || (a || c) || !c;
.....
}
```

При обчисленні значення *y* порядок обчислень буде наступним. На першому кроці буде обчислюватись вираз, що стоїть в круглих дужках. Він має щонайвищий пріоритет. Значення виразу (a || c) матиме true (оскільки a = true, c = false) відповідно до таблиці істинності. На другому кроці буде обчислено !c. Оскільки c = false, то !c = true. Потім буде обчислено значення виразу a && b. Воно прийме значенні true, оскільки a = true і b = true. А потім зліва направо виконуватимуться операції OR. Перша операція || (OR) дасть значення true. Друга операція true || true дасть значення true. Отже, в результаті виконання логічних операцій *y* прийме значення true.

Логічні операції &&(AND), ||(OR) і операції відношення використовуються для запису складних умовних виразів.

Приклад 1. Записати умову, що точка *x* не належить відрізку ab, тобто a>x>b.

У С (С++) це запишеться таким чином:

```
x < a || x > b.
```

Приклад 2. Записати умову, що точка *x* належить відрізку ab, тобто a<x<b.

```
x >a && x < b.
```

У логічних виразах можна використовувати круглі дужки, які мають щонайвищий пріоритет. Крім того, круглі дужки дозволяють зробити вираз більш зрозумілим і зручним для читання.

Особливість виконання логічних операцій && (AND) і || (OR) полягає в наступному.

Якщо при обчисленні результату операції (вир.1) && (вир.2) значення лівого виразу (вир.1) буде false, то значення (вир.2) не обчислюється, оскільки його значення на результат операції не зробить ніякого впливу. Результат операції буде false.

Це торкається операції (вир.1) || (вир.2). У цьому випадку другий операнд (вир.2) не обчислюється, якщо значення лівого операнда (вир.1) приймає значення true.

5. Порозрядні (побітові) операції

Ці операції можна виконувати над будь-якими цілочисельними операндами. Останні розглядаються в цьому випадку як набір окремих бітів. Результатом виконання порозрядних операцій буде ціле число.

До порозрядних операцій відносяться:

& - AND;

| - OR;

^ - XOR;

~ - NOT;

<< - зсув вліво;

>> - зсув управо.

Порозрядні операції. При виконанні порозрядних операцій кожний розряд (біт) одного операнда комбінується з однойменним розрядом (бітом) іншого операнда, даючи розряд результату.

При виконанні одномісної порозрядної операції ~ (NOT) – бітом результату є інверсія відповідного біта операнда.

Приклад. Необхідно встановити старший розряд змінної типу char рівним нулю.

Для вирішення цієї задачі використовуємо порозрядну операцію & (AND).

```
 void main ()
{
 char ch = ‘A’;
 ch = ch & 127;
.....
}
```

Величина ‘A’ в двійковому коді записується таким чином:

1 1 0 0 0 0 0 1

Число 127 в двійковій системі числення запишеться таким чином:

0 1 1 1 1 1 1 1

Результат виконання порозрядної операції & (AND).

```
‘A’     1 1 0 0 0 0 0 1
        &
 127    0 1 1 1 1 1 1 1
        ---------------
        0 1 0 0 0 0 0 1
```

Операція виконується над кожним розрядом окремо відповідно до таблиці істинності.

Порозрядні операції зручні для організації зберігання в стислому вигляді про стан on / of (включено / вимкнено). В одному байті можна зберігати 8 таких ознак (прапорів). Якщо змінна ch є сховищем таких ознак, то можна перевірити, в якому стані знаходиться кожний з розрядів байта.

Це робиться за допомогою операції & (AND) і підбором відповідної константи в двійковій системі числення.

Наприклад. Перевірити, чи знаходиться прапор, що міститься в третьому розряді змінної ch, в стані on. Ця перевірка ґрунтується на двійковому представленні числа 4:

4 ----------→ 0 0 0 0 0 1 0 0.

Записавши логічний вираз

Ch & 4 == 4

і перевіривши істинно воно або помилково, визначаємо стан цього розряду.

Якщо необхідно перевірити 7-й розряд, то підбираємо константу так, щоб одиниця була в 7-у розряді байта, а в останніх розрядах були б нулі.

0 1 0 0 0 0 0 0 ----------→ 64.

Тоді логічний вираз матиме вигляд: Ch & 64 == 64.

Операції зсуву. Операції зсуву >> і << застосовуються тільки до цілочисельних операндів (змінних). В результаті виконання цих операцій зсовуються вліво (вправо) всі розряди лівого операнда на число позицій, визначених виразом, що стоїть праворуч від знаку операції. Розряди, що звільняються, заповнюються нулями. Форма операції зсуву має вигляд:

value >> число позицій зсуву;

value << число позицій зсуву.

Приклад.

```
    void main ()
{
  int x = 9;//  у двійковій системі  9 -> 0 0 0 0 1 0 0 1
  x = x << 3;  // результат виконання операції  x = 0 1 0 0 1 0 0 0
  x = x >> 5;  // результат виконання операції  x = 0 0 0 0 0 0 1 0
.....
}
```

Використовуючи скорочений запис операцій зсуву фрагмент програми матиме вигляд:

```
    void main ()
{
  int x = 9;//  у двійковій системі  9 -> 0 0 0 0 1 0 0 1
  x <<= 3;  // результат виконання операції  x = 0 1 0 0 1 0 0 0
  x >>= 5;  // результат виконання операції  x = 0 0 0 0 0 0 1 0
.....
}
```

При використанні операцій зсуву необхідно мати на увазі наступні особливості їх виконання.

При *зсуві вліво* розряди першого операнда зсовуються (переміщаються) вліво (у бік старших розрядів) на задане другим операндом число позицій. Старші розряди, що виявилися за межами розрядної сітки втрачаються. Молодші розряди заповнюються нулями.

Результат *зсуву вправо* залежить від того, чи є лівий операнд знаковим або без знаковим. Розряди операнда переміщаються вправо на задане число позицій. Молодші розряди втрачаються. Якщо операнд – ціле із знаком (signed), відбувається розширення знакового (старшого) розряду:

- якщо операнд позитивний (його знак дорівнює нулю), то позиції, що звільнилися, заповнюються нулями;

- якщо операнд негативний (знак операнда - одиниця), то розряди, що звільнилися, заповнюються одиницями.

Приклад.

```
    void main ()
{
  int x = 9;//  у двійковій системі  9 -> 0 0 0 0 1 0 0 1
signed int y = -9; //  у двійковій системі  -9 -> 1 1 1 1 0 11 1
  x >>= 3;  // результат виконання операції  x = 0 0 0 0 0 0 0 1
  y >>= 3;  // результат виконання операції  y = 1 1 1 1 1 1 1 0
.....
}
```

Зауваження. *Зсув вліво* еквівалентний *множенню* на відповідний ступінь двійки, а *зсув управо – діленню* на ступінь двійки. Наприклад.

```
namber = namber <<4;
```

умножає namber на 2^4 =16, тобто на 16.

6. Операція умова ?:

Операція умова – єдина операція мови С (С++), в якій беруть участь три операнди. Ця операція має вигляд:

```
(вир.1) ? (вир.2) : (вир.3);
```

Операція виконується таким чином. Обчислюється вираз (вир.1). Якщо цей вираз має значення true, то обчислюється вираз (вир.2). Результатом операції буде значення виразу (вир.2).

Якщо значення (вир.1) – false, то обчислюється вираз (вир.3) і його значення буде результатом операції. У будь-якому випадку обчислюється тільки один з виразів: (вир.2) або (вир.3).

Приклад 1. Визначити найбільше з двох чисел *a i b.*

```
Max_ab = (a > b) ? a : b;
```

Приклад 2. Визначити абсолютну величину *x*:

```
abs_x = (x > 0) ? x : -x;
```

7. Операція sizeof

Операція sizeof (результат операції – розмір в байтах) має дві форми:

```
sizeof (тип);
sizeof (вираз).
```

Результатом цієї операції є цілочисельне значення величини типу або виразу в байтах. При використанні другої форми значення виразу не обчислюється, а визначається величина виразу в байтах.

Приклади запису операції:

```
sizeof (short  int);
sizeof (x);
```

Тип void не може використовуватися в операції sizeof (тип).

Інші операції будуть розглянуті у міру їх використовування.

Далі ми побачимо, що деякі знаки операцій мають декілька смислових значень. Так знак операції & має два значення:

- в бінарній операції (два операнди) – це порозрядне AND;

- в унарній операції (один операнд) – це взяття адреси.

Питання для само контролю.

1. На мові С (С++) запис **A** і **a** – це:
    1. Ім'я однієї і тієї ж змінної
    2. Імена двох змінних **A** і **a**
    3. Все залежить від контексту програми
2. Ідентифікатор у мові С(С++) - це:
    1. Довільна послідовність латинських букв (великих і малих), цифр і символу підкреслення , що починається з букви або символу підкреслення
    2. Довільна послідовність латинських букв (великих і малих), цифр і, що починається з букви
    3. Довільна послідовність латинських букв (великих і малих), спеціальних символів і цифр ,що починається з букви
3. У мові С(С++) для визначення дійсних змінних використовується наступний запис:
    1. x : real;
    2. x -> float;
    3. float x;
4. Складовий оператор у мові С(С++) записується в середині:
    1. Begin end;
    2. ( )
    3. { }

5. У мові С (С++) базових типів даних:
    1. 5
    2. 3
    3. 4
6. Змінні типу double і float у мові С(С++) – це:
    1. Ціле число і число з плаваючою крапкою
    2. Це числа з плаваючою крапкою
    3. Це число з плаваючою крапкою і ціле число
7. У двійковій системі числення записано число 1001. У десятковій системі – це:
    1. 1001
    2. 9
    3. 17
8. У С(С++) операція ділення записується так:
    1. A : B
    2. A / B
    3. A \ B
9. У мові С(С++) літерний тип даних визначається так:
    1. char
    2. void
    3. real
10. Виконується фрагмент програми на мові С(С++)

```
int  a;          a = 4/3;
```

Укажіть результат виконання операції:

11. Виконується фрагмент програми на С

```
int  a, i=0;
a = (i++) + (++i);
```

Укажіть результат виконання операції:

12. Виконується фрагмент програми:

```
int  a = 5, b =3, x ;
 x = a % b;
```

Укажіть результат виконання операції:

13. У програмі на C(C++) записаний вираз &x. Це:
    1. Операція логічного AND
    2. Операція порозрядного AND
    3. Операція взяття адреси
14. Виконується фрагмент програми на мові C (C++) :

```
int x = 1;
x *= 3;
```

    Яке значення прийме змінна x:
15. На C (C++) записаний вираз (float) x/2. Це:
    1. Визначення змінної x
    2. Операція «приведення типу»
    3. Така конструкція в C (C++) не можлива
16. У мові C(C++) для обчислення тригонометричної функції tgX використовується функція:
    4. sin (x) / cos (x)
    5. tg (x)
    6. tan (x)
17. У мові C(C++) використовується стандартна функція pow (x,y). Це:
    4. Функція для обчислення залишку від розподілу x на y
    5. Функція для зведення x у ступінь y
    6. Такої функції в C немає

Лекція 5
УПРАВЛЯЮЧІ КОНСТРУКЦІЇ МОВИ C (C++)

Мета лекції. Вивчити оператори умовного і безумовного переходу, особливості їх конструкції і виконання.

Основні питання лекції.

1. Структури потоку управління.
2. Умовний оператор if …else.
3. Оператор вибору switch.
4. Безумовний оператор.

1. Структури потоку управління.

Мову C (C++) називають мовою структурного програмування, не тільки тому, що в цій мові є поняття структури даних, а в основному завдяки структурованому

потоку управління обчисленнями. Потік управління можна визначити як алгоритм переходу від поточного оператора до наступного. Існує три основні потоки управління.

1. *Послідовна структура* – наступним виконується оператор, розташований безпосередньо після поточного.
2. *Структура вибору* – має декілька операторів; залежно від виконання деякої умови вибирається тільки один з них, інші ігноруються.
3. *Структури повторення* – поточний оператор виконується деяке число разів підряд до тих пір, поки не буде задоволена умова його завершення.

З цих трьох структур можна будувати скільки завгодно складні конструкції, оскільки вони підкоряються правилу *суперпозиції.* Це означає, що на місце будь-якого оператора деякої конструкції можна підставити будь-яку конструкцію. При цьому, іноді, останню необхідно заключити в операторні дужки. У мові С (С++) – це фігурні дужки { }. Будь-яка послідовність операторів, заключна у фігурні дужки, з погляду потоку управління, вважається єдиним оператором.

Раніше розглянуті приклади програм реалізовували послідовну структуру потоку управління. Такі програми називаються лінійними. Вони володіють досить скромними можливостями. Справжня програма повинна «ухвалювати рішення», тобто змінювати послідовність виконання операторів залежно від виконання деяких умов. Прикладом може служити розгалуження в процесі обчислень при виконанні (або не виконанні) деякої умови.

Засобами мови С (С++), за допомогою яких реалізується обчислювальний процес, що розгалужується, є *умовні оператори.* Розглянемо їх.

2. Умовний оператор if …else.

Умовний оператор реалізує структуру вибору. Форма запису оператора має вигляд:

if (умова) ореператор1; else оператор 2;

Якщо умова виконується (приймає значення «істина» - true), то виконується *оператор 1*. Якщо умова не виконується (приймає значення «неправда» - false) – виконується *оператор 2*.

Схема алгоритму виконання оператора if …else приведена на Рис. 1.

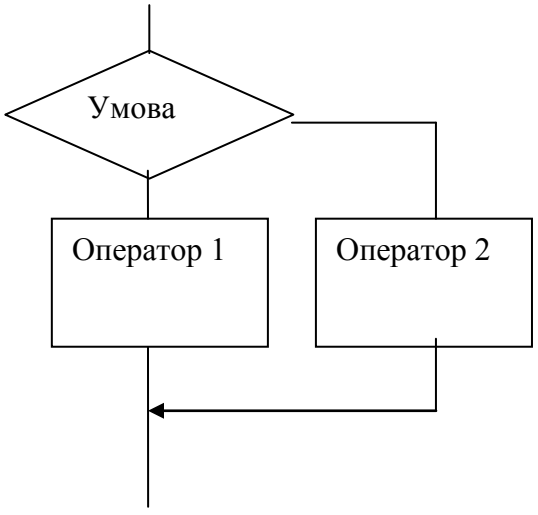


Рис.1. Схема алгоритму виконання оператора

Приклад запису умовного оператора.

```
if (a > b) max_ab = a; else  max_ab = b;
```

Як було сказано вище, замість одиночного оператора завжди можна поставити *блок* з декількох операторів, заключного в операторні (фігурні) дужки. Іншими словами, можлива наступна синтаксична форма запису оператора:

```
if (умова)
{оператори блоку if};
else
{оператори блоку else};
```

Приклад. Є два числа a і b. Якщо a > b, то поміняти їх місцями; інакше – їм присвоїти нулі.

```
if (a > b)
 {
   temp = a;
  a = b;
  b = temp;
 }
else
 {
  a = 0;
  b = 0;
 }
```

При записі оператора if друга його частина, тобто конструкція else, може бути відсутня. Тоді, якщо умова не виконується (приймає значення false), виконується наступний оператор програми.

Приклад.

```
if (a > b) {temp = a;  a = b;  b = temp; }
//… наступний оператор програми.
```

Відповідно до правила суперпозиції можна будувати вкладені if …else. Тобто в операторі

```
if (умова) ореператор1; else оператор 2;
```

замість опер.1 і опер.2 можна ставити будь які оператори мови С (С++), у тому числі і оператор if....else.

Наприклад.

```
if (умова) опер.1;
else  if (умова) опер.2;
else  if (умова) опер.3;
.....
else  опер.N;
```

У такій конструкції умови операторів if перевіряються з верху вниз. Як тільки яка-небудь з умов приймає значення «істинно», виконується оператор, наступний за цією умовою, а вся решта конструкції буде пропущена.

Розглянемо приклад. Задано три числа a, b, c. Необхідно визначити найбільше з них.

Фрагмент програми матиме вигляд.

```
max = c;
if ( a > b )
{
  if ( a > c )
  max = a;
}
else
 if ( b > c )
 max = b;
```

Умова оператора if може бути скільки завгодно складним виразом. Воно може містити операції відношення (операції порівняння), арифметичні операції, логічні операції, тощо. Але частіше за все воно містить операції порівняння. Вираз вважається істинним (true), якщо умова виконується, і помилковим – інакше.

Приклади запису виразів.

```
 if ( x )… -  // якщо x не дорівнює нулю;
 if ( !x ) … - //  якщо x  дорівнює нулю;
 if ( b = = з ) … - якщо b дорівнює c;
 if ( b != з )… - якщо b  не дорівнює c;
 if (x >= a && x <= b)… - x  належить відрізку ab.
```

Зауваження. При записі виразів досить часто замість знака = = (дорівнює) записують знак = (присвоєння). Не слід забувати цієї різниці.

4. Оператор вибору switch.

Часто виникають ситуації, коли обчислення необхідно проводити не по двох гілках, як в операторі if …else, а по декількох (>=3).

У цьому випадку можна використовувати послідовно вкладені оператори if …else. Але використовування такої конструкції робить програму практично не читаною. Для подібних випадків у мові C (C++) існує спеціальний оператор вибору switch. Оператор switch записується таким чином.

```
switch (вираз)
  {
   case  константний вираз 1: група операторів;
   case  константний вираз 2: група операторів;
    .....................................
   default: група операторів;
  }
```

Конструкція switch є своєрідним перемикачем. Працює він таким чином. Спочатку обчислюється значення «виразу» в операторі switch. Потім набуте значення послідовно порівнюється з кожним з константних виразів. При збігу значень управління передається на відповідну групу операторів, тобто відбувається

виконання відповідності операторів, наступних за двокрапкою. Якщо значення «виразу» не співпадає ні з одним з константних виразів, то управління передається на групу операторів, визначених ключовим словом default, або на наступний після конструкції switch оператор програми, якщо група default відсутня.

Під *групою операторів* мається на увазі просто один оператор або декілька довільних операторів. Група тут не зобов'язана бути блоком, тобто брати її в операторні дужки не потрібно.

Розглянемо приклад.

```
#include <iostream.h>
void main()
{
        char ch;
        cout << "Vvedit zagolovnu bucvu alfavitu :"<<endl;
        cin >> ch;
        if( ch >='A' && ch<='Z')
                switch (ch)
                {
                 case 'A':
                        cout<<"Avramenko"<<endl;
       case 'B':
                        cout<<"Bugaenko"<<endl;
       case 'V':
                        cout<<"Voloshin"<<endl;
                 case 'G':
                        cout<<"Gogol"<<endl;
                 default:
                        cout<<"Zuev"<<endl;
                }
         else
                cout<<"Vvedite zagolovnu bucvu !"<<endl;
}
```

Після запуску програми і введення букви B на екрані з'явиться результат:

```
Vvedit zagolovnu bucvu alfavitu :
B
Bugaenko
Voloshin
Gogol
Press any key to continue
```

Ми бачимо, що виконалися всі оператори, починаючи з мітки B.

У конструкції switch є одна особливість. Якщо знайдена мітка case, співпадаюча із значенням виразу, що перевіряється, то виконується група операторів даного case . Після чого управління передається наступному по порядку case . В результаті будуть виконані всі оператори в конструкції switch.

Якщо це небажано, то в кінці групи операторів case необхідно поставити оператор break. Він перериває виконання конструкції switch і передає управління наступному оператору програми.

Приклад використовування оператора break.

```
#include <iostream.h>
void main()
{
        char ch;
        cout << "Vvedit zagolovnu bucvu alfavitu :"<<endl;
        cin >> ch;
        if( ch >='A' && ch<='Z')
                switch (ch)
                {
                        case 'A':   cout<<"Avramenko"<<endl;break;
                        case 'B':   cout<<"Bugaenko"<<endl;break;
                        case 'V':   cout<<"Voloshin"<<endl;break;
                        case 'G':   cout<<"Gogol"<<endl;break;
                        default:    cout<<"Zuev"<<endl;
                }
        else
                cout<<"Vvedite zagolovnu bucvu !"<<endl;
}
```

Після запуску програми і введення букви В на екран буде виведено тільки одне прізвище.

```
Vvedite zagolovnu bucvu alfavitu :
B
Bugaenko
Press any key to continue
```

5. Безумовний оператор.

Безумовний перехід до якого-небудь оператора програми здійснюється за допомогою оператора безумовного переходу. Оператор має наступну структуру:

```
goto мітка;
```

Мітка є ідентифікатором з розташованим за ним символом двокрапки :

```
A:, label: .
```

Мітками позначають який-небудь оператор, на який повинен бути здійснений безумовний перехід.

Приклад фрагмента програми з використанням оператора goto.

```
#include <iostream.h>
void main()
{
        // опис змінних
        .............
        if (x>=0)
        {
        // виконання якісь операторів прграми
        .............
        goto A;
        }
```

.....

A: cout <<"Sum = "<< S << endl;

}

Як тільки виконання програми досягне оператора goto **A**, управління буде передано оператору cout, оскільки він помічений міткою A.

Висновок. Умовний оператор реалізує структуру вибору. Форма запису оператора має вигляд:

if (умова) ореператор1; else оператор 2;

Якщо умова виконується (приймає значення «істина» - true), то виконується *оператор 1*. Якщо умова не виконується (приймає значення «неправда» - false) – виконується *оператор 2*.

Питання для самоконтролю.

1. В операторі **if** (умова) необхідно перевірити умову на рівність змінної нулю. Перевірку умови необхідно записати так:
   1. if (x)
   2. if (!x)
   3. if (x=0)
2. Записаний оператор

   if(a>b) x=a else x=b.

   Скільки помилок зроблено при запису оператора:
3. Поточний оператор виконується де - яке число раз дотиpоки не буде задоволена умова його за вершення. Це:
   1. Послідовна структура
   2. Структура вибору
   3. Структура повторення
4. Є два числа **a** і **b.** Якщо **a>b,** то поміняти їх місцями. Який з операторів записаний правильно:
   1. if (a>b) { c=a; a=b; b=c };
   2. if (a>b) { c=a; b=c; a=b };
   3. if (a>b) { b=c; a=b; c=a };
5. В операторі if перевіряється умова if ((x>=a) &&(x<=b)). Перевірка цієї умови означає:
   1. **x** належить відрізку a,b
   2. **x** не належить відрізку a,b
   3. Так записувати умову не можна
6. Укажіть правильний запис оператора умовного переходу:
   1. IF(x>1) THEN y=x; ELSE y:=0;
   2. IF(x>1)y=x ELSE y=0;
   3. IF(x>1)y=x; ELSE y=0;
7. У загальному вигляді оператор умовного переходу має вигляд IF(умова)S1;ELSE S2; S1 і S2 це:
   1. Будь-які оператори мови С
   2. Оператори присвоювання
   3. Спеціальні оператори мови С
8. У програмі на мові С записаний оператор:

   IF(умова) S;

   Якщо умова не виконується, то який оператор буде виконаний:
   1. Оператор S
   2. Наступний оператор програми
   3. Відбудеться перехід на кінець програми
9. У загальному вигляді оператор умовного переходу має вигляд:

   IF(умова) S1; ELSE S2;

   Якщо умова не виконується, то який оператор буде виконаний:

1. Оператор S1
2. Оператор S2
3. Наступний оператор програми

10. Укажіть правильний запис оператора безумовного переходу:
    1. GOTO A1;
    2. GOTO A1,A2,A3;
    3. GOTO 1;
11. Записаний оператор:
    if(X>0){ X = X-1; Y= 0; } else Y= Y+1
    Скільки помилок у приведеному операторі?
12. Укажіть на можливість такого запису оператора
    if(1< X < 3) S1; else S2;
    1. Можливо
    2. Можливо в окремих випадках
    3. Не можливо
13. Яке значення матиме змінна Z після виконання операторів:
    X=1; Y= 1; Z =0;
    if(X>0) if(Y>0) Z=1; else Z=2;
14. В операторі if необхідно записати умову, що "X не належить відрізку [A,B]". Укажіть на правильний запис цієї умови:
    1.if(A > X > B)
    2.if((X>A)&&(X>B))
    3.if((X<A)and(X>B))

Лекція 6
ОПЕРАТОРИ ЦИКЛУ

Мета лекції. Вивчити оператори циклу, особливості їх конструкції та виконання.

Основні питання лекції.

1. Оператор циклу while
2. Оператор циклу do… while
3. Оператор циклу for…
4. Оператори break и continue

Якщо обчислювальний процес містить багатократні обчислення по одних і тих же математичних залежностях, але для різних значень змінних, то такий процес називається *циклічним*.

Наприклад. Необхідно обчислити значення виразу

$$y = ax + \sin 2x$$

для $x = x_1, x_2, x_3, x_4, x_5$.

У даному прикладі обчислення проводяться по одній і тій же формулі, але для різних значень змінної x. При $x = x_1$ обчислюється y_1, при $x = x_2$ обчислюється y_2 і т.д. Обчислення будуть повторені 5 разів.

Багато разів повторювані частини такого процесу називають *тілом циклу*, а змінну, яка змінюється при виконанні дій тіла циклу – *параметром циклу*. В даному прикладі

x – параметр циклу;

y = ax + sin2x – тіло циклу.

Алгоритм циклічних структур обов'язково повинен включати наступні дії.

1. Підготовка до циклу – присвоєння початкових значень параметру циклу.
2. Перевірку умови виконання тіла циклу.
3. Тіло циклу – дії, які повторюються в циклі при різних значеннях параметра циклу.
4. Зміна значення параметра циклу.

У залежності від того, яким чином здійснюється модифікація (зміна) параметра циклу і управління циклом, в мові С (С++) для організації циклічних ділянок програм є наступні оператори.

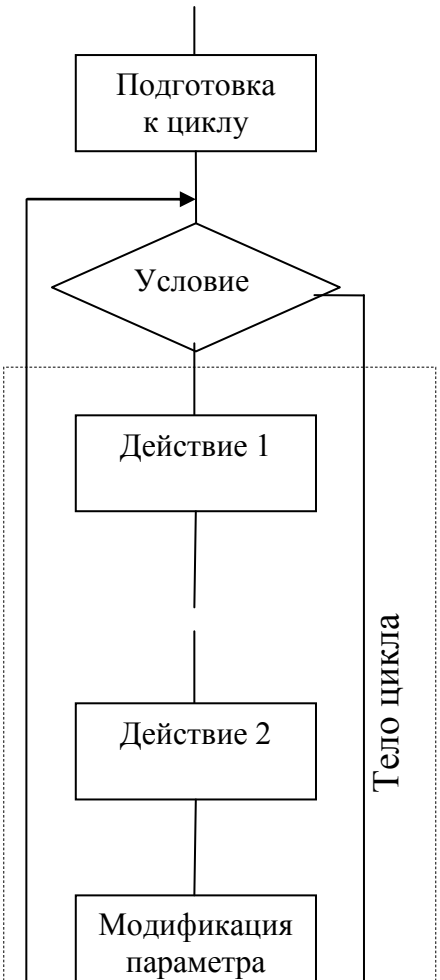
1. Оператор циклу з передумовою – оператор while.
2. Оператор циклу з післяумовою – оператор do… while.
3. Оператор циклу з параметром – оператор for…

Розглянемо ці оператори.

1. Оператор циклу while

Цей оператор є самим загальним. Він широко використовується при програмуванні задач, коли число повторень циклу наперед не відоме.

Конструкція повторення (цикл) в схемах алгоритмів має вигляд.



Вихід з циклу

Рис.7.1. Конструкція повторення

Оператор циклу while записується таким чином.

while (Умова) Оператор або блок;

де

while – «поки», службове слово.

умова – як і в інших операторах управління обчислювальним процесом, є просто виразом.

Оператор циклу while виконується таким чином. Параметру циклу присвоюється початкове значення. Потім перевіряється умова. Тіло циклу виконується до тих пір, поки умова приймає значення «істина», тобто умова виконується. Коли умова прийме значення «неправда», відбудеться вихід з циклу.

У операторі while *спочатку* перевіряється умова, а потім виконується тіло циклу. Звідси і назва оператора – цикл з передумовою.

Приклад. Розробити алгоритм і програму обчислення значення функції:

$y = ax + \sin(\pi x)$,

якщо x змінюється від початкового значення x_n до кінцевого x_k з кроком h.

Типи і структури даних

a – константа, наприклад, a = 2 – тип int;

π – константа, позначимо pi = 3.14 – тип float;

x_n – проста змінна, позначимо її xn – тип float;

x_k – проста змінна, позначимо її xk – тип float;

h – проста змінна, позначимо її dx – тип float;

x – проста змінна, позначимо її x – тип float;

y – проста змінна, позначимо її біля – тип float.

Схема алгоритму обчислень матиме наступний вигляд (Рис.7.2).

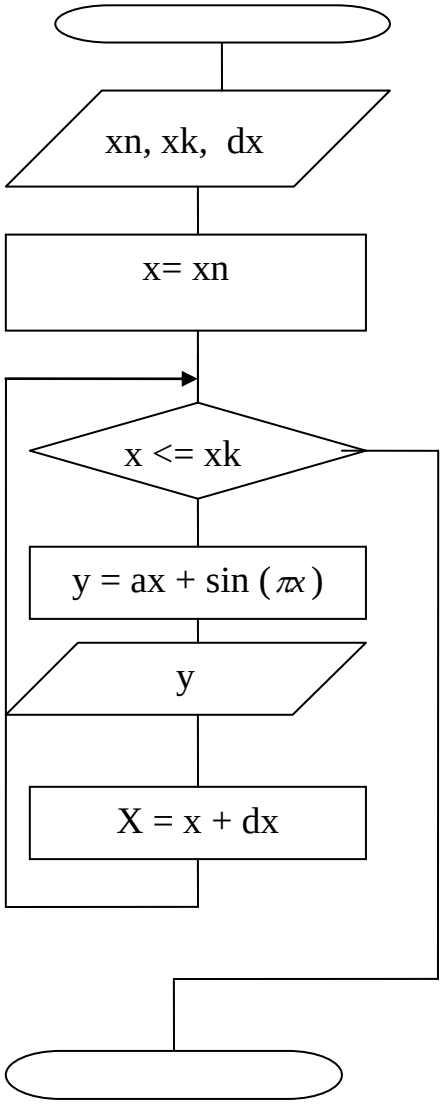


Рис.7.2. Схема алгоритму обчислень

Програма буде мати вигляд.

```
#include <iostream.h>
#include <math.h>
void main()
{
        const int a=2;
        const double pi=3.14;
        double xn, xk, dx, x, y;
        cout<<"Vvedite dannye xn,xk,dx:"<<endl;
        cin>>xn>>xk>>dx;
        cout<<endl;
        x = xn;
        while(x <= xk)
        {
                y = a*x + sin(pi*x);
                cout<<"y= "<<y<<endl;
                x=x+dx;
        }
```

```
}
```
Результати виконання програми приведені нижче.

```
Введіть значення даних xn, xk, dx :
0.1 1.0 0.1

y= 0.508866
y= 0.987528
y= 1.40874
y= 1.75086
y= 2
y= 2.15135
y= 2.20967
y= 2.18882
y= 2.11038
y= 2.00159
Press any key to continue
```

2. Оператор циклу do… while

Цей оператор також використовується в тих випадках, коли число (кількість) повторень циклу наперед не відоме, але є необхідність виконати тіло циклу хоча б один раз. В цьому операторі умова виконання циклу перевіряється після виконання тіла циклу.

Синтаксис оператора наступний:

```
do
  {
  Один оператор або блок;
   }
while (умова);
```

Схема алгоритму виконання оператора do…while ,буде мати наступний вигляд.

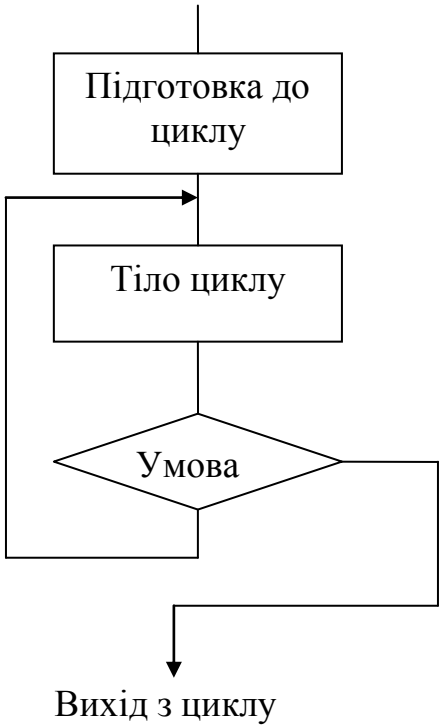


Рис.7.3. Схема алгоритму виконання оператора do…while

При запису оператора фігурні дужки необов'язкові. Але вони ставляться завжди для кращої читаності програми. Оператор do…while – це оператор циклу з післяумовою. Яка б умова не стояла в кінці оператора, тіло циклу (набір операторів у фігурних дужках) виконається обов'язково хоча б один раз.

Оператор виконується таким чином. Параметру циклу присвоюється початкове значення і виконується тіло циклу. Потім перевіряється умова виходу з циклу. Якщо умова «істина», то управління передається на початок циклу. Якщо «неправда», то цикл завершується і управління передається наступному оператору програми.

Запишемо програму для попереднього прикладу.

```
#include <iostream.h>
#include <math.h>
void main()
{
        const int a=2;
        const double pi=3.14;
        double xn, xk, dx, x, y;
        cout<<"Vvedite dannye xn,xk,dx:"<<endl;
        cin>>xn>>xk>>dx;
        cout<<endl;
        x = xn;
 do
        {
        y = a*x+sin(pi*x);
        cout<<"y= "<<y<<endl;
```

```
        x=x+dx;
        }
while (x<=xk);
}
```

Після запуску програми і введення початкових даних н екрані дисплея буде знаходитись така інформація.

```
Vvedite dannye xn,xk,dx:
0.1 1.0 0.1

y= 0.508866
y= 0.987528
y= 1.40874
y= 1.75086
y= 2
y= 2.15135
y= 2.20967
y= 2.18882
y= 2.11038
y= 2.00159
Press any key to continue
```

Розглянемо ще один приклад використання операторів while і do…while .

Скласти програму обчислення коріння рівняння f (x) = 0 з точністю ε. Відрізок локалізації коріння [a,b] – відомий. Для обчислення коріння рівняння f (x) = 0 використовуємо метод простої ітерації. Він полягає в тому, що по і- му наближенню коріння знаходиться і + 1 наближення за формулою:

$$x_{i+1} = f(x_i), \quad i = 0, 1, 2, 3 .$$

Процес продовжується до тих пір, поки відносна погрішність для двох послідовних наближень не стане менше ε.

$$| (x_{i+1} - x_i) / x_i | < \varepsilon.$$

Для початкового наближення коріння x0 приймається, що

$$x_0 = (a,b) / 2.$$

Складемо програму обчислення коріння рівняння f (x) = 0 при заданому початковому наближенні коріння x0 і точності ε.

$$f(x) = x - \sin x - 0.25 = 0; \; x0 = 1.25; \; \varepsilon = 10\text{-}6.$$

Рішення. Використовуючи метод простої ітерації, можна записати

$$x_1 = 0.25 + \sin x_0.$$

У загальному вигляді це співвідношення запишеться таким чином:

$$x_{i+1} = 0.25 + \sin x_i.$$

Процес ітерації продовжуватиметься до тих пір, поки

$$| (x_{i+1} - x_i) / x_i | < \varepsilon.$$

Введемо наступні ідентифікатори

X_0 -> x0

X_i -> xs

x_{i+1} -> xn
ε -> eps.

Схема алгоритму матиме наступний вигляд (Рис.7.4).

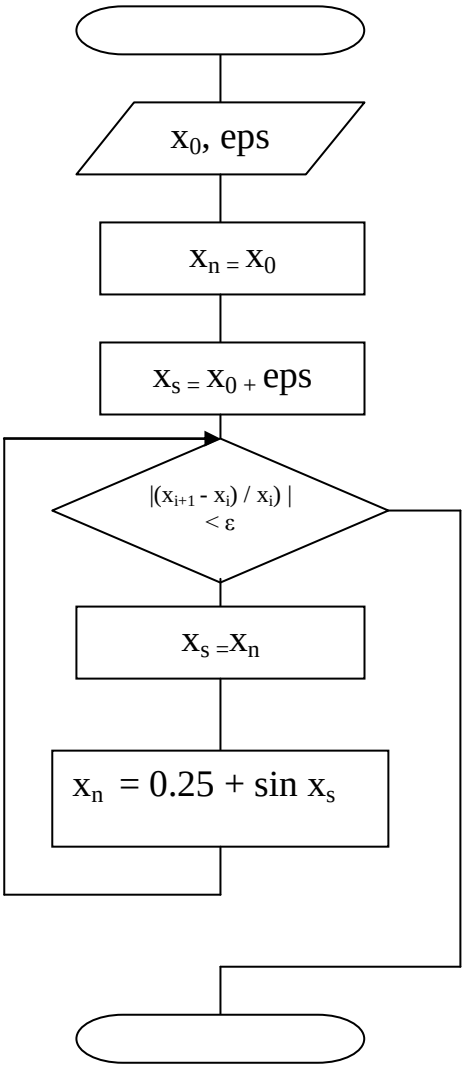


Рис.7.4. Схема алгоритму обчислень кореня рівняння.

Программа

```
#include <iostream.h>
#include <math.h>
void main()
{
        double x0,xn,xs,eps,k;   // k – кількість ітерацій
        cout<<" Введіть x0 і eps:  "<<endl;
        cin>>x0>>eps;
        cout<<endl;
        xn=x0;
        xs=x0+eps;
        k=0;
        while(fabs((xn-xs)/xn)>eps)
```

```
        {
            xs=xn;
            xn=0.25+sin(xs);
            k+=1;
        }
        cout<<"Корінь X=  "<<xn<<endl;
        cout<<"Кільк.ітер. = "<<k<<endl;
        cout<<endl;
}
/* Введіть x0 і eps:
1.25 1E-6

Корінь X=  1.17123
Кільк. ітер.= 13*/

/* Введіть x0 і eps:
1.25 1E-4

Корінь X=  1.17127
Кільк. ітер. = 8*/

/* Введіть x0 і eps:
1.19 1E-4

Корінь X=  1.17129
Кільк. ітер. = 6*/
```

3. Оператор циклу for…

Цей оператор використовується в тих випадках, коли кількість повторень тіла циклу наперед відомо.

Цикл for… самий універсальний зі всіх циклів мови С (С++). Основний синтаксис оператора циклу for… має наступний вигляд:

```
for ([ініціалізація]; [умова]; [модифікація])
    оператор або блок операторів;
```

Квадратні дужки говорять про те, що дана секція в операторі може бути опущена.

Іноді оператор циклу for записується у такому вигляді

```
for ([вып.1]; [вып.2]; [вып.3])
    оператор або блок операторів;
```

У схемах алгоритмів цикл for. позначається символом *модифікація* (Рис.7.5).

На схемі алгоритму приведені наступні позначення:

P – параметр циклу;

N1, N2 – межі зміни параметра циклу;

step – крок зміни параметра циклу (якщо крок не вказаний, то він дорівнює 1) .

Параметр циклу P, межі зміни N1, N2 і крок step повинні мати один і той же тип.

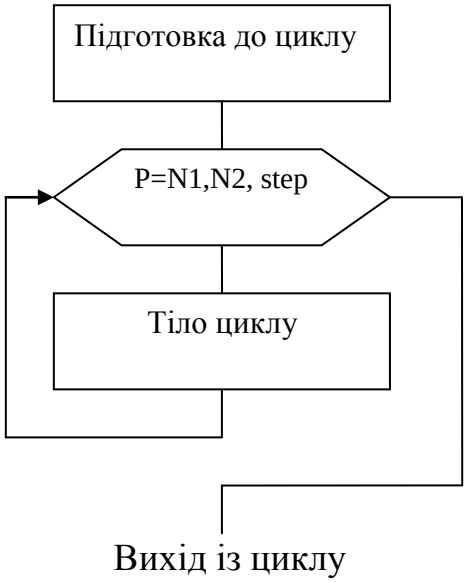


Рис.7.5. Схема алгоритму оператора циклу for.

Оператор циклу for. виконується таким чином. На першому кроці виконується ініціалізація параметра циклу. Ініціалізація використовується для присвоєння початкового значення параметру циклу. Секція ініціалізація може містити будь-який вираз. Ініціалізація проводиться тільки один раз на початку виконання оператора циклу.

Потім обчислюється вираз «умова». Умова – це звично умовний вираз, який визначає, коли цикл має бути завершений. Якщо воно «істина», то виконується тіло циклу. Інакше – відбувається вихід із циклу і управління передається наступному оператору програми.

Після виконання тіла циклу проводиться модифікація (зміна) параметра циклу на певну величину. Після модифікації параметра циклу управлінні передається заголовку циклу і все повторюється спочатку.

Приклад запису оператора циклу for.

```
for ( i=0; i<10; i++ ) cout << i <<endl;
```

У результаті виконання цього оператора будуть надруковані в стовпчик цифри від 0 до 9.

У даному прикладі параметром циклу for… є змінна i – лічильник. На початку циклу лічильник (змінна i) ініціалізується значенням 0. Потім виконується тіло циклу (cout << i <<endl;) і перевіряється, чи не досяг лічильник значення 10. Після кожного виконання тіла циклу лічильник (i) збільшується на одиницю. Як тільки і

стане рівним 10, тіло циклу пропускається і управління передається наступному оператору програми.

Для друку цих цифр у зворотному порядку можна використати наступний оператор:

```
for ( i=9; i<=0; i-- )  cout << i <<endl;
```

Для видачі на екран парних чисел використовують такий оператор:

```
for ( i=0; i<10; i+=2 ) cout << i <<endl;
```

У якості параметра циклу можна використовувати дані типу char:

```
unsigned char ch;
for ( ch='A'; ch<=Z; ch++ ) cout << ch<<endl;
```

Як видно з синтаксичного опису оператора будь-яку секцію заголовку оператора циклу for… можна опустити. Але роздільники – крапки з комою – повинні бути присутні обов'язково. Якщо пропущена «умова», то цикл виконуватиметься нескінченно. Приведемо три приклади нескінченних циклів.

```
for (i=0; ; i++) cout<<" Нескінчений цикл " << endl;
for (i=1; 1; i++) cout<<" Нескінчений цикл " << endl;
for (; ; ) cout<<" Нескінчений цикл " << endl;
```

Розглянемо приклади використовування оператора циклу for.

Приклад. Розробити алгоритм і програму обчислення суми цілих чисел, починаючи з 10.

Схема алгоритму матиме вигляд (Рис.7.6).

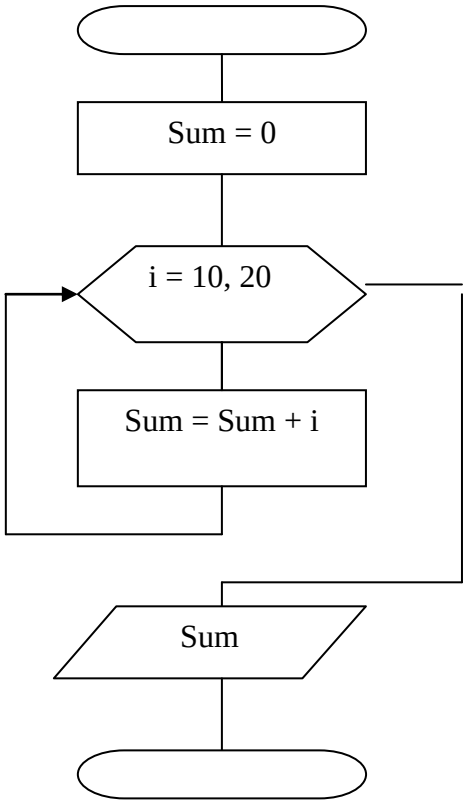


Рис.7.6. Схема алгоритму обчислення суми

Програма

```
#include <iostream.h>
void main()
{
        int i,Sum=0;
        for (i=10; i<20; i++) Sum=Sum + i;
        cout <<"SUMMA = "<<Sum<<endl;
}
```

Після запуску програми на виконання на екрані з'явиться результат.

```
        SUMMA = 145
```

Приклад обчислення факторіалу.

Факторіал –це добуток цілих чисел від 1 до n. Тоді:

$$n! = 1*2*3*4* \ldots n = \prod_{j=1}^{n} j .$$

При обчисленні факторіалу початковому значенню добктку необхідно привласнити 1. Потім, при кожному виконанні тіла циклу добуток умножатимемо на j.

Тоді алгоритм обчислення факторіалу можна записати таким чином.

1. P = 1 (підготовка до циклу);
2. P = P * j (тіло циклу).

Програма.

```
#include<iostream.h>
void main()
{
        int j,n;
        unsigned long fak=1;
        cout<<"Введіть ціле число:"<<endl;
        cin>>n;
        if (n>=1 && n<=30)
        {
                for(j=1; j<=n; j++) fak =fak*j;

                cout<<"fak= "<<fak<<endl;
        }
        else
                cout<<"Вы ввели неприпустиме число !"<<endl;
}
```

Після запуску програми на виконання на екрані з'явиться результат.

```
Введіть ціле число:
5
fak= 120
```

4. Оператори break і continue

Коли оператор break зустрічається у середині оператора циклу, то відбувається негайний вихід з циклу і перехід до виконання оператора, наступного за оператором циклу.

Приклад.

```
#include <iostream.h>
void main()
{
	unsigned char ch;
	for (; ;) //
	{
		cout<<"Введітье будь-який символ : "<<endl;
		cin >> ch;
		if(ch=='Q') break;//
		cout<<ch<<endl;
	}
	cout<<"Вихід з циклу"<<endl;
}
```

Після запуску програми на виконання на екрані з'явиться результат.

```
Введіть будь-який символ:
?
?
Введіть будь-який символ:
A
A
Введіть будь-який символ:
Q
Вихід з циклу
Press any key to continue
```

Оператор continue впливає тільки на оператори тіла циклу. Якщо він зустрівся в операторі циклу, то він передає управління на початок циклу. Іншими словами, тут достроково завершується не сам цикл, а його поточна ітерація. Наприклад. Видати на екран числа кратні *семи.*

Програма.

```
#include <iostream.h>
void main()
{
	int i;
	for (i=1;i<40;i++)
	{
		if(i%7) continue;
		cout<<i<<endl;
	}
}
```

Після запуску програми на виконання на екрані з'явиться результат.

7
14
21
28
35

Висновок. Якщо обчислювальний процес містить багатократні обчислення по одних і тих же математичних залежностях, але для різних значень змінних, то такий процес називається *циклічним*.

Алгоритм циклічних структур обов'язково повинен включати наступні дії.

5. Підготовка до циклу – присвоєння початкових значень параметру циклу.
6. Перевірку умови виконання тіла циклу.
7. Тіло циклу – дії, які повторюються в циклі при різних значеннях параметра циклу.
8. Зміна значення параметра циклу.

Питання для самоконтролю.

1. Укажіть на можливість запису оператора у такому вигляді:
   for (i=1; i<10; i++) i=i+1;
   1. Можливо
   2. Можливо в окремих випадках
   3. Не можливо
2. У програмі записана послідовність операторів
   N=5; for (i=1; i<=N; i++) N=N-1;
   Скільки разів виконається оператор циклу.
3. Скільки помилок зроблено при запису оператора циклу
   for (i=1, i< N, i++) Y=Y+1;
4. У програмі записана послідовність операторів
   y= 0; for(i=0.1; i<=0.9; i++) y+=sin(x);
   Укажіть на можливість запису такого оператора for...
   1. Можливо
   2. Не можливо
   3. Можливо в окремих випадках
5. У програмі записаний оператор
   for(i=1; i<=10; i+=2) s=s+1;
   Укажіть на можливість такого запису оператора:
   1. Можливо
   2. Не можливо
   3. Можливо в окремих випадках
6. Виконується оператор
   for(i=1;i<=3;i++)for(j=1;j<=6;j++) s;
   Скільки разів виконається оператор s:
   1. 6
   2. 9
   3. 18
7. Виконуються оператори
   X=0; N=2; for(i=1;i<=N;i++) X=X+N-1;
   Яке значення матиме змінна X після виходу з циклу?
8. У мові С(С++) оператор do… while

1. Оператор циклу з передумовою
2. Оператор циклу з післяумовою
3. Оператор циклу з параметром

9. У мові С(С++) в операторі while послідовність операторів (блок)
   1. Потрібно брати в операторні дужки
   2. Не потрібно
   3. На розсуд автора програми
10. У мові С(С++) записаний оператор

```
for (i = 0; i < 10; i+=2)  cout << i << “  “;
```

На екран дисплея буде видано:
   1. Послідовність парних чисел
   2. Послідовність непарних чисел
   3. Помилка про невірний запис оператора for.
11. У програмі записаний оператор

```
for (i = 0; 1;  i++)  cout << i << “  “;
```

Скільки разів виконається оператор циклу:
   1. Один раз
   2. Двічі
   3. Нескінченне число раз
12. Записаний фрагмент програми

```
int  sum = 0; int i,j;
for (i=10; j=2; i<20; i++, j=j+1)  sum +=i;
```

Чи можливий такий запис оператора for.
   1. Можлива
   2. Не можлива
   3. Можлива тільки в Visual C++
13. Використання оператора break в тілі циклу, викликає:
   1. Негайне закінчення циклу
   2. Дострокове завершення поточної ітерації циклу
   3. Оператор break можна використовувати тільки в конструкції switch
14. У програмі записаний оператор

```
for (i=10; i<20; i++); sum +=i.
```

Скільки помилок зроблено при запису оператора for.

15. На С++ записаний фрагмент програми

```
int  i;
for( i = 0; i <= 100; i++) if ( i*i >= 2)  break;
```

Скільки разів виконається оператор циклу.

Лекція 7,8

ОБРОБКА МАСИВІВ

Мета лекції. Вивчити особливості обробки одновимірних масивів у мові С (С++).

Основні питання лекції.

1. Опис масиву в програмі
2. Обробка масиву

До сих пір ми оперували з типами даних, які носять назву базових і відносяться до скалярних типів. На основі цих типів даних у мові С (С++) будуються інші типи даних. Масив – одна з найпростіших і відомих структур даних.

Під масивом у мові С (С++) розуміють набір даних одного і того ж типу, зібраних під одним ім'ям. Окрема одиниця таких даних, що входить у масив, називається елементом масиву. У якості елемента масиву можуть виступати дані будь-якого типу (один тип даних для всіх елементів масиву). Кожний елемент масиву визначається ім'ям масиву і його порядковим номером в масиві (індексом). Індекс елементу масиву – завжди ціле число.

Масиви бувають одновимірними і багатовимірними. Розглянемо одновимірні масиви.

1. Опис масиву в програмі

Використовуванню масиву в програмі передує його оголошення (опис), тобто резервування під масив певної кількості комірок пам'яті.

При оголошенні масиву указується тип елементів масиву, ім'я масиву і його розмір. Оголошення масиву в програмі – наступне.

```
Тип Імя масиву [розмір];
```

Наприклад.

```
int Mas [20];
```

Тут визначений (описаний) масив, що складається з 20-ти цілих чисел і в пам'яті буде зарезервовано місце для 20-ти цілочисельних елементів масиву.

Звернення до елементів масиву здійснюється за ім'ям масиву, з указівкою індексу (номера елементу масиву) в квадратних дужках.

```
Mas[17]= 11;
x = Mas[13];
y = Mas[19];
```

У мові С (С++) елементи масиву нумеруються, починаючи з *нуля.* Перший елемент має індекс 0, другий – індекс 1 і т.д.

Якщо оголошений масив

```
int Mas[100];
```

то елементи масиву матимуть наступні індекси

```
Mas[0], Mas[1] . Mas[99].
```

Тоді запис типу:

```
x=m[13];
y=m[19];
```

означає, що змінній x буде присвоєно значення 14-го елементу, змінній y – значення 20-го елементу масиву.

Масив, як і змінну, можна ініціалізувати при оголошенні. Значення для послідовних елементів масиву відділяються один від одного комами і поміщаються у фігурні дужки. Наприклад.

```
int Temp[6]= {2, 4, 7, 11, 12, 13};
```

Якщо в списку ініціалізації значень вказано менше ніж розмір масиву, то має місце часткова ініціалізація. При такому оголошенні після останнього значення у виразі ініціалізації, після останнього значення для наочності ставлять кому.

```
int Temp[6]= {2, 4, 7,};
```

Елементам Temp[0], Temp[1], Temp[2] будуть присвоєні значення 2, 4, 7, а елементи масиву, що залишилися, ініціалізації не одержать.

У мові С (С++) не перевіряється вихід індексу за межі масиву. Якщо масив m[100] цілочисельний масив:

```
int m[100];
```

а в програмі вказано

```
x=m[200];
```

то повідомлення про помилку не буде, змінній x буде присвоєно якесь то значення.

Розглянемо приклади обробки масивів.

2. Обробка масиву

При обробці масиву всі дії в програмі виконуються над елементами масиву, а не над масивом у цілому, як об'єктом. При обробці масиву індекс елементу може бути заданий або його значенням, або виразом:

```
A[4],  F[i+k+1];
```

Над масивом можна виконувати наступні дії:

1. Введення масиву в пам'ять ПК.
2. Виведення масиву на екран дисплея.
3. Присвоєння певних значень елементам масиву.
4. Копіювання масиву.
5. Перестановка елементів масиву.
6. Пошук елементу в масиві.
7. Сортування масиву.

Введення і виведення масиву в пам'ять ПК.

Як було сказано вище, при обробці масиву всі дії виконуються над елементами масиву. Отже, якщо заданий масив, наприклад

```
int Mas [19];
```

то для введення масиву в пам'ять ПК, необхідно організувати по елементне введення цього масиву. Іншими словами, для введення масиву в пам'ять ПК, необхідно організувати цикл. Якщо відомий розмір масиву, то необхідно використовувати оператор циклу for…

Виведення масиву здійснюється аналогічно введенню.

Приклад 1. Заданий масив із N цілих чисел. Необхідно ввести цей масив в пам'ять ПК, визначити його розмір і видати на екран початковий масив і його розмір.

Програма.

```
#include <iostream.h>
void main()
{
	const int N=20;
	int size, l, i;
	int Mas[N];
	cout<<"Введіть розмір масиву  l :"<<endl;
	cin>>l;
	cout<<endl<<"Введіть масив :"<<endl;
	for(i=0; i<l; i++) cin>>Mas[i]; cout<<endl;
	size=sizeof(Mas)/sizeof(Mas[0]);
	for (i=0; i<l; i++) cout<<Mas[i]<<' '; cout<<endl;
	cout<<"size= "<<size<<endl;
}
```

Після запуску програми і введення початкових вигляд екрану буде наступний:

```
Введіть розмір масиву l:
5
Ведіть масив :
1 2 3 4 5

1 2 3 4 5
size= 20
Press any key to continue
```

Присвоєння і копіювання масивів

Елементам масиву можуть бути присвоєні значення виразів. Елементи масиву і значення виразів повинні мати один і той же тип.

Наприклад. float A[3];

тоді

```
A[0]= 3.5;
A[1]= 0;
A[2]= a*x + b;
```

Копіювання – це присвоєння значень елементів одного масиву елементам іншого масиву. При копіюванні обидва масиви повинні мати однаковий розмір і тип елементів. Копіювання масиву A в B матиме вигляд.

```
for (i=0; i<N; i++) B[i]= A[i];
```

Перестановка елементів масиву

При обробці масиву виникає необхідність перестановки його елементів. У найпростішому випадку – це перестановка елементів, що стоять поруч. Ця задача вирішується з використанням допоміжної змінної, яка грає роль буфера.

Наприклад, необхідно поміняти місцями елементи A[i] і A[i+1]. Введемо додаткову змінну C того ж типу, що і елементи масиву.

Алгоритм перестановки буде наступним (Рис.8.1).

1. Елемент A[i] пересилаємо в C
2. Елемент A[i +1] пересилаємо на місце A[i].
3. Елемент A[i], який знаходиться в C, пересилаємо на місце A[i+1].

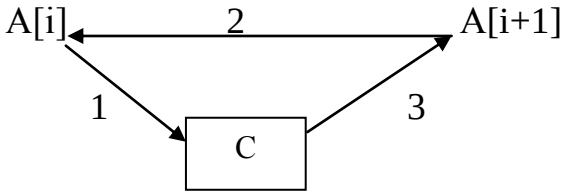


Рис.8.1. Алгоритм перестановки

Фрагмент програми перестановки двох елементів масиву матиме наступний вигляд.

```
C = A[i];
A[i]= A[i+1];
A[i+1]= C;
```

Пошук елемента масиву

Пошук елементу масиву полягає у виділенні з масиву окремих його елементів або виділення деяких елементів і їх кількості і т.д. Пошук може проводитися за зразком або за правилом.

Пошук за зразком полягає в наступному. Задається значення деякої змінної (зразок) і всі елементи масиву (або частина елементів) порівнюються зі значенням цієї змінної(зразком).

Пошук за правилом проводиться на основі перевірки деяких умов, яким повинні відповідати або елемент масиву, або група елементів.

Розглянемо приклад пошуку елементу за зразком.

Приклад 2. Заданий масив з N довільних чисел. Необхідно визначити кількість елементів, які більше заданого числа q, їх суму і порядкові номери. На екран видати початковий масив, суму елементів масиву, їх кількість і порядкові номери.

Алгоритм пошуку за зразком приведений на Рис.8.2.

Визначимо типи і структури даних, які використовуватимуться в програмі.

N – максимальний розмір масиву, з яким може працювати програма. Задамо його константою, рівною, наприклад, 50.

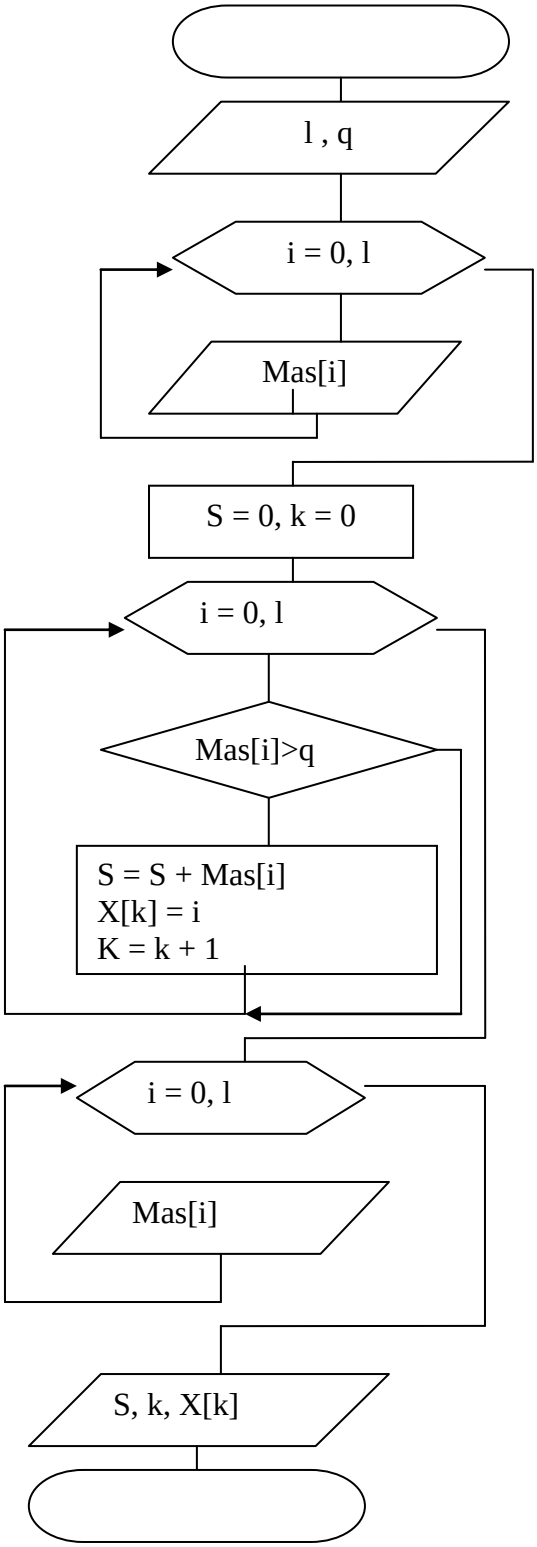


Рис.8.2. Алгоритм пошуку за зразком

l – реальний розмір масиву, який оброблятимемо. Його значення введемо з клавіатури, наприклад, 5.

q – зразок, за яким здійснюється пошук елементів масиву.

S – змінна для накопичення суми елементів масиву, більших q.

до – змінна для накопичення кількості елементів, більших q.

i – параметр циклу.

Mas[N] – масив з N довільних чисел.

X[k] – масив для зберігання номерів елементів початкового масиву, більших q.

Програма матиме вигляд.

```
#include <iostream.h>
void main()
{
	const int N=50;
	int i,l,k;
	float q,S;
	float Mas[N];
	int X[N];
	cout<<"Введіть розмір масиву l і q:"<<endl;
	cin>>l>>q;
	cout<<"Введіть масив:"<<endl;
	for(i=0; i<l; i++) cin>>Mas[i];
	cout<<endl;
	S=0; k=0;
	for(i=0; i<l; i++)
		if(Mas[i]>q)
		{
			S=S+Mas[i];
			X[k]=i;
			k=k+1;
		}
		cout<<"Масив:"<<endl;
		for(i=0; i<l; i++) cout<<Mas[i]<<' ';
		cout<<endl;
		cout<<"S= "<<S<<endl;
		cout<<"k= "<<k<<endl;
		cout<<"Номери:"<<endl;
		for(i=0; i<k; i++) cout<<X[i]<<' ';
		cout<<endl;
}
```

Запуск програми і введення початкових даних.

```
Введіть розмір масиву l і q:
5  3.7
Введіть масив:
3.9  2.5  3.6  6.2  5.0
```

Результат рішення задачі.

```
Масив:
3.9  2.5  3.6  6.2  5
S = 15.1
k = 3
Номери:
0 3 4
```

Типовим прикладом пошуку елементу за правилом є пошук максимального (мінімального) елементу масиву. Пошук максимального (мінімального) елементу полягає в наступному.

Значення першого елементу масиву присвоюється якій-небудь допоміжній змінній. Значення інших елементів порівнюються із значенням цієї змінної. Якщо значення поточного елементу масиву більше (менше) значення допоміжної змінної, то їй присвоюється значення цього елементу. Після перегляду всіх елементів масиву в допоміжній змінній знаходитиметься максимальний (мінімальний) елемент масиву.

Приклад 3. Заданий масив X з N довільних чисел. Знайти максимальний елемент масиву і його номер.

Схема алгоритму матиме наступний вигляд (Рис.8.3).

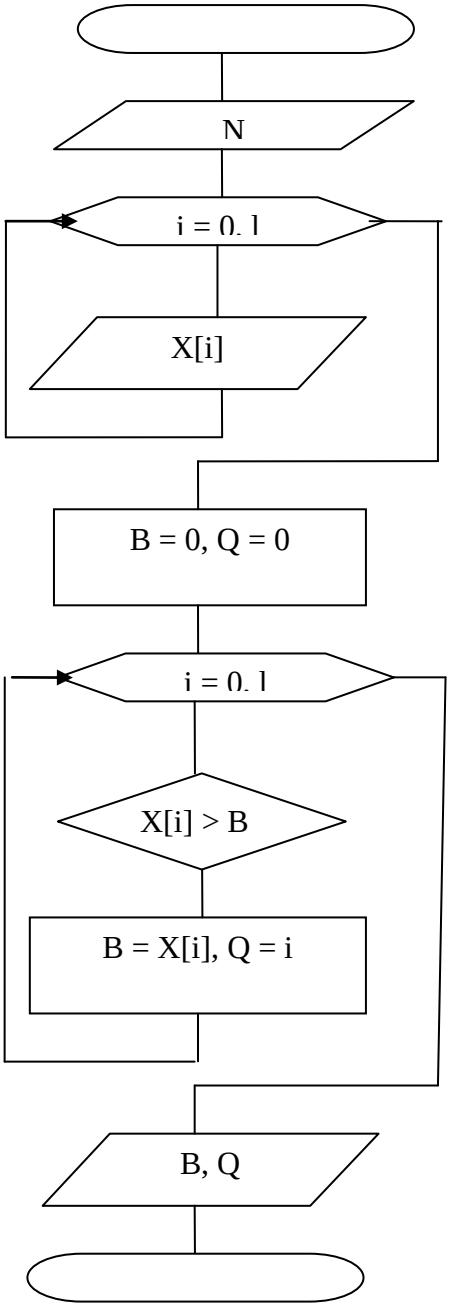


Рис.8.3. Схема алгоритму пошуку максимального елементу

Введемо дві допоміжні змінні.

B – для зберігання максимального елементу масиву.

Q – для зберігання номера (індексу) максимального елементу масиву.

Визначимо інші типи і структури даних.

N – максимальний розмір масиву, з яким працюватиме програма, наприклад, N = 50.

X[N] – масив з N довільних чисел.

i – параметр циклу.

l – реальний розмір масиву, що вводиться.

Підготовка до циклу включатиме:

B = X[0]; Q = 0;

Програма матиме наступний вигляд.

```
#include <iostream.h>
void main()
{
        const int N=50;
        int i,l,Q;
        float B,X[N];
        cout<<"Введіть розмір масиву:"<<endl;
        cin>>l;
        cout<<"Введіть масив:"<<endl;
        for(i=0; i<l; i++) cin>>X[i];
        cout<<endl;
        B=X[0];
        Q=0;
        for(i=0; i<l; i++)
                if(X[i]>B)
                {
                        B=X[i];
                        Q=i;
                }
        cout<<"Max= "<<B<<endl;
        cout<<"Його номер: "<<Q<<endl;
}
```

Після запуску програми на виконання і введення початкових даних, екран буде мати наступний вигляд.

```
Введіть розмір масиву:
6
Введіть масив:
1.9  5.6  1.8  2.0  0.4  3.4

Max= 5.6
Його номер: 1
```

Сортування масиву

Сортування масиву – це упорядкування елементів масиву за певними правилами. Наприклад, необхідно упорядкувати масив за збільшенням значень його елементів.

Ця задача розв'язується шляхом багатократного пошуку максимального елементу і пересилки його в упорядкований масив.

Розглянемо сортування масиву методом парної перестановки елементів масиву.

Елементи масиву $x_0, x_1, x_2, x_3, \ldots x_{n-1}$ розглядуються зліва направо і по черзі порівнюються пари елементів

x_0 і x_1; x_1 і x_2; x_2 і x_3 ;. x_{n-2} і x_{n-1}.

Якщо $x_{i-1} > x_i$, то робиться перестановка цих елементів. Після того, як переглянутий увесь масив, максимальний елемент буде розташований на останньому місці масиву справа, тобто він матиме індекс n-1. При другому перегляді масиву ця процедура повторюється, тільки для масиву $x_0, x_1, x_2, x_3, \ldots x_{n-2}$. При третьому перегляді – для масиву $x_0, x_1, x_2, x_3, \ldots x_{n-3}$. Процес сортування закінчується, якщо виконується одна з наступних умов.

1. Якщо при черговому перегляді масиву не було зроблено жодної перестановки елементів масиву.
2. Якщо в масиві залишився один елемент.

Приклад 4. Розробити алгоритм і програму сортування елементів масиву X[N] у порядку зростання значень його елементів.

Введемо позначення.

P – права межа масиву X[N], який є переглядається.

X[P]-> – дія, яка означає парну перестановку елементів масиву $x_0, x_1, x_2, x_3, \ldots x_p$.

Обчислювальний процес буде складним циклічним з розгалуженням.

Цикл по P буде зовнішнім. Для підготовки до виконання циклу параметру циклу P присвоюємо значення L-1 (де L – реальний розмір масиву, який сортується). У кожному циклі проводитимемо огляд масиву, парне порівняння і перестановку елементів масиву $x_0, x_1, x_2, x_3, \ldots x_p$. Після кожного перегляду значення параметра циклу P (межу масиву, що переглядається в черговий раз) зменшуватимемо на 1.

Процес перегляду чергової частини масиву теж носить циклічний характер – це буде внутрішній цикл for… з параметром i (i = 1, 2, 3 … P).

Процес сортування може бути закінчений раніше, якщо при черговому перегляді частини масиву не було зроблено жодної перестановки. Це означає, що частина масиву, що залишилася – вже упорядкована. Для перевірки цієї умови введемо ознаку перестановок R.

$$R = \begin{cases} 0 - \text{перестановок не було} \\ 1 - \text{перестановки були} \end{cases}$$

Для виконання перестановок елементів масиву введемо додаткову змінну C.

Крім того, підрахуємо загальну кількість перестановок елементів масиву при сортуванні. Кількість перестановок накопичуватимемо в змінній k.

При розробці алгоритму використовуватимемо покроковий метод. 1-й крок – це загальна схема дій при сортуванні, тобто реалізація принципу «що зробити». 2-й і подальші кроки – детальна розробка окремих блоків схеми алгоритму, тобто реалізація принципу «як зробити».

1-й крок 2-й крок

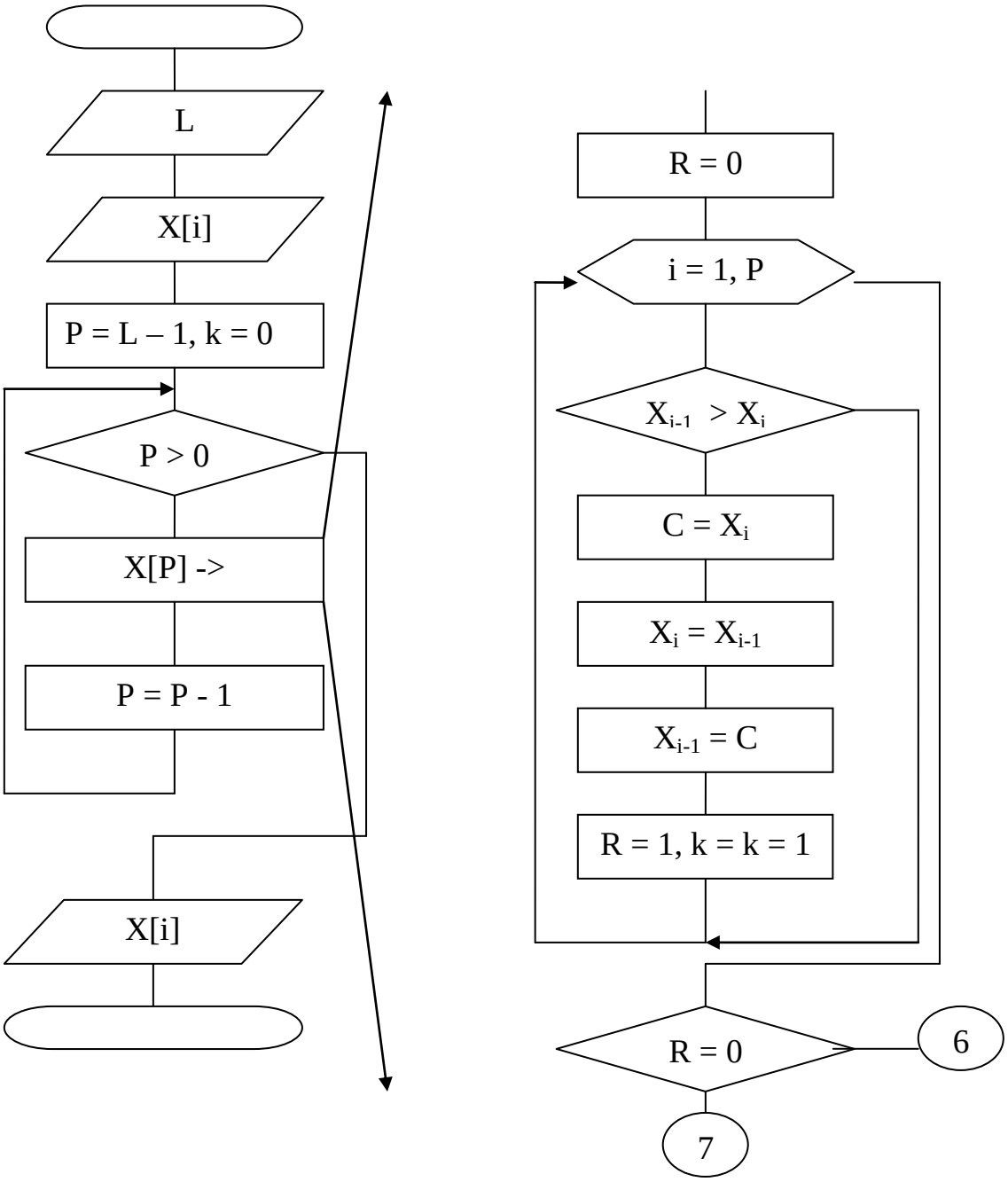


Рис 8.4. Схема алгоритму сортування масиву

Програма сортування масиву методом парних перестановок матиме наступний вигляд.

```
#include <iostream.h>
void main()
{
        const int N=50;
        int i,L,P,R,k;
        float C,X[N];
        cout<<"Введіть розмір масиву  L: "<<endl;
        cin>>L;
   cout<<"Введіть масив: "<<endl;
        for(i=0; i<L; i++) cin>>X[i];
        P=L;
```

```
        k=0;
        while (P>0)
        {
                R=0;
                for(i=0; i<P; i++)
                        if(X[i-1]>X[i])
                        {
                                C=X[i];
                                X[i]=X[i-1];
                                X[i-1]=C;
                                R=1;
                                k=k+1;
                                }
                if(R==0) goto A;
                P=P-1;
        }
A: cout<<"     Упорядкованій масив         "<<endl;
  for(i=0; i<L; i++) cout<<X[i]<<' ';
  cout<<endl;
  cout<<"k=  "<<k<<endl;
}
```

Після запуску програми на виконання екран буде мати наступний вигляд.

```
Введіть розмір масиву  L:
6
Введіть масив:
    2.9  5.2  1.8  7.3  3.9  2.6
    Упорядкованій масив
1.8  2.6  2.9  3.9  5.2  7.3
k=  8
```

Є і інші програми сортування масиву. Самою поширеною програмою є програма, яка реалізує метод «пухирця».

```
#include <iostream.h>
void main()
{
        const int N=50;
        int i,j,l,R,k;
        float C,X[N];
        cout<<" Введіть розмір масиву  l: "<<endl;
        cin>>l;
   cout<<" Введіть масив: "<<endl;
        for(i=0; i<l; i++) cin>>X[i];
        k=0;
```

```
        for(i=l-1; i>0; i--)
        {
                R=0;
                for(j=0; j<i; j++)
                        if(X[j]>X[j+1])
                        {
                                C=X[j];
                                X[j]=X[j+1];
                                X[j+1]=C;
                                R=1;
                                k=k+1;
                        }
                        if(R==0) goto A;
        }
A: cout<<"     Упорядкованій масив        "<<endl;
  for(i=0; i<l; i++) cout<<X[i]<<' ';
  cout<<endl;
  cout<<"k=  "<<k<<endl;
}
```

Вигляд екрану після виконання програми.

```
Введіть розмір масиву  l :
6
Введіть масив:
4.8  3.6  1.9  3.7  2.5  3 9
Упорядкованій масив
1.9  2.5  3.6  3.7  3.9  4.8
k=  10
```

Висновок. Під масивом у мові С (С++) розуміють набір даних одного і того ж типу, зібраних під одним ім'ям. Окрема одиниця таких даних, що входить у масив, називається елементом масиву. У якості елемента масиву можуть виступати дані будь-якого типу (один тип даних для всіх елементів масиву). Кожний елемент масиву визначається ім'ям масиву і його порядковим номером в масиві (індексом). Індекс елементу масиву – завжди ціле число.

При оголошенні масиву указується тип елементів масиву, ім'я масиву і його розмір. Оголошення масиву в програмі – наступне.

Тип Імя масиву [розмір];

При обробці масиву всі дії в програмі виконуються над елементами масиву, а не над масивом у цілому, як об'єктом. При обробці масиву індекс елементу може бути заданий або його значенням, або виразом:

A[4], F[i+k+1];

Над масивом можна виконувати наступні дії:

1. Введення масиву в пам'ять ПК.
2. Виведення масиву на екран дисплея.

3. Присвоєння певних значень елементам масиву.
4. Копіювання масиву.
5. Перестановка елементів масиву.
6. Пошук елементу в масиві.
7. Сортування масиву.

Питання для самоконтролю.

1. Обробляється масив X. В програмі обробки масиву зустрівся оператор
   if (X > 3) B=X;
   Скільки помилок зроблено при запису цього оператора
2. У мові С(С++) визначення масиву здійснюється таким чином
   1. int mas [1..20];
   2. mas : array[1..20] int;
   3. int mas [20];
3. У програмі на С (С++) зустрівся оператор x = m [13];
   Це означає, що змінній x буде привласнено значення:
   1. 12-го елементу
   2. 13-го елементу
   3. 14-го елементу
4. У програмі на С (С++) при обробці масиву всі дії виконуються:
   1. Над масивом в цілому
   2. Над елементами масиву
   3. Тільки над індексами елементів масиву
5. У програмі на С (С++) необхідно ввести з клавіатури n елементів масиву.
   Укажіть яким оператором це можна зробити:
   1. for (i=0; i<=n; i++) cin>>x[i];
   2. for (i=0; i< n; i++) cin>>x[i];
   3. for (i=1; i<= n; i++) cin>>x[i];
6. У програмі на С (С++) двовимірний масив визначається таким чином:
   1. float A [0..2] [0..3];
   2. float A [2, 3];
   3. float A [2][3];
7. У програмі на С (С++) записаний оператор
   for(i=0; i<l; i++) for (j=0; j<l; j++) cout<<a[i][j]<<endl;
   На екран двовимірний масив буде виведений:
   1. У вигляді рядка
   2. У вигляді стовпця
   3. У вигляді матриці
8. Масив:
   1. Має ім'я
   2. Не має імені
   3. Імена мають тільки елементи масиву
9. Заданий масив X = {X1, X2 ..., XN}. Укажіть, яким оператором можна вивести цей масив на екран:
   1. cout<<X;
   2. for (i=0; i<n; i++) cout <<X;
   3. for (i=0; i<n; i++) cout <<X[i];
10. У мові С(С++) визначення масиву здійснюється таким чином
    4. int mas];
    5. mas [20] ;
    6. int mas [20];
11. У програмі на С (С++) визначений масив char mas [10];
    Це означає, що під масив в пам'яті буде зарезервоване:
    1. 9 байтів

2. 10 байтів
3. 11 байтів

12. У програмі на С (С++) визначений масив int mas [10];
Це означає, що масив містить:
1. 9 елементів
2. 10 елементів
3. 11 елементів

13. У програмі на С (С++) зустрівся оператор x = m [13];
Це означає, що змінній x буде привласнено значення:
1.12-го елементу
2.13-го елементу
3.14-го елементу

14. У програмі на С (С++) визначений масив int A [6] = { 2, 4, 7 };
Чи можливе таке визначення масиву:
1. Можливо
2. Не можливо
3. Все залежить від майстерності програміста

15. У програмі на С (С++) визначений масив int A [100] і в програмі
записаний оператор x = A [200]. При виконанні цього оператора:
1. Буде видано повідомлення про помилку
2. Не буде видано повідомлення про помилку і змінній x буде привласнено якесь значення
3. Відбудеться зависання програми

16. У програмі на С (С++) при обробці масиву індекс елементу масиву може бути заданий:
1. Значенням
2. Виразом
3. Не має значення як

17. Є масиви A і B однакового розміру і їх елементи
мають тип int. Чи можливо в мові С (С++) копіювання масиву A в B операцією B = A:
1. Можливо
2. Не можливо
3. Можливо, якщо елементи масиву матимуть тип char

18. У програмі на С (С++) необхідно ввести з клавіатури n елементів масиву.
Укажіть яким оператором це можна зробити:
1. for (i=0; i<=n; i++) cin>>x[i];
2. for (i=0; i< n; i++) cin>>x[i];
3. for (i=1; i<= n; i++) cin>>x[i];

Лекція 9

БАГАТОВИМІРНІ МАСИВИ

Мета лекції. Вивчити особливості обробки двовимірних масивів у мові С (С++).

Основні питання лекції.

1. Опис двохвимірного масиву в програмі
2. Обробка двохвимірного масиву

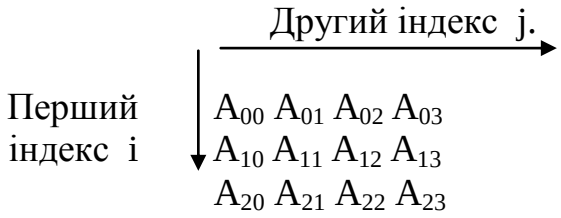
1. Опис двохвимірного масиву в програмі

. Найпростішим багатовимірним масивом є двовимірний масив – матриця або масив одновимірних масивів.

Оголошення двовимірного масиву в програмі має наступний вигляд:

Тип ІмяМас [Розмір 1] [Розмір 2];

Двовимірний масив int A[3] [4] можна представити у вигляді матриці.



Перший індекс – це номер рядка, другий індекс – номер стовпця.

У пам'яті машини двовимірний масив розташовується безперервно по рядках, тобто

A[0][0], A[0][1], A[0][2], A[0][3], … A[2][3].

Багатовимірний масив ініціалізується в порядку зміни з правого індексу (задом на перед). Спочатку відбувається присвоєння початкових значень всім елементам останнього (правого) індексу, потім попереднього і т.д. до самого лівого індексу.

Найпростіший спосіб ініціалізації наступний. При оголошенні масиву потрібно вказати у фігурних дужках список ініціалізаторів.

int A[2][3]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

або

int A[2][3]= {{1, 2, 3, 4} {5, 6, 7, 8} {9, 10, 11, 12}};

Для багатовимірних масивів при ініціалізації дозволяється опускати тільки величину першої розмірності.

int A[ ][3]= {{1, 2, 3, 4} {5, 6, 7, 8} {9, 10, 11, 12}};

Кількість ініціалізаторів не зобов'язана співпадати з кількістю елементів масиву. Якщо ініціалізаторів менше, то елементи, що залишилися, не визначені.

2. Обробка двохвимірного масиву

Над двовимірними масивами можна виконувати такі ж дії як і над одновимірними масивами: введення, виведення, присвоєння, копіювання, перестановку елементів і сортування.

Розглянемо приклади програм, що виконують деякі дії над двовимірними масивами.

Введення, виведення і присвоювання

При обробці двовимірних масивів перегляд елементів масиву необхідно проводити як по стовпцях так і по рядках. Для цієї мети необхідно використовувати вкладені цикли for…. Зовнішній цикл – перегляд рядків, внутрішній цикл – перегляд стовпців.

Приклад 1. Задана матриця цілих чисел A розмірністю M x N. Необхідно рядки матриці A присвоїти стовпцям матриці B. Одержану матрицю B вивести на екран.

Програма матиме вигляд.

```
#include <iostream.h>
void main()
{
  const int M=10,N=10;          //максимальний розмір матриці
  int i,j,l,k,A[M][N],B[M][N];  //опис змінних і матриць A і B
  cout<<"Введіть розмір матриці l і k:"<<endl; // Виведення запрошення
  cin>>l>>k;          //введення реальних розмірів матриці
  cout<<"Введіть матрицю:"<<endl;   // Виведення запрошення
  for(i=0; i<l; i++)
    for(j=0; j<k; j++) cin>>A[i][j];   //Введення матриці
         for(i=0; i<l; i++)
              for(j=0; j<k; j++) B[j][i]=A[i][j];  //Копіювання матриці A в B
cout<<"    Матриця B     "<<endl;     // Виведення пояснення
for(i=0; i<l; i++)
  {
     for(j=0; j<k; j++)
     cout<<B[i][j]<<'  ';       //Виведення матриці B
     cout<<endl;
   }
}
```

Після запуску програми на виконання екран дисплея буде мати вигляд.

```
Введіть розмір матриці l і k:
3 3
Введіть матрицю:
1 2 3
4 5 6
7 8 9
    Матриця B
1 4 7
2 5 8
3 6 9
```

Пошук максимального (мінімального) елемента масиву

Приклад 2. Задана матриця розміром 4x5 довільних чисел. Знайти максимальний (мінімальний) елемент матриці і його номер (номер рядка і номер стовпця). На екран видати початкову матрицю, максимальний (мінімальний) елемент і його номер.

При розробці програми використовуватимемо наступні змінні.

A[4][5] –матриця розміром 4x5;

B – змінна, в якій буде знаходитись максимальний елемент;

k – номер рядка матриці;

l – номер стовпця матриці.

Підготовка до циклу передбачає наступні дії.

B = A[0][0];

k = 0; l = 0;

Зовнішній цикл по i – перегляд рядків матриці, внутрішній цикл по j – перегляд стовпців. Схема алгоритму матиме наступний вигляд (Рис.9.1).

Програма.

```
#include <iostream.h>
void main()
{
  int i,j,l,k;          // опис змінних
  float A[4][5],B; //опис матриці A і змінної B
  cout<<"Введіть матрицю 4x5:"<<endl; // Виведення запрошення
  for(i=0; i<4; i++)
     for(j=0; j<5; j++) cin>>A[i][j]; //Введення матриці
     B=A[0][0];                       //Підготовка до циклу
     l=0; k=0;
       for(i=0; i<4; i++)
          for(j=0; j<5; j++)
            if(A[i][j]>B)              //Перегляд елементів матриці
             {
                B=A[i][j];    //Занесення в B максимального елементу
                 k=i;                  //Визначення його номера
                 l=j;
              }
  cout<<endl<<"Початкова матриця:"<<endl; // Виведення пояснення
  for(i=0; i<4; i++)
   {
     for(j=0; j<5; j++) cout<<A[i][j]<<' ';       //Виведення матриці
     cout<<endl;
   }
  cout<<endl<<"Max= "<<B<<endl;       //виведення Max елементу
  cout<<"Його номер :"<<endl;          // Виведення пояснення
  cout<<"Рядок №:"<<k<<endl;           // Виведення номеру рядка
  cout<<"Стовпчик №:"<<l<<endl;        //Виведення номеру стовпця
}
```

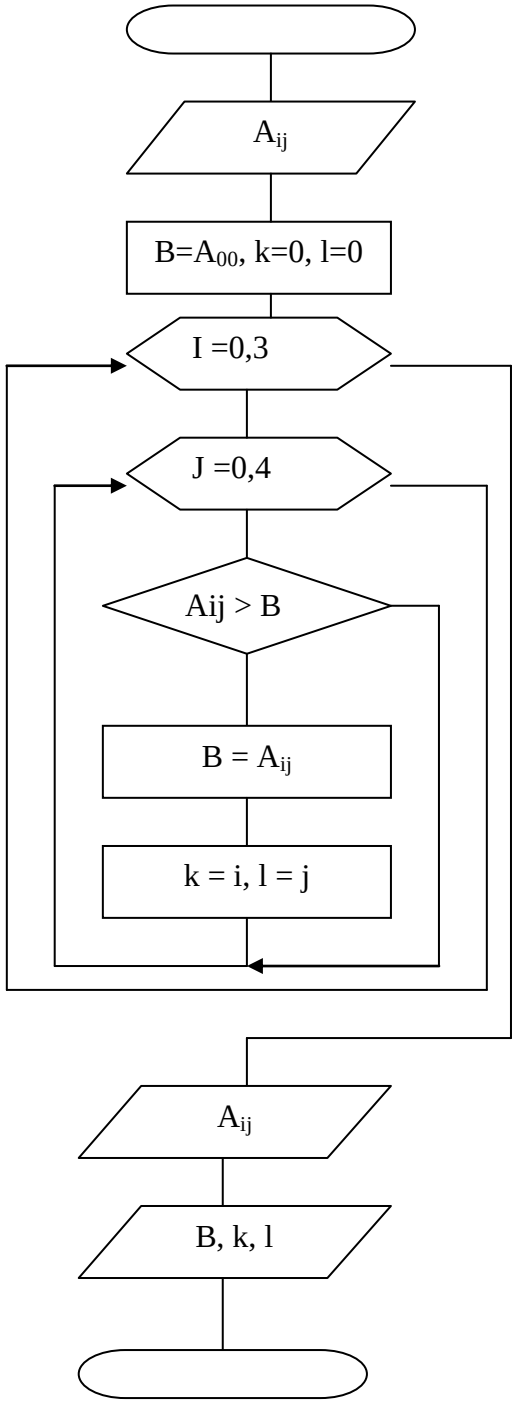


Рис.9.1. Схема алгоритму копіювання матриці

Вигляд екрану після введення матриці і виконанні програми має наступний вигляд.

Введіть матрицю 4x5:
1 2 3 4 5
9 7 3 7 2
12 7 4 9 3
9 5 3 22 5

Початкова матриця:
1 2 3 4 5

```
9 7 3 7 2
12 7 4 9 3
9 5 3 22 5

Max= 22
Його номер:
Рядок №:3
Стовпчик №:3
```

Розглянемо ще один приклад обробки двовимірного масиву.

Приклад 3. Задана матриця A розміром M x N цілих позитивних чисел. Сформувати масив R, елементи якого складають мінімальні елементи кожного рядка матриці A. В одержаному масиві R знайти суми парних і непарних елементів масиву. На екран видати початкову матрицю, одержаний масив R і суми парних і непарних елементів масиву R.

Введемо додаткові змінні.

S1 – змінна для зберігання суми непарних елементів;

S2 – змінна для зберігання суми парних елементів;

l і до – реальні розміри матриці, що вводиться.

Програма матиме вигляд.

```
#include <iostream.h>
void main()
{
	const int M=10,N=10; //Максимальний розмір матриці
	int A[M][N],R[M]; //Визначення матриці і рядка
	int i,j,k,l,S1,S2;   //опис змінних
	cout<<"Введіть розмір матриці lik:"<<endl;
	cin>>l>>k; //Введення реального розміру матриці
	cout<<"Введіть матрицю:"<<endl;
	for(i=0; i<l; i++)
		for(j=0; j<k; j++) cin>>A[i][j];//Введення матриці
	for(i=0; i<l; i++)
	{
		R[i]=A[i][0];     //Взначення
		for(j=0; j<k; j++)  //мінімального елементу
			if(A[i][j]<R[i]) R[i]=A[i][j];  //кожного рядка
	}
	S1=0;
	S2=0;
	for(i=0; i<l; i++)
	{
		if(R[i]%2==0) S2=S2+R[i]; //Визначення  суми парних
		else S1=S1+R[i]; //Визначення суми непарних
	}
	cout<<endl<<"Матриця A:"<<endl;
```

```
    for(i=0; i<l; i++)
    {
        for(j=0; j<k; j++)
            cout<<A[i][j]<<' ';//Виведення матриці
          cout<<endl;
    }
    cout<<endl<<"Вектор R:"<<endl;
    for(i=0; i<l; i++) cout<<R[i]<<' '; //Виведення рядка R
    cout<<endl;
    cout<<"S1= "<<S1<<endl; //Виведення суми S1
    cout<<"S2= "<<S2<<endl;  //Виведення суми S2

}
```

Вид екрану після запуску програми на виконання і виконання програми буде мати наступний вигляд.

```
Введіть розмір матриці l і k:
4 5
Введіть матрицю:
1 2 3 4 5
6 7 8 9 10
6 4 2 8 3
9 6 4 2 7

Матриця A:
1 2 3 4 5
6 7 8 9 10
6 4 2 8 3
9 6 4 2 7

Вектор R:
1 6 2 2
S1= 1
S2= 10
```

Висновок. Оголошення двовимірного масиву в програмі має наступний вигляд:

Тип ІмяМас [Розмір 1] [Розмір 2];

Двовимірний масив int A[3] [4] можна представити у вигляді матриці.

Перший індекс – це номер рядка, другий індекс – номер стовпця.

Над двовимірними масивами можна виконувати такі ж дії як і над одновимірними масивами: введення, виведення, присвоєння, копіювання, перестановку елементів і сортування.

Питання для самоконтролю.

1. У програмі на С (С++) двовимірний масив визначається таким чином:
   1. float A mas[2] [3];
   2. float A [2, 3];
   3. float A [2][3];
2. У пам'яті машини двомірний масив розташовується:
   1. Безперервно рядками
   2. Безперервно стовпцями
   3. У вигляді звичайної матриці

3 У програмі на С (С++) записані наступні визначення

int i, j, k, l ; int a[i][j], b[k][l];

Чи можливе таке визначення масивів:

1. Можливо
2. Не можливо
3. Можливо, якщо елементів масиву буде <=100

4. У програмі на С (С++) записаний оператор

for(i=0; i<l; i++] for (j=0; j<l; j++) cout<<a[i][j]<<” “;

На екран двовимірний масив буде виведений:

1. У вигляді рядка
2. У вигляді стовпця
3. У вигляді матриці

5. У програмі на С (С++) записаний оператор

```
for(i=0; i<l; i++)
    {for ( j=0; j<l; j++) cout<<a[i][j]; cout<<endl;}
```

На екран двовимірний масив буде виведений:

1. У вигляді рядка
2. У вигляді стовпця
4. У вигляді матриці

6 Виконується оператор

for(i=0; i<n; i++)for (j=0; j<n; j++) cout<<a[0][j];

Що буде видано на екран дисплея:

1. Перший рядок матриці, повторений n раз
2. Перший стовпець матриці, повторений n раз
3. Нормальна матриця

Лекція 10
ОБРОБКА РЯДКІВ

Мета лекції. Вивчити особливості обробки рядків у мові с С(С++).

Основні питання лекції.

1. Опис рядків.
2. Введення – виведення рядків.
3. Функції обробки рядків.
4. Рядки.
5. Функції перетворення типів.

1. Опис рядків

У мові С відсутній тип даних – рядок символів (string). Рядок представляється як одновимірний масив, елементи якого мають тип char.

Отже, символьний рядок – це одновимірний масив типу char, що закінчується нульовим байтом.

Для нульового байта визначена спеціальна символьна константа “\0”. Це слідує враховувати при визначенні відповідного масиву символів. Так, якщо рядок повинен містити N символів, то у визначенні масиву слід вказати N + 1 елемент.

Наприклад, визначення

```
char str[11];
```

означає, що рядок містить 10 символів, а останній байт зарезервований для нульового байта. І якщо його розглядати як звичний одновимірний масив, то він складатиметься з 10 елементів типу char.

У якості символів можуть використовуватися.

1. Прописні букви латинського і російського алфавітів.
2. Рядкові букви латинського і російського алфавітів.
3. Цифри від 0 до 9.
4. Символи пунктуації . ; тощо.
5. Символьні константи.
6. Управляючі символи.
7. Пробіл.
8. Шістнадцятеричні цифри.

Символьні масиви при їх визначенні можуть ініціалізуватися як звичайний масив

```
char str[12]={‘B’,’O’,’R’,’L’,’A’,’N’,’D’,’ ‘,’C’,’+’,’+’};
```

а можуть – як символьний рядок. Символьний рядок – це послідовність символів, укладених в подвійні лапки.

```
char str[12]=“BORLAND C++”;
```

Відмінність цих двох способів полягає в тому, що в другому випадку автоматично буде доданий ще і нульовий байт. До того ж другий спосіб коротше.

Для виділення місця в пам'яті під символьний масив необхідно вказати кількість символів у рядку або вказати явно більший розмір масиву.

```
char str[80]= “Це ініціалізація масиву символів”;
```

У даному випадку вказаний розмір масиву 80, хоча для розміщення цього рядка необхідно було вказати 35 (з урахуванням нульового байта).

Ініціалізувати символьний масив можна і без вказівки його розміру.

char str[ ]= “Це ініціалізація масиву символів”;

У цьому випадку компілятор сам визначить необхідний розмір пам'яті під цей масив.

Розглянемо питання, пов'язані з введенням і виведенням рядків умові С (С++).

2. Введення - виведення рядків

У мові С (С++) при роботі з рядками можна використовувати оператори запису в потік >> і виводу з потоку <<. Але оператор введення >> ігнорує пробіли, що вводяться

Приклад.

```
#include <iostream.h>
void main()
{
        char str[80];
        cout<<"Введіть рядок < 80 символів:"<<endl;
        cin>>str;
        cout<<endl<<"Ви ввели рядок:"<<endl;
        cout<<str<<endl;
        cout<<endl<<" Введіть  ще рядок < 80 символів:"<<endl;
        cin>>str;
        cout<<endl<<" Ви ввели рядок:"<<endl;
        cout<<str<<endl;
}
```

Після виконання Програми екран буде мати вигляд..

```
Введіть рядок < 80 символів:
PPPPPPPPPPPP

Ви ввели рядок:
PPPPPPPPPPPP

Введіть ще рядок < 80 символів:
AAAAA  BBBB  CCCCC

Ви ввели рядок:
AAAAA
Press any key to continue
```

Тому, при роботі з рядками замість оператора введення в потік >> краще використовувати функцію getline (). Функція getline () використовує два аргументи:

- перший аргумент указує на рядок, в який здійснюється введення;
- другий – число символів, підлягаючі введенню.

Приклад.

```
#include <iostream.h>
void main()
{
        char str[80];
        cout<<" Введіть рядок < 80 символів:"<<endl;
        cin.getline(str,30);
        cout<<endl;
        cout<<" Ви ввели рядок:"<<endl;
        cout<<str<<endl;
}
```

Вид екрану після роботи програми.

```
Введіть рядок < 80 символів:
AAAA BBBB CCCC

Ви ввели рядок:
AAAA BBBB CCCC
Press any key to continue
```

При використанні функції getline () другим слід указувати число, менше або рівне розміру оголошеного символьного рядка.

Оголошений в програмі рядок str може прийняти 80 символів. У функції getline (str,30) вказано число 30. Якщо вводити рядок з 35 символів, то введеться рядок из30 символів. Решта символів буде відкинута.

3. Функції обробки рядків.

Для роботи з рядками існує спеціальні функції, опис яких знаходиться в заголовному файлі string.h.

Розглянемо функції, що використовуються найбільш часто .

Визначення довжини рядка

Дуже часто при роботі з рядками необхідно знати, скільки символів містить рядок. Для отримання інформації про довжину рядка використовується функція strlen (). Виклик функції має вигляд:

```
strlen (str);
```

Функція повертає значення на одиницю менше ніж відводиться під масив (без урахування нульового байта).

Приклад.

```
#include <iostream.h>
#include <string.h>
void main()
{
        char str[]="ABCDEFGHIJK";
        int до;
        k=strlen(str);
        cout<<"k= "<<k<<endl;
```

```
}
```

Вид екрану після роботи програми.

```
k= 11
Press any key to continue
```

Копіювання і приєднання рядків

Значення рядків можуть копіюватися з одного рядка в інший. Копіювання здійснюється за допомогою наступних функцій.

Функція strcpy (S1,S2) використовується для побайтного копіювання рядка S2 в рядок S1. Копіювання припиняється досягши нульового байта. Тому довжина рядка S1 повинна бути достатньо великою, щоб в неї помістився рядок S2.

Приклад.

```
char str [20];
ctrcpy (str " Перевірка копіювання");
```

Даний фрагмент програми пояснює процес копіювання в рядок str рядка " Перевірка копіювання".

Функція ctrncpy () відрізняється від функції ctrcpy () тим, що включає ще один параметр. Він указує кількість символів, не більше яких буде скопійовано. Функція має вигляд:

```
ctrncpy (S1, S2, n);
```

де n – ціле без знаку.

Якщо довжина S1 менше довжини S2, то відбувається урізання символів.

Приклад.

```
#include <iostream.h>
#include <string.h>
void main()
{
    char S1[]="0123456789012345";
    char S2[]="abcdef";
    strncpy(S2,S1,4);
    cout<<"S2= "<<S2<<endl;
    cout<<"S1= "<<S1<<endl;
}
```

У результаті виконання програми на екран буде видано:

```
S2= 0123ef
S1= 0123456789012345
Press any key to continue
```

Тобто, з рядка S1 в рядок S2 будуть скопійовани 4 перших символа, розмістив їх на початку короткого рядка S2.

Приєднання (Конкатенація) рядків досить часто використовується для утворення нового рядка символів. Для цієї мети використовуються функції

```
strcat (S1, S2) і strncat (S1, S2, n);
```

Функція strcat (S1, S2) приєднує рядок S2 до рядка S1 і поміщає її в масив, де знаходився рядок S1. Рядок S2 не змінюється. Знов одержаний рядок S1 автоматично завершується нульовим байтом.

Приклад.

```
#include <iostream.h>
#include <string.h>
void main()
{
        char S1[20],S2[20];
        strcpy(S1, "Hello ");
        strcpy(S2, "World!");
        cout<<"S1= "<<S1<<endl;
        cout<<"S2= "<<S2<<endl;
        strcat(S1, S2);
        cout<<endl<<"S1= "<<S1<<endl;
        cout<<"S2= "<<S2<<endl;
}
```

Результат виконання програми.

```
S1= Hello
S2= World!

S1= Hello, World!
S2= World!
Press any key to continue
```

Функція strncat (S1, S2, n) також здійснює приєднання рядків, проте приєднує лише вказану в третьому параметрі кількість символів.

Приклад.

```
#include <iostream.h>
#include <string.h>
void main()
{
        char S1[80]="Для продовження  ";
        char S2[80]="нажати кнопку  ОК !";
        strncat(S1,S2,7);
        cout<<S1<<endl;
}
```

На екран буде виведений рядок:

```
Для продовження  нажати
Press any key to continue
```

Порівняння рядків

У бібліотеці функцій string.h є функції, які виконують посимвольне порівняння двох рядків.

Функція strcmp (S1, S2) порівнює рядки S1 і S2. Після порівняння рядків дана функція повертає одне з наступних значень:

<0 – якщо рядок S1 менше ніж S2;
=0 – якщо рядки еквівалентні;
>0 – якщо S1 більше, ніж S2.

Ця функція проводить порівняння рядків, розрізняючи прописні і рядкові букви.

Приклад.

```
#include <iostream.h>
#include <string.h>
void main()
{
        char S1[]="Помилка відкриття Бази";
        char S2[]="Помилка відкриття бази";
        int до;
        k= strcmp(S1,S2);
        cout<<"k= "<<k<<endl;
}
```

Змінної k буде присвоєно негативне значення (-1). Рядок S1 менше рядка S2 з тієї причини, що прописні букви мають код символів менше ніж ті ж рядкові букви.

```
k= -1
Press any key to continue
```

Функція stricmp (S1, S2) порівнює рядки S1 і S2 не розрізняючи регістра символів.

Функція strncmp (S1, S2, n) проводить порівняння певного числа (n) перших символів двох рядків. Регістр символів при цьому враховується.

Функція strnicmp (S1, S2, n) проводить порівняння певного числа (n) перших символів двох рядків. Регістр символів при цьому не враховується.

Приклад.

```
#include <iostream.h>
#include <string.h>
void main()
{
        char S1[]="Помилка відкриття Базі";
        char S2[]="Помилка відкриття бази";
        int до;
        k= strcmp(S1,S2);
        cout<<"strcmp"<<endl;
        cout<<"k= "<<k<<endl;
        k= stricmp(S1,S2);
```

```
        cout<<"stricmp"<<endl;
        cout<<"k= "<<k<<endl;
        k= strncmp(S1,S2, 12);
        cout<<"strncmp"<<endl;
        cout<<"k= "<<k<<endl;
        k= strnicmp(S1,S2,12);
        cout<<"strnicmp"<<endl;
        cout<<"k= "<<k<<endl;
}
```

Результат виконання програми.

```
strcmp
k= -1
stricmp
k= -1
strncmp
k= 0
strnicmp
k= 0
Press any key to continue
```

Окрім розглянутих функцій є велика кількість інших функцій обробки рядків.:

- функції перетворення рядків (перетворення елементів символьного рядка з одного регістра в іншій);
- функції обігу рядків (міняє порядок проходження символів в рядку на зворотний) ;
- функції пошуку символів (одного або групи символів в рядку);
- пошук підрядків.

4. Рядки

Стандартна бібліотека мови С++ містить опис типу string. Використовуючи цю бібліотеку, можна оголошувати і застосовувати змінні, що містять рядки. Для цього достатньо включити у програму наступні директиву і оператор.

```
#include <string>
using namespace std;
```

Можна оголосити рядкову змінну title і ініціалізувати її порожнім рядком.

```
string title;
```

Рядкову змінну можна ініціалізувати рядковим літералом, написавши оператор

```
string title = "Рядки С++";
```

Згодом, змінній title можна присвоїти інший рядок, використовуючи оператор присвоєння.

```
title = "Гекльберрі Фінн";
```

Цей рядок складається з 16 символів. У цьому випадку говорять, що довжина рядка title рівна 16. Для обчислення поточної довжини рядка можна застосовувати або функцію length(), або функцію size().

Посилатися на окремі символи рядка можна за допомогою індексів, наче рядок є масивом. Таким чином, у попередньому прикладі комірка title[0] містить символ ‘Г’, а комірка title[15] - символ 'н'.

Рядки можна порівнювати, використовуючи звичайні оператори порівняння. Причому, можна перевіряти не тільки рівність, але і який з рядків передує іншому. Рядки упорядковуються в порядку слідування ASCII-кодів їх символів. Таким чином, всі наведені нижче відношення є істинними.

```
"dig"  <  "dog"
"Star"  <  "star"  (оскільки  'S'  <  's')
"start"  >  "star"
"d"  >  "abc"
```

Рядки можна конкатенувати, використовуючи оператор «+». У результаті утворюється новий рядок, що складається з двох частин: першого і другого рядка, записаних послідовно. Наприклад, якщо у програмі помістити оголошення

```
string strl  =  "Com";
```

то оператори

```
string str2  = strl +  "puter";
strl  +=  "puter";
```

присвоять змінним strl і str2 рядок "Computer". Аналогічно, до рядка можна приписати окремий символ.

```
Strl  +=  'S1;
```

Використовуючи функцію substr(позиція, довжина), можна маніпулювати частинами рядка. Його перший аргумент визначає позицію початку підрядка (пам'ятайте, що перший символ рядка знаходиться на *нульовій* позиції). Другий аргумент задає довжину підрядка. Наприклад, результатом виклику

```
title.substr (10, 4)
```

є слово "Фінн".

Щоб виконати введення і виведення рядків мови С++, слід використовувати сучасну версію бібліотеки iostream. Для цього необхідно включити у програму наступні директиву і оператор:

```
#include <iostream>
using namespace std;
```

Ця версія бібліотеки iostream працює точно так, як і стара бібліотека iostream.h. Отже, оператор «<<» можна використовувати для виведення на екран як літерального рядка, так і рядкової змінної. Наприклад, фрагмент програми

```
title = "Рядки С++";
cout << "\"" << title << "\ "\ n";
```

виведе на екран рядок "Рядки С++". Зверніть увагу на спеціальний символ «\ "», що стоїть усередині літерального рядка. Він призначений для виведення на екран подвійної лапки. Оператор «<<» виводить на екран весь рядок, включаючи пропуски.

У символьну змінну можна зчитувати рядок символів. Якщо оператор

```
cin >> title;
```

прочитає рядок введення Гекльберрі Фінн

він присвоїть змінній title рядок "Гекльберрі". Пробільні символи у вхідному рядку переривають операцію читання. Для введення рядка разом з пробільними символами слід використовувати функцію getline(cin, title).

5. Функції переиворення типів

Функції перетворення типу використовуються для перетворення чисел, введених у вигляді символьних рядків, у числове уявлення, виконання певних математичних операцій над ними і зворотне перетворення в рядок символів.

Ці функції розміщуються в заголовному файлі stdlib.h.

Функції atof(), atoi(), atol() – перетворюють рядок символів в число типу float, int, long.

Функції ecvt(), fcvt(), gcvt() – перетворюють число з плаваючою точкою типу double в рядок символів.

ecvt() – десяткова крапка і знак числа не включаються в одержаний рядок;

fcvt() – округляє набуте значення до заданого числа цифр;

gcvt() – включає символ десяткової крапки.

Функції itoa(), ltoa(), ultoa() – перетворюють числа типу int, long і unsigned long в рядок символів.

Функції strtod() і strtol() – перетворюють рядок символів в число типу double і long.

Розглянемо використання деяких з цих функцій.

Функції

int atoi (const char *ptr);

long atol (const char *ptr);

Перетворюють рядок символів, на яку указує ptr в число типу int.

Приклад.

```
#include <iostream.h>
#include <stdlib.h>
void main()
{
    char S1[]="7000";
    int до;
    long l;
    k=atoi (S1);
    l=atol (S1);
    cout<<"k= "<<k<<endl;
    cout<<"l= "<<l<<endl;
}
```

Результакт виконання програми.

```
k= 7000
l= 7000
Press any key to continue
```

Функції зворотного перетворення itoa і ltoa проводять перетворення чисел int і long в рядок.

char_*ltoa (long num, char *str, int radix);

char *itoa (long num, char *str, int radix);

Тут число num перетворюється в рядок str з урахуванням системи числення – radix.

Приклад.

```
#include <iostream.h>
#include <stdlib.h>
void main()
{
    int num=98765;
    char S1[10];
    itoa(num,S1,10);
    cout<<"num= "<<num<<endl;
    cout<<"S1= "<<S1<<endl;
}
```

Результат виконання програми.

```
num= 98765
S1= 98765
Press any key to continue
```

Функція gcvt має прототип

char *gcvt (double val, int sig, char *buf);

перетворює число val типу double в рядок і поміщає її в буфер buf (int sig – число цифр, що підлягають перетворенню).

Якщо число цифр, що підлягають перетворенню, менше числа, вказаного в sig, то в перетвореному числі вказується знак і десяткова крапка. Молодші розряди дробової частини відкидаються. Інакше - число перетворюється в експоненціальну форму.

Приклад.

```
#include <iostream.h>
#include <stdlib.h>
void main()
{
        char S1[10];
        double num;
        int sig=4;
        num=3.547;
        gcvt(num,sig,S1);
        cout<<"S1= "<<S1<<endl;
        num=-843.7105;
       gcvt(num,sig,S1);
        cout<<"S1= "<<S1<<endl;

        num=0.135E4;
       gcvt(num,sig,S1);
        cout<<"S1= "<<S1<<endl;
}
```

Вигляд екрану після виконання програми.

```
S1= 3.547
S1= -843.7
S1= 1350
Press any key to continue
```

Висновок. У мові С відсутній тип даних – рядок символів (string). Рядок представляється як одновимірний масив, елементи якого мають тип char.

Отже, символьний рядок – це одновимірний масив типу char, що закінчується нульовим байтом.

У мові С відсутній тип даних – рядок символів (string). Рядок представляється як одновимірний масив, елементи якого мають тип char.

Отже, символьний рядок – це одновимірний масив типу char, що закінчується нульовим байтом.

Питання для самоконтролю.

1. У програмі на С (С++) визначений масив char A [11]. Це означає, що рядок містить:
    1. 10 символів
    2. 11 символів
    3. 12 символів
2. У мові С (С++) для копіювання рядків використовується функція:
    1. strlen( )
    2. strcpy( )
    3. strcat( )

3 У мові С (С++) конкатенація рядків – це:

1. Копіювання одного рядка в інший
2. Порівняння двох рядків
3. Приєднання одного рядка до іншого

Лекція 11
ПОКАЖЧИКИ

Мета лекції. З'ясувати поняття покажчика з адресацією пам'яті. Вивчити особливості використання покажчиків в програмах на С (С++).

Основні питання лекції.

6. Поняття покажчика.
7. Розіменування покажчиків.
8. Операції з покажчиками.
9. Покажчики і масиви

1. Поняття покажчика.

Змінна базового типу або похідного типу займає в пам'яті певну область (кількість комірок). Її місцеположення в пам'яті визначається адресою. При оголошенні змінної для неї резервується місце в пам'яті. Розмір зарезервованої пам'яті залежить від типу даної змінної.

Для доступу до вмісту виділеної пам'яті служить його ім'я (ідентифікатор). Для того, щоб узнати адресу конкретної змінної, служить операція *узяття адреси.* Синтаксис операції:

&ІЗ

Тобто перед ім'ям змінної (ІЗ) ставиться знак &. Розглянемо приклад.

```
#include <iostream.h>
void main ()
 {
        int Var1=4000;
        int Var2=300;
        cout<<»Znachenie Var1= «<<Var1<<endl;
        cout<<»Adres Var1= «<<&Var1<<endl;
        cout<<»Znachenie Var2= «<<Var2<<endl;
        cout<<»Adres Var2= «<<&Var2<<endl;
}
```

Після запуску програми на виконання на екрані з'явиться наступна інформація:

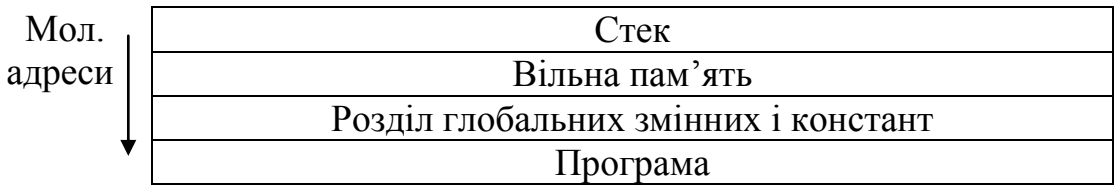
```
Znachenie Var1= 4000
Adres Var1= 0x0012FF7C
Znachenie Var2= 300
Adres Var2= 0x0012FF78
Press any key to continue
```

Адреси змінних записані в шістнадцятеричній системі числення. Ознакою 16-ричної константи є ознака 0x.

Представлення чисел в різних системах числення представлені в таблиці.

Десяткове число	8 – ричне число	16 – ричне число	Двійкове число
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	3	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111

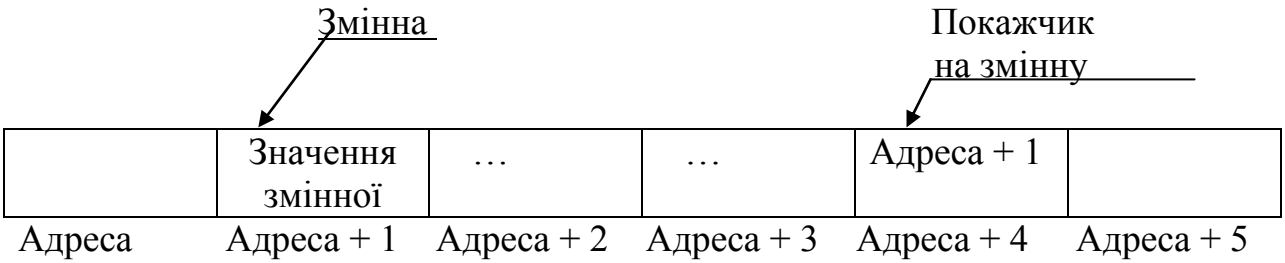
У мові С (С++) прийнято наступний розподіл пам’яті.



Адреси локальних змінних розміщуються в стеку. Тому їх адреси слідують у зворотному порядку (стек росте у напрямі молодших адрес). Різниця в адресах Var1 і Var2 завжди буде однакова і при 4-х байтовому представленні чисел типу int складе 4 байти.

У мові С (С++) є можливість здійснювати безпосередній доступ до пам’яті. Для цього передбачений спеціальний тип змінних – *покажчики.*

Покажчик – є змінною, яка містить адресу деякого об’єкту. Об’єктом може бути: змінна базового або похідного типа або функція. У загальному випадку – це просто ціле число!



Якщо змінна буде покажчиком, то вона повинна бути оголошена в програмі. Покажчик в програмі оголошується таким чином:

ТипОб’єкту *Ідентифікатор;

Тут «ТипОб'єкту» визначає тип даних, на які посилається покажчик з ім'ям «Ідентифікатор». Символ * (зірочка) означає, що наступна за нею змінна *є покажчиком.* При оголошенні покажчика під нього резервується 4 байти.

Приклади оголошення покажчиків.

```
Char *ch;
int *temp, i *z;
float f *ptr;
```

Тут оголошені покажчики ch, temp, z, ptr, змінні і типу int і f типу float.

Оскільки покажчик є посиланням на деяку область пам'яті, йому може бути присвоєна тільки адреса змінної, а не саме її значення.

Розглянемо приклад оголошення і ініціалізації покажчика.

```
#include <iostream.h>
void main()
{
        float c=57.97;
        float *pc=&c;
        long temp=340;
        long *plong;
        plong=&temp;
        cout<<c<<' '<<&c<<endl;
        cout<<»     «<<pc<<endl;
        cout<<temp<<' '<<&temp<<endl;
        cout<<»   «<<plong<<endl;
}
```

Результат виконання програми.

```
57.97 0x0012FF7C
        0x0012FF7C
340 0x0012FF74
     0x0012FF74
Press any key to continue
```

У приведеній програмі оголошується змінна с типу float і ініціалізується значенням 57.97. Потім оголошується покажчик на дані типу float – pc, значення якого призначається рівним адресі змінної с. Потім оголошується змінна temp типу long і покажчик plong на той же тип. Після чого проводиться ініціалізація покажчика plong адресою змінної temp.

Зауваження. При написанні програм бажано дотримуватися угод про імена – відповідно до яких перший символ в імені змінної указує на її тип. Наприклад, можна використовувати n для int, f для float, d для double. З урахуванням цієї угоди імена покажчиків починатимемо з букви p.

2. Розіменування покажчиків.

Покажчики допомагають здійснювати безпосередній доступ до пам'яті. Для того, щоб набути (прочитати) значення, записане за адресою, яка знаходиться в

покажчику, використовують операцію непрямого звернення або розіменування (*). Для цього використовується ім’я покажчика із зірочкою перед ним.

Приклад.

```
#include <iostream.h>
void main()
{
        float x=10.1,y;
        float *pf;
        pf=&x;
        y=*pf;
        cout<<»x= «<<x<<endl;
        cout<<»y= «<<y<<endl;
}
```

Після виконання програми на екран буде видана наступна інформація.

```
x= 10.1
y= 10.1
Press any key to continue
```

3. Операції з покажчиками.

Мова С (С++) дозволяє працювати з покажчиками так, якби вони були змінними стандартних типів. Проте операції над покажчиками вимагають знання деяких особливостей. Розглянемо ці особливості.

Операція присвоювання

Покажчики одного і того ж типу можуть використовуватися в операціях присвоювання, як і інші будь-які змінні.

Приклад.

```
#include <iostream.h>
void main()
{
        int x=10;
        int *p *g;
        p=&x;
        g=p;
        cout<<»p= «<<p<<endl;
        cout<<»g= «<<g<<endl;
        cout<<»x= «<<x<<»    *g= «<<*g<<endl;
}
```

Після виконання програми на екран буде видана наступна інформація.

```
P= 0x0012FF7C
g= 0x0012FF7C
x= 10    *g= 10
Press any key to continue
```

Арифметичні операції

Як і над іншими типами даних, над покажчиками можна виконувати арифметичні операції: складання і віднімання. Але арифметичні операції над покажчиками мають свої особливості. До таких особливостей виконання операцій відноситься наступне.

P1 + n (P1-n) – обчислення покажчика, віддаленого від заданого на визначене в n число елементів. Тут n – ціле.

Хай покажчик P1 має значення 2000 і указує на ціле. Тоді, в результаті виконання оператора

P1 + 3;

значення покажчика P1 буде дорівнюватись 2012 (якщо під int відводиться 4 байти).

Загальна формула для обчислення значення покажчика після виконання операції P = P + n, матиме вигляд:

P = P + n*k;

де k – кількість байт пам'яті базового типу покажчика.

Іншою операцією є операція *віднімання*.

P2 – P1;

Це обчислення числа елементів між покажчиками.

Операції інкремента (++) і декремента (--) є окремим випадком складання і віднімання.

Приклад.

```
#include <iostream.h>
void main()
{
	int *p;
	int x;
	p=&x;
	cout<<»p= «<<p<<endl;
	cout<<»  «<<++p<<endl;
}
```

```
p= 0x0012FF78 // значення  p;
   0x0012FF7C // значення ++p;
Press any key to continue
```

Після виконання цієї програми ми бачимо, що при операції ++p значення покажчика p збільшилося не на 1, а на 4. Це правильно, оскільки нове значення покажчика повинне указувати не на наступну адресу пам'яті, а на адресу наступного цілого.

Інші арифметичні операції над покажчиками заборонені.

Порівняння покажчиків

Покажчики можна порівнювати, застосовуючи всі 6 операцій порівняння.

P1 = = P2 – порівняння на рівність;

P1 != P2 – порівняння на нерівність;

P1 < P2 – менше;

P1 < = P2 – менше або рівно;
P1 > P2 – більше;
P1 > = P2 – більше або рівно.

Порівняння p < g означає, що адреса, що знаходиться в p, менше адреси, що знаходиться в g.

Операція sizeof

Як і до будь-якої змінної або типу даних, до покажчиків можна застосовувати операцію визначення розміру – sizeof. В сучасних ПК покажчик має розмір 4 байти (32 двійкові розряди) і може адресувати

2^{32} = 4 Гбайт пам'яті.

До покажчиків можна застосовувати не тільки операцію sizeof але і однойменну функцію.

Приклад.

```
#include <iostream.h>
void main()
{
        unsigned long A=546213;
        bool T=false;
        unsigned long *pA=&A;
        bool *pT=&T;
        cout<<sizeof pA<<endl;
        cout<<sizeof (pA)<<endl;
        cout<<sizeof pT<<endl;
        cout<<sizeof (pT)<<endl;
}

4
4
4
4
Press any key to continue
```

4. Покажчики і масиви

У мові C (C++) є сильний зв'язок між покажчиками масивами. У мові прийнято, що ім'я масиву – це адреса пам'яті, починаючи з якої розташований масив. Іншими словами, ім'я масиву – це адреса першого (нульового) елементу масиву.

Якщо оголошений масив

```
int  Mas[10];
```

то Mas є покажчиком на масив, точніше на перший елемент масиву

Mas ⟶ Mas [0];

Якщо ми запишемо:

```
int  Mas[10];
int *pM;
pM = Mas;
```

або

```
int  Mas[10];
int *pM;
pM = Mas[0];
```

то ці оператори приведуть до одного і того ж результату.

Для того, щоб набути значення 6-го елементу масиву Mas, можна написати

Mas [5] або *(Mas+5).

Результат буде один і той же.

Відповідність між двома формами виразів настільки строга, що C++ розглядає елемент масиву Mas[n] як *(Mas[0]+n).

Розглянемо приклад програми, яка вводить в пам'ять в клавіатури цілочисельний масив, обчислює суму його елементів і виводить на екран початковий масив і суму його елементів

Без вживання покажчиків ця програма матиме наступний вигляд.

```
#include <iostream.h>
void main()
{
      const int M=50;
      int i, l, S, Mas[M];
      cout<<"Введіть розмір масиву l:"<<endl;
      cin>>l;
      cout<<"Введіть масив:"<<endl;
      for (i=0; i<l; i++) cin>>Mas[i];
      cout<<endl;
      S=0;
      for (i=0; i<l; i++) S=S+Mas[i];
      for (i=0; i<l; i++) cout<<Mas[i]<<' ';
      cout<<endl<<"S= "<<S<<endl;
}
```

Результат рішення задачі.

```
Введіть розмір масиву l:
5
Введіть масив:
1 2 3 4 5

1 2 3 4 5
S= 15
Press any key to continue
```

Запишемо цю програму з використанням покажчиків.

```
#include <iostream.h>
void main()
{
```

```
    const int M=50;
    int i, l, S, Mas[M];
    int *pMas;
    pMas=Mas;
    cout<<" Введіть розмір масиву l:"<<endl;
    cin>>l;
    cout<<" Введіть масив:"<<endl;
    for (i=0; i<l; i++, pMas++) cin>>*pMas;
    cout<<endl;
    S=0;
    pMas=pMas-l;
    for (i=0; i<l; i++, pMas++) S=S+*pMas;
    pMas=pMas-l;
    for (i=0; i<l; i++, pMas++) cout<<*pMas<<' ';
    cout<<endl<<"S= "<<S<<endl;
}
```

Результат рішення задачі матиме наступний вигляд

```
Введіть розмір масиву l:
7
Введіть масив:
1 2 3 4 5 6 7

1 2 3 4 5 6 7
S= 28
Press any key to continue
```

Програма майже не змінилася і виконує такі ж операції як і попередня. Проте в С++ покажчики використовуються частіше ніж робота з індексами.

Висновок. У мові С (С++) є можливість здійснювати безпосередній доступ до пам’яті. Для цього передбачений спеціальний тип змінних – *покажчики.*

Покажчик – є змінною, яка містить адресу деякого об’єкту.

Оскільки покажчик є посиланням на деяку область пам’яті, йому може бути присвоєна тільки адреса змінної, а не саме її значення.

Покажчики допомагають здійснювати безпосередній доступ до пам’яті. Для того, щоб набути (прочитати) значення, записане за адресою, яка знаходиться в покажчику, використовують операцію непрямого звернення або розіменування (*). Для цього використовується ім’я покажчика із зірочкою перед ним.

Питання для самоконтролю.

1. У програмі на С (С++) є такий фрагмент:

```
unsigned  int  Var1 = 4000;     cout << &Var1;
```

Що буде видане на екран:

1. Значення Var1, тобто 4000

2. Адреса, за якою записано значення Var1
3. Повідомлення про помилку

2. Укажіть правильне визначення покажчика в С (С++) :
   1. *float pf;
   2. float*pf;
   3. float pf*;
3. У програмі на С (С++) є такий фрагмент:

```
char ch = 'y';   char *pch = &ch;
```

Чи можлива така ініціалізація покажчика:
   1. Можлива
   2. Не можлива
   3. Такої конструкції в С (С++) немає
4. У програмі на С (С++) є такий фрагмент:

```
float x = 10.1, y; float * pf;
pf = &x; y = *pf;
```

Яке значення прийме y:
   1. y = 10.1
   2. У змінну **y** запишеться адреса, за якою знаходиться значення **x**
   3. Буде видано повідомлення про помилку з вказівкою на останній рядок
5. Чи можна у мові С (С++) виконувати арифметичні операції над покажчиками:
   1. Можна
   2. Не можна
   3. Можна виконувати тільки операцію присвоювання

6 У програмі на С (С++) зустрівся такий фрагмент

```
int i, mas[ i ]; int *pmas=mas;
```

Що визначає останній оператор:
   1. Розіменування покажчика
   2. Опис покажчика
   3. Опис покажчика і привласнення йому початкової адреси масиву
7. В оголошенні змінних зустрівся такий запис int *px[7]. Це:
   1. Оголошення покажчика
   2. Оголошення масиву покажчиків
   3. Таку конструкцію в мові С використовувати не можна
8 У програмі зустрівся такий фрагмент

```
int y=1, b; int *px;
px= &y; b=*px;
```

Чому дорівнює значення b?
9. У програмі на С (С++) є такий фрагмент

```
int *prt;   prt= new int [100];
```

Останній оператор визначає, що це:
   1. Покажчику prt привласнено значення 100
   2. Покажчику привласнено початкову адресу динамічного масиву
   3. Покажчику привласнено значення змінної new
10. У програмі є такий фрагмент

```
int *prt;
for (i=0; i<100; i++) cout<<*(prt+i)<<" ";
```

Це виведення на екран дисплею:
   1. Значень елементів якогось масиву
   2. Значень покажчиків
   3. Таку конструкцію у мові С (С++) використовувати не можна

11. Укажіть на можливість такого оголошення покажчика

```
int ** pprt;
```

1. Можливо
2. Не можливо
3. Все залежить від змісту програми

12 У програмі є такий фрагмент

```
int  S1[20];
int *pS1=S1;
……
pS1++;
……
```

На скільки байтів зміниться значення pS1

13. Кожній змінній програми пам'ять може виділятися статично. Цей процес здійснюється при:
    1. Набору програми на екрані дисплея
    2. Компіляції програми
    3. Виконанні програми
14. Змінним програми пам'ять може виділятися динамічно. Цей процес здійснюється при:
    1. Набору програми на екрані дисплея
    2. Компіляції програми
    3. Виконанні програми
15. Виділення динамічної пам'яті під змінну здійснюється за допомогою оператора:
    1. new
    2. main
    3. case

Лекція 12
ФУНКЦІЇ

Мета лекції. Вивчити особливості побудови функцій у мові С (С++) і їх використання в програмах.

Основні питання лекції.

1. Визначення (опис) функції.
2. Прототипи функцій.
3. Виклик функцій.
4. Область дії і область видимості змінних

Принципи програмування на мові С (С++) засновані на понятті функції. Функція – це самостійна одиниця програми, яка створена для розв'язання конкретної задачі.

У загальному вигляді, програма на С++, складається з головної функції main() і набору окремих функцій.

```
#include <iostream.h>
void main()             //головна функція
{                       // початок тіла головної функції
        funccion 1();   // виклик функції 1
        funccion 2();   // виклик функції 1
        funccion 3();   // виклик функції 1
}                               //кінець головної функції
funccion 1();
{
Тіло функції 1;
}
funccion 2();
{
Тіло функції 2;
}
funccion 3();
{
Тіло функції 3;
}
```

1. Визначення (опис) функції.

Кожна функція, яку передбачається використовувати в програмі, повинна бути визначена (описана). Основна форма опису функції має вигляд.

```
Тип ІмяФункциі (список параметрів)
{
  Тіло функції
}
```

Визначення (опис) функції складається із *заголовка* функції і *тіла* функції.

У заголовку Тип визначає тип значення, яке повертає функція. Якщо тип не вказаний, то за умовчанням передбачається, що функція повертає ціле значення (тип int).

Список параметрів складається з переліку *типів і імен параметрів*, розділених комами. Функція може не мати параметрів, але круглі дужки необхідні завжди.

У списку параметрів для *кожного параметра* повинен бути вказаний *тип*.

```
func ( int x, int y, float z ) // правильний список параметрів
func ( int x,  y, float z ) // не правильний
```

Якщо функція повертає значення, відмінне від типу void, то в тілі функції обов'язково повинен бути присутній оператор return з параметром того ж типу, що і значення, що повертається.

Оператор return має два варіанти використання.

1. Цей оператор викликає негайний вихід з поточної функції і повернення в ту програму. , що її викликала .
2. Цей оператор використовується для повернення значення функції

Якщо значення, що повертається, не використовується надалі в програмі, оператор return слідує без параметра або взагалі може бути опущений. У цьому випадку повернення в програму здійснюється після досягнення дужки, що закривається.

Приклад визначення (опису) функції зведення числа a в натуральний ступінь b.

$$P = a^b; \qquad P = a*a*a*a \ldots a = \prod_{i}^{b} a;$$

```
float step(float a, iтt b)
{
        int i;
        float P;
        if(a<0) return (-1);
        P=1;
        for (i=0; i<b; i++) P *= a;
        return P;
}
```

Інший приклад – функція для знаходження найбільшого з двох чисел.

```
Max(int a, int b)
{
        int m;
        if(a>b) m=a;
        else m=b;
        return m;
}
```

У разі, коли оператора return в тілі функції немає або за ним немає значення, то значення, що повертається функцією, невідоме (не визначено). Якщо функція повинна повертати значення, але не робить цього, компілятор видає попередження.

Усі функції, що повертають значення, можуть використовуватися у виразах мови С (С++).

Коли функція не повертає ніякого значення, вона повинна бути описана як функція типу void (порожня).

2. Прототипи функцій.

Особливістю мови С (С++) є те, що для створення правильного машинного коду функції компілятору необхідно повідомити (до першого виклику функції) тип результату, що повертається, а також кількість і типи аргументів. Для цієї мети в С++ використовується поняття *прототипу функції*. Прототип функції задається таким чином.

Тип ІмяФункйії (список параметрів);

Використання прототипу функції є *оголошенням* функції. Частіше за все прототип функції співпадає із заголовком функції. На відміну від заголовка функції прототип *закінчується крапкою з комою*.

При оголошенні функції компілятору важливо знати *ім'я* функції, *кількість і тип* параметрів, також *тип результату*, що повертається. При цьому, імена формальних параметрів не грають ніякої ролі.

Тому прототип функції може бути виглядати так:

int func (int a, float b. float c);

int func (int, float, float);

Два ці оголошення рівносильні.

3. Виклик функцій.

Для того, щоб функція виконувала певні дії в програмі, вона повинна *бути викликана.* При зверненні до функції вона виконує поставлену задачу. Після закінчення роботи функції, вона повертає у якості результата деяке значення.

Виклик функції здійснюється шляхом вказівки в програмі її імені (ідентифікатора), за яким в круглих дужках слідує список аргументів, розділених комами.

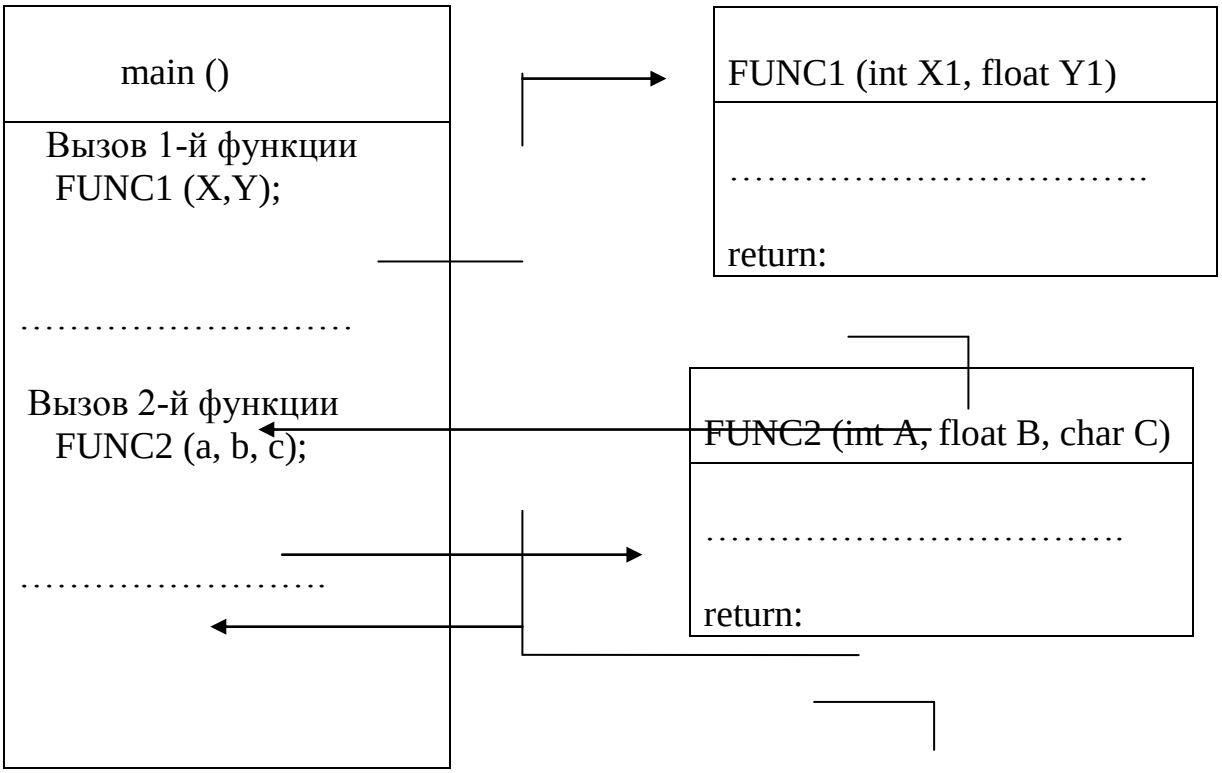
ІмяФункциі (арг.1, арг.2, арг.N);

Кожний аргумент функції є змінною, виразом або константою. Вони передаються в тіло функції для подальшого використовування в обчислювальному процесі. Список аргументів може бути порожнім.

Функція може викликати інші функції (одну або декілька). А ті, у свою чергу, проводити виклик третіх і т.д. Крім того, функція може викликати саму себе. Це явище в програмуванні називається рекурсією.

Будь-яка програма на С (С++) *обов'язково* включає головну функцію main (). З цієї функції починається виконання програми.

Розглянемо схематично порядок виклику функції.



Програма починає виконуватися з функції main () до виклику функції FUNC1 (X,Y). Після чого, управління передається у функцію FUNC1(). На місце параметра X1 передається значення змінної X, а на місце змінної Y1 передається значення Y. Далі виконується тіло функції FUNC1 () до тих пір, поки не зустрінеться оператор return, який передає управління назад в тіло функції main ().

_Після цього продовжується виконання функції main () до тих пір, поки не відбудеться виклик функції FUNC2 (a, b, c). При виклику цієї функції на місце змінної A передається значення a, на місце змінної B – значення змінної b і на місце C – значення змінної c. Після виконання тіла функції FUNC2() управління знов передається в тіло функції main ().

Аргументи, які вказані в заголовку функції, носять назву *формальних*. В тілі функції вони не приймають ніяких значень.

FUNC1 (int X1, float Y1) – тут X1 і Y1 формальні параметри.

Аргументи, які вказані в імені функції при її виклику, називаються фактичними.

FUNC1 (X,Y) – тут X і Y фактичні параметри.

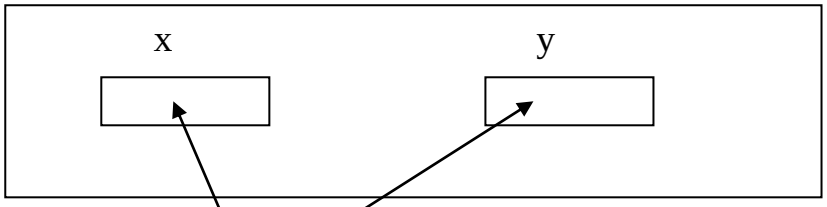
Фактичні параметри приймають конкретні значення і вони передаються формальним параметрам.

У мові C (C++) є особливість, зв'язана з тим, що всі аргументи функції передаються по значенню.

При трансляції функції в стеку виділяється місце для її формальних параметрів. В це виділене місце заносяться значення фактичних параметрів, тобто значення параметрів при виклику функції. Далі функція використовує ці параметри .

Є функція float f1 (float x, float y).

Стек



Виклик функції f1 (a,b)

Розглянемо приклад програми з використанням функції. Хай це буде функція, яка обчислює факторіал.

y = n !;

Цей вираз можна записати таким чином.

$y = n ! = 1*2*3*4* \ldots *n = \prod_{j=1}^{n} j;$

Тоді алгоритм обчислення добутку запишеться у вигляді:

1. y = 1;
2. y = y*j.

Програма функції обчислення факторіалу матиме вигляд.

```
int fac(int n)
{
    int j,y;
    y=1;
    for(j=1; j<=n; j++) y=y*j;
    return y;
}
```

Розробити програму для обчислення числа сполучень із N елементів по M.

$R = C_N^M = \frac{N!}{M!(N-M)!};$

Програма обчислення значення величини R матиме вигляд.

```
// число сполучень
    #include <iostream.h>
    int fac(int x);    //прототип функції обчислення факторіалу
    void main()
    {
        int N,M,R;
        cout<<"Vvedite N i M :"<<endl;
        cin>>N>>M;     // введення значень  N і M
        R=fac(N)/(fac(M)*fac(N-M)); //звернення до функції   cout<<endl<<"R=
"<<R<<endl;              // виведення значення R
    }
    int fac(int x) //  функція обчислення факторіалу
    {
        int j,y;
```

```
        y=1;
        for(j=1; j<=x; j++) y=y*j;
        return y;
}
```

Вид екрану при виконанні програми.

```
Vvedite N i M :
5 3

R= 10
Press any key to continue
```

4. Область дії і область видимості змінних

Область дії змінної – це правила, які встановлюють, які дані доступні у даному місці програми.

У мові С (С++) кожна функція – це окремий блок програми. Увійти в тіло функції не можна інакше, як через виклик даної функції.

З точки зору області дії змінних розрізняють три типи змінних:

- локальні;
- глобальні;
-формальні.

Правила області дії визначають, *де* кожна з них може застосовуватися.

Локальні змінні **–** це змінні, оголошені у середині блоку, зокрема у середині функції. Локальна змінна доступна у середині блоку, в якому вона оголошена. Блок відкривається фігурною дужкою і закривається фігурною дужкою. Область дії локальної змінної – б л о к.

Приклад.

```
#include<iostream.h>
void f (void); //прототип функції f()
void main() //головна функція  main()
{
        int i=1; //оголошення змінної  i  в головній функції
        f (); // виклик функції  f ()
        cout<<"В main() i= "<<i<<endl; // друк значення  i в main()
}
void f(void) //опис функції f ()
{
        int i;
        i=10;// змінній і присвоюємо значення 10
        cout<<"В f() i= "<<i<<endl;// друк значення i  в f ();
}
```

Вид екрану після виконання програми.

```
У f() i= 10
У main() i= 1
```

Press any key to continue

Приклад показує, що при виклику функції f() значення i, оголошене в main (), не змінилося.

Формальні змінні – це змінні, оголошені при описі функції як її аргументи. Формальні параметри використовуються в тілі функції, як локальні змінні. Область дії формальних параметрів – б л о к, яким є тіло функції.

Глобальні змінні – це змінні, оголошені поза якою-небудь функцією. Вони можуть бути використані в будь-якому місці програми. Область дії глобальної змінної – вся програма.

Приклад оголошення глобальної змінної.

```
#include <iostream.h>
int var;  // var – глобальна змінна
void main ()
{
.....
```

Розглянемо ще один приклад розробки програми з використанням функцій.

Імовірність обслуговування заявки в системі масового обслуговування (СМО) визначається за формулою.

$$P= 1 - \frac{\alpha^n}{n!\sum_{k=0}^{n}\frac{\alpha^k}{k!}};$$

де:

α – потік заявок на обслуговування;

n – кількість приладів обслуговування.

Введемо додаткову змінну S.

$$S = \sum_{k=0}^{n}\frac{\alpha^k}{k!};$$

Для розв'язання поставленої задачі необхідно розробити програми двох функцій: обчислення факторіалу і зведення в ступінь.

Функція обчислення факторіалу у нас вже є.

Розробимо програму зведення в ступінь.

$$Y = C^R = C*C*C* \ldots *C = \prod_{j=0}^{R} C;$$

Програма функції зведення в ступінь матиме вигляд.

```
float -й(float c, int R)              // заголовок функції
{
        int j;                        //параметр циклу
        float y;                      // змінна y
         y=1;                         // присвоєння y початкового значення
        for(j=0; j<=R; j++)           // організація циклу для обчислення y
         {
                if(j= =0) y=1;
```

```
            else y=y*c;                 // обчислення  y
        }
        return y;                       // повернення обчисленого значення  y
}
```

Програма обчислення імовірності матиме вигляд.

```
#include <iostream.h>
float st (float c, int R);          //прототип функції зведення в ступінь
int fac (int x);                    // прототип функції обчислення факторіалу
void main ()                        // головна функція main ()
{
        int N,k;
        float A,S,P;
        cout<<"Введіть A і N :"<<endl; // введення  A і N
        cin>>A>>N;
        S=0;
        For (k=0; k<=N; k++) S = S+st (A,k) / fac (k); // обчислення  S
                        P = 1- st (A,N) / (fac (N)*S);          //обчислення P
        cout<<endl<<"P= "<<P<<endl; /
}
float st (float c, int R) // функція зведення в ступінь
{
        int j;
        float y
        for(j=0; j<=R; j++)
        {
                if(j= =0) y=1;
                else y=y*c;
        }
        return y;
}
int fac(int x)  // функція обчислення факторіалу.
{
        int j,y;
        y=1;
        for(j=0; j<=x; j++)
        {
                if(j= =0) y=1;
                else y=y*j;
        }
        return y;
}
```

Вид екрану після виконання програми.

```
ВведітьA і N :
2  4
```

```
P= 0.904762
Press any key to continue*/
```

Із програми видно, що функції fac () і st () використовують одні і ті ж змінні j і y. Тут чітко виконується принцип дії і видимості локальних змінних.

Висновок. Принципи програмування на мові С (С++) засновані на понятті функції. Функція – це самостійна одиниця програми, яка створена для розв'язання конкретної задачі.

У загальному вигляді, програма на С++, складається з головної функції main() і набору окремих функцій.

Питання для самоконтролю.

1. При зверненні до функції реальні і формальні параметри повинні співпадати:
   1. По кількості і місцю
   2. По кількості, типу і місцю
   3. По кількості і типу
2. Звернення до функції здійснюється :
   1. По заголовку функції
   2. За ім'ям функції
   3. За допомогою спеціального оператора
3. Функції можуть бути:
   1. Тільки з параметрами
   2. Тільки без параметрів
   3. З параметрами і без параметрів
4. У програмі використовуються дві незалежні функції F1 і F2.
   У якому порядку вони повинні бути описані в розділі прототипів:
   1. Спочатку F1, а потім F2
   2. Спочатку F2, а потім F1
   3. Не має значення в якому порядку
5. У масиві A необхідно визначити Max елемент і його номер. Чи можливо цей блок оформити функцією:
   1. Можливо
   2. Можливо в окремих випадках
   3. Не можливо

6 У програмі описано дві функції F1 і F2. Чи можливо в них використовувати одні і ті ж змінні:
   1. Можливо
   2. Не можливо
   3. Можливо в окремих випадках
7. Окремий блок програми доцільно оформити функцією, якщо в ньому є :
   1. Один результат
   2. Два і більш результату
   3. Не має значення скільки результатів

8 У програмі описана функція int f1(int x);
   Укажіть на можливість такого звернення до функції:

```
r := f1(n)/(f1(m)*f1(n-m));
```

   1. Можливо
   2. Не можливо
   3. Можливо в окремих випадках

Лекція 13
ФУНКЦІЇ І МАСИВИ

Мета лекції. Вивчити особливості використання функцій при обробці масивів

Основні питання лекції.

1. Функції і масиви.
2. Глобальні змінні.
3. Рекурсивні функції.
4. Класи пам'яті.

1. Функції і масиви.

Якщо в якості аргументу функції використовується масив, то необхідно вказати адресу початку масиву і його розмір.

Тоді заголовок функції, яка обробляє масив, необхідно записати таким чином. Наприклад.

```
float func (float mas[n]);
float func (float mas[ ], int n);
float func (float *mas, int n);
```

Виклик функції func () з основної програми запишеться таким чином.

```
func (arr, k);
```

Розглянемо приклад.

Заданий масив A з N довільних чисел. Сформувати новий масив B, кожний елемент якого дорівнює додатку від ділення елементу масиву A на його максимальний елемент. На екран видати початковий масив A, його максимальний елемент і масив B. Пошук максимального елементу масиву A оформити функцією.

Визначимо спочатку функцію знаходження максимального елементу масиву.

```
float max(float arr[ ],int n)   // заголовок функції
{
        int j;                        // параметр циклу
        float C=arr[0];          // змінній C присвоюємо нульовий елемент
        for(j=0; j<n; j++)
        if (arr[j]>C) C=arr[j];   // перегляд масиву і визначення Max
                return 3;            // повернення Max елементу
}
```

Програма рішення поставленої задачі матиме вигляд.

```
#include <iostream.h>
float max(float arr[],int n);
void main()
{
        const int N=50;             //максимальний розмір масиву
        int i,l;                        //параметр циклу і реальний розмір масиву
        float A[N],M;                  // масив A[N] і змінна M
        cout<<"Введіть розмір масиву l:"<<endl;
        cin>>l;                          // введення реального розміру масиву
```

```
        cout<<"Введіть масив:"<<endl;
        for(i=0; i<l; i++) cin>>A[i];        // введення масиву A
        cout<<endl;
        M=max(A,l);                          // звернення до функції max()
        cout<<"Початковий масив "<<endl;
        for(i=0; i<l; i++) cout<<A[i]<<' '          ;//друк початкового масиву
        cout<<endl<<"Max= "<<M<<endl;      //друк Max елементу масиву
        cout<<"Масив B "<<endl;
        for(i=0; i<l; i++) cout<<A[i]/M<<' ';     // друк масиву B
        cout<<endl;
}

float max(float arr[],int n)      // функція пошуку максимального елементу
{
        int j;
        float C=arr[0];
        for(j=0; j<n; j++)
        if (arr[j]>C) C=arr[j];
                return 3;
}
```

Вигляд екрану після виконання програми.

```
Введіть розмір масиву l:
5
Введіть масив:
2  7  13  9  6

Початковий масив
2  7  13  9  6
Max= 13
Масив B
0.153846  0.538462  1  0.692308  0.461538
Press any key to continue
```

2. Глобальні змінні.

Функція повертає одне значення. Проте іноді виникає необхідність, щоб функція обчислювала декілька значень. Для таких цілей необхідно використовувати глобальні змінні.

Розглянемо приклад використання глобальних змінних.

Заданий масив A з N довільних чисел. Знайти максимальний елемент цього масиву і його номер. На екран видати максимальний елемент і його номер. Пошук максимального елементу масиву і його номера оформити функцією.

Визначимо спочатку функцію знаходження максимального елементу і його номера.

```
float Max(float arr[],int n)                //заголовок функції
```

```
{
        int j;                                  // параметр циклу
        B=arr[0];  // глобальній змінній B присвоюємо нульовий елемент
        for(j=0; j<n;j++)
                if(arr[j]>B)
        {
                B=arr[j];               // визначаємо максимальний елемент
                k=j;                // його номер заносимо в глобальну змінну k
         }
        return 0;                   // функція не повертає ніяких значень
}
```

Програма для вирішення поставленої задачі має вигляд.

```
// Використання глобальних змінних
#include <iostream.h>
float Max(float arr[], int n);      //прототип функції Max()
float B;                            // глобальна змінна B
int k;                              // глобальна змінна k
void main()
{
        const int N=50;             //максимальний розмір масиву
        int i,l;                    //параметр циклу і реальний розмір масиву
        float mas[N];               // масив mas[N]
        cout<<"Введіть розмір масиву  l:"<<endl;
        cin>>l;                         // введення реального розміру масиву
        cout<<endl<<"Введіть масив:"<<endl;
        for(i=0; i<l; i++) cin>>mas[i];       // введення масиву mas[i]
        cout<<endl;
        Max(mas,l);                         //звернення до функції Max()
        cout<<"Max= "<<B<<endl;          //друк Max елементу масиву
        cout<<"Його №   "<<k<<endl;   //друк № елементу масиву
}

float Max(float arr[],int n)  //Функція пошуку max елементу і його №
{
        int j;
        B=arr[0];
        for(j=0; j<n;j++)
                if(arr[j]>B)
        {
                B=arr[j];
                k=j;
        }
        return 0;
}
```

Вигляд екрану після виконання програми.

```
Введіть розмір масиву  l:
5

Введіть масив:
3 9 12 7 5

Max= 12
Його №  2
Press any key to continue
```

3. Рекурсивні функції.

У мові С (С++) функції можуть викликати самі себе. Функція називається рекурсивною, якщо оператор в тілі функції містить виклик цієї ж функції.

Класичний приклад рекурсивної функції – це обчислення факторіалу числа N.

$$N! = 1*2*3*4.*N = \prod_{j=1}^{N} j;$$

Програма має вигляд.

```
int fak(int n)
{
	int a;
	if(n= =1) return 1;
	a=fak (n-1)*n;     //функція викликає саму себе
	return a;
}
```

Виклик функції в рекурсивній функції не створює нову копію функції, а створює в пам'яті нові копії локальних змінних і параметрів. З рекурсивної функції необхідно передбачити вихід.

Розглянемо приклад використовування рекурсивної функції.

$$R=C_N^M = \frac{N!}{M!*(N-M)!};$$

У програмі використовуємо дві функції обчислення факторіалу. Одна – класична fac(), друга – рекурсивна fak().

Програма матиме наступний вигляд.

```
#include <iostream.h>
int fac(int k); // прототип класичної функції
int fak(int n); // прототип рекурсивної функції
void main()
{
	int N,M,R,R1;
	cout<<"Введіть N і M:"<<endl;
	cin>>N>>M;
	cout<<endl;
	R=fac(N)/(fac(M)*fac(N-M));//обчислення R класичною функцією
```

```
        cout<<"R= "<<R<<endl;
        R1=fak(N)/(fak(M)*fak(N-M));// обчислення R1 рекурсивною функцією
        cout<<"R1= "<<R1<<endl;
}

int fac(int k) //класична функція
{
        int j,y;
        y=1;
        for(j=1;j<=k; j++)  y=y*j;
        return y;
}

int fak(int n) //рекурсивна функція
{
        int a;
        if(n= =1) return 1;
        a=fak(n-1)*n;
        return a;
}
```

Вид екрану після виконання програми.

```
Введіть N і M:
7 3

R= 35
R1= 35
Press any key to continue
```

Як бачимо, результат – однаковий.
Рекурсія буває:
- простою – якщо функція в тілі містить виклик самої себе;
- непрямою – якщо функція викликає іншу функцію, а та у свою чергу викликає першу.

4. Класи пам’яті.

У мові С (С++) є інструмент, що дозволяє управляти механізмами використання пам'яті. Цей інструмент – класи пам'яті. Кожна змінна належить до одного з 4-х класів пам'яті. Вони описуються наступними ключовими словами.

auto – автоматична;
extern – зовнішня;
static – статична;
register – регістрова.

Тип пам'яті вказується модифікатором – ключовим словом, що стоїть перед типом змінної.

```
static int sum;
register int plus;
```

Якщо модифікатора пам'яті перед типом змінної при її оголошенні *немає,* то за умовчанням вона належить класу auto. Тому цей модифікатор практично ніколи не використовується.

Змінні auto мають локальну область дії. Вони відомі тільки у середині блоку, в якому вони визначені. Інші функції можуть використовувати ці ж імена. Змінна auto створюється (тобто їй відводиться місце в пам'яті) при вході у функцію. При виході з функції змінна auto пропадає, а область пам'яті, в якій знаходилася ця змінна, вважається вільною і може використовуватися для інших цілей. Змінні auto зберігаються в оперативній пам'яті.

Регістрові змінні зберігаються в регістрах. Доступ до змінних, що зберігаються в регістрах, швидше, ніж до тих, що зберігаються в оперативній пам'яті. Регістрові змінні аналогічні змінним auto. Регістрова пам'ять процесора невелика. І якщо доступних регістрів немає, то вона стає змінною auto. Опис регістрової змінної.

```
register float a;
```

Зовнішня змінна відноситься до глобальних змінних. Вона може бути оголошена як зовні, так і усередині функції. Наприклад.

```
void f (void)
{
  extern int j;       //оголошення  j усередині функції
.....
}
```

Поява ключового слова extern пов'язана з модульністю програм на С (С++). На С (С++) можна складати багато файлову програму і роздільно компілювати кожний файл. Якщо в одному з файлів описати зовні тіла функції глобальну змінну

```
float global;
```

то для неї виділяється місце в пам'яті в розділі глобальних змінних і констант. Якщо використовувати цю глобальну змінну в іншому файлі, то при роздільній компіляції без *додаткового оголошення* змінної компілятор не знатиме, що це за змінна. Використання оголошення

```
extern float global;
```

не приведе до виділення пам'яті, а повідомляє компілятор, що така змінна описана в іншому файлі.

Оголошення зовнішньої змінної може бути як поза функцією, так і у середині неї.

Якщо це ж ім'я без слова extern оголошено у середині функції, то під цим ім'ям буде створена інша змінна типу auto.

Статичні змінні. Область дії локальної статичної змінної – вся програма. Місце в пам'яті під локальні статичні змінні виділяється на початку роботи програми в розділі глобальних змінних. Проте область видимості локальних статичних змінних така ж, як і у auto. Значення статичних змінних зберігається від

одного виклику функції до іншого. Локальні статичні змінні ініціалізується нулем. При цьому, опис з ініціалізацією

```
static int M1=10;
```

викликає однократну ініціалізацію змінної M1 при виділенні під неї місця. При подальших викликах функції, в якій описана ця змінна, ініціалізації не відбувається. Це дозволяє використовувати таку змінну для лічильника числа викликів функції.
Приклад.

```
#include <iostream.h>
void tristan(void);
void main()
{
        int i;
        for(i=1; i<=3;i++)
        {
                cout<<"visov # "<<i<<endl;
                tristan();
        }
}

void tristan(void)
{
        int auto_l=1;
        static int stat_l=1;
        cout<<"auto_l= "<<auto_l++<<endl;
        cout<<"stat_l= "<<stat_l++<<endl;
}
```

Вигляд екрану після виконання програми.

```
visov # 1
auto_l= 1
stat_l= 1
visov # 2
auto_l= 1
stat_l= 2
visov # 3
auto_l= 1
stat_l= 3
Press any key to continue
```

Аналіз результатів показує, при вході у функцію змінна auto_l *кожного разу* ініціалізується *одиницею,* а змінна stat_l ініціалізується тільки один раз при першому входженні у функцію.

Можна описати також глобальну зовнішню статичну змінну, тобто описати змінну типу static поза будь-якою функцією. Відмінність зовнішньої змінної від зовнішньої статичної змінної полягає в області їх дії. Звична зовнішня змінна може

використовуватися функціями в будь-якому файлі. В той час як зовнішня статична змінна тільки функціями того файлу, де вона описана.

Приведемо наступну таблицю.

Клас пам'яті	Ключове слово	Час присутності	Область дії
Автоматичний	auto	Тимчасово	Блок
Регістровий	register	Тимчасово	Блок
Статичний локальний	static	Постійно	Блок
Статичний глобальний	static	Постійно	Файл
Зовнішній	txtern	Постійно	Програма

І ще раз! В програмі може бути описано декілька змінних з одним і тим же ім'ям, але тільки в різних блоках.

Висновок. Якщо в якості аргументу функції використовується масив, то необхідно вказати адресу початку масиву і його розмір.

Тоді заголовок функції, яка обробляє масив, необхідно записати таким чином. Наприклад.

```
float func (float mas[n]);
float func (float mas[ ], int n);
float func (float *mas, int n);
```

Виклик функції func () з основної програми запишеться таким чином.

```
func (arr, k);
```

У мові С (С++) є інструмент, що дозволяє управляти механізмами використання пам'яті. Цей інструмент – класи пам'яті. Кожна змінна належить до одного з 4-х класів пам'яті. Вони описуються наступними ключовими словами.

auto – автоматична;
extern – зовнішня;
static – статична;
register – регістрова.

Тип пам'яті вказується модифікатором – ключовим словом, що стоїть перед типом змінної.

```
static int sum;
register int plus;
```

Якщо модифікатора пам'яті перед типом змінної при її оголошенні *немає,* то за умовчанням вона належить класу auto. Тому цей модифікатор практично ніколи не використовується.

Питання для самоконтролю.

1 У якості аргументу функції використовується одномірний масив із 10 елементів – Mas[10]. Звернутися до функції можна так:
   1. func (int Mas[10])
   2. func (int Mas[], 10)
   3. Не має значення як.

2. У якості аргументу функції використовується двомірний масив W[3][3].
Укажіть неправильне звернення до функції:
   1. func (int W[3][3]);
   2. func (int W[][3]);
   3. func (int W[][]);

3. Коли функція не повертає ніякого значення, чи можна її використовувати у виразах мови С (С++)
   1. Можна
   2. Не можна
   3. Можна ,коли вираз приймає певне значення

4. Коли функція не повертає ніякого значення, вона повинна бути оголошена як функція типу:
   1. int
   2. void
   3. char

5. Параметри функції – це змінні, які
   1. Оголошені в функції
   2. Оголошені поза функцією
   3. Не має значення де

6. Укажіть правильний запис заголовка функції
   1. func (int x, y, float z)
   2. func (x, int y, float z)
   3. func (int x, int y, float z)

7 Оператор return:
   1. Викликає вихід з функції
   2. Використовується для повернення значення функції
   3. І перше і друге

8. У тілі функції оператор return
   1. Може бути відсутній
   2. Повинен бути обов'язково
   3. Все залежить від призначення функції

9 Коли в тілі функції відсутній оператор return, вихід з функції здійснюється:
   1. Коли виконається останній оператор функції
   2. У даному випадку виникне зависання програми
   3. Оператор return повинен бути в тілі функції обов'язково

10. Якщо функція не має аргументів, то в прототипі такої функції необхідно:
   1. Нічого не писати у дужках функції
   2. У дужках записати службове слово int
   3. У дужках записати службове слово void

11 Локальна змінна – це змінна, яка оголошена
   1. У тілі даної функції
   2. Поза даною функцією
   3. Яка вказана в якості параметра у заголовку функції

12. У заголовку функції записуються
   1. Формальні параметри
   2. Фактичні параметри
   3. Глобальні змінні

13. При зверненні до функції реальні параметри повинні збігатися з формальними
   1. По місцю
   2. По місцю і типу
   3. По місцю, типу і кількості

Лекція 14
СТРУКТУРИ

Мета лекції. Вивчити особливості побудови структур та їх використання при обробці даних.

Основні питання лекції.

1. Оголошення структур.
2. Масиви структур
3. Покажчики на структури

Окрім відомих нам типів даних мова С (С++) дозволяє створювати ще 5 типів даних.

1. Структури ().
2. Об'єднання ().
3. Поля бітів ().
4. Перелічений тип ().
5. За допомогою оператора створювати нове ім'я (псевдонім) для вже існуючого типу.

Розглянемо ці типи даних.

1. Оголошення структур.

Структура об'єднує деякий набір різнотипних та/або однотипних даних, сукупність яких розглядається як абсолютно новий, *призначений для користувача тип* даних.

Структура оголошується за допомогою ключового слова

struct

за яким слідує ім'я (необов'язково) для створення нового типу даних. Далі у фігурних дужках слідують змінні, які з'єднані структурою. Ці змінні називаються членами, елементами або полями структури. Оголошення поля складається з вказівки типу і імені змінної.

```
struct NewType
{
  type 1  Name 1:
  type 2  Name 2;
 .....
 type N  Name N;
};
```

Опис структури *завжди* закінчується символом крапка з комою.

Приклад.

```
struct student
{
 char name [10];
 int  kurs;
 char group [4];
```

```
};
```

Таке визначення вводить *новий створений тип даних*, який називатимемо *структурним типом*. У даному прикладі у цього структурного типу є конкретне ім'я – student.

Поки ніяка змінна не оголошена. Виділення пам'яті під змінну не відбулося. Під ім'ям student створений новий тип даних. Говорять, що заданий шаблон структури і визначений новий тип struct student.

Для того, щоб оголосити конкретні змінні *типу* struct student, необхідно записати

```
struct student stud1, stud2;
```

або

```
student stud1, stud2;
```

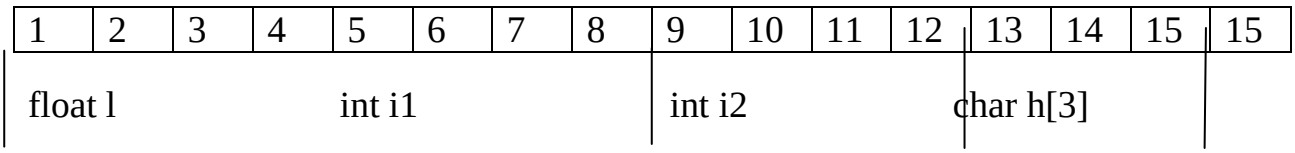
Тепер оголошено дві змінні структурного типу stud1 і stud2. Тепер компілятор виділить для них місце в пам'яті. Під кожну з них виділяється безперервна ділянка пам'яті.

Наприклад, є змінна структурного типу

```
struct str
{
  float L;
  int i1;
  int i2;
  char ch[3];
}STR;
```

Змінна STR буде розміщена в пам'яті таким чином.

Байти



Всі елементи розміщуються в пам'яті підряд і спільно займають в пам'яті стільки місця, скільки відведено структурі в цілому.

Завдання шаблона структури і оголошення змінної може проводиться в одному операторі (як це показано в попередньому прикладі). Тут одночасно задається структура з ім'ям str і оголошується змінна структурного типа STR.

Для звернення до об'єктів, що входять як елементи в конкретну структуру, використовуються складові імена. Загальною формою складового імені елементу структури є наступна конструкція.

```
Ім’яСтруктури . Ім’яЕлементу
```

Наприклад, для певної вище змінної структурного типа STR оператор

```
STR . i1 = 25;
```

присвоїть змінній i1 значення 25.

Для виведення значення змінної i1 структури STR можна використати оператор

```
cout<<STR . i1;
```

Для введення значень змінної i2 структури STR можна використовувати оператор

```
cin>>STR . i2;
```

При визначенні структур можлива їх ініціалізація, тобто завдання елементам структури початкових значень.

Наприклад.

```
struct student
 {
  char name [10];
  int  kurs;
  char group [4];
 }stud1;

strcpy ( stud1 . name " Петренко І. З " };
stud1 . kurs =1;
strcpy (stud1 . group, "РК11");
```

Як видно з прикладу, цілочисельним і логічним змінним проводиться звичайне присвоєння. А заповнення рядкових елементів структури за допомогою функції роботи з рядками strcpy (копіювання рядків).

Ініціалізацію елементів структури можна здійснювати і таким чином.

```
struct str
{
  float L;
  int i1;
  int i2;
  char ch[3];
}STR {100, 20, 30, 'a', 'b', 'c'};
```

Ініціалізація може бути проведена при оголошенні змінної структурного типу. Значення указуються у фігурних дужках через кому.

Необхідно ще раз звернути увагу на відмінність імені конкретної структури (у наших прикладах stud1, stud2, STR) від імені структурного типу (у нашому випадку student, str).

З ім'ям структурного типу не пов'язаний ніякий конкретний об'єкт. Тому з його допомогою не можна сформувати адресу звернення до елементу структури.

Буде помилкою використовувати ім'я структурного типу для іменування елементу структури

```
Str . L = 800; // помилкова конструкція;
STR . L = 800; // правильна конструкція.
```

2. Масиви структур

Один запис типу структури практично ніколи не використовується. Як правило, структурні дані об'єднуються в масиви.

Оголошення масиву структур відрізняється від звичайних масивів лише тим, що указується знов створений тип структури.

Таким чином, для створення відомості про 30 студентів навчальної групи можна оголосити наступний масив.

```
struct student
{
  char name [15];
  int  kurs;
  char group [4];
} stud1[30];
```

Або таким чином:

```
struct student stud1[30];
```

Доступ до елементів такого масиву здійснюється звичайним способом, наприклад, по індексу.

```
stud[0].group = “PK11”;
cin>>stud1[4].name;
cout<<stud1[1].kurs<<endl;
```

Розглянемо приклад.п

Є відомість про атестацію навчальної групи з дисципліни Інформатика. Розробити програму для визначення середнього бала за групу і видачі на екран списку студентів, які одержали за атестацію оцінки «добре» і «відмінно».

VED

N	FIO	OC
1	Іванов	3
2	Петров	4
3	Ткач	5
..	.....	...
M	Чумак	3

Стандартна відомість розрахована на 35 студентів. Отже, максимальний розмір масиву може бути M = 35. В кожній групі кількість студентів буде різною (тобто k).

Розглянемо початкові дані для розв’язання поставленої задачі.

VED – масив структур розміром M = 35.

Елементи структури.

N – номер по порядку, проста змінна типу int;

FIO – масив символів; максимальна кількість символів в полі FIO хай буде дорівнюватись 15;

OC – оцінка, проста змінна, тип int.

Результат

SRB – середній бал, проста змінна, тип float.

Розробка алгоритму.

У програмі необхідно виконати наступні дії.

1. Обчислити середній бал навчальної групи за атестацію

$$SRB = \frac{1}{k}\sum_{i=0}^{k} OC_i$$

Введемо позначення

$$S = \sum_{i=0}^{k} OC_i ; \quad \text{тоді} \quad SRB = S / k;$$

2. Здійснити пошук студентів за умовою (ОСі >= 4) і вивести на екран прізвища студентів, які мають оцінки за атестацію 4 і 5.

Програма матиме наступний вигляд.

```
#include <iostream.h>
void main()
{
const int M=35;                    // максимальний розмір масиву
      struct attest                // ім'я структурного типу attest
      {
            int N;                 // номер по порядку
            char FIO[15];          // прізвище, ім'я, по батькові
            int OC;                // оцінка
      }VED[M];                     // ім'я масиву структур
      int i, k;                    // параметр циклу і кількість студентів у групі
      float S, SRB;                // змінна S  і середній бал
      cout<<"Введіть k :"<<endl;
      cin>>k;                      // введення числа студентів у групі
      cout<<"Заповніть   VED :"<<endl;
      cout<<"N   "<<"FIO        "<<"OC  "<<endl;          / структура відомості
      for(i=0; i<k; i++)
      {
            cin>>VED[i].N>>VED[i].FIO>>VED[i].OC;     //введення масиву
            cout<<endl;
      }
      S=0;
      for(i=0; i<k; i++) S = S+VED[i].OC;                  //обчислення суми
      SRB = (float) S/k;                                   // обчислення середнього балу
      cout<<"SRB= "<<SRB<<endl;                            // виведення на екран SRB
      for(i=0; i<k; i++)                                   // перегляд масиву структур
            if (VED[i].OC>=4)                              // перевірка умови
                  cout<<VED[i].FIO<<"  "<<VED[i].OC<<endl;
      // виведення на екран списку студентів, що відповідають критерію
}
```

Вид екрану після виконання програми

```
Введіть  k:
3
```

```
Заповніть  VED :
N   FIO        OC
1   Ivanov      4

2   Petriv      3

3   Tkach       4

SRB= 3.66667
Ivanov  4
Tkach  4
Press any key to continue
```

3. Покажчики на структури

Оскільки ім'я структурного типу володіє правами імен типів, то можна визначити покажчик на структуру таким чином.

ІмʼяСтруктТипу *ІмʼяПокажчикаНаСтруктуру.

Наприклад.

```
struct complex
{
  float x;
  float y;
} C1;
```

Оголошення покажчика матиме вигляд.

```
struct complex *pcom;
```

Покажчик можна ініціалізувати. Значенням покажчика на структуру повинна бути адреса структури того ж типу.

Для нашого прикладу ініціалізація покажчика матиме вигляд.

```
pcom = &C1;
```

тобто покажчику pcom присвоюється адреса змінної C1. Набути значення елементу x змінної C1 можна таким чином.

```
(*pcom) .x;
```

Використання покажчиків на структуру зустрічається часто. Тому окрім (*pcom) .x є ще один спосіб. У мові C (C++) є спеціальна операція «стрілка» ->. Вона уживається замість операції крапка.

Замість (*pcom) .x ми можемо записати

```
pcom ->x;
```

Цей спосіб звичайно і застосовується.

Зауваження. Необхідно звернути увагу на круглі дужки. Операція розіменування * повинна відноситися тільки до імені покажчика, а не до повної складової адреси елементу структури.

Таким чином, наступні три вирази еквівалентні.

```
(*pcom) .x
pcom ->x;
```

C1.x;

У якості елементів структури можна використовувати масиви, структури і масиви структур

Наприклад. Нехай оголошення змінних мають вигляд.

```
struct fio
{
  char F[20];
  char I[12];
  char Про[20];
}
struct student
{
  struct fio name;
  int kurs;
  char group[4];
} stud1;
```

Тут структура fio входить до складу структури stud1. Для звернення до елементу F структури name треба записати.

stud1 . name . F = “Іваненко”;

Приклад програми.

```
#include <iostream.h>
#include <string.h>
void main()
{
	struct fio
	{
		char F[20];
		char I[12];
		char Про[20];
	};
	struct student
	{
		struct fio name;
		int kurs;
		char group[4];
	} stud1;
	strcpy(stud1.name.F, "Ivanenko");
	strcpy(stud1.name.I, "Petro");
	strcpy(stud1.name.O, "Olegovich");
	stud1.kurs =1;
	strcpy(stud1.group, "PK11");
	cout<<stud1.name.F<<endl;
	cout<<stud1.name.I<<endl;
	cout<<stud1.name.O<<endl;
	cout<<"kurs ->"<<stud1.kurs<<endl;
```

```
    cout<<"group ->"<<stud1.group<<endl;
}
```

Вигляд екрану після виконання програми.

```
Ivanenko
Petro
Olegovich
kurs ->1
group ->PK11
Press any key to continue
```

Висновок. Структура об'єднує деякий набір різнотипних та/або однотипних даних, сукупність яких розглядається як абсолютно новий, *призначений для користувача тип* даних.

Структура оголошується за допомогою ключового слова

struct

за яким слідує ім'я (необов'язково) для створення нового типу даних. Далі у фігурних дужках слідують змінні, які з'єднані структурою. Ці змінні називаються членами, елементами або полями структури. Оголошення поля складається з вказівки типу і імені змінної.

```
struct NewType
{
  type 1  Name 1:
  type 2  Name 2;
 .......
 type N  Name N;
};
```

Опис структури *завжди* закінчується символом крапка з комою.

Питання для самоконтролю.

01. 1. 1.

Структура – це сукупність:

1. Однотипних даних
2. Різнотипних даних
3. Різнотипних і (або) однотипних даних

2. Опис структури повинен обов’язково закінчуватися

1. Крапкою
2. Комою
3. Крапкою з комою

3. Оголошення структури здійснюється за допомогою ключового слова

1. switch
2. string
3. struct

4. Визначена структура

struct attest {int N, char f[20], int OC};

Скільки помилок зроблено при визначенні структури?

5. Визначена структура

struct attest {int N; char F[20]; int OC;}VED[32];

Це:

1. Масив структур
2. Змінна VED[32] типу struct attest
3. Такої конструкції в C немає

6. Визначена структура

struct attest {int N; char F[20]; int OC}VED[32];

Скільки помилок зроблено при визначенні структури?

7. Структура – це:

1. Вбудований тип даних
2. Тип даних користувача
3. Такого типу даних у мові C немає

8. Доступ до конкретного елемента структури здійснюється за допомогою операції:

1. Крапка
2. Двокрапка
3. Дефіс

9. У програмі на C зустрівся оператор strcpu (stud.name, “Іванов В.П.”); Це визначає:

1. На екран дисплея буде видано “Іванов В.П.”
2. Елементу структури stud. name буде привласнено значення Іванов В.П.
3. Таку конструкцію використовувати не можна.

10. Визначена структура:

struct. student {char name[20]; int kurs; char group[3];} stud1;

Це визначає, що:

1. Задано шаблон структури
2. Оголошена змінна stud1 типу struct. student
3. Ніяка змінна не оголошена

11. Визначена структура:

struct. student {char name[20]; int kurs; char group[3];} stud1;

і в програмі зустрівся оператор cout<<stud1.group; Що буде видано на екран:

1. stud1.group
2. Номер групи студента
3. Повідомлення про помилку

12. Чи можливо структури об’єднувати у масив:

1. Можливо
2. Не можливо
3. Можливо, коли елементи структури мають один і той же тип

13. Визначена структура:

struct. student {char name[20]; int kurs; char group[3];} stud1.kurs[20];

Цей оператор створить у пам’яті:

1. 20 змінних stud1
2. 20 змінних структури з шаблоном struct. student і іменами stud1. kurs [0], stud1. kurs [1] і т.д.
3. Таку конструкцію використовувати не можна

14. У програмі зустрівся такий фрагмент:

struct first { int a; char b;} C1; C1.a=1;

Укажіть значення поля a структури C1:

15. У програмі зустрівся такий фрагмент: struct first { int a; char b;} C1[10];

Якщо ви бажаєте звернутися до поля a 5-го елементу масиву структур, то необхідно записати :

1. C1[4]. name
2. C1[5]. name

3. C1.name [4]

16. Якщо оголошені дві змінні типу структури з одним шаблоном, чи можливо виконати таку опера цію:
stud1 = stud2:

1. Можливо
2. Не можливо
3. Можливо, коли поля цих змінних мають один і той же тип

17. Чи можливо виконувати операцію присвоювання змінних типу структури, шаблони яких описані під різними іменами (шаблони ідентичні):

1. Можливо
2. Не можливо
3. Можливо, коли поля цих змінних мають один і той же тип

Лекція 15

СТРУКТУРИ І ФУНКЦІЇ

Основні питання лекції.

1. Основні визначення.
2. Функції і масиви структур.
3. Покажчики на масиви структур

1. Основні визначення.

Структура, на відміну від масиву об'єднує, як правило, декілька змінних різного типу.

Приклад визначення структури.

```
struct student
{
  char name [20];
  int kurs;
  char group [4];
};
```

При визначенні структури ми задаємо шаблон структури і визначаємо новий тип даних struct student. Для оголошення змінної даного типу необхідно записати.

```
struct student stud1;
```

У цьому випадку stud1 є змінною структурного типу struct student. Ця змінна може бути локальною, глобальною, а також параметром функції. Отже, структуру або її елемент можна використовувати як параметр функції.

Наприклад.

```
func1 (stud1.group);
```

У якості параметра можна передати по значенню всю структуру.

Розглянемо приклад використання структури у якості параметра функції.

```
#include <iostream.h>
void f(struct str param);   //прототип функції
struct str                  //шаблон структури
{
        int x;
        char y;
};
void main()
{
        struct str arg;         // створення змінної arg типу struct str
        arg.x=1;                // присвоєння елементу x значення 1
        arg.y='A';              // присвоєння елементу y значення  A
        f(arg);                 // виклик функції
}
void f(struct str param)     //заголовок функції
{
        cout<<param . x<<endl; //виведення елементу структури x на екран
        cout<<param . y<<endl; //виведення елементу структури y на екран
```

```
}
```

Вигляд екрану після виконання програми.

```
1
A
Press any key to continue
```

Можна також створити покажчик на структуру і передавати аргумент типу структури за посиланням.

Якщо ми передаємо структуру за значенням, то всі елементи структури заносяться в стек. При передачі за посиланням в стек заноситься тільки адреса структури. Копіювання всієї структури в стек не відбувається.

Нехай є структура.

```
struct student
{
  char name[20];
  int kurs;
  char group;
} stud1;
```

Оголосимо покажчик на структуру

```
struct student *pst
pst = &stud1;
```

Набути значення елементу kurs змінній stud1 можна таким чином.

```
stud1.kurs;
(*pst) . kurs;
pst ->kurs;
```

Розглянемо приклад використовування покажчика на структуру у якості параметра функції.

```
#include <iostream.h>
    struct str          //опис шаблона структури
    {
      int x;
      char y;
    } arg;
void F(struct str *param);   //прототип функції
void main()
{
    struct str *pstr;   // створення покажчика *pstr на структуру
    pstr=&arg;          //ініціалізація покажчика адресою структури
    pstr->x=1;          // присвоєння елементу x значення 1
    pstr->y='A';        // присвоєння елементу y значення  A
    F(pstr);            // виклик функції
}
void F(struct str *param)  //заголовок функції
{
  cout<<"arg.x= "<<param->x<<endl;//виведення елементу структури x на екран
```

```
  cout<<"arg.y= "<<param->y<<endl;//виведення елементу структури y на екран
}
```

Вигляд екрану після виконання програми.

```
arg.x= 1
arg.y= A
Press any key to continue
```

2. Функції і масиви структур.

Якщо в якості аргументу функції використовується масив структур то необхідно вказати початкову адресу масиву і його розмір.

Розглянемо приклад використання масиву структур у якості параметру функції.

Є відомість атестації студентів навчальної групи з дисципліни Інформатика. Розробити програму для розрахунку середнього балу за групу і видачі на екран списку студентів, які за атестацію одержали оцінки задовільно і менше. Обчислення середнього балу оформити функцією.

VED

N	F	IN	OC
1	Іванов	І.П.	3
2	Петров	М.Ф.	2
3	Ткач	Т.П.	3
..	.....		...
M	Чумак	А.І	4

Стандартна відомість розрахована на 35 студентів. Отже, максимальний розмір масиву може бути M = 35. В кожній групі кількість студентів буде різною (тобто k).

Розглянемо початкові дані для розв’язання поставленої задачі.

VED – масив структур розміром M = 35.

Елементи структури.

N – номер за порядком, проста змінна типу int;

F – прізвище, масив символів; максимальна кількість символів в полі F хай буде дорівнюватись 15;

IN – ініціали, масив символів; максимальна кількість символів в полі IN хай буде дорівнюватись 4;

OC – оцінка, проста змінна, тип int.

Результат

SRB – середній бал, проста змінна, тип float.

У програмі необхідно виконати наступні дії.

1. Обчислити середній бал навчальної групи за атестацію.

2. Здійснити пошук студентів за умовою (OCi < 4) і вивести на екран прізвища студентів, які мають оцінки за атестацію менше 4.

Функція обчислення середнього балу повинна виконувати наступні дії

$$srb = \frac{1}{k}\sum_{i=0}^{k} OC_i$$

Введемо позначення

$$S = \sum_{i=0}^{k} OC_i ; \quad \text{тоді} \quad srb = S / k;$$

Програма функції обчислення середнього балу матиме наступний вигляд.

```
        float sb(struct attest dim[], int n)                    //заголовок функції
{
        int j, S;                                       // змінні параметру циклу і суми
        float srb;              // змінна для зберігання значення середнього балу
        S=0;                                    // присвоєння сумі початкового значення
        for(j=0;j<n; j++) S=S+dim[j].OC;                        // обчислення суми
        srb=(float) S/n;                                // обчислення середнього балу
        return srb;                     // повернення обчисленого середнього балу
}
```

Програма матиме наступний вигляд.

```
#include <iostream.h>
const int M=35;                         //максимальний розмір масиву структур
struct attest
{
        int N;
        char F[15];
        char IN[4];
        int OC;
} VED[M];                                                       //масив структур
float sb(struct attest dim[], int n);                           //прототип функції
void main()
{
        int i, k;                       //параметр циклу і кількість студентів у групі
        float SRB;                                              // середній бал
        cout<<"Введіть  k ;"<<endl;
        cin>>k;                                 // введення кількості студентів у групі.
        cout<<"Заповніть  VED :"<<endl;
        cout<<"N  "<<" F      "<<"IN   "<<"OC"<<endl;
        for(i=0; i<k; i++)                                      // введення масиву структур
        {
                cin>>VED[i].N;
                cin>>VED[i].F;
                cin>>VED[i].IN;
                cin>>VED[i].OC;
        }
        cout<<endl;
        SRB=sb(VED,k);          // виклик функції обчислення середнього балу
        cout<<"SRB= "<<SRB<<endl;       //Виведення на екран середнього балу
```

```
            for(i=0; i<k; i++)                          // перегляд відомості
            if(VED[i].OC<4)                             // перевірка умови
            cout<<VED[i].F<<" "<<VED[i].IN<<"   "<<VED[i].OC<<endl;
                    // виведення на екран студентів, які відповідають критерію
}
float sb(struct attest dim[], int n)          // функція обчислення середнього балу
{
      int j, S;
      float srb;
      S=0;
      for(j=0;j<n; j++) S=S+dim[j].OC;
      srb=(float) S/n;
      return srb;
}
```

Вигляд екрану після виконання програми.

```
Введіть k ;
4
Заповніть VED :
```

N	**F**	**IN**	**OC**
1	Іванов	І.П.	3
2	Петрів	М.Ф.	2
3	Ткач	Т.П.	3
4	Чумак	А.І.	4

```
SRB= 3
Іванов І.П.      3
Петрів М.Ф.    2
Ткач Т. П.       3
Press any key to continue
```

3. Покажчики на масиви структур

При роботі з масивами структур у якості параметра функції використовують покажчик на масив структур. Для нашого прикладу покажчик на масив структур буде визначений таким чином.

Визначимо масив структур VED[M].

```
      const int M=35;
      struct attest
      {
            int N;
            char F[20];
            char IN[4];
            int OC;
      } VED[M];
```

Покажчик на цей масив :

```
struct attest *mas;
mas=VED;
```

З урахуванням сказаного прототип функції sb матиме вигляд.

```
float sb(struct attest *dim, int n);
```

А звернення до функції запишеться таким чином.

```
SRB=sb(mas,k);
```

Програма функції матиме вигляд.

```
float sb(struct attest *dim, int n)
{
    int j, S;
    float srb;
    S=0;
    for(j=0;j<n; j++) S=S+(dim+j) ->OC;
    srb=(float) S/n;
    return srb;
}
```

Програма попередньої задачі з урахуванням використання покажчиків запишеться таким чином.

```
#include <iostream.h>
float sb(struct attest *dim, int n);
void main()
{
const int M=35;
struct attest
{
    int N;
    char F[20];
    char IN[4];
    int OC;
} VED[M];

    int i, k;
    float SRB;
    cout<<"Введіть  k :"<<endl;
    cin>>k;
    cout<<"Заповніть  VED :"<<endl;
```

```
        cout<<"N  "<<" Fam      "<<"IN   "<<"OC"<<endl;
        for(i=0; i<k; i++)
        {
                cin>>VED[i].N;
                cin>>VED[i].F;
                cin>>VED[i].IN;
                cin>>VED[i].OC;
        }
        cout<<endl;
        struct attest *mas;
        mas=VED;
        SRB=sb(mas,k);
        cout<<"SRB= "<<SRB<<endl;
          for(i=0; i<k; i++)
          if(VED[i].OC<4)
          cout<<VED[i].F<<""<<VED[i].IN<<"   "<<VED[i].OC<<endl;

}
float sb(struct attest *dim, int n)
{
        int j, S;
        float srb;
        S=0;
        for(j=0;j<n; j++) S=S+(dim+j) ->OC;
        srb=(float) S/n;
        return srb;
}
```

Вигляд екрану після виконання програми

```
Введіть k ;
4
Заповніть VED :
N   F          IN        OC
1   Іванов    І.П.       3
2   Петрів    М.Ф.       2
3   Ткач      Т.П.       3
4   Чумак      А.І.       4

SRB= 3
Іванов І.П.       3
Петрів М.Ф.       2
Ткач Т. П.        3
Press any key to continue
```

Висновок. Якщо ми передаємо структуру за значенням, то всі елементи структури заносяться в стек. При передачі за посиланням в стек заноситься тільки адреса структури. Копіювання всієї структури в стек не відбувається.

Якщо в якості аргументу функції використовується масив структур то необхідно вказати початкову адресу масиву і його розмір.

Питання для самоконтролю.

1. У якості параметра функції можна використовувати тільки:
   1. Усю структуру
   2. Тільки елемент структури
   3. Усю структуру або елемент структури
2. Чи можливо створювати покажчик на структуру:
   1. Можливо
   2. Не можливо
   3. можливо, коли є масив структур
3. Оголошена структура:

```
struct ddr { float x; float y;} C1, C2;
struct ddr *a;  Це:
```

   1. Оголошення покажчика a
   2. Оголошення змінної a типу struct ddr
   3. Таку конструкцію використовувати не можна
4. У програмі є такий фрагмент:

```
struct ddr { float x; float y;} C1;
struct ddr *a;     a=&C1;
```

Щоб звернутися до елементу x структури C1 треба записати:

   1. (*a).x
   2. a -> x
   3. Не має значення як
5. Чи можливо у якості елемента структури використовувати іншу структуру:
   1. Можливо
   2. Не можливо
   3. Можливо, коли поля цих структур мають один і той же тип
6. Оголошена структура:

```
struct ddr { int x; char y;} C1;
```

Скільки байт памяті буде виділено під змінну C1:

Лекція 16
ВВЕДЕННЯ-ВИВЕДЕННЯ ДАНИХ В Сі

Мета лекції. Вивчити особливості побудови операторів введення-виведення в програмах на мові С.

Основні питання лекції.

1. Функция printf ().
2. Функція scanf ().

У даний час дуже багато програм написано на мові С. Умові С є свої особливості введення-виведення даних. Розглянемо ці особливості.

У мові відсутні багато вбудованих в транслятор засобів, властивих розвиненим мовам програмування. В ньому немає вбудованих методів доступу до файлів, немає операцій введення-виведення даних в їх звичному розумінні. Виконання цих задач забезпечується бібліотекою підпрограм.

Функції введення-виведення зберігаються в заголовному файлі stdio.h. Щоб зв'язати програму користувача із стандартною бібліотекою введення-виведення, необхідно в програмі передбачити директиву препроцесора

include <stdio.h>

У програмі на С автоматично відкриваються три файли для введення-виведення.

stdin – стандартний файл введення (з клавіатури);

stdout – стандартний файл виведення (на дисплей);

stderr – стандартний файл повідомлень про помилки.

У стандартній бібліотеці введення-виведення міститься велике число функцій забезпечующих ведення-виведення. Розглянемо деякі з них.

Функції printf () і scanf () здійснюють форматоване введення і виведення на консоль. Форматоване введення-виведення означає, що функції можуть читати і виводити дані в різному форматі, яким можна управляти.

1. Функція printf ()

Функція printf() служить для виведення значень змінних. Її параметрами є *управляючий рядок* (або рядок форматів), укладений в подвійні лапки, і об'єкти (імена змінних), значення яких необхідно вивести на екран.

Виклик функції printf() має наступний вигляд.

printf(Управляючий рядок, [список змінних]);

Управляючий рядок містить два типи інформації:

- символи, які безпосередньо виводяться на екран;
- специфікатори перетворення (команди формату), що визначають, в якому вигляді виводити на екран значення аргументів.

Наприклад, щоб вивести на екран деяке повідомлення, його необхідно помістити як параметр функції, укладене в лапки.

printf (“ Цікаве повідомлення \ n “);

Тут, в управляючому рядку містяться тільки символи, які виводяться на екран: «Цікаве повідомлення». Комбінація \ n означає перехід на новий рядок.

Це повідомлення можна вивести на екран таким чином.

```
printf(“Цікаве“);
printf(“повідомлення“);
printf(“\ n “);
```

Якщо необхідно вивести на екран результати обчислювань, тобто значення яких-небудь змінних, то в командному рядку указуються специфікатори перетворення.

Синтаксис специфікаторів перетворення має наступний вигляд.

%[ознака] [поле] [точність] [розмір] символ типу

Обов'язковими елементами специфікатора є тільки початковий знак % і символ типу.

Символ типу може приймати різні значення.

Символ	Тип	Опис
d	Ціле	Виводити як десяткове ціле із знаком
i	Ціле	Те ж, що і d
o	Ціле	Виводити як восьмирічне без знаку
u	Ціле	Виводити як десяткове ціле без знаку
x	Ціле	Виводити як шістнадцятирічне
X	Ціле	Виводити як шістнадцятирічне
f	Дійсне	Виводити десяткове число з плаваючою крапкою [-] dddd . dddd
e	Дійсне	Виводити десяткове число в експоненціальній формі [-] d . dddd e [+|-] dd
E	Дійсне	Те ж, що і e, але із заміною e на E
g	Дійсне	Використовувати форму f або e залежно від величини числа і ширини поля
G	Дійсне	Те ж, що і g, але форма f або E
C	Символ	Вивести одиночний символ
s	Рядок	Вивести рядок
n	Покажчик	Аргумент-покажчик на змінну типу int. В неї записується кількість виведених до даного моменту символів
p		Вивести покажчик
%		Вивести символ відсотка %

Величини a, b,c – цілі десяткові числа. Видати їх значення на екран.

```
printf (“a, b, c дорівнюються : %d%d%d \n”, a, b, c);
```

Кожний модифікатор перетворення відповідає одному з аргументів, які записуються за управляючим рядком. Між ними встановлюється взаємно однозначна відповідність.

Приклад.

```
printf (“%c = %d \n”, g, g);
```

Тут значення змінної g виводиться спочатку як символ алфавіту, а після знаку дорівнює – як числове значення..

Наступний приклад – проста, але цілком закінчена програма на мові С

```
#include <stdio.h>
void main()
{
    int a,b,c;
    a=5;
    b=7;
    c=a+b;
    printf("Summa= %d \n", з);
}
```

Вид екрану після виконання програми.

```
Summa= 12
Press any key to continue
```

Наступним елементом специфікатора перетворення є «розмір», який уточнює тип аргументу. Для уточнення типу аргументу використовуються модифікатори l, L, h.

%ld – друк long int (коротке ціле);

%Ld – друк long double;

%hd – друк chort unsigned (коротке ціле).

Елементи «поле», «точність» визначають характеристики поля, відведеного для зображення аргументу. В таблиці приведені варіанти елементів «поле» і «точність».

Елемент	Символ	Опис
Поле	Число	Мінімальна ширина поля виводу
Точність	Число	Для рядків – макс. число символів, що виводяться Для цілих – мін. число цифр, що виводяться Для речовинних – число цифр дробової частини

Між знаком % і символом, який задає тип перетворення, може стояти число. Воно указує на якнайменше поле, що відводиться для друку. Якщо рядок або число *більше* цього поля, то рядок або число друкуються повністю, ігноруючи ширину поля.

Приклад.

```
#include <stdio.h>
void main()
{
    int a=27182;
    short  b=273;
    printf(" a=%.3d\n b=%2d\n",a,b);
```

```
}
```

Вигляд екрану після виконання програми.

```
 a=27182
 b=273
Press any key to continue
```

Оскільки обидва числа більше, ніж ширина поля, то числа надруковані повністю.

Щоб вказати число десяткових знаків після цілого числа, ставиться крапка і ціле число, яке вказує на кількість десяткових знаків. Коли такий формат застосовується до цілого числа або до рядка, те число, що стоїть після крапки, указує на максимальну ширину поля видачі.

Елемент «ознака» може приймати наступні значення.

Елемент	Символ	Опис
	-	Вирівняти по лівому краю
	0	Заповнити вільні позиції нулями
Ознака	+	Завжди виводити знак числа
	пробіл	Вивести пробіл на місці знаку, якщо знак +
	#	Вивести 0 перед восьмирічним або 0x перед шістнадцятирічним значеннями

Програми, написані на мові С повинні видавати інформацію в зручному для читання вигляді. Для цього необхідно вільно маніпулювати всіма перерахованими можливостями форматування при виведенні інформації.

Розглянемо програму виведення інформації функцією printf ().

```
#include <stdio.h>
#include <conio.h>
void main()
{
        double p=27182.81828;
        int j=255;
//Виведення цифр. Виведення знаку
        printf("Тест форматування цілих: %13.4d %+8d\n", j,j);
//Виведення по лівому краю. Заповнення нулями
        printf("Форматування цілих:   %- +13d%08d\n", j,j);
        printf("Тест          %#13o       %#8.6x\n", j,j);
        printf("\nТест  e  i  f:  %13.7e      %8.2f\n", p,p);
}
```

Вигляд екрану після виконання програми.

```
Тест форматування цілих:    0255            +255
Форматування цілих:         +255        00000255
Тест:                       0377         0x0000ff
Тест    e  i  f:      2.7182818e+004       27182.82
Press any key to continue
```

Рекомендація ! Введіть цю програму і по експериментуйте з її текстом, міняючи ознаки і параметри поля виводу.

2. Функція scanf ().

Функція scanf () є основною функцією введення даних з клавіатури. Функція читає дані із стандартного файлу введення, за умовчанням – з клавіатури. Вона здійснює введення даних будь-якого стандартного (вбудованого) типу і автоматично перетворює введене число в заданий формат.

Функція scanf () побудована за таким же принципом, що і функція printf (). Звернення до функції має вигляд.

scanf (Управляючий рядок, [список аргументів]);

Оскільки scanf () повинна передавати числа, що вводяться, програмі, що звернулася до неї, остання повинна забезпечити її адресами змінних, яким будуть присвоюватися значення. Адресу одержують за допомогою операції «взяття адреси» &. Наприклад &x &B.

Управляючий рядок містить три види символів:

- специфікатори перетворення;
- пробіли;
- інші символи.

Для функції scanf () специфікатор перетворення має вигляд.

%[ознака] [поле] [розмір] символ типу;

Обов'язковими елементами специфікатора є знак відсотка % і символ типу числа, що вводиться.

%c – читання символу;
%d – читання десяткового цілого;
%i – читання десяткового цілого;
%e – читання числа типу float
%h – читання числа short int;
%o – читання восьмирічного числа;
%s – читання рядка;
%x – читання шістнадцятирічного числа;
%p – читання покажчика;
%n – читання покажчика в збільшеному форматі.

Приклад.

```
int a;
scanf("%d", a);
```

Елемент «розмір» уточнює тип аргументу. Для уточнення типу аргументу використовується модифікатор l. Наприклад.

ld – посилання на long;
lo – посилання на long;
lx – посилання на long;

lf – посилання на double;

le – посилання на double.

Елемент «поле» указує найбільшу ширину поля, яка підлягає читанню. Наприклад.

```
scanf(“%5s”, str);
```

Специфікатори перетворення у функції указують на необхідність прочитати при введенні перші 5 символів. При введенні “ABCDEFGIJK”, буде введене тільки ABCDE. Решта символів буде проігнорована.

Роздільниками між двома числами, що вводяться, є символ пробілу, табуляції або нового рядка. Отже, символ пробілу в управляючому рядку дає команду пропустити один або декілька пробілів в потоці введення. Якщо необхідно ввести два числа a і b, які відповідно дорівнюють 5 і 7, то при наборі з клавіатури вони повинні бути розділені пробілом.

```
5  7
```

Тоді управляючий рядок функції scanf () матиме наступний вигляд.

```
int a, b;
scanf(“%d  %d”, &a &b);
```

У даному прикладі управляючий рядок приписує функції scanf () ввести десяткове ціле число, яке треба помістити в змінну a. Потім просунутися до наступного символу, на що указує пропуск після першого специфікатора %d і з цієї точки почати введення нового цілого десяткового числа, яке потім помістити в змінну b.

Ще приклад.

```
int a, b, c;
scanf(“%d \n%d\n %d” &a &b &c);
```

У цьому випадку вхідний потік повинен виглядати таким чином.

```
5
7
11
```

Як «ознака» може стояти символ *. Цей символ дає команду прочитати дані вказаного типу, але не присвоювати це значення. Так

```
scanf(%d %*c%d” &a &b);
```

при введенні 50 + 20 присвоїть змінній a значення 50, змінній b - значення 20, а символ + буде прочитаний і проігнорований.

Однієї з особливостей функції scanf () є можливість завдання множини пошуку. Множина пошуку визначає набір символів, з якими порівнюватимуться читані функцією символи з вхідного потоку. Функція scanf () читає символи з вхідного потоку до тих пір, поки вони зустрічаються в множині пошуку. Як тільки символ, який вводиться, не зустрівся в множині пошуку, функція scanf () переходить до наступного специфікатора формату. Множина пошуку визначається набором символів, поміщених у квадратні дужки. Перед дужкою, що відкривається, ставиться знак %. Специфікатор %[…] указує, що рядок містить будь-які символи, які перераховані в квадратних дужках, і закінчується будь-яким з символів, відмінних від них.

Можна також визначити символи, які не входять у множину пошуку. Перед першим із них ставиться знак ^. Специфікатор %[^...] говорить, що рядок обмежується будь-яким з символів, поміщених в дужки.

Приклад.

```
char ch;
char str[20];
int i, j, number, ext;
float x;
// Вхідний рядок   PHONE65201X4133
Scanf (“PHON %c % f % * з % d” &ch &x &ch &ext);
//  ch=’e’, x = 65201.0, x4133

Scanf(“PHONE %2d %3d %c %2f “ &I &j &ch &ext);
//  I = 65, j = 201, ch = ‘x’, x = 41.0

Scanf(“%[^x]%c %d “, str &ch &ext);
//  str = “PHONE65201”, ch = ‘x’,ext = 4133

Scanf( “PHONE %[0123456789] %c %d”,  str &ch &ext
//  str = “65201”, ch = ‘x’, ext = 4133
```

І на закінчення приведемо простішу програму, яка вводить два числа, обчислює їх суму і видає результат на екран.

```
#include <stdio.h>
void main()
{
        int a,b,c;
        scanf("%d  %d" &a &b);
        з = a + b;
        printf("Summa = %d \n", з);
}
```

Вигляд екрану після виконання програми.

```
5 7
Summa = 12
Press any key to continue
```

Висновок. У даний час дуже багато програм написано на мові С. Умові С є свої особливості введення-виведення даних. Розглянемо ці особливості.

У мові відсутні багато вбудованих в транслятор засобів, властивих розвиненим мовам програмування. В ньому немає вбудованих методів доступу до файлів, немає операцій введення-виведення даних в їх звичному розумінні. Виконання цих задач забезпечується бібліотекою підпрограм.

Функції введення-виведення зберігаються в заголовному файлі stdio.h. Щоб зв'язати програму користувача із стандартною бібліотекою введення-виведення, необхідно в програмі передбачити директиву препроцесора

include <stdio.h>

Функції printf () і scanf () здійснюють форматоване введення і виведення на консоль. Форматоване введення-виведення означає, що функції можуть читати і виводити дані в різному форматі, яким можна управляти.

Питання для самоконтролю.

1. Сформулюйте призначення функції printf ().
2. Сформулюйте призначення функції scanf ().
3. Яка інформація зберігається в управляючому рядку функції printf ().
4. Яка інформація зберігається в управляючому рядку функції scanf ().
5. Визначте специфікатори перетворення функції printf ().
6. Визначте специфікатори перетворення функції scanf ().
7. Які символи перетворення Ви знаєте.

Лекція 17,18
ФУНКЦІЇ МОВИ С (С++) ПРИ РОБОТІ З ФАЙЛАМИ

Мета лекції. Визначити і вивчити особливості використання функцій при роботі з файлами.

Основні питання лекції.

1. Файли і потоки.
2. Створення, відкриття і закриття файлу
3. Запис (читання) даних у файл
4. Позиціонування файлу

1. Файли і потоки.

При ускладненні програм виникає необхідність зберігати і одержувати інформацію, використовуючи файли.

Операції введення-виведення в мові С організовані за допомогою бібліотечних функцій.

Крім того, мова С++ передбачає власне об'єктно-орієнтоване введення-виведення. С++ надає набір класів файлових потоків, за допомогою яких можна легко виконувати операції обміну даними з файлами.

Розглянемо організацію обміну даними з файлами, використовуючи бібліотечні функції.

Важливо зрозуміти, що таке *файл* (file) і що таке *потік* (stream), а так само яка різниця між цими поняттями.

Система введення-виведення мови С(С++) підтримує інтерфейс, не залежний від того, який насправді використовується фізичний пристрій введення-виведення. Іншими словами, С (С++) підтримує абстрактний рівень взаємодії між програмою і фізичним пристроєм. *Ця абстракція і називається потоком.* Потік – це логічний пристрій, який зв'язує програму з фізичним пристроєм (терміналом, дисководом і ін.). Оскільки потоки не залежать від фізичних пристроїв, то одна і та ж функція може записувати інформацію на диск або інший пристрій.

Файл в мові С (С++) – це поняття, яке може бути прикладене до всього: від файлу на диску до терміналу.

У мові С (С++) існує два типи потоків:

- текстовий (text);
- двійковий (binary).

Текстовий потік – це послідовність символів. Проте може не бути взаємно однозначної відповідності між символами, які передаються в потоці і виводяться на екран. Серед символів в потоці може бути символ повернення каретки, перехід на новий рядок, символ кінця рядка і ін.

Двійковий потік – це послідовність байтів, які взаємно однозначно відповідають тому, що знаходиться на зовнішньому носії.

Не всі файли однакові. Наприклад, з файлу на диску можна вибрати 5-й запис або замінити 10-й запис. В той час у файл, пов'язаний з друком, інформація може передаватися тільки послідовно. Це ілюструє найголовнішу відмінність між потоками і файлами: всі *потоки однакові,* чого не можна сказати про файли.

Потоки, що використовуються в програмах, логічно діляться на:

- вхідні, з яких *читається* інформація;
- вихідні, в які *вводиться* інформація;
- двонаправлені, допускаючі як читання, так і запис.

Відповідно до особливостей «пристрою», до якого «приєднаний» потік, потоки прийнято ділити на:

- стандартні;
- консольні;
- рядкові;
- файлові.

Якщо символи потоку в сукупності утворюють символьний масив (рядок) в основній пам'яті, то це рядковий потік.

Якщо інформація розміщується на зовнішньому носії даних, то говорять про файловий потік або просто файлі.

Стандартні і консольні потоки відповідають передачі даних від клавіатури і до дисплея.

Дотепер у всіх програмах виконувався обмін із стандартними потоками.

cin – стандартний вхідний потік, пов'язаний з клавіатурою;

cout – стандартний вихідний потік, пов'язаний з екраном дисплея.

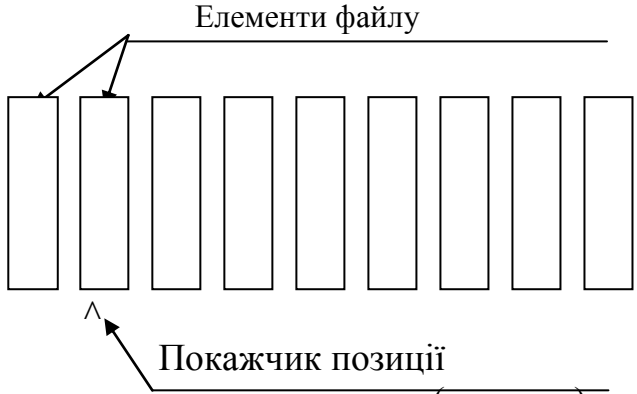
Використовуючи операції включення (записи) даних в потік << і витягання даних з потоку >> виконувався обмін даними з дисплеєм і клавіатурою ПК. Для цих цілей в програму необхідно було включити заголовний файл iostream.h.

Розглянемо особливості обміну даними з файлами.

2. Створення, відкриття і закриття файлу

Бібліотека введення-виведення включає засоби для роботи з послідовними файлами. Логічно послідовний файл можна представити як іменований ланцюжок (стрічку, рядок) байтів, що має початок і кінець. Послідовний файл відрізняється від файлу з іншою організацією тією властивістю, що читання з файлу (або запис у файл) ведуться байт за байтом від початку до кінця.

У кожний момент часу позиції у файлі, звідки виконується читання (або запис), визначаються значеннями покажчика позиції запису і читання файлу.



Установка покажчика запису (читання) на потрібні байти виконується або автоматично, або за рахунок явного управління їх положенням. В бібліотеці введення-виведення є відповідні засоби.

При роботі з файлами необхідно указати на існування наступних процедур:

- створення файлів;

- знищення файлів;
- пошук файлів на зовнішньому носії;
- відкриття файлу;
- читання з файлу або запис даних у файл;
- позиціонування файлу;
- закриття файлу.

Усі перераховані дії можуть бути виконані за допомогою засобів бібліотеки введення-виведення.

Операція відкриття файлу пов'язує потік з певним файлом. Операція закриття файлу – розриває цей зв'язок.

Кожний потік, пов'язаний з файлом, має *управляючу структуру*, звану FILE. Вона описана в заголовному файлі stdio.h.

При обробці даних, які зберігаються у файлі, програмі необхідно мати доступ до даного файлу. Це робиться за допомогою введення спеціальної файлової змінної. Через цю змінну і ведеться надалі вся робота з файлом в програмі.

У мові С (С++) зв'язуючою ланкою між файлом і потоком в системі введення-виведення є покажчик на файл. Покажчик на файл – це покажчик на структуру типа FILE.

Щоб оголосити покажчик на файл, використовується оператор.

```
FILE *fptr;
```

Наприклад.

```
FILE *F1;
```

Крім того, в заголовному файлі stdio.h визначені також наступні функції.

Функція	Опис функції
fopen()	Відкрити файл
fclose()	Закрити файл
fseek()	Перемістити (встановити) покажчик позиції файлу
feof()	Повертає значення «істина», якщо досягнутий кінець файлу
ferror()	Повертає значення «неправда», якщо знайдена помилка
fread()	Читає блок даних з потоку.
fwrite()	Записує блок даних в потік
rewind()	Встановлює покажчик позиції файлу на початок
remove()	Знищує файл

При відкритті файлу функція fopen() виконує дві дії:

- відкриває файл і пов'язує його з потоком;
- повертає покажчик, асоційований з цим файлом.

Прототип функції:

```
FILE *fopen ( char *filename, char mode);
```

де

char *filename – рядок, що містить повне ім'я файлу на диску;

char *mode – рядок, що містить режим файлу, що відкривається.

Можливі режими відкриття файлу.

“r” – відкрити для читання;
“w” – створити для запису;
“a” – відкрити для додавання в існуючий файл;
“rb” – відкрити двійковий файл для читання;
“wb” – створити двійковий файл для запису;
“ab” – відкрити двійковий файл для додавання;
“r+” – відкрити файл для читання і запису;
“w+” – створити файл для читання і запису;
“a+” – відкрити для додавання;
“r+b” – відкрити двійковий файл для читання і запису;
“w+b” – створити двійковий файл для читання і запису;
“a+b” – відкрити двійковий файл для додавання;
“rt” – відкрити текстовий файл для читання;
“wt” – створити текстовий файл для запису;
“at” – відкрити текстовий файл для додавання;
“r+t” – відкрити текстовий файл для читання і запису;
“w+t” – створити текстовий файл для читання і запису;
“a+t” – відкрити текстовий файл для додавання.

Для створення файлу для запису з ім'ям C:\\Book1.dat необхідно записати:

```
FILE *F1;
F1= fopen (“C:\\Book1.dat”, “w”);
```

Проте при відкритті файлу необхідно провести перевірку відкриття файлу.

Тоді відкриття файлу матиме наступний вигляд.

```
FILE *F1;
if ((F1=fopen (“C:\\Book1.dat”, “w”)) = = NULL)
{
  cout << “Не можу відкрити файл!” << endl;
  exit(1);
}
```

Цей метод визначає помилку при відкритті файлу. Константа NULL визначена в stdio.h. Функція exit(1) визначена у файлі stdlib.h і припиняє виконання програми, одиницю повертає в операційну систему. Перед припиненням програми вона закриває всі відкриті файли, звільняє буфери і виводить всі необхідні повідомлення на екран.

Якщо файл відкривається для запису, то існуючий файл знищується і створюється новий.

При відкритті файлу для читання, вимагається, щоб він існував.

У разі відкриття файлу для читання і запису існуючий файл не знищується, проте створюється, якщо він не існує.

Після читання даних з файлу (або запису даних у файл) він повинен бути закритий. Закриття файлу проводиться таким чином.

```
fclose (F1);
```

3. Запис (читання) даних у файл

Запис даних в потік проводиться функцією fwrite(), а читання - функцією fread(). Ці функції мають наступні прототипи.

```
unsigned fread (void *buf, int bytes, int c, FILE *fptr);
unsigned fwrite (void *buf, int bytes, int c, FILE *fptr);
```

де:

buf – покажчик на буфер пам'яті, звідки відбуватиметься обмін з файлом:

bytes – довжина кожної одиниці запису (читання) в байтах;

c – кількість одиниць запису, яка буде прочитана (записана);

fptr – покажчик на відповідний файл.

Приклад. В пам'яті ПК є масив з 12 довільних чисел. Записати цей масив у файл за адресою C:\\BOOK.dat. Прочитати цей масив з файлу і видати його на екран дисплея.

Програма запису масиву у файл, читання масиву з файлу і відображення його на екрані дисплея матиме наступний вигляд.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
void main()
{
	float S; //визначення буфера обміну і покажчика на буфер
	float *pS; //визначення покажчика на буфер
		float Mas[12]; // визначення масиву
	int i;
	pS=&S; // присвоєння адреси буфера покажчику на буфер
	cout<<"Введіть Mas :"<<endl;
	for(i=0; i<12; i++) cin>>Mas[i]; //введення масиву
	FILE *F1; //визначаємо покажчик на файл F1
	// відкриваємо файл C:\\BOOK1.dat в режимі запису
	if((F1=fopen("C:\\BOOK.dat", "w+b"))==NULL)
	{
		cout<<"Ne mogu otkrit file !"<<endl;
		exit(1);
	}
	//читання даних з файлу
		for(i=0; i<12; i++)
	{
		S=Mas[i]; // запис елемента масиву в буфер
		fwrite (pS, 4, 1, F1); //запис однієї порції з 4-х байтів у файл

	}
		fclose(F1); //закриття файлу F1
int l;
float Mas1[12]; // Визначення масиву Mas1
FILE *F2; //визначаємо покажчик на файл F2
// відкриваємо файл C:\\BOOK1.dat в режимі читання і запису
if((F2=fopen("C:\\BOOK.dat", "r+b"))==NULL)
```

```
{
        cout<<"Ne mogu otkrit file1 !"<<endl;
        exit(1);
}
i=0;
        while(i<12)  //читання даних з файлу
{
        fread(pS, 4, 1, F2);  //читання однієї порції з 4-х байтів у буфер S
        Mas1[i]=S;  //запис з буфера S у масив
        i+=1;  // рахуємо кількість прочитаних чисел з файлу
}
l=i; //визначаємо кількість прочитаних чисел з файлу
fclose(F2);  //закриття файлу F1
cout<<endl;
for(i=0; i<l; i++) cout<<Mas1[i]<<'  '; //відображення масиву на екрані
cout<<endl;
}
```

Вигляд екрану після виконанні програми.

```
Введіть   Mas :
1 2 3 4 5 6 7 8 9 10 11 12

1 2 3 4 5 6 7 8 9 10 11 12
Press any key to continue
```

При читанні даних з файлу розмір файлу часто невідомий. Для визначення кінця файлу служить функція feof(). Вона має наступний прототип.

```
int feof (FILE *fptr);
```

Функція повертає значення «істина», якщо досягнутий кінець файлу і «неправда» - інакше.

Програма читання даних з файлу C:\\BOOK1.dat з використанням функції feof() має наступний вигляд.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
void main()
{
        float S; //визначення буфера обміну і покажчика на буфер
        float *pS;   //визначення  покажчика на буфер
        float Mas[9]; // визначення масиву
        int i;
        pS=&S; // присвоєння адреси буфера покажчику на буфер
        cout<<"Введіть  Mas :"<<endl;
        for(i=0; i<9; i++) cin>>Mas[i]; //введення масиву
        FILE *F1; //визначаємо покажчик на файл F1
        // відкриваємо файл C:\\BOOK1.dat в режимі  запису
```

```
        if((F1=fopen("C:\\BOOK1.dat", "w+b"))==NULL)
        {
                cout<<"Ne mogu otkrit file1 !"<<endl;
                exit(1);
        }
        for(i=0; i<9; i++)
        {
        S=Mas[i]; // запис елемента масиву в буфер
        fwrite(pS, 4, 1, F1); //запис однієї порції з 4-х байтів у файл
        }
        fclose(F1); //закриття файлу F1
int l;
float Mas1[9]; // Визначення масиву Mas1
FILE *F2;  //визначаємо покажчик на файл F2
// відкриваємо файл C:\\BOOK1.dat в режимі  читання і запису
if((F2=fopen("C:\\BOOK1.dat", "r+b"))==NULL)
{
        cout<<"Ne mogu otkrit file2 !"<<endl;
        exit(1);
}
i=0;
//читання даних з файлу доки не наступить кінець файлу
while(!feof(F2))
{
        fread(pS, 4, 1, F2); //читання однієї порції з 4-х байтів у буфер S
        Mas1[i]=S;  //запис з буфера S у масив
i+=1;  // рахуємо кількість прочитаних чисел з файлу

}
l=i-1;  //визначаємо кількість прочитаних чисел з файлу
fclose(F2); //закриття файлу F2
cout<<endl;
for(i=0; i<l; i++) cout<<Mas1[i]<<' '; //відображення масиву на екрані
cout<<endl;
}
```

Вигляд екрану після виконання програми.

```
Введіть  Mas :
1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9
Press any key to continue
```

5. Позиціонування файлу

Функція rewind () встановлює покажчик позиції файлу на початок файлу. Прототип цієї функції має вигляд.

```
void rewind (FILE *fptr);
```

Звернення до функції має вигляд.

```
rewind (F2);
```

Читання і запис у файл не обов'язково робити послідовно. Можна дістати доступ безпосередньо до потрібного байта. Для цих цілей використовується функція fseek(), яка встановлює покажчик позиції в потрібне місце.

Прототип цієї функції має вигляд.

```
int fseek (FILE *fptr, long num, int mode);
```

де:

fptr – покажчик на відповідний файл;

num – кількість байт від точки відліку для установки поточної позиції покажчика файлу;

mode – визначає точку відліку.

Точка відліку	Значення mode
1. Початок файлу	0
2. Поточна позиція	1
3. Кінець файлу	2

Розглянемо приклад. У файлі C:\\BOOK1.dat записано 9 чисел: 1 2 3 4 5 6 7 8 9. Необхідно прочитати дані з цього файлу, починаючи з 3-го числа (тобто з 3), визначити суму прочитаних чисел і додати її у файл C:\\BOOK1.dat. Потім прочитати знов одержаний масив з файлу і видати його на екран дисплея.

Програма матиме вигляд.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
void main()
{
    const int n=30; //максимальний розмір масиву
    float S, *pS; //визначення буфера обміну і покажчика на буфер
    float Mas1[n],Sum;// визначення масиву і суми
    int i,l;
    pS=&S; // присвоєння адреси буфера покажчику на буфер
    FILE *F2;//визначаємо покажчик на файл F2
    // відкриваємо файл C:\\BOOK1.dat в режимі читання і запису
    if((F2=fopen("C:\\BOOK1.dat", "r+b"))==NULL)
    {
        cout<<"Не можу  відкрити file1 !"<<endl;
        exit(1);
```

```
        }
i=0;
//установка покажчика позиції файлу через 8 байтів с початку файлу
fseek(F2, 8, 0);
//читання даних з файлу доки не наступить кінець файлу
while(!feof(F2))
{
        fread(pS,  4, 1, F2); //читання однієї порції з 4-х байтів у буфер S
        Mas1[i]=S; //запис з буфера S у масив
        i+=1; // рахуємо кількість прочитаних чисел з файлу
}
l=i-1; //визначаємо кількість прочитаних чисел з файлу
fclose(F2); // закриття файлу
cout<<endl;
for(i=0; i<l; i++) cout<<Mas1[i]<<'  '; // відображення масиву на єкрані
cout<<endl;
Sum=0;
for(i=0; i<l; i++) Sum=Sum+Mas1[i]; // обчислення суми
cout<<"Sum= "<<Sum<<endl; // відображення суми на екрані
FILE *F3;//визначаємо покажчик на файл F3
// відкриваємо файл C:\\BOOK1.dat в режимі додавання
F3=fopen("C:\\BOOK1.dat", "a");
S=Sum;// пересилаємо суму в буфер
fwrite(pS, 4, 1, F3); \\записуємо суму з буферу в файл
fclose(F3);  // закриття файлу F3

        FILE *F4; //визначаємо покажчик на файл F4
        // відкриваємо файл C:\\BOOK1.dat в режимі читання і запису
        if((F4=fopen("C:\\BOOK1.dat", "r+b"))==NULL)
        {
        cout<<"Не можу відкрити  file2 !"<<endl;
                exit(1);
        }
i=0;
//читання даних з файлу доки не наступить кінець файлу
while(!feof(F4))
{
        fread(pS,  4, 1, F4); );//читання однієї порції з 4-х байтів у буфер S
        Mas1[i]=S; //запис з буфера S у масив
        i+=1;  // рахуємо кількість прочитаних чисел з файлу
}
l=i-1; //визначаємо кількість прочитаних чисел з файлу
fclose(F4);  // закриття файлу F4
cout<<endl;
for(i=0; i<l; i++) cout<<Mas1[i]<<'  '; // відображення масиву на єкрані
cout<<endl;
```

```
}
```
Вигляд екрану після виконання програми.
```
3 4 5 6 7 8 9
Sum= 42

1 2 3 4 5 6 7 8 9 42
Press any key to continue
*/
```

Висновок. При ускладненні програм виникає необхідність зберігати і одержувати інформацію, використовуючи файли.

Операції введення-виведення в мові С організовані за допомогою бібліотечних функцій.

При роботі з файлами необхідно указати на існування наступних процедур:

- створення файлів;
- знищення файлів;
- пошук файлів на зовнішньому носії;
- відкриття файлу;
- читання з файлу або запис даних у файл;
- позиціонування файлу;
- закриття файлу.

Усі перераховані дії можуть бути виконані за допомогою засобів бібліотеки введення-виведення.

Питання для самоконтролю.

1. Для введення даних у файл необхідно наступне оголошення:
   1. ofstream name_file;
   2. ofstream (“filename.txt”);
   3. ofstream name_file (“filename.txt”);
2. Щоб виконати операцію виведення даних з файлу необхідно наступне оголошення:
   1. ifstream name_file (“filename.dat”);
   2. iostream name_file (“filename.dat”);
   3. fstream name_file (“filename.dat”);
3. При читанні даних з файлу кінець файлу визначається функцією:
   1. eol()
   2. eof()
   3. eok()
4. Функція eof() повертає значення 0, коли
   1. Кінець файлу ще не наступив
   2. Кінець файлу наступив
   3. Ця функція нічого не повертає
5. Для визначення кінця файлу записаний оператор while (! name_file.eof())
   Цикл буде виконуватись
   1. Поки функція eof() повертає неправду (0)
   2. Коли функція eof() повертає істину (1)
   3. Таку конструкцію використовувати не можна
6. При завершенні роботи з файлом його треба закрити функцією:
   1. close();
   2. close.file_name;
   3. file_name.close();

7. При читанні масивів чи структур з файлу використовується функція:
   1. read
   2. fread
   3. ifread
8. У програмі зустрілось визначення ifstream koord (“koord1.dat”);
   Із якого файлу буде прочитана інформація:
   1. koord
   2. koord1
   3. Тут ім’я файлу не вказано
9. Функція write використовується для:
   1. Виводу масиву (структури) у файл
   2. Виводу масиву (структури) з файлу
   3. Виводу масиву (структури) на екран
10. Для виведення структури
    struct stats{char name[64]; int age; float salary;}worker;
    треба записати оператор ddd.write (…, sizeof(…));
    Що необхідно записати у дужках функції sizeof():
    1. worker
    2. stats
    3. Не має значення що
11. Для виведення структури
    struct stats{char name[64]; int age; float salary;}worker;
    треба записати оператор ddd.write ((char*)&worker, sizeof(stats);
    Що визначає &worker :
    1. Початкову адресу буфера, з якого структура буде виводитись у файл
    2. Розмір буфера
    3. Ім’я буфера
12. Заголовочний файл fstream.h визначає класи об’єктів:
    1. ifstream
    2. ofstream
    3. Обидва класи
13. Записано оператор
    ddd.write ((char*)&worker; sizeof(stats)
    Скільки помилок зроблено при запису оператора?
14. При читанні структури з файлу:
    struct stats{char name[64]; int age; float salary;}worker;
    1. Необхідно ініціалізувати поля структури
    2. Цього не треба робити
    3. Все залежить від досвіду програміста

Перелік рекомендованої літератури

Основна

1. Глушаков С.В., Коваль А.В., Смирнов С.В. Язык программирования С++. Учебный курс. Харьков, «Фолио», 2003, – 500 с.
2. Березин Б.И. Березин С.Б. Начальный курс С и С++. – М.: ДИАЛОГ-МИФИ, 2004. – 288 с.
3. Панішев А.В., Подоляка О.О., Подоляка О.Н. Комп'ютерно-тренажерний практикум з програмування на мові С++. Навчальний посібник для самостійної роботи студентів. Житомир, ЖДТУ, 2002, -- 205 с.
4. Онуфрей Ю.Є., Подоляка О,О. Методичні рекомендації до виконання лабораторних робіт з мови С (С++). Харків, ХНАДУ, 2006 – 112 с. (електронний варіант – портал ХНАДУ).
5. Берковський В.В., Левтеров А.І., Костікова М.В., Онуфрей Ю.Є., Подоляка О.О., Попеленко А.А. Програмування в середовищі С (С++). Збірник задач. Харків, ХНАДУ, 2007 – 224 с. (електронний варіант – портал ХНАДУ).

Додаткова

6. Подбельский В.В. Язык С++. 5-е издание. М: «Финансы и статистика»,2004, -- 560 с.

Навчальне видання

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни “Обчислювальна техніка та програмування”
за напрямом підготовки – 6.051001,
“Метрологія та інформаційно-вимірювальні технології. ”
(Розділ 2. “Мова програмування С (С++)”)

Укладачі: ОНУФРЕЙ Юрій Євграфович
ПОДОЛЯКА Оксана Олександрівна
ТІМОНІН Володимир Олексійович

Відповідальний за випуск А. І. Левтеров
Редактор
Комп’ютерна верстка

Підписано до друку
Формат 60×84 1/16. Папір офсетний. Гарнітура Times New Roman.
Друк RISO. Умовн. друк. арк. Обл.-вид. арк.
Замовлення № . Тираж прим. Ціна договірна.

Видавництво ХНАДУ, 61002, м. Харків-МСП, вул. Петровського, 25

Свідоцтво державного комітету інформаційної політики, телебачення та радіомовлення України про внесення суб’єкта видавничої справи до державного реєстру видавництв, виготівників і розповсюджувачів видавничої продукції, серія ДК №897 від 17.04.2002 р.