

ЛЕКЦІЯ 2

1. ОПЕРАТОРИ УПРАВЛІННЯ
2. ПЕРШІ АЛГОРИТМИ. ЧИСЛОВІ ПОСЛІДОВНОСТІ
3. МАСИВИ В МОВІ С.
4. АЛГОРИТМ ПОШУКУ МАКСИМАЛЬНОГО ЕЛЕМЕНТУ МАСИВУ

ОПЕРАТОРИ УПРАВЛІННЯ.

1. Умовні оператори *if*, *if-else* та *switch*.
2. Оператори циклу *for*, *while* та *do-while*.
3. Оператор безумовного переходу *goto*.

УМОВНІ ОПЕРАТОРИ *if*, *if-else*

Повна форма

if (умова) оператор; else оператор;

Якщо значення умови є істинним, то виконується оператор (складений оператор - блок), що йде за умовою. Якщо умова є хибною, то виконується оператор, що йде за ключовим словом *else*. У запису оператора *if* друга частина (тобто оператор *else*) може бути відсутньою. Тоді, якщо умова є хибною, виконуються наступний оператор програми.

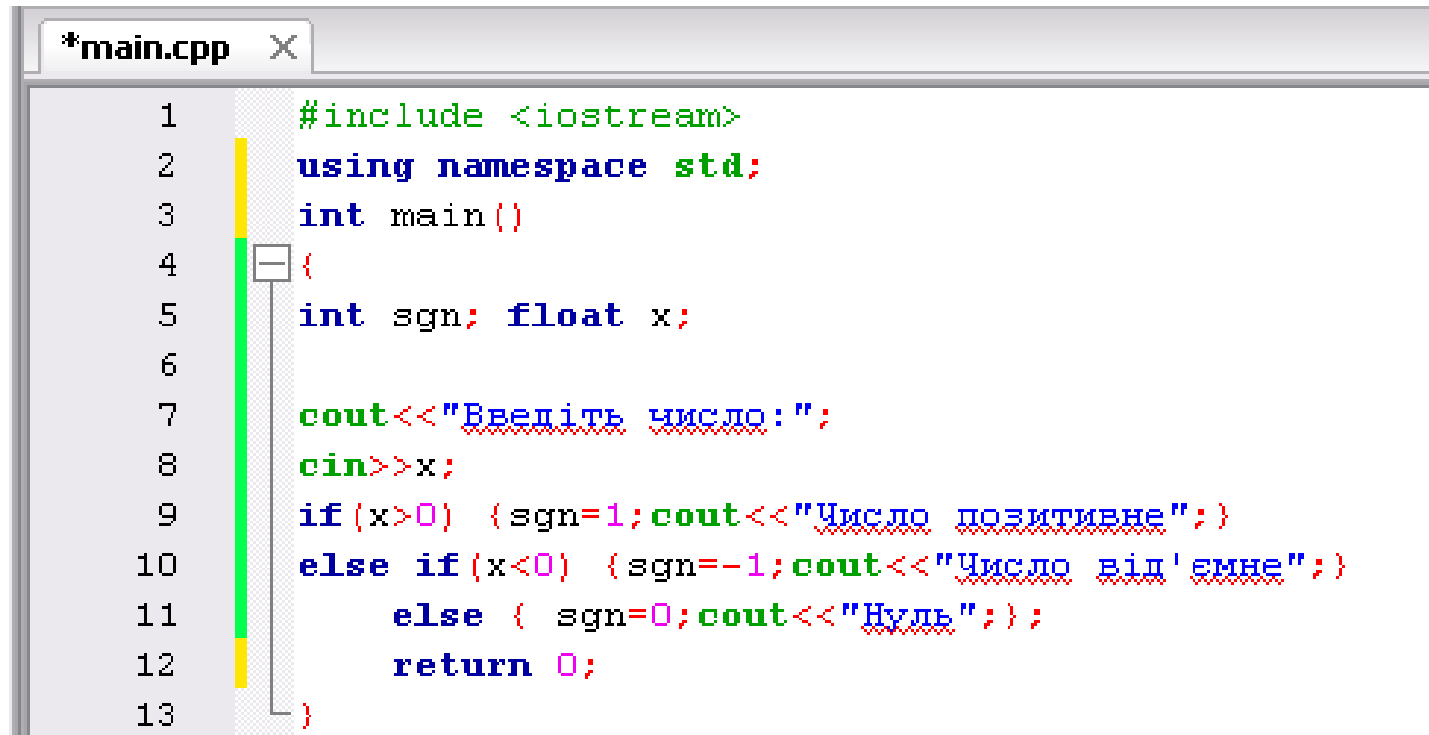
Часто є необхідність використовувати конструкцію *if-else-if*:

```
if (умова) оператор;  
else if (умова) оператор;  
else if (умова) оператор;  
...  
else оператор;
```

У цій формі умови операторів *if* перевіряються зверху донизу. Як тільки будь-яка з умов прийме істинне значення, буде виконуватись оператор, що йде за цією умовою, а решту конструкції буде проігноровано.

У мові C++ як умова може використовуватись довільний вираз. В операторі *if* лише перевіряється, чи є значення цього виразу ненулевим (істинним) або нульовим (хибним).

Приклад



```
*main.cpp X
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int sign; float x;
6
7      cout<<"Введіть число:";
8      cin>>x;
9      if(x>0) {sign=1;cout<<"Число позитивне";}
10     else if(x<0) {sign=-1;cout<<"Число від'ємне";}
11         else { sign=0;cout<<"Нуль";}
12         return 0;
13     }
```

Вкладеним оператором *if* називають наступну конструкцію:

```
if(x)  
    if (y) оператор1;  
    else оператор2;
```

У такій формі незрозуміло, до якого з операторів *if* відноситься *else*. У мові С оператор *else* асоціюється з найближчим *if* у відповідному блоці. В останньому прикладі *else* відноситься до *if*(*y*). Для того, щоб віднести *else* до оператору *if*(*x*), необхідно відповідним чином розставити операторні дужки:

```
if(x) {  
    if (y) оператор1;  
}  
else оператор2;
```

ЦИКЛИ

Цикли є необхідними, коли нам потрібно повторювати деякі дії декілька разів, як правило, поки виконується певна умова. У мові C++ є три оператори циклу: *for*, *while* та *do-while*.

ЦИКЛ FOR

Основна форма циклу *for* має наступний вигляд :
for (ініціалізація ; перевірка умови; змінювання) оператор;

У загальній формі:
for (вираз1 ; вираз2 ; вираз 3) оператор;

Приклади:

```
for (;;) cout<<"Нескінченний цикл";  
for (i=1;1;i++) cout<<"Нескінченний цикл";  
for (i=10;i>6;i++) cout<<"Нескінченний цикл";
```

ЦИКЛИ WHILE та DO-WHILE

ЦИКЛ WHILE. Основна його форма :

while (умова) оператор;

де оператор може бути простим, складеним або пустим оператором.

"Умова " є виразом. Цикл виконується доти, доки умова є істинною. Коли умова прийме значення «хибне», програма передає управління наступному оператору програми. Аналогічно, як й у циклі *for*, у циклі *while* спочатку перевіряється умова, а потім виконується оператор.

Це так званий цикл з передумовою.

ЦИКЛ DO WHILE

На відміну від попередньо описаних операторів циклу у циклі *do-while* умова перевіряється наприкінці оператора циклу.

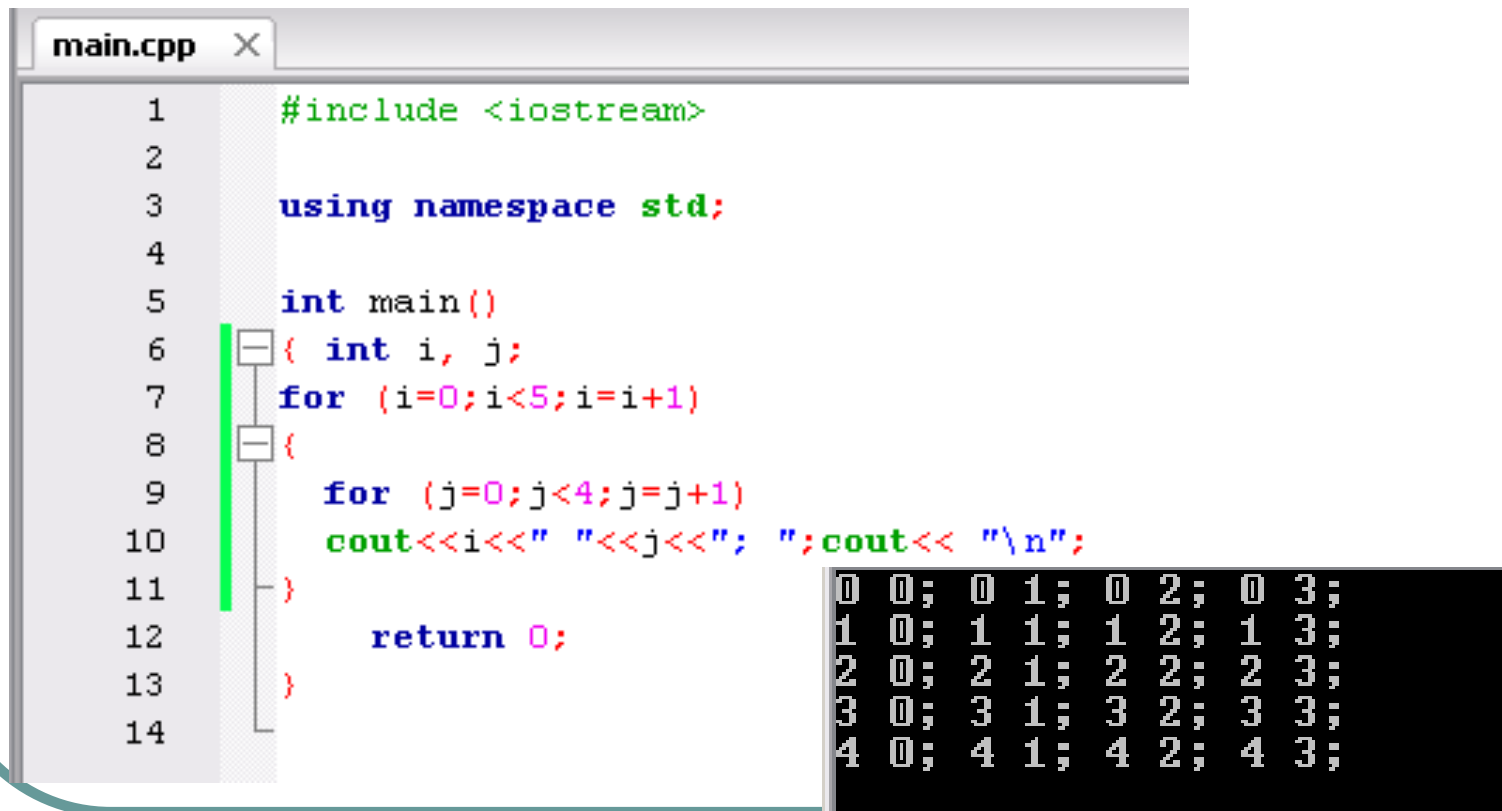
Основна форма оператора do-while :

```
do {  
    послідовність операторів  
} while (умова);
```

Це так званий цикл з постумовою.

ВКЛАДЕНІ ЦИКЛИ

В середині одного циклу задано другий оператор циклу. Приклад Виведення на екран номеру рядку та стовпця у двовимірній таблиці



```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  { int i, j;
7    for (i=0;i<5;i=i+1)
8    {
9      for (j=0;j<4;j=j+1)
10     cout<<i<<" "<<j<<" "; cout<<"\n";
11   }
12   return 0;
13 }
14
```

0 0; 0 1; 0 2; 0 3;
1 0; 1 1; 1 2; 1 3;
2 0; 2 1; 2 2; 2 3;
3 0; 3 1; 3 2; 3 3;
4 0; 4 1; 4 2; 4 3;

АЛГОРИТМИ ДЛЯ ОБРОБКИ ЧИСЛОВИХ ПОСЛІДОВНОСТЕЙ

Задача 1. Знайти n -й ($n=4$) член числової послідовності

$$x_n = x_{n-1} + x_{n-2}; x_0 = 1; x_1 = 2.$$

Розв'язок

$$x_2 = x_1 + x_0;$$

$$x_3 = x_2 + x_1$$

$$x_4 = x_3 + x_2$$

...

ПРОГРАМУВАННЯ

```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {int x0,x1,x2,i;
7   /** Четвертий елемент числової послідовності  $x(n)=x(n-1)+x(n-2)$  */
8   x0=1; x1=2; i=2;
9   while (i<=4)
10  {
11      x2=x0+x1;
12      x0=x1;
13      x1=x2;
14      i=i+1;
15  }
16  cout<<"i="<<i-1<<"\n";
17  cout<<x2;
18  return 0;
19  }
20
```

ПРОГРАМУВАННЯ

Задача 2. Знайти усі члени числової послідовності з задачі 1, які є меншими заданого числа m ($m=100$).

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {int x0,x1,x2,i;
7   /** Знаходження всіх елементів числової послідовності  $x(n)=x(n-1)+x(n-2)$ ,
8    які є меншими 100*/
9    x0=1; x1=2;i=2;
10   do
11   {
12       x2=x0+x1;
13       x0=x1;
14       x1=x2;
15       cout<<"i="<<i<<" ";
16       i=i+1;
17       cout<<"x2="<<x2<<"\n";
18   }
19   while (x2<100);
20   return 0;
21 }
```

"D:\CPP\c21\bin\Debug\p2019 1.exe"

```
i=2 x2=3
i=3 x2=5
i=4 x2=8
i=5 x2=13
i=6 x2=21
i=7 x2=34
i=8 x2=55
i=9 x2=89
i=10 x2=144
```

Process returned 0 (0x0) execution time : 0.169 s
Press any key to continue.

МАСИВИ

Відносяться до складних типів даних. Будуються на основі скалярних типів (поки відомі два з них: *float* та *int*).

Масив – набір даних одного типу, зібраних під одним іменем. Кожен елемент масиву визначається іменем масиву та порядковим номером цього елемента – індексом.

Основна форма оголошення **масиву розмірності n** така:

тип <ім'я масиву>[розмір1][розмір2]...[розмір n]

Найчастіше використовуються **одновимірні масиви**:

тип <ім'я масиву>[розмір];

де тип - базовий тип елементів масиву,

розмір - кількість елементів одновимірного масиву.

При описі **двовимірного** масиву оголошення має наступний вид:

тип <ім'я масиву> [розмір 1][розмір2];

У цьому описі оголошення двовимірного масиву можливо трактувати як оголошення масиву масивів, тобто масив розміру [розмір2], елементами якого є одновимірні масиви <ім'я масиву>[розмір1].

Розмір масиву може задаватися константою чи константним виразом. Не можна задати масив змінного розміру. Для цього існує окремий механізм, називаний динамічним виділенням пам'яті.

ОДНОВИМІРНІ МАСИВИ

У мові C++ індекс завжди починається з нуля. Коли ми говоримо про перший елемент масиву, маємо на увазі елемент з індексом 0. Якщо ми оголосили масив

```
int a[100];
```

це значить, що масив містить 100 елементів від `a[0]` до `a[99]`.

Посилання на *i*-й елемент масиву `a`: `a[i]`;

Для одновимірного масиву легко підрахувати, скільки байт у пам'яті буде займати цей масив:

кільк. байт = <розмір базового типу> * <кільк. елементів>.

У мові C++ під масив завжди виділяється неперервне місце в оперативній пам'яті.

УВАГА!

У мові C++ не перевіряється вихід індексу за межі масиву. Якщо масив `a[100]` описаний як цілочисельний масив, що має 100 елементів, а ви в програмі вкажете `a[200]`, то повідомлення про помилку не буде видано, а як значення елемента `a[200]` буде видане деяке число, що займає відповідну пам'ять на відстані 100 елементів типу `int` від елементу `a[99]`.

Повернемось до задачі виводу на екран елементів числової послідовності, які є меншими 100.

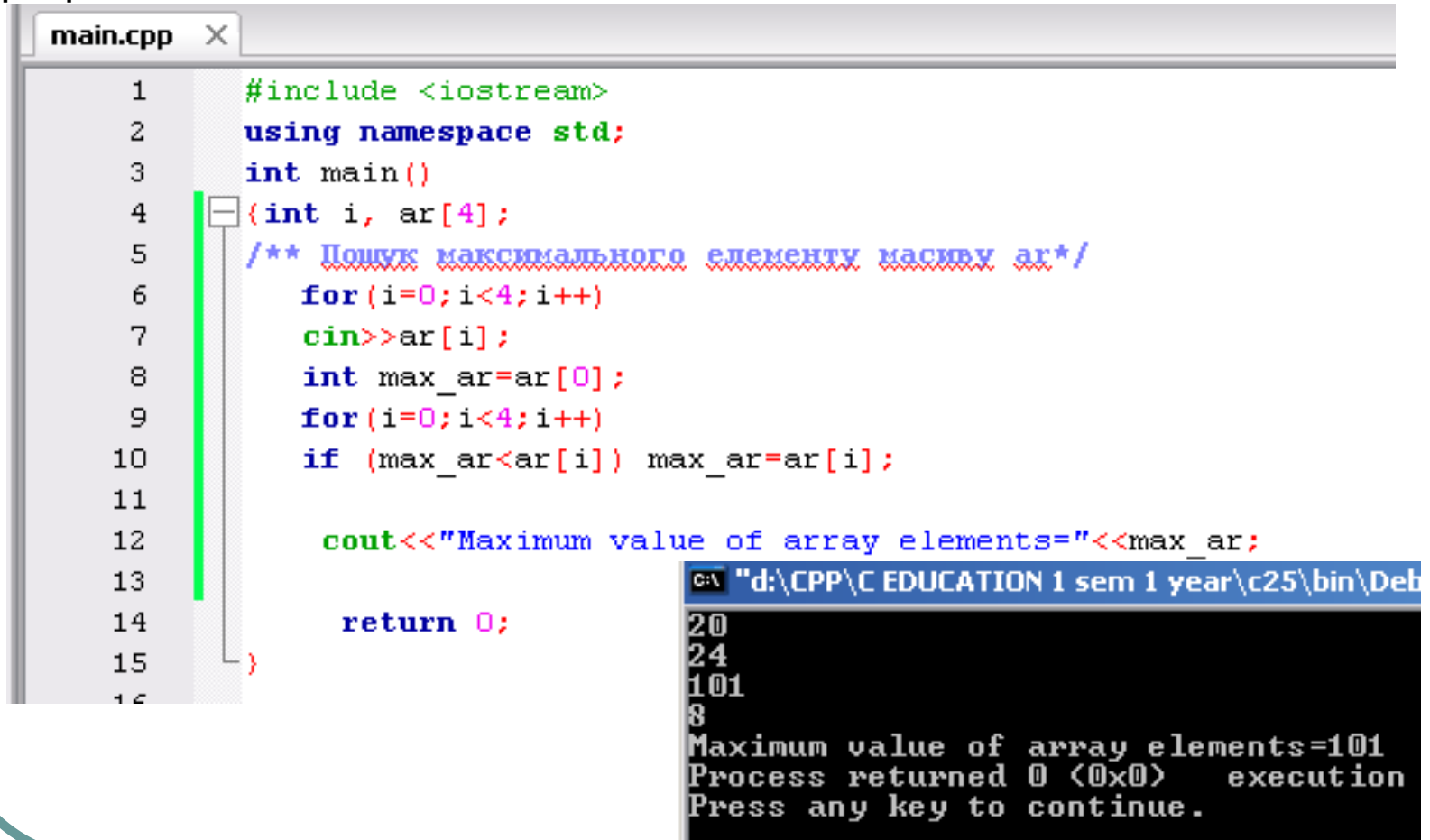
```
3  int main()
4  {int x0,x1,x2,i;
5  int a[10];
6  /** Знаходження усіх елементів числової послідовності  $x(n)=x(n-1)+x(n-2)$ ,
7   які є меншими 100 та запис їх до масиву a*/
8   for(i=0;i<10;i++) a[i]=0;
9   x0=1; x1=2;
10  a[0]=x0; a[1]=x1;
11  i=1;
12  do
13  {
14      x2=x0+x1;
15      x0=x1; x1=x2;
16      i=i+1;
17      a[i]=x2;
18  }
19  while (x2<100);
20  for(i=0;i<10;i++)
21      if (a[i]<100) cout<<a[i]<<" ";
22  return 0;
23 }
```

```
G:\ "d:\CPP\C EDUCATION 1 sem 1 year...
1 2 3 5 8 13 21 34 55 89
Process returned 0 (0x0)
Press any key to continue.
```

АЛГОРИТМ ПОШУКУ МАКСИМАЛЬНОГО ЗНАЧЕННЯ В МАСИВІ

- M1. Обрати як поточне значення ідентифікатору `max_ar` будь-який з елементів масиву `ar`, наприклад перший.
- M2. Лічильник елементів циклу $i = 0$
- M3. Порівняти поточне значення ідентифікатору `max_ar` з поточним значенням елементу масиву `ar[i]`.
- M4. Якщо $a[i] > \text{max_ar}$, оновити поточне значення `max_ar`: `max_ar = a[i]`
- M5. $i = i + 1$
- M6. Якщо $i < \text{розміру масиву}$, перейти на M3, інакше закінчити алгоритм.

Програма.



The image shows a C++ program in a code editor window titled 'main.cpp'. The code is as follows:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {int i, ar[4];
5  /** Пошук максимального елементу масиву ar*/
6      for(i=0;i<4;i++)
7          cin>>ar[i];
8      int max_ar=ar[0];
9      for(i=0;i<4;i++)
10         if (max_ar<ar[i]) max_ar=ar[i];
11
12         cout<<"Maximum value of array elements="<<max_ar;
13
14         return 0;
15     }
```

Below the code editor, a terminal window shows the execution output:

```
C:\ "d:\CPP\C EDUCATION 1 sem 1 year\c25\bin\Deb
20
24
101
8
Maximum value of array elements=101
Process returned 0 (0x0)   execution
Press any key to continue.
```

ДЯКУЮ ЗА УВАГУ!