

Лекція 4. Інструкції C++.



План на сьогодні

1

Що таке керуючі інструкції?

2

Умовні інструкції **if; else if; else**

3

Умовні інструкції **switch**

4

Цикли **for; while; do while**

5

Інструкція **break**

6

Інструкція **continue**



Що таке керуючі інструкції?



Що таке керуючі інструкції?

Керуючі інструкції — це команди в програмуванні, які змінюють послідовність виконання інших інструкцій у програмі. Вони дозволяють виконувати або повторювати різні частини коду залежно від певних умов.

Види інструкцій C++

Галужені

керують вибором гілки
виконання

if	умовна інструкція
switch	перемикач
continue	інструкція продовження
break	інструкція виходу
return	інструкція повернення
goto	Інструкція переходу

Циклічні

керують повторенням
виконання

for	покроковий цикл
while	цикл з передумовою
do-while	цикл з передумовою

Види інструкцій C++

Лінійні

<code>;</code>	порожня інструкція
<code>{ }</code>	блок
<code>expr;</code>	вираз

Усі інструкції, крім блоку, завершуються символом `;`;

Кожна інструкція може бути помічена міткою, яку розпізнає `goto`

`goto label;`

...

...

`label: statement;`

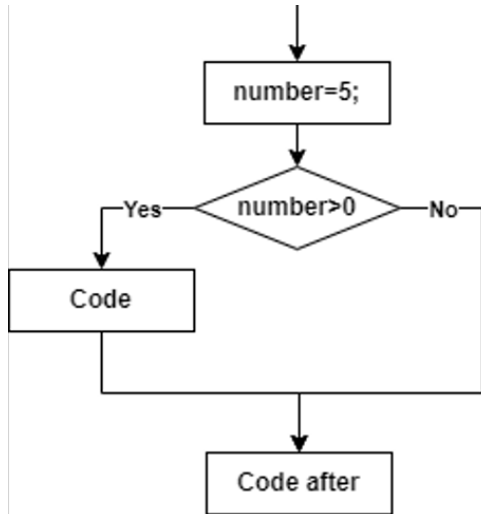
Умовні інструкції

if..else if..else



Умовна інструкція **if**

```
if (condition)
{
    // body of if statement
}
```



```
int number = 5;

if (number > 0) {
    // code
}

// code after if
```

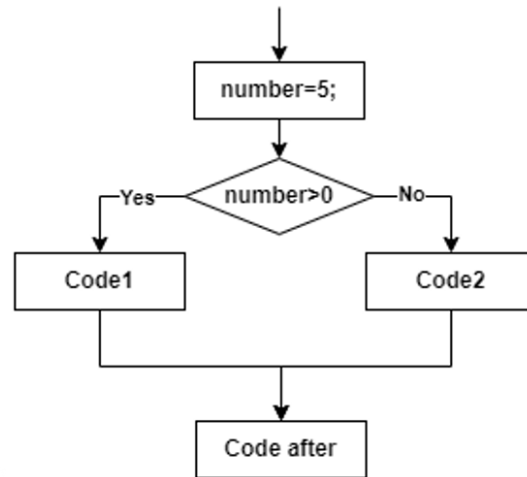
```
int number = 5;

if (number < 0) {
    // code
}

// code after if
```


Умовна інструкція **if..else**

```
if (condition)
{
    // block of code if condition
    // is true
}
else
{
    // block of code if condition
    // is false
}
```



Condition is true

```
int number = 5;

if (number > 0) {
    // code
}
else {
    // code
}

// code after if...else
```

Condition is false

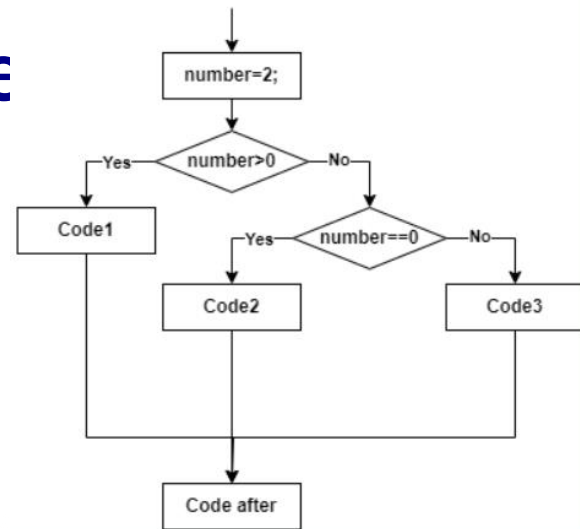
```
int number = 5;

if (number < 0) {
    // code
}
else {
    // code
}

// code after if...else
```

Умовна інструкція **if..else if..else**

```
if (condition1)
{
    // block of code if condition1
    // is true
}
else if (condition2)
{
    // block of code if condition2
    // is true
}
else
{
    // block of code if condition
    // is false
}
```



1st Condition is true

```
int number = 2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}
//code after if
```

2nd Condition is true

```
int number = 0;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}
//code after if
```

All Conditions are false

```
int number = -2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}
//code after if
```

Вкладена інструкція if..else

```
// outer if statement
```

```
if (condition1)
```

```
{
```

```
  // statements
```

```
  // inner if statement
```

```
  if (condition2)
```

```
  {
```

```
    // statements
```

```
  }
```

```
else
```

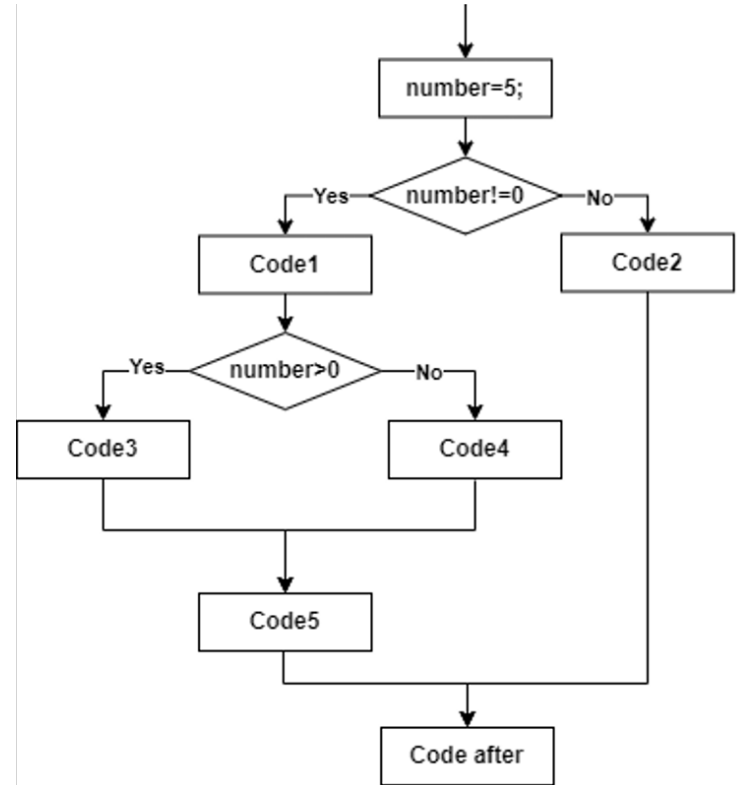
```
{
```

```
  // statements
```

```
}
```

```
// statements
```

```
}
```



Умовні інструкції switch



Умовна інструкція **switch**

```
switch (expression)
```

```
{
```

```
  case constant1:
```

```
    // code to be executed if
```

```
    // expression is equal to constant1;
```

```
    break;
```

```
  case constant2:
```

```
    // code to be executed if
```

```
    // expression is equal to constant2;
```

```
    break;
```

```
  .
```

```
  .
```

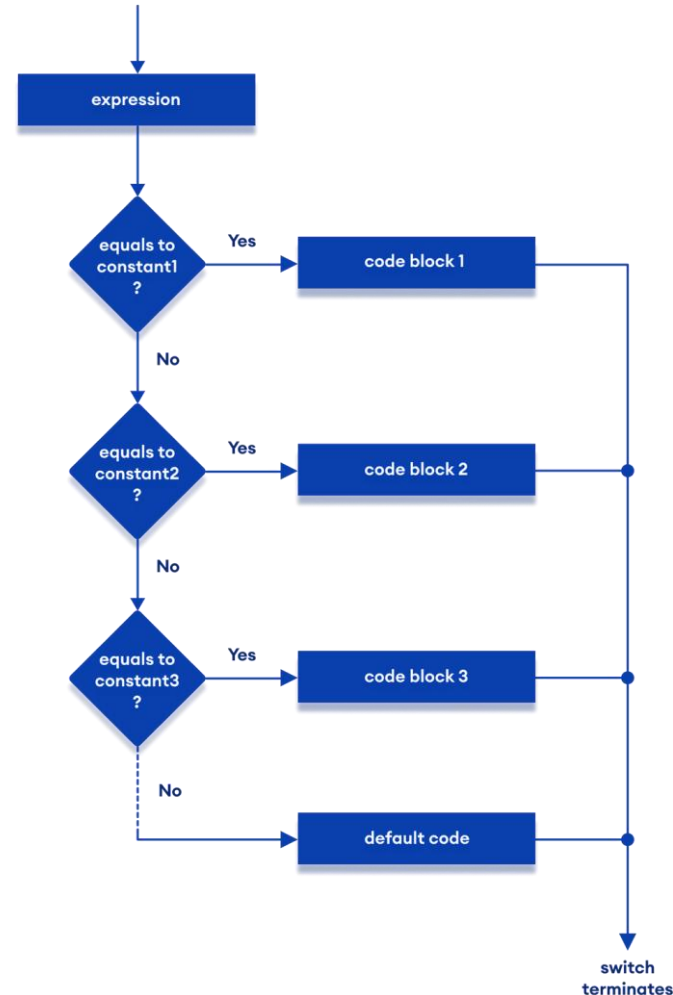
```
  .
```

```
  default:
```

```
    // code to be executed if
```

```
    // expression doesn't match any constant
```

```
}
```



Умовна інструкція **switch**

- Результат **expression** повинен бути інтегрального типу (**int**, **char**, **bool**)
- **constant1**, **constant2**, ... – повинні бути константними виразами

Алгоритм виконання **switch**:

1. обчислення **expression** - лише один раз на початку виконання
2. співставлення отриманого результат зі значенням **constant1**, **constant2**, ... у заданому порядку
3. при першому співпадінні значень (**constant_i == expression**) виконуються інструкції цього блоку з міткою **constant_i**,
інакше виконуються інструкції **за замовчуванням** (після мітки **default**), якщо вони наявні.
1. Якщо **break** (або **return**) відсутні серед інструкцій виконуваного блоку, то після його завершення почнеться виконання інструкцій наступного варіанту.

Цикл for



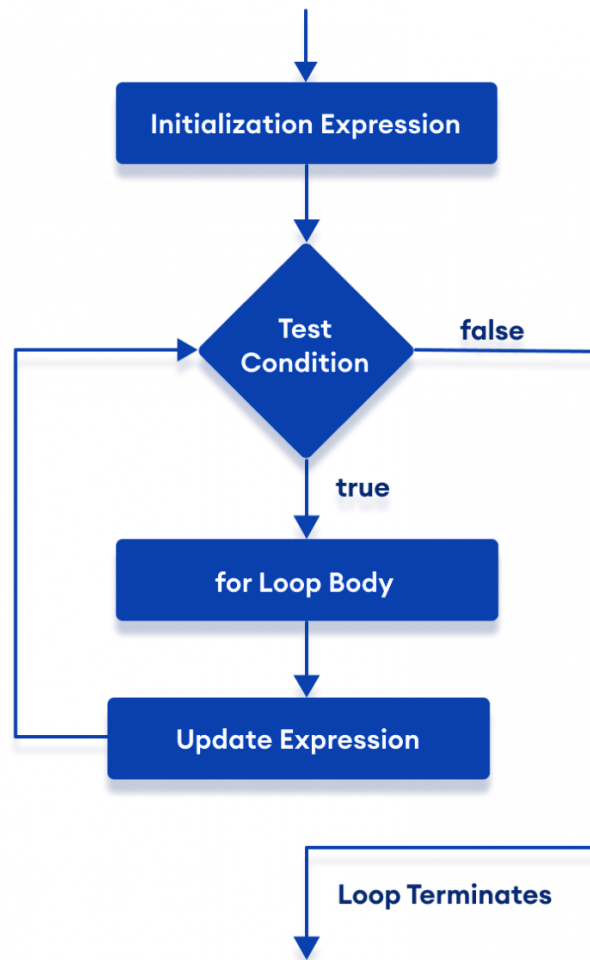
Цикл for

```
for (initialization; condition; update)
{
    // body of-loop
}
```

- **initialization** — ініціалізує змінні та виконується тільки один раз
- **condition** — якщо істинна, виконується тіло циклу for якщо хибна, цикл for завершується
- **update** — оновлює значення ініціалізованих змінних і знову перевіряє умову

Відсутність **condition** еквівалентна **true**

```
for ( ; ; ) // вічний цикл
{
    // body of-loop
}
```



Приклад табулювання функції

```
1 #include <iostream>
2 #include <iomanip> // For std::setw
3 #include <cmath>   // For std::sin
4
5 using namespace std;
6
7 int main() {
8     double start, end, step;
9
10    // User input for start, end, and step values
11    std::cout << "Enter the start value (in radians): ";
12    std::cin >> start;
13    std::cout << "Enter the end value (in radians): ";
14    std::cin >> end;
15    std::cout << "Enter the step size (in radians): ";
16    std::cin >> step;
17
18    // Print the table header
19    std::cout << std::fixed << setprecision(4);
20    std::cout << "x\tsin(x)\n";
21    std::cout << "-----\n";
22
23    // Tabulate the function values
24    for (double x = start; x <= end; x += step) {
25        std::cout << std::setw(6) << x << "\t" << sin(x) << "\n";
26    }
27
28    return 0;
29 }
30
```

```
Enter the start value (in radians): 0
Enter the end value (in radians): 1
Enter the step size (in radians): 0.1
x      sin(x)
-----
0.0000 0.0000
0.1000 0.0998
0.2000 0.1987
0.3000 0.2955
0.4000 0.3894
0.5000 0.4794
0.6000 0.5646
0.7000 0.6442
0.8000 0.7174
0.9000 0.7833
1.0000 0.8415
```

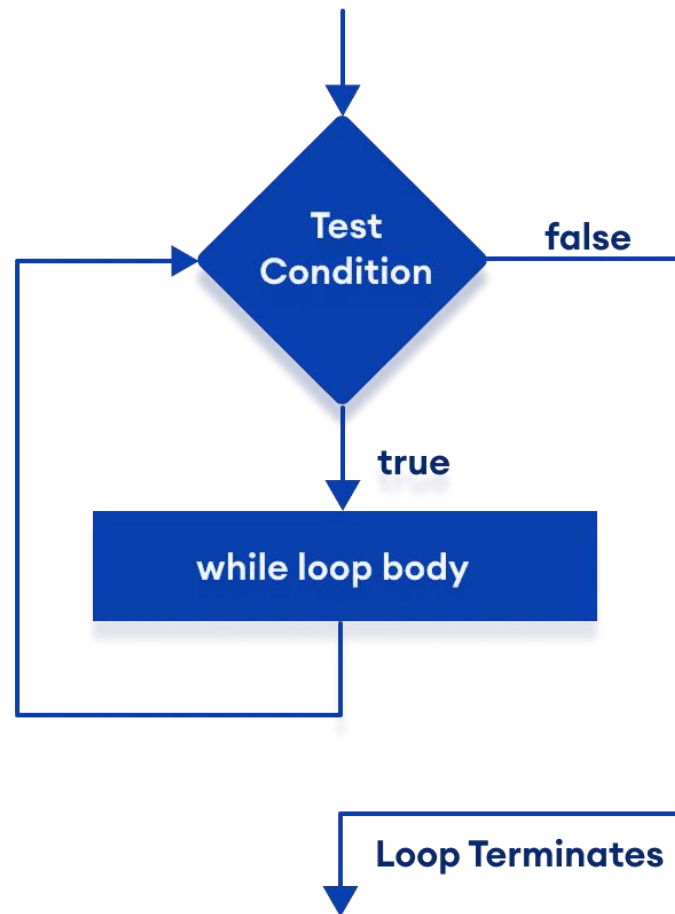
Цикл while



Цикл `while`

```
while (condition)
{
    // body of the loop
}
```

- Цикл `while` перевіряє умову.
- Якщо умова істинна, виконується код всередині циклу `while`.
- Потім умова перевіряється знову.
- Цей процес продовжується, поки умова є істинною.
- Коли умова стає хибною, цикл завершується.
- Наявність `condition` обов'язкова
- Під час виконання блоку циклу (або обчислення `condition`) компоненти `condition` мають модифікуватися – інакше "вічний" цикл.



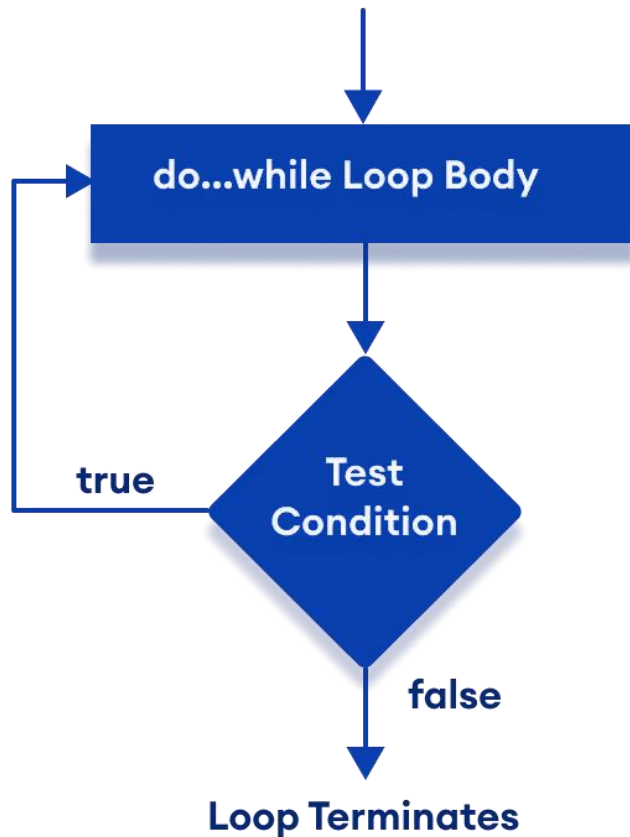
Цикл do..while



Цикл do..while

```
do  
{  
    // body of loop;  
}  
while (condition);
```

- Спочатку виконується тіло циклу. Потім перевіряється умова.
- Якщо умова істинна, тіло циклу всередині оператора do виконується знову.
- Умова перевіряється ще раз.
- Якщо умова істинна, тіло циклу всередині оператора do виконується знову.
- Цей процес продовжується, доки умова не стане хибною. Після цього цикл припиняється.



Інструкція break

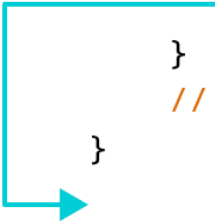


Інструкція **break**

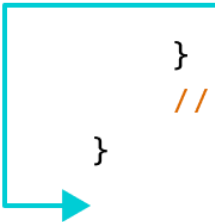
break;

Оператор **break** завершує цикл, коли він зустрічається.

```
for (init; condition; update) {  
    // code  
    if (condition to break) {  
        break;  
    }  
    // code  
}
```

A blue line starts from the 'break;' statement, goes left, then down, then right, ending in an arrow pointing to the right, indicating the exit from the loop.

```
while (condition) {  
    // code  
    if (condition to break) {  
        break;  
    }  
    // code  
}
```

A blue line starts from the 'break;' statement, goes left, then down, then right, ending in an arrow pointing to the right, indicating the exit from the loop.

Інструкція continue

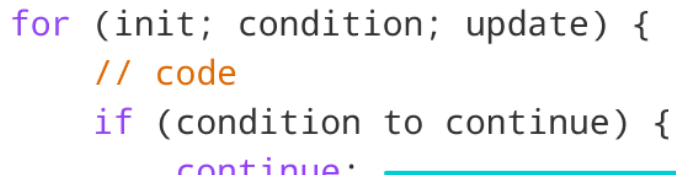


FACULTY OF APPLIED
MATHEMATICS AND
INFORMATICS
LVIV UNIVERSITY

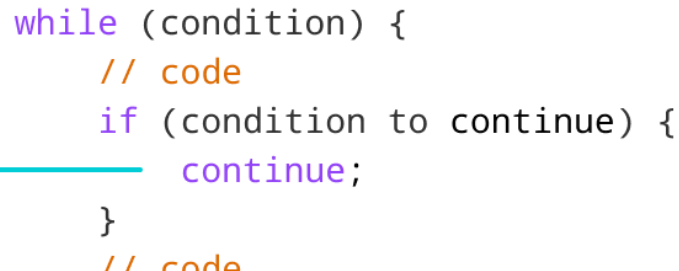
Інструкція **continue**

continue;

Оператор **continue** використовується для пропуску поточної ітерації циклу, після чого керування програмою переходить до наступної ітерації.



```
for (init; condition; update) {  
    // code  
    if (condition to continue) {  
        continue;  
    }  
    // code  
}
```



```
while (condition) {  
    // code  
    if (condition to continue) {  
        continue;  
    }  
    // code  
}
```

Дякую!