ОРГАНІЗАЦІЯ РОЗГАЛУЖЕНЬ. УМОВНИЙ ОПЕРАТОР ТА ОПЕРАТОР ВИБОРУ

Блок команд

```
{
  // команда 1;
  // команда 2;
  ...
  } // після блоку команд ; не ставимо
```

Реалізація розгалужень

Розгалуженням називається вибір програмою певної групи команд залежно від виконання певної умови, при цьому виконується лише одна з гілок алгоритму.

- безумовний перехід (`goto`);
- умовний перехід (`if`);
- вибір варіанта (`switch`).

Оператор безумовного переходу

```
int test;
goto Go_;//оператор goto
test = 5;
test ++;
Go_: //мітка
test = 10;
```

Хороший стиль – уникати команди goto

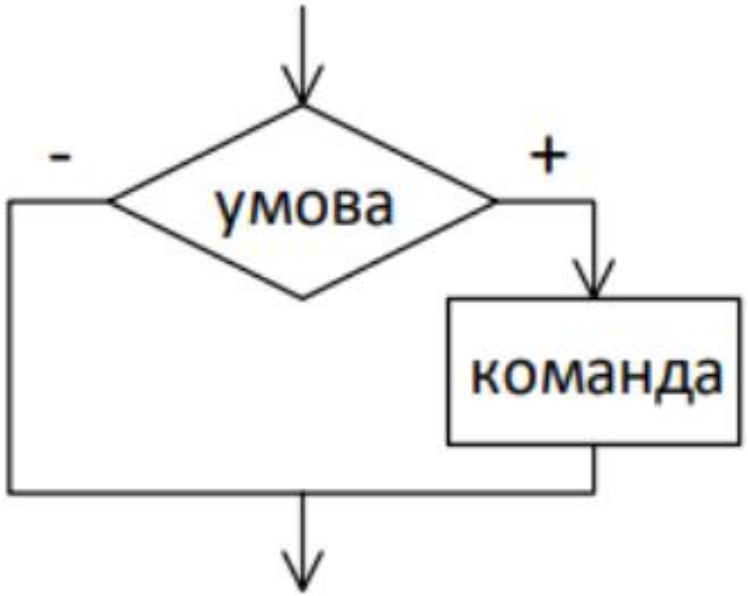
```
// можна
    int i;
    for (i = 0; i < 10; i++)
    {
        if (i == 3)
            goto STOP;
    }
STOP:
    cout << "i = " << i << endl;
```

```
// можна, але не рекомендується
    int i = 4;
    goto IN_FOR;
    for (i = 0; i < 10l i++)
    {
    IN_FOR:
        cout << i << endl;
    }
```

Оператор розгалуження

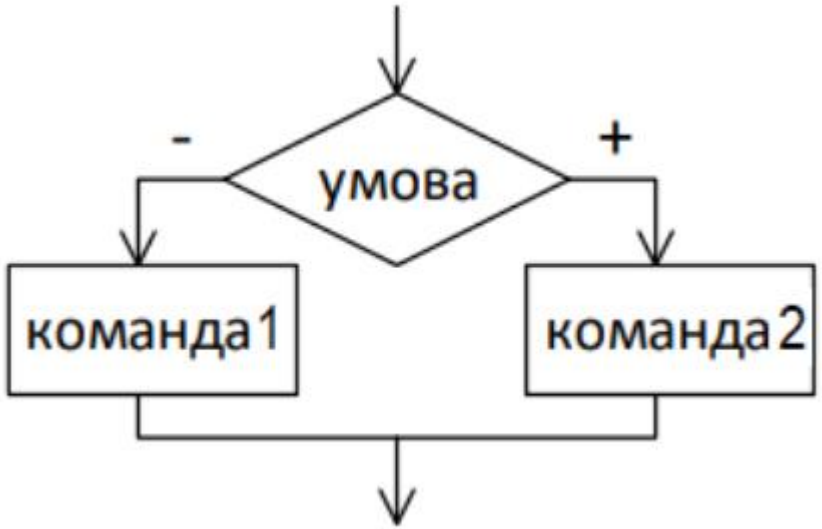
Скорочена форма оператора if

```
if (<умова>) <команда>;
```



Повна форма

```
if (<умова>) <команда 2>;
    else <команда 1>;
```




```
if (<умова>)
{
<команда 1>;
<команда 2>;
}
else
{
<команда 3>;
<команда 4>;
}
```

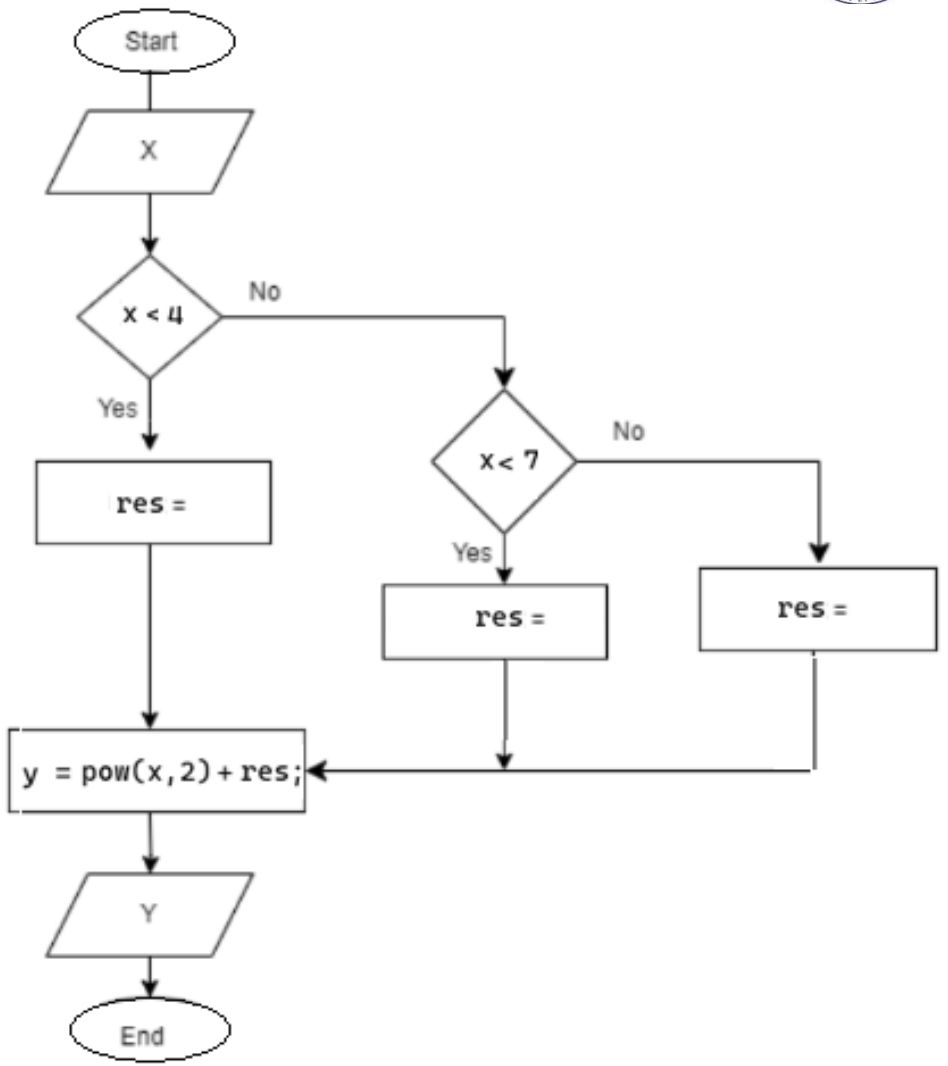
Умова перевірки певної цілої змінної k на ненульове значення

if(k!=0) або if(k)

if(k==0) або if(!k)

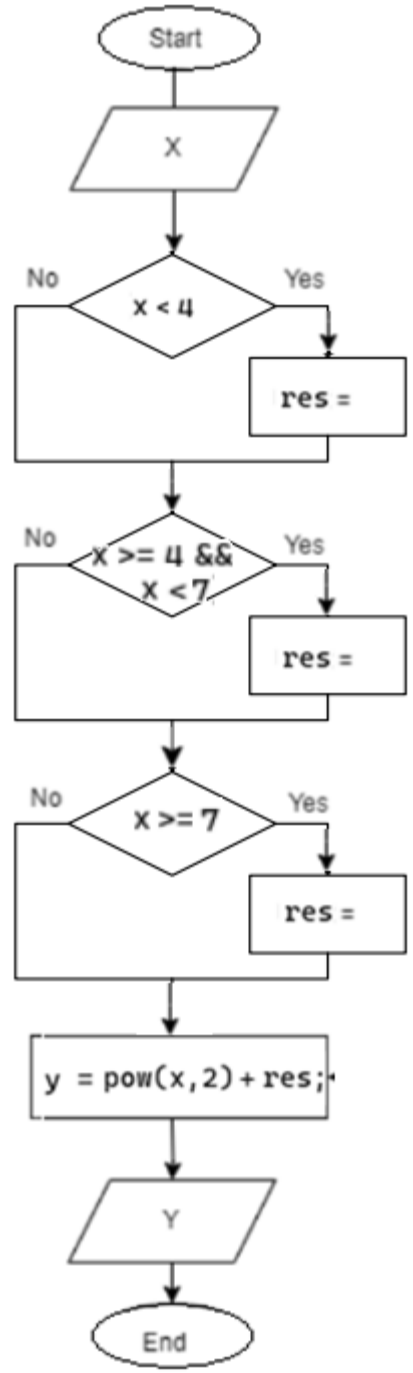
```cpp
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

int main() {
	//Оголошення локальних змінних
	int x;
	float y, res;
	//зчитування даних
	cout << ("Enter X / integer: ")<< endl;
	cin >>(x);
	//обчислення
	if (x < 4) {
		res = ((4 * pow(x, 7)) - (pow(x, 5)) + (pow(x, 3)) - 2);
	}
	else {
		// (x >= 4 and x < 7)
		if (x < 7)
		{
			res = ((atan((abs(x) + 1) / 2)) + 8.3 * x);
		}
		//(x > 7)
		else
			res = (log10(abs(2 * x + exp(4 * x + 1))));
	}
	y = pow(x, 2) + 1 + res;
	//виведення результату
	cout<< "Resul = "<< setprecision(5)<< y;
	return 0;
	}
```



```cpp
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

int main() {
	//Оголошення локальних змінних
	int x;
	float y, res;
	//зчитування даних
	cout << ("Enter X / integer: ")<< endl;
	cin >>(x);
	//обчислення
	if (x < 4) {
		res = ((4 * pow(x, 7)) - (pow(x, 5)) + (pow(x, 3)) - 2);
	}
	//(4 <= x < 7)
	if (x >= 4 && x < 7) // (x >= 4 and x < 7)
	{
		res = ((atan((abs(x) + 1) / 2)) + 8.3 * x);
	}
	//(x > 7)
	if (x >= 7) {
		res = (log10(abs(2 * x + exp(4 * x + 1))));
	}
	y = pow(x, 2) + 1 + res;
	//виведення результату
	cout<< "Resul = "<< setprecision(5)<< y;
	return 0;
}
```



Типові помилки при організації розгалужень

- *Присвоєння замість порівняння на збіг*

= замість ==:

```
int x;
/* ... */
if (x = 1)
    cout << "x == 1" << endl;
else
    cout << "x != 1" << endl;
```

- *Ланцюжок порівнянь*

$$2 < x < 5.$$

Якщо напишемо цю команду буквально:

```
if (2 < x < 5)
    cout << "x в діапазоні від 2 до 5" << endl;
else
    cout << "x поза діапазоном від 2 до 5" << endl;
```

помилка!

- *Вкладений* `if` *замість складної умови*

Не можна замість

```
int x, y, z;
// ...
if (x > 0 && y > 0)
    z = 1;
else
    z = 2;
```

використовувати

```
int x, y, z;
// ...
if (x > 0)
{
    if (y > 0)
        z = 1;
}
else
    z = 2;
```

Тернарна операція

<умова> ? < вираз 1> : < вираз 2>;

```
int i = 1, j = 2;
int k = i > j ? 3 : 4;
cout << k << endl;
```

```
int i = 1, j = 2, k;
if (i > j) k = 3;
else k = 4;
cout << k << endl;
```

```
int x, z, y;
cout << "Please, input x=";
cin >> x;
(x <= -4) ? y = x * x - 8 : (x >= 0) ? y = 2 - x : y = 3 * x - 2;
cout << "\n" << y;
```

Оператор вибору варіантів

`switch` дозволяє вибрати один з варіантів розгалуження

```
switch (<вираз>)
{ case <значення_мітка_1> : <послідовність_операторів_1>;
  break;
  ...
  case <значення_мітка_n> : <послідовність_операторів_N>;
  break;
  [ default: <послідовність_операторів>;
    break;]
}
```

вираз повинен бути цілого або символьного типу

оператор break здійснює вихід із switch

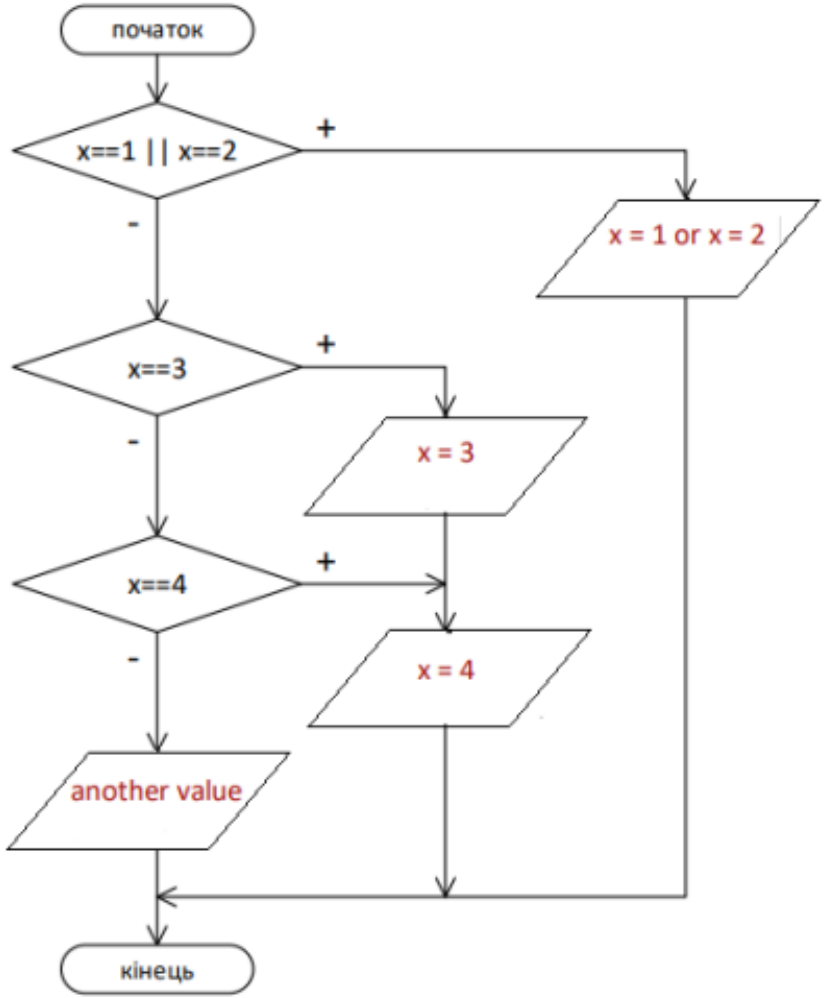
Якщо не спрацювала жодна перевірка – `default`

за введеним номером дня тижня вивести його назву

```
int num;
cout << "Input number from 1 to 7.\nnum =";
cin >> num;
switch (num)
{
case 1: cout << "mondey"; break;
case 2: cout << "tuesday"; break;
case 3: cout << "wednesday"; break;
case 4: cout << "thursday"; break;
case 5: cout << "friday"; break;
case 6: cout << "saturday"; break;
case 7: cout << "sunday"; break;
default: cout << "Error!"; break;
}
```

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

 int main()
{
        int x;
        switch (x)
        {
        case 1:
        case 2:
          cout << "x = 1 or x = 2" << endl;
          break;
        case 3:
          cout << "x = 3" << endl;
        case 4:
          cout << "x = 4" << endl;
          break;
        default:
          cout << "x has another value" << endl;
        }
        return 0;
}
```



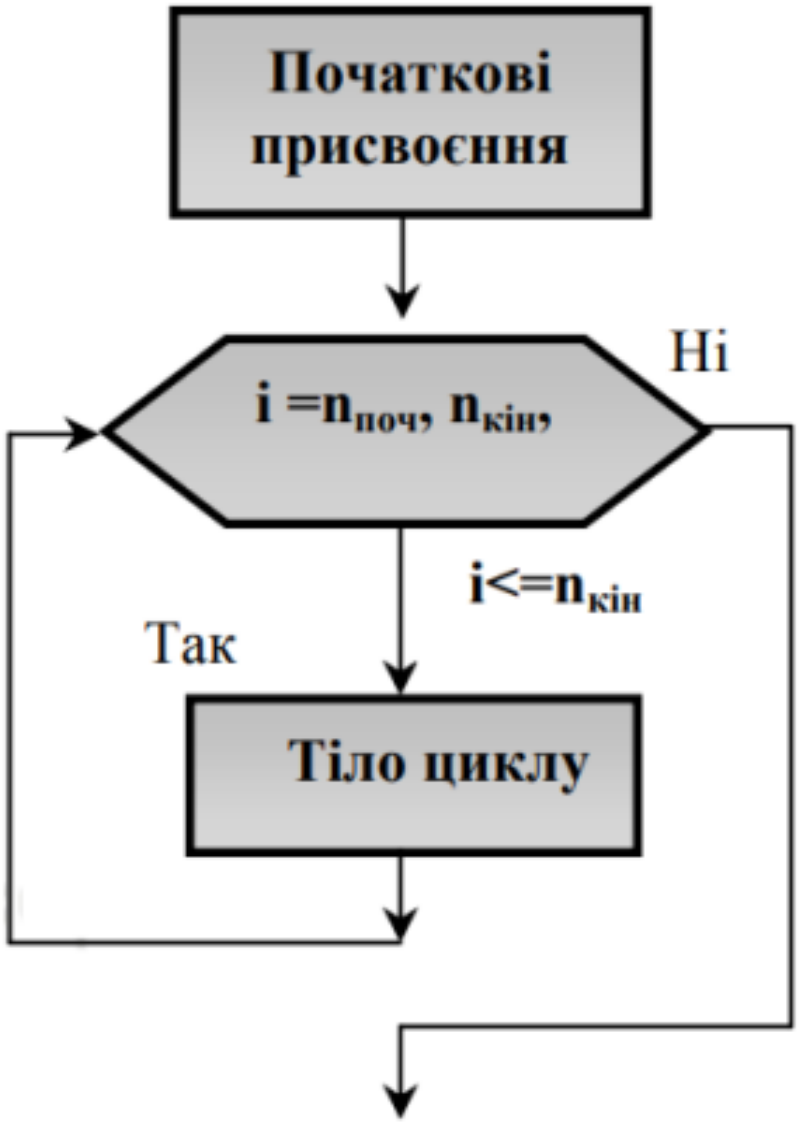
ОРГАНІЗАЦІЯ ЦИКЛІВ.
КЕРУЮЧІ ОПЕРАТОРИ В ЦИКЛАХ

цикл з параметром	відома кількість повторень.
цикл з передумовою	кількість повторень наперед невідома немає явно вираженого кроку зміни параметра
цикл з післяумовою	

Оператор циклу з параметром

```
for (<ініціалізація>; <умова>; <модифікація>)
<тіло_циклу>;
```

```
for  (<ініціалізація>; <умова>; <модифікація>)
{
<оператор_1>;
<оператор_2>;
...
<оператор_N>; }
```



Перевірка умови виконується на початку кожної ітерації циклу.

обчислення суми перших десяти чисел

```
int s = 0;
for (int i = 1; i <= 10; i++)
    s += i;
```

або

```
for (int s = 0, i = 10; i >= 1; i--)
    s += i;
```

або

```
for (int s = 0, i = 1; i <= 10; s += i++);
```

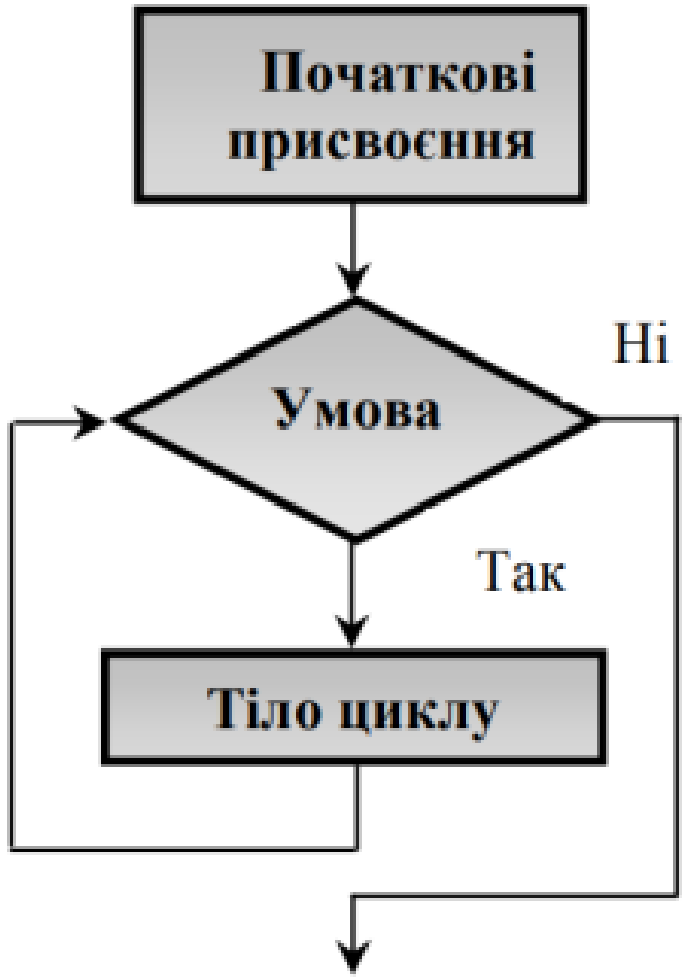
вкладені цикли

```
for (k = 1; k <= 10; k++)
  {
      for (i = 1; i <= 10; i++)
          {
             for (m = 1; m <= 10; m++)
                 {   }
           }
   }
```

Коли певний блок відсутній, тоді вираз цього блока пропускається, але крапка з комою (;) обов'язково має залишитись.

Оператор циклу з передумовою

```
while (<умова>)
  { <тіло_циклу> };
```



обчислення суми всіх непарних чисел у діапазоні від 10 до 100

```
int s = 0, i = 11;
while (i < 100)
{
    s += i;
    i += 2;
}
```

Оператор циклу з післяумовою

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

завжди виконується принаймні один раз незалежно від істинності чи хибності умови.

Обчислити суму всіх непарних чисел у діапазоні від 10 до 100

```
int s = 0, i = 11;
do {
    s += i; i += 2;
   }
while (i < 100);
```

Оператори переривання виконання циклів

Для завчасного виходу з циклу використовують оператори

`break` (вихід з конструкції)

goto (безумовний перехід)

return (вихід з поточної функції)

```
int k = 1;
for (int i = 0; i < m - 1; i++)
{
    for (int j = i + 1; j < m; j++)
        if (i == m - j) break;
        if (j == m) k++;
}
```

```
for (i = 0; i < 100; i++)
    for (j = 0; j < 100; j++)
     {
...
        if (j - i) < 0) { break; break; }
...
     }
```

Всередині вкладених операторів `do-while`, `for`, `while` чи `switch` оператор `break` завершує лише той оператор, якому він належить, не можна використовувати для виходу з декількох вкладених циклів.

Для *переходу до наступної ітерації циклу* всередині операторів циклів

`for`, `while`, `do-while` — оператор `continue`

```
for (int i = 3; i < 7; i++)
    if (i == 5) continue; // Пропустити елемент, рівний 5
    else
        cout << i; // Результат: 3  4  6  7
```

+ оператори `return` та `goto`.

Дякую за увагу

Лектор:
кандидат фіз.-мат. наук, доцент
Шаклеіна Ірина
iryna.o.shakleina@lpnu.ua
кафедра ІСМ, ІКНІ