# Оператори управління. Оператори вибору

У лекції розглядаються оператори управління: оператори вибору, оператори повторення, оператори передачі управління (переходу). Розглядаються синтаксичні правила, приклади використання.

**Мета:** Дати базові знання з використання операторів управління.

# Типи операторів управління

*Оператори вибору:*

***if***

***if / else***

***switch***

*Оператори повторення:*

***while***

***do / while***

***for***

*Оператори передачі управління (переходу):*

***break***

***continue***

***return***

***goto***

# *if (якщо)*
# Оператор з єдиним вибором або умовний оператор з єдиним вибором

Оператор з єдиним вибором виконує зазначену дію, якщо умова є істиною і пропускає ( не виконує) цю дію в іншому випадку.

# Синтаксис оператора if

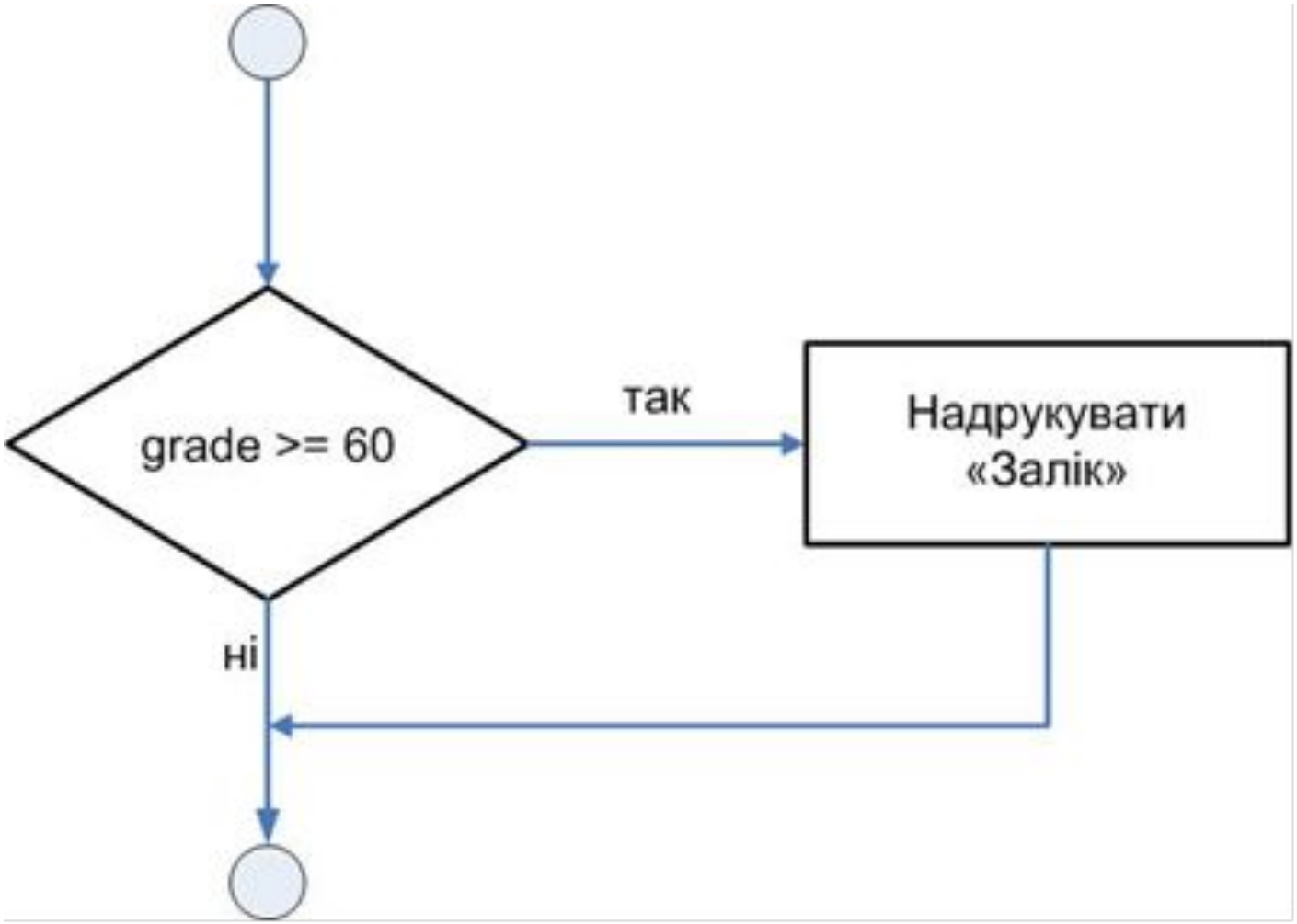
```
if (умовний вираз)
оператор;
```

```
if (умовний вираз)
{   оператор1;
     оператор2;
    …
}
```

# Блок-схема оператора з єдиним вибором

ЯКЩО оцінка студента більше чи дорівнює 60 балів

Надрукувати "Залік"



# Запис оператора мовою С++

```
int  grade;
…
if (grade  >= 60   )
cout<<"Залік";
…
```

# *if/else (якщо/інакше)*
# Оператор з подвійним вибором або умовний оператор з подвійним вибором

Оператор з подвійним вибором виконує одну дію, якщо умова є істиною і виконує іншу дію в іншому випадку. Оператор здійснює вибір між двома різними діями.

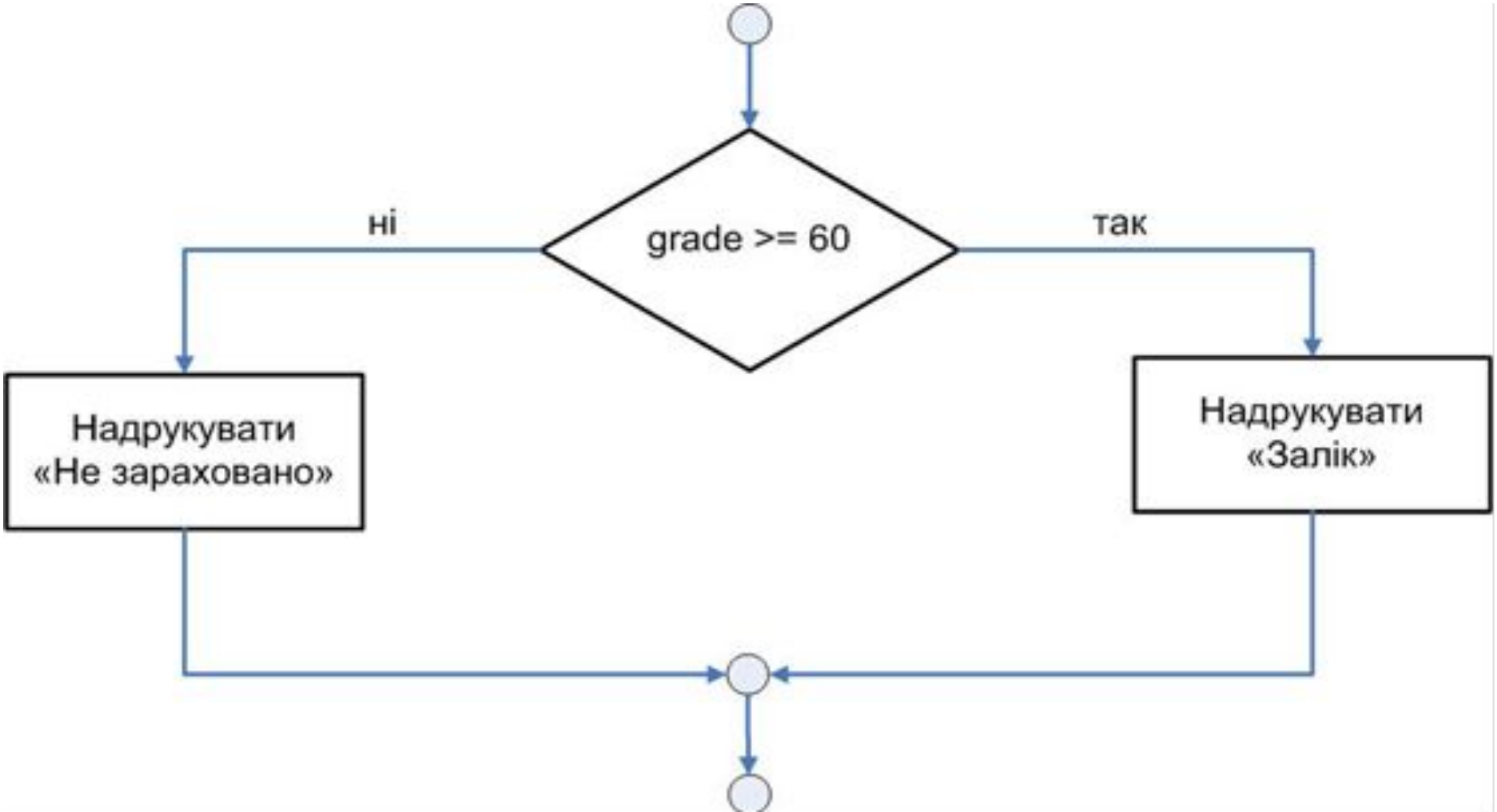
# Синтаксис оператора if/else

```
if (умовний вираз)
оператор1;
else
оператор2;
```

```
if (умовний вираз)
{   оператор1;
     оператор2;
}
else
{   оператор3;
     оператор4;
}
```

# Блок-схема оператора з подвійним вибором

ЯКЩО оцінка студента більше чи дорівнює 60 балів
Надрукувати «Залік» ІНАКШЕ Надрукувати «Не зараховано»



# Запис оператора мовою С++

```
int  grade;
…
if ( grade >= 60 )
cout << "Залік";
else
cout<<"Не зараховано”;
…
```

# *Умовний оператор ?*

Умовний оператор (?:) є близьким до оператора if/else. Це тернарний оператор, який має три операнди. Перший операнд є умовою, другий операнд дорівнює значенню виразу у випадку, якщо умова істинна, а третій операнд дорівнює значенню виразу, якщо умова помилкова.

# Синтаксис оператору ?

Умова ? Вираз1 : Вираз2;

# Приклад використання оператору ?

ЯКЩО оцінка студента більше чи дорівнює 60 балів
Надрукувати «Залік» ІНАКШЕ Надрукувати «Не зараховано»

```
int  grade;
…
grade>=60? cout <<"Залік" : cout <<"Не зараховано"  ;
```

або

```
  cout <<(grade>=60? "Залік" : "Не зараховано" );
```

# Приклад використання оператору ?

```
int x=10;
y=x>9 ? 100 : 200;
```

```
int x=10;
if (x>9)
  y=100;
    else
     y=200;
```

# *Вкладені оператори if/else (якщо/інакше)*

Вкладені оператори використовуються для організації множинного вибору. Оператор здійснює вибір між декількома різними діями.

# Синтаксис вкладених операторів if/else

```
if (умовний вираз1) оператор1;
else
    if(умовний вираз2) оператор2;
    else
        if(умовний вираз3) оператор3;
…
        else операторN;
```

# Приклад використання вкладених операторів if/else

```
int  grade;
…
if (grade >= 100 || grade < 0)
    cout<<“Error”<<endl;
    else if (grade >= 90)
        cout<<“Excellent”<<endl;
        else if (grade >=75)
            cout<<“Good”<<endl;
            else if (grade>=60)
                cout<<“Not bad”<<endl;
                else
                    cout<<“Bad”<<endl;
```

# *Оператор множинного вибору switch*

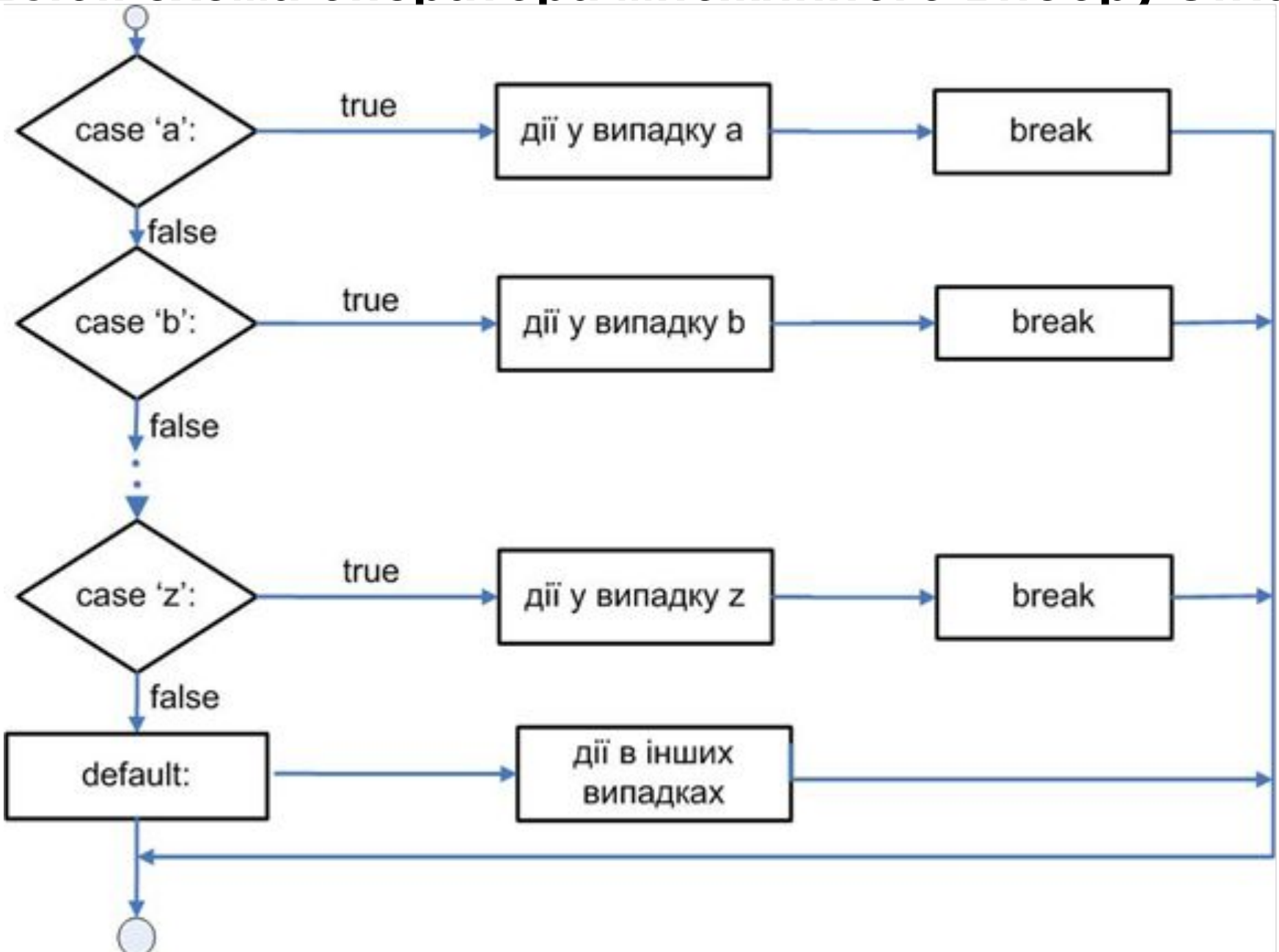
Оператор множинного вибору призначений для вибору гілки обчислень виходячи із значення виразу управління. Значення виразу має бути таким, щоб його можна було представити **цілим числом** або **символом**.

Значення виразу управління порівнюється із значенням списку цілих або символьних констант.

# Синтаксис оператору switch

```
switch (вираз управління)
{
    case константа1:
            оператор(и);
            break;
    case константа2:
            оператор(и);
            break;
…
    default:
        оператор(и);
}
```

# Блок-схема оператора множинного вибору switch



# Приклад використання оператора switch

```
int auto_class, price;
cout<<”Input class_auto: 1, 2, 3”<<endl;
cin>>auto_class;
switch (auto_class)
{
      case 1:
            cout<<”Легковий автомобіль”;
            price=20;
            break;
      case 2:
            cout<<”Автобус”;
            price=40;
            break;
      case 3:
            cout<<”Вантажна”;
            price=45;
            break;
      default:
cout<<”Невідомий транспортний засіб”;
}
```

# Оператори управління. Оператори повторення та передачі управління

У лекції розглядаються оператори управління: оператори вибору, оператори повторення, оператори передачі управління (переходу). Розглядаються синтаксичні правила, приклади використання.

**Мета:** Дати базові знання з використання операторів управління.

# *Оператор повторення (циклу) з передумовою while*

Оператор повторення дозволяє визначити дію, яка повинна повторюватися, поки зазначена умова залишається істинною.

Приклад опису повторних дій при відвідуванні магазину.

***ПОКИ*** *є елементи в моєму списку покупок*
***Зробити*** *наступну покупку і викреслити її зі списку*

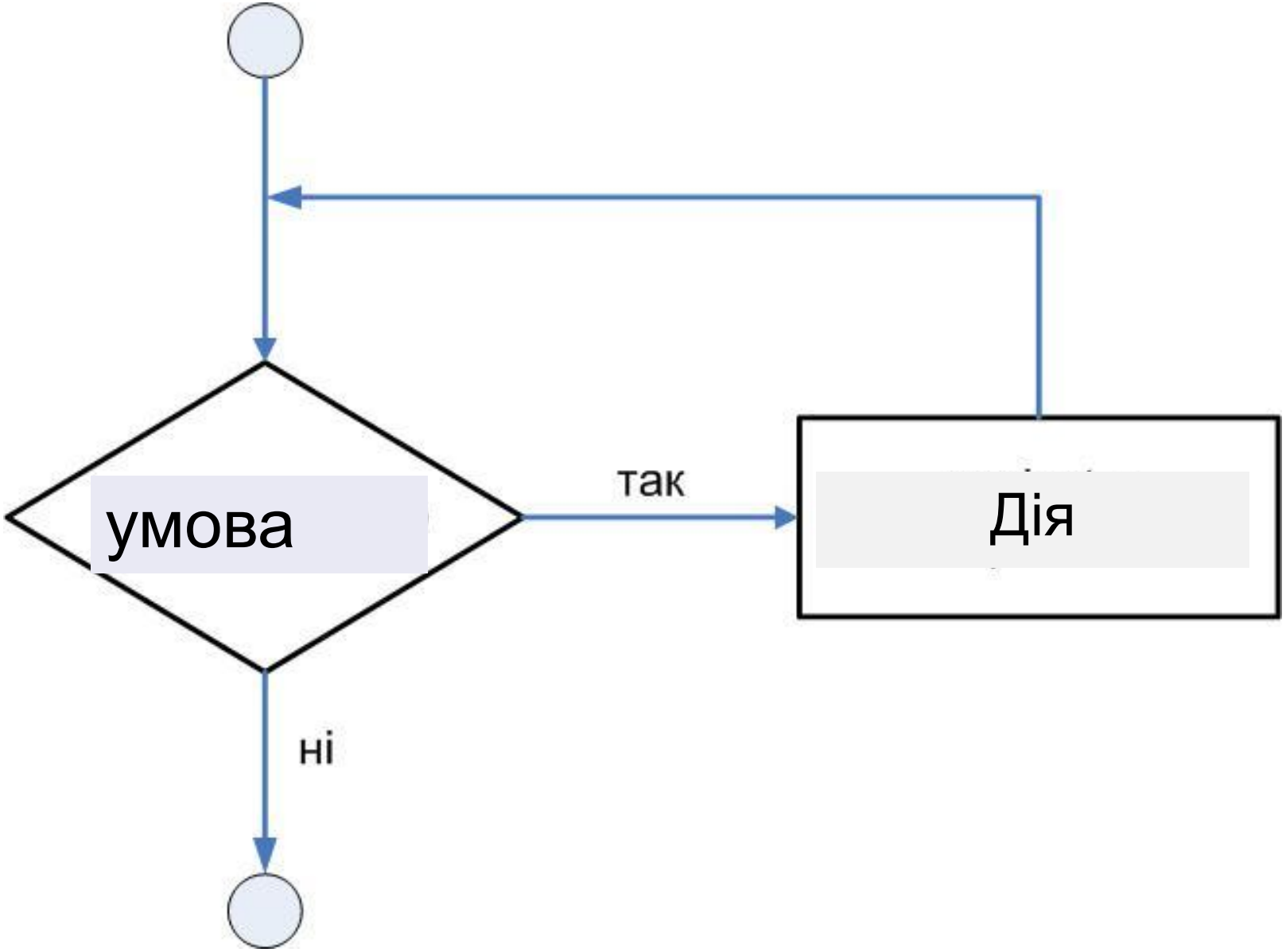
# Синтаксис оператору while

**while** (умова виконання циклу )
оператор;

```
while (умова виконання циклу )
{
оператор1;
оператор2;
…
}
```

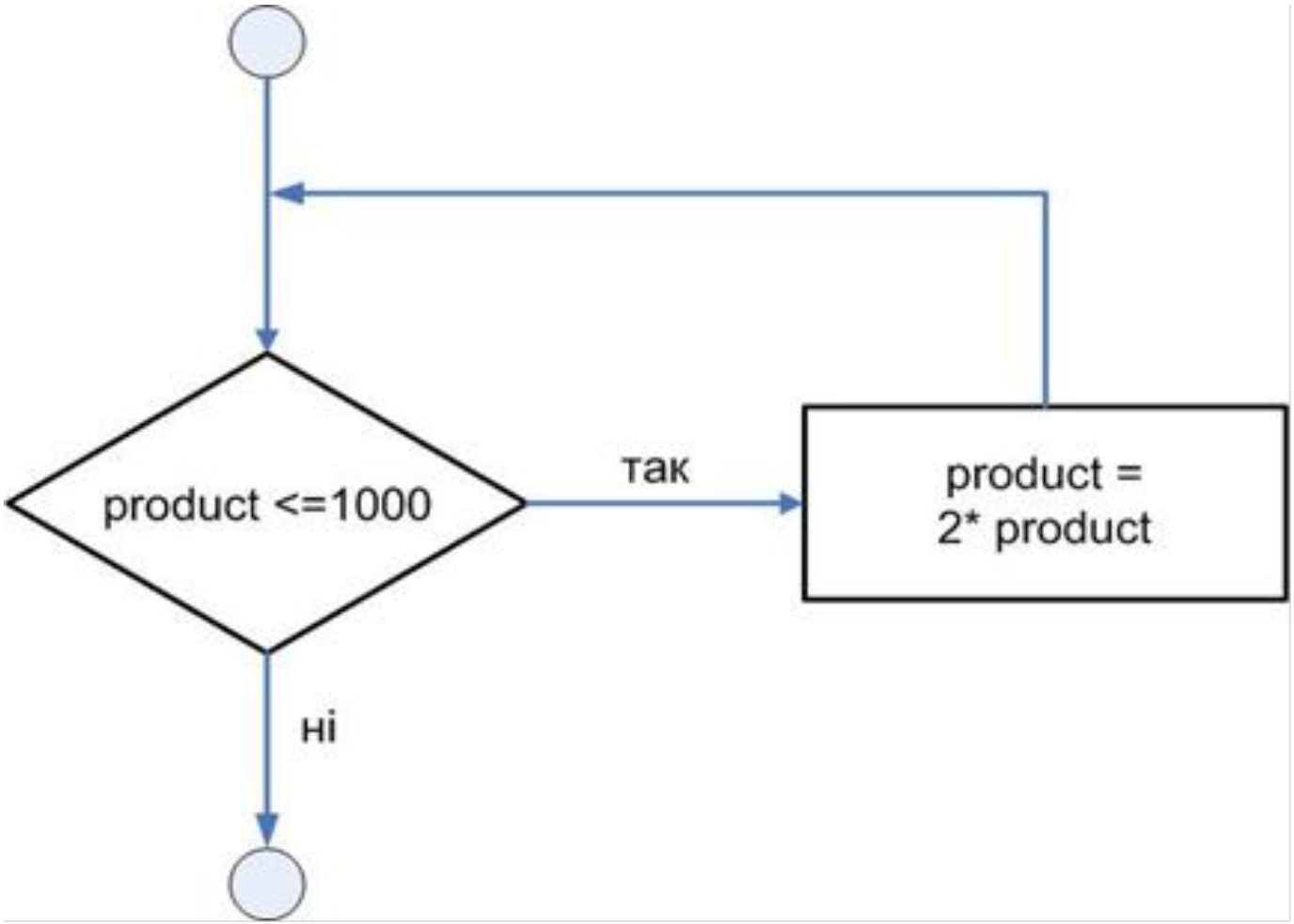
Оператор або оператори укладені у фігурні дужки називають **тілом циклу**. Цей оператор (оператори) повторюються до тих пор, поки умова продовження циклу є істиною. Коли **while** завершується, виконання програми продовжується з оператора, що слідує за оператором **while.**

# Блок-схема оператора повторення while



# Приклад використання оператора while

```
int product = 2;
while (product <= 1000)
product = 2*product;
cout<< product;
```



# Приклад використання оператора while

```
int  product  =  2;
while   (product >2)
product  =  2*product;
cout<< product;
```

# Приклад використання оператора while

```
int  i  =  1;
while   (i  <=   10)
   {
    cout<< i<<“  “;
    i++;
    }
cout<<endl<< i<<endl;
```

# Приклад використання оператора while

Якщо у тілі структури while не передбачається дія, яка приводить до того, що згодом умова while стане хибною, виконання подібного оператору повторення ніколи не перерветься — така помилка називається «зациклення».

```
int  i  =  1;
while   (i  <=   10)
   {
    cout<< i<<“  “;
    }
cout<<endl<< i<<endl;
```

# *Оператор повторення (циклу) з післяумовою do/while*

Оператор повторення дозволяє визначити дію, яка повинна повторюватися, поки зазначена умова залишається істинною.

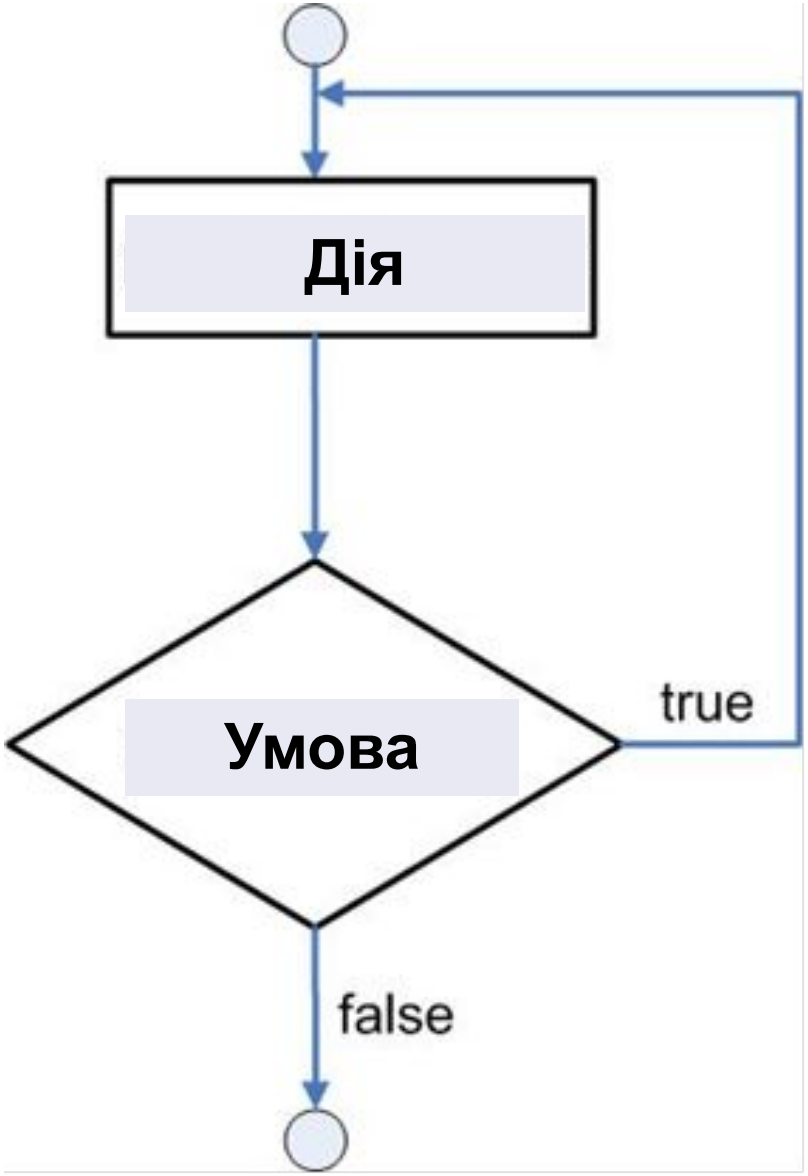
У операторі **do/while** перевірка умови продовження циклів виконується після виконання тіла циклу, отже тіло циклу буде виконане принаймні один раз.

# Синтаксис оператору while

```
do
 {
    оператори;
 }   while  (умова виконання циклу);
```

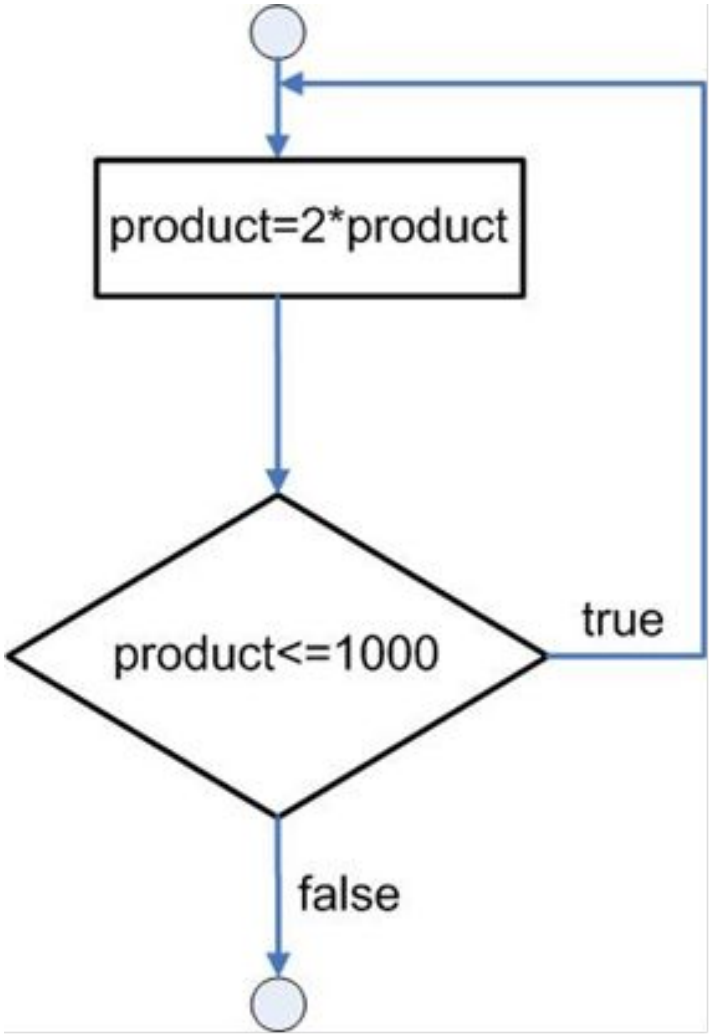
Оператор або оператори укладені у фігурні дужки називають **тілом циклу**. Цей оператор (оператори) повторюються до тих пор, поки умова виконання циклу є істиною.

# Блок-схема оператора повторення while



# Приклад використання оператора while

```
 int  product  =  2;
do
{
product  =  2*product;
} while  (product  <=  1000);
cout<< product;
```



# Приклад використання оператора while

```
 int  product  =  2;
do
{
product  =  2*product;
} while   (product >2);
cout<< product;
```

# Приклад використання оператора while

```
int i = 1;
do {
    cout << i << "  ";
  } while ( ++i <= 10 );
cout <<endl<<i<< endl;
```

# *Оператор повторення (циклу) з лічильником for*

Оператор повторення **for** використовується для повторення дій, які управляються змінною циклу (лічильником).

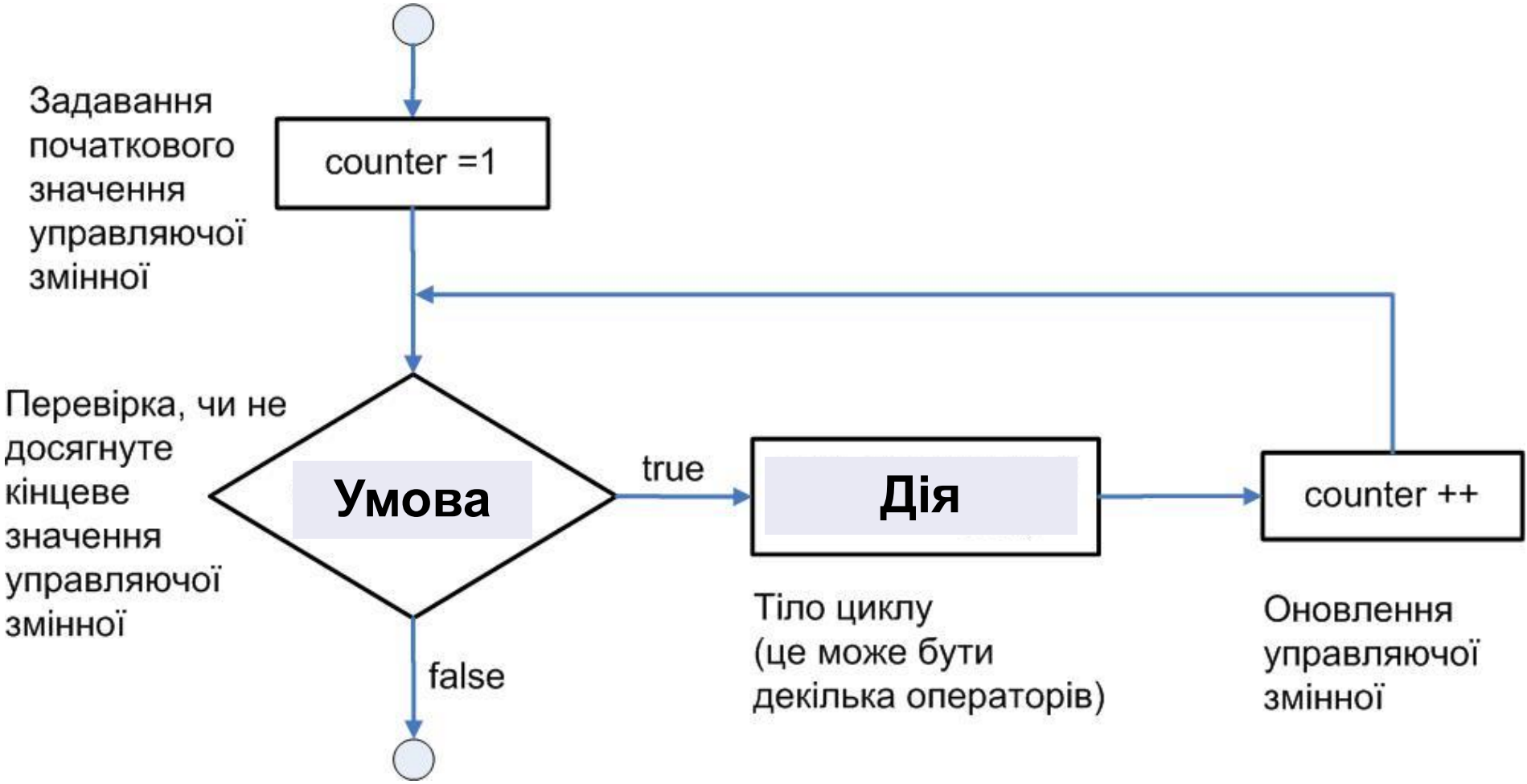
# Синтаксис оператору for

**for**( ініціалізація лічильника; умова виконання циклу; оновлення лічильника)
оператор;

**for**(ініціалізація лічильника; умова виконання циклу; оновлення лічильника)
{
оператори;
}

Оператор містить **заголовок циклу** та **тіло циклу**, яке повторюється визначену у заголовку циклу кількість разів.

# Блок-схема оператора повторення for

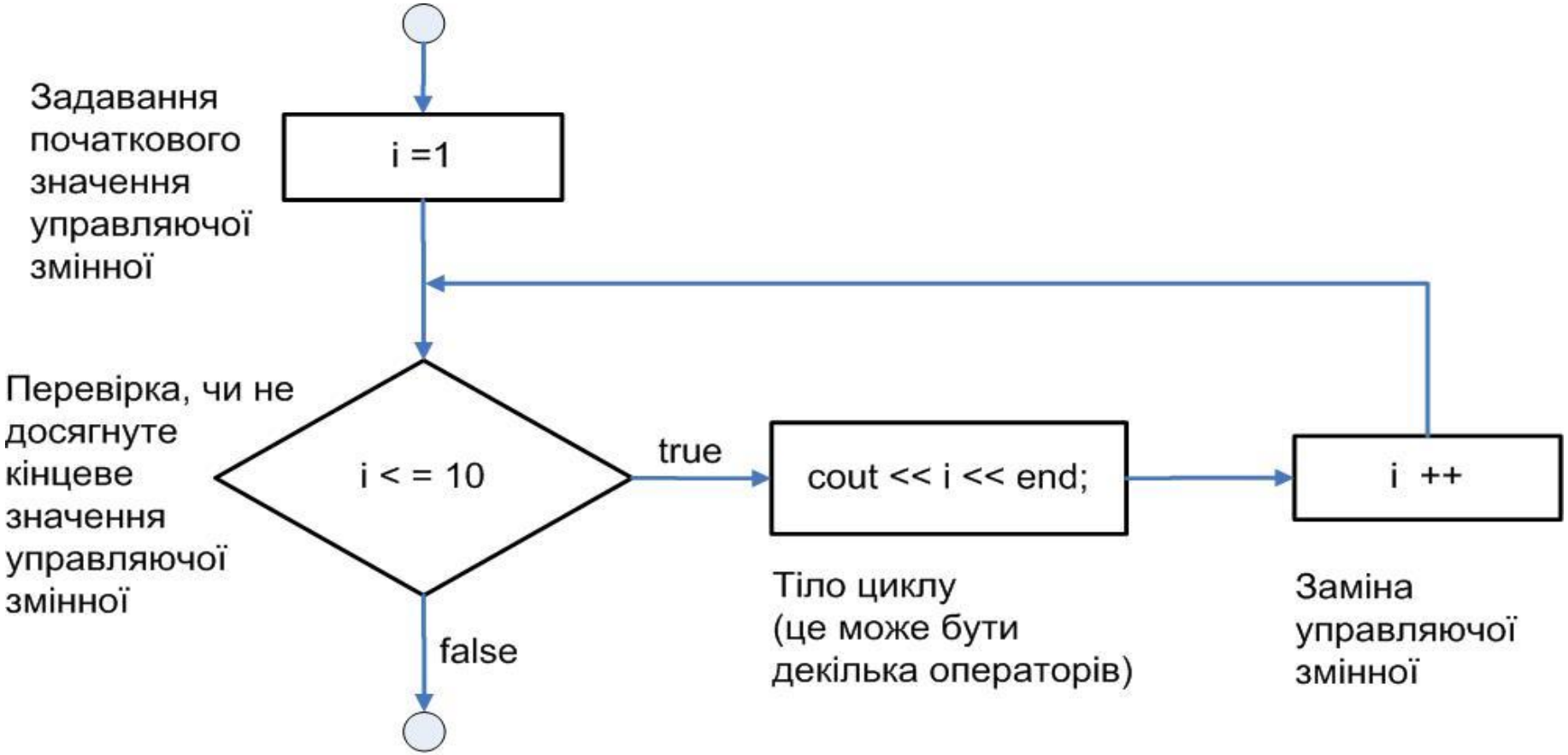


Цикл виконується в такому порядку:

1. Встановлення початкового значення змінної управління циклом (лічильника);
2. Перевірка умови виконання циклу, якщо умова є хибною, то переходимо до п. 5;
3. Виконання дій всередині циклу, тобто тіла циклу;
4. Оновлення (приріст) значення змінної циклу та перехід до п.2.
5. Закінчення циклу.

# Приклад використання оператора for

```
for(int i=1;   i<=10;   i++)
cout<<i<<endl;
```




Мал. 4.7. Блок-схема типової структури повторення **for**

# Приклад використання оператора for

```
for(int counter=l;   counter<=10;   counter++)
    cout<<counter*2<<”  “;
```

---

```
for(int counter=2;  counter<=10;   counter=counter+2)
    cout<<counter<<”  “;
```

---

```
for(int counter=l;   counter<=10;   counter++)
    cout<<“Hello!  “;
```

# Приклад використання оператора for

«Приріст» оператора for може бути негативним, у цьому випадку відбувається не збільшення, а зменшення змінної циклу.

```
for(int i=10;  i<=0;   i=i-1)
    cout<<i<<”  “;
```

---

| | |
|---|---|
| i = i + 1 | i = i - 1 |
| i +=1 | i -=1 |
| ++i | --i |
| i++ | i-- |

# Приклад використання оператора for

Змінна управління циклу може бути оголошеною поза межами циклу.

```
int i;
for(i=l;   i<=10; i++)
cout<<i<<”  “;
cout <<endl<<i<< endl;
```

Вирази *„ініціалізація змінної циклу”* та *„оновлення змінної циклу” можуть* представлятися як списки виразів, розділені комами.

В одному операторі for може бути декілька управляючих змінних, котрим треба задавати початкове значення і які треба змінювати.

```
int i, j;
for(i=1, j=1;   i<=10, j<5;   i++, j++)
cout<<i*j<<”  “;
```

Три вирази в заголовку оператора for є необов’язковими. Якщо вираз „*умова виконання циклу”* відсутній, то умова продовження циклу завжди істинна й таким чином утворюється нескінченно повторюваний цикл.

Може бути відсутній вираз „*ініціалізація змінної циклу”,* якщо початкове значення лічильника задане в іншому місці програми.

Може бути відсутній і вираз „*оновлення змінної циклу”,* якщо збільшення змінної здійснюється в тілі циклу або якщо збільшення не потрібно.

## Нескінченний цикл.

*for (; ; )*

*оператор;*

***Порожній цикл***

```
for(int counter=1;   counter<=10;   counter++) ;
cout<<counter<<“  “;
```

Розміщення крапки з комою відразу після правої закриваючої дужки заголовка for робить тіло структури порожнім оператором.

Такий цикл for з порожнім тілом виконується зазначену кількість разів, не роблячи нічого, крім оновлення змінної циклу.

Початкове значення, умова продовження циклу і оновлення змінної циклу можуть містити **арифметичні вирази**.

```
int x = 2, y = 10;
for(int j=x;   j<=4*x*y;   j+=y/x)
  cout<<j<<”  “;
```

Цей оператор еквівалентний оператору

```
for (int j=2;   j<=80;   j+=5)
cout<<j<<”  “;
```

# Оператори переходу (передачі управління)
## break, continue, return, goto

Оператори **break** і **continue, return, goto** змінюють потік управління, тобто порядок виконання програми.

# Оператор **break**

Оператор **break** може використовуватися в операторах циклу **while, for, do/while** або в тілі оператору вибора **switch.**

Звичайне призначення оператора **break** — достроково перервати цикл чи пропустити частину оператора switch, що залишилася.

# Приклад використання **break**

```
#include <iostream>
using namespace std;
void main() {
int x;
for ( x = 1; x <= 10; x++ )
{
   if ( x ==5 )
   break;
cout << x << "   ";
}
cout << endl<< "x == " << x;     }
```

# Оператор **continue**

Оператор **continue** може використовуватися в операторах повторення **while, for, do/while.**

Цей оператор дозволяє пропускати частину тіла циклу, яка залишилася, і починати виконання наступної ітерації циклу. В циклах **while** і **do/while** негайно після виконання оператора **continue** здійснюється перевірка умови продовження циклу. У циклі for виконується вираз приросту, а потім здійснюється перевірка умови продовження.

# Приклад використання **continue**

```
#include <iostream>
using namespace std;
void main() {
int x;
for ( x = 1; x <= 10; x++ )
{
   if ( x ==5 )
 continue;
cout << x << "   ";
}
cout << endl<< "x == " << x;     }
```

# Оператор **return**

Оператор **return** можна використовувати в будь-якому місці функції. Коли зустрічається оператор **return** відбувається закінчення виконання відповідної функції**.**

Звичайне призначення оператора **return** — достроково закінчити виконання функції.

# Приклад використання **return**

```
#include <iostream>
using namespace std;
void main() {
int x;
for ( x = 1; x <= 10; x++ )
{
   if ( x ==5 )
 return;
cout << x << "   ";
}
cout << endl<< "x == " << x;     }
```

# Оператор **goto**

Оператор goto може використовуватися в будь-якому місці функції для переходу на виконання фрагменту коду, який помічено міткою. Мітка – це ідентифікатор, після якого ставиться двокрапка. Мітка  та оператор goto має знаходитися в одній функції. Загальна форма оператора goto є такою:

**goto** мітка;

…

.

мітка**:**

# Приклад використання **goto**

```
int x=1;
loop1:
 cout<<x<<endl;
 x++;
if(x<=10) goto loop1;
```