

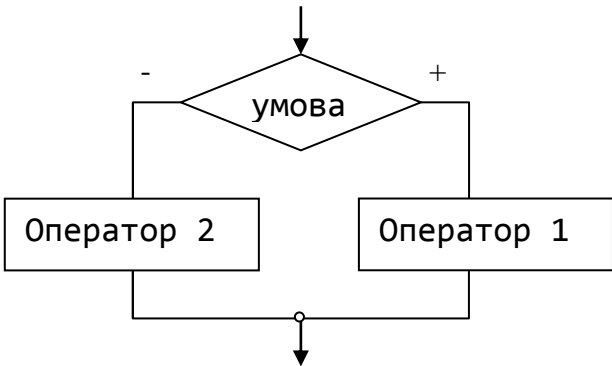
РЕАЛІЗАЦІЯ АЛГОРИТМІВ ІЗ РОЗГАЛУЖЕННЯМ

Розгалуженням називається алгоритмічна конструкція, що дозволяє виконавцеві алгоритму вибирати ту чи іншу послідовність дій залежно від певних умов.

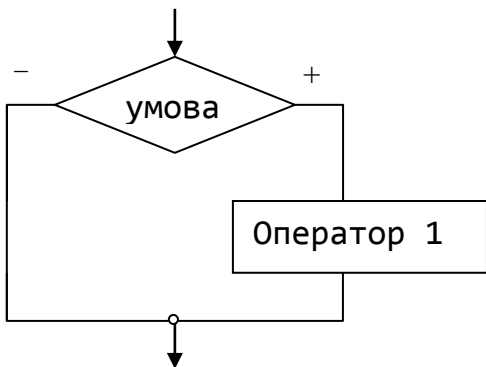
1. Умовний оператор

Алгоритмічна конструкція вибору з двох альтернатив, дозволяє виконавцеві алгоритму вибирати один із двох варіантів дій залежно від істинності деякої умови. У мові C# така конструкція реалізується *умовним оператором* (*оператором розгалуження*). Існують дві форми для даного оператора *повна* та *скорочена*.

Повна форма

Програмна структура	Аналог на мові блок-схем	Приклад
<pre>if (<умова>) <оператор1>; else <оператор2>;</pre>		<pre>if(x>y) max=x; else max=y;</pre>

Скорочена форма

Програмна структура	Аналог на мові блок-схем	Приклад
<pre>if (<умова>) <оператор1>;</pre>		<pre>if(x!=0) z=1/x;</pre>

Приклад. Визначити чи належить значення дійсної змінної x проміжку $[0,1]$ та вивести відповідне повідомлення.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // введення вихідних даних
            Console.Write("x=");
            double x = double.Parse(Console.ReadLine());
            // реалізація алгоритму розв'язання задачі
            if (x >= 0 && x <= 1)
                Console.WriteLine("належить");
            else
                Console.WriteLine("не належить");
            //
            Console.ReadKey();
        }
    }
}

```

Результат роботи програми:

x=0,3

належить

Гілки деякого розгалуження можуть містити інші розгалуження. У цьому випадку виникає *вкладеність умовних операторів*.

Приклад. Скласти програму для обчислення значення функції

$$f(x) = \begin{cases} 0, & \text{якщо } x \leq 0, \\ x^2 - x, & \text{якщо } 0 < x \leq 1, \\ x^2 - \sin \pi x^2, & \text{якщо } x > 1. \end{cases}$$

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // введення вихідних даних

```

```

        Console.WriteLine("x=");
        double x = double.Parse(Console.ReadLine());
        // реалізація алгоритму розв'язання задачі
        double f;
        if (x <= 0)
            f = 0;
        else
            if (x <= 1)
                f = x * x - x;
            else
                f = x * x - Math.Sin(Math.PI * x * x);
        // вивід результату
        Console.WriteLine("f={0}", f);
        //
        Console.ReadKey();
    }
}

```

Результат роботи програми:

x=0,5

f=-0,25

2. Складений оператор

Часто виникає потреба у розгалуженнях, гілки яких містять більше ніж один оператор. У цьому випадку застосовують *складений оператор* або *операторні блоки*. *Складений оператор* – це оператор, який об'єднує декілька операторів в одну логічну групу.

Загальне правило запису складеного оператора:

```

{
    <оператор 1>;
    <оператор 2>;
    .....
    <оператор n>;
}

```

В даній конструкції “{” – відкриваюча операторна дужка; “}” – закриваюча операторна дужка. Складений оператор визначається як єдиний оператор. Його можна вставляти в довільне місце програми, де дозволено використання одного простого оператора.

Якщо в умовному операторі при виконанні чи невиконанні умови необхідно виконати декілька операторів, то необхідно ці оператори помістити в складений оператор.

Повна форма

Програмна структура	Аналог на мові блок-схем	Приклад
<pre> if (<умова>) { <оператор1.1>; <оператор1.N>; } else { <оператор2.1>; <оператор2.M>; } </pre>		<pre> if(x>y) { max=x; min=y; } else { max=y; min=x; } </pre>

Скорочена форма

Програмна структура	Аналог на мові блок-схем	Приклад
<pre> if (<умова>) { <оператор1>; <операторN>; } </pre>		<pre> if(x>0) { z=1/x; l=y/x; } </pre>

Приклад. Знайти максимальне та мінімальне із двох дійсних чисел.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            double b,d;
            Console.Write("b=");
            b = double.Parse(Console.ReadLine());
        }
    }
}
        
```

```

        Console.Write("d=");
        d = double.Parse(Console.ReadLine());
        double max, min;
        if (b > d)
        {
            max = b;
            min = d;
        }
        else
        {
            max = d;
            min = b;
        }
        Console.WriteLine("max={0} min={1}", max, min);
        Console.ReadLine();
    }
}

```

Результати роботи програми:

```

b=3
d=8

max=8 min=3

```

3. Умовний оператор (?:)

Якщо значення виразу може дорівнювати одному із двох значень в залежності від виконання чи невиконання деякої умови, то можна скористатися умовним оператором (?:).

Загальний вигляд умовного оператора (?:)	Аналог з використанням умовного оператора if
<змінна>=<умова>?<значення1>:<значення2>;	if (<умова>) <змінна>=<значення1>; else <змінна>=<значення2>;

Приклад. Знайти максимальне із двох дійсних чисел.

З використанням умовного оператора (?:)	З використанням умовного оператора if
max = (x>y) ? x : y;	if (x>y) max=x; else max=y;

4. Оператор вибору switch

Оператор switch дозволяє передавати керування одному з декількох операторів в залежності від значення виразу, який називають *селектором*

вибору. У якості селектора вибору може бути вираз цілого типу, типу `char`, перелікового типу або типу `string`.

Загальне правило запису	Приклад Вводиться оцінка – цифра, вивести оцінку прописом (селектор вибору цілого типу).
<pre>switch (<селектор вибору>) { case <константа1> : <оператор1>; break; case <константа2> : <оператор2>; break; case <константаN> : <операторN>; break; default : <оператор N+1>; break; }</pre>	<pre>using System; using System.Collections.Generic; using System.Linq; using System.Text; namespace ConsoleApplication1 { class Program { static void Main(string[] args) { int mark; Console.Write("Mark = "); mark = int.Parse(Console.ReadLine()); switch (mark) { case 2: Console.WriteLine("Незадовільно"); break; case 3: Console.WriteLine("Задовільно."); break; case 4: Console.WriteLine("Добре"); break; case 5: Console.WriteLine("Відмінно"); break; default: Console.WriteLine("Неправильна оцінка."); break; } Console.ReadKey(); } } }</pre>

Якщо для декількох констант вибору необхідно виконати одні і ті ж дії, тоді ці константи вибору записують послідовно одна за одною, після чого вказується загальна дія.

Приклад. З клавіатури вводиться оцінка у національній шкалі, необхідно вивести повідомлення про те, чи зараховано студенту залік.

<pre>using System; using System.Collections.Generic; using System.Linq; using System.Text; namespace ConsoleApplication1 { class Program { static void Main(string[] args) { int mark;</pre>
--

```

        Console.WriteLine("Mark = ");
        mark = int.Parse(Console.ReadLine());
        switch (mark)
        {
            case 1:
            case 2: Console.WriteLine("Незараховано.");
                    break;
            case 3:
            case 4:
            case 5: Console.WriteLine("Зараховано.");
                    break;
            default: Console.WriteLine("Неправильна оцінка.");
                    break;
        }
        Console.ReadKey();
    }
}

```

Приклад. З клавіатури вводиться колір помідора, на екран необхідно вивести у якому він стані (росте, дозріває чи можна зірвати). При розв’язанні цього завдання використаємо оператор `switch`, у якому селектор і константи вибору є величинами типу `string`.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string color;
            Console.WriteLine("Color = ");
            color = Console.ReadLine();
            switch (color)
            {
                case "Green" : Console.WriteLine("Ще росте");
                            break;
                case "Yellow": Console.WriteLine("Дозріває.");
                            break;
                case "Red": Console.WriteLine("Можна зірвати");
                            break;
                default: Console.WriteLine("Помідор-мутант.");
                            break;
            }
            Console.ReadKey();
        }
    }
}

```

Приклад. З клавіатури вводиться буква у нижньому регістрі, з'ясувати, чи є буква голосною. При розв'язанні цього завдання використаємо оператор `switch`, у якому селектор вибору та константи вибору типу `char`.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            char c;
            Console.Write("Input letter= ");
            c = (char)Console.Read();
            switch (c)
            {
                case 'a' :
                case 'o' :
                case 'y' :
                case 'и' :
                case 'i' :
                case 'e' : Console.WriteLine("Голосна");
                           break;
                default: Console.WriteLine("Приголосна");
                           break;
            }
            Console.ReadKey();
        }
    }
}
```

Питання для самоконтролю

1. Що називають розгалуженням?
2. Які форми умовного оператора розрізняють у мові C#?
3. Який загальний вигляд умовного оператора у повній формі?
4. Який загальний вигляд умовного оператора у скороченій формі?
5. У яких випадках використовують складений оператор?
6. Який загальний вигляд умовного оператора (`?:`)?
7. Який загальний вигляд оператора вибору `switch`?
8. У яких випадках використовують оператор вибору `switch`?

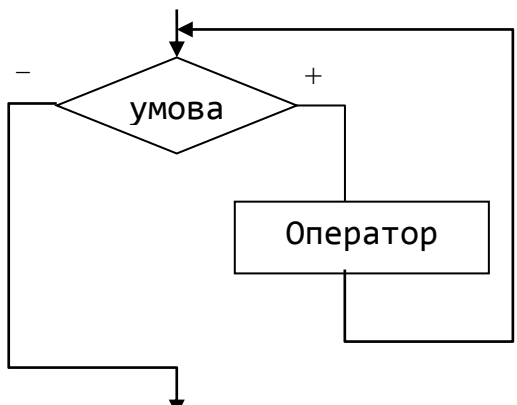
РЕАЛІЗАЦІЯ ЦИКЛІЧНИХ АЛГОРИТМІВ

1. Оператори циклу

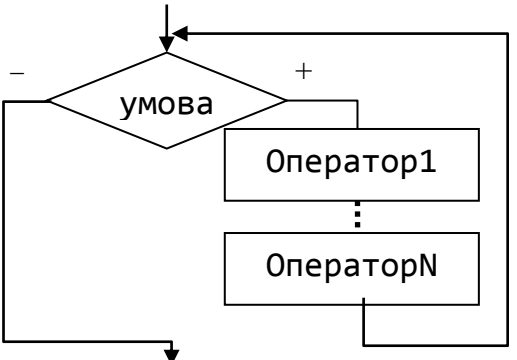
Часто постає потреба виконати один і той самий оператор декілька разів. Для цього застосовують *оператори циклів*. Цикл складається із *заголовка* і *тіла*. У заголовку циклу зазначається умова завершення циклу, а тіло циклу являє собою оператор, який потрібно виконати декілька разів. Кожне виконання оператора тіла циклу називається його *ітерацією*.

1.1. Оператор циклу while

Оператор `while` циклічно виконує своє тіло до тих пір, поки умова виконується (логічний вираз приймає значення `true`).

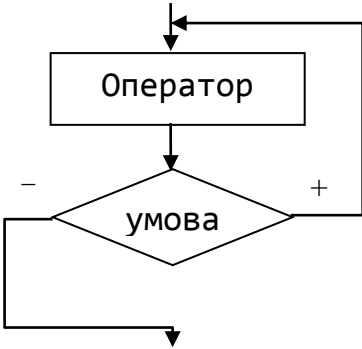
Програмна структура	Аналог на мові блок-схем	Приклад. Знайти суму перших n натуральних чисел.
<code>while (<умова>) <оператор>;</code>		<pre>int sum=0; int i=1; while(i<=n) sum=sum+i++;</pre>

Якщо тіло циклу складається з більше ніж одного оператора, то необхідно використати складений оператор.

Програмна структура	Аналог на мові блок-схем	Приклад. Знайти суму і добуток перших n натуральних чисел.
<pre>while (<умова>) { <оператор1>; <операторN>; }</pre>		<pre>int sum=0; int mult=1; int i=1; while(i<=n) { sum=sum+(i++); mult=mult*(i++); }</pre>

1.2. Оператор циклу do-while

Оператор циклу do-while відрізняється від оператора while тим, що перевірка умови виконується не до, а після виконання інструкції (оператора).
У операторі циклу do-while тіло циклу виконається принаймні один раз.

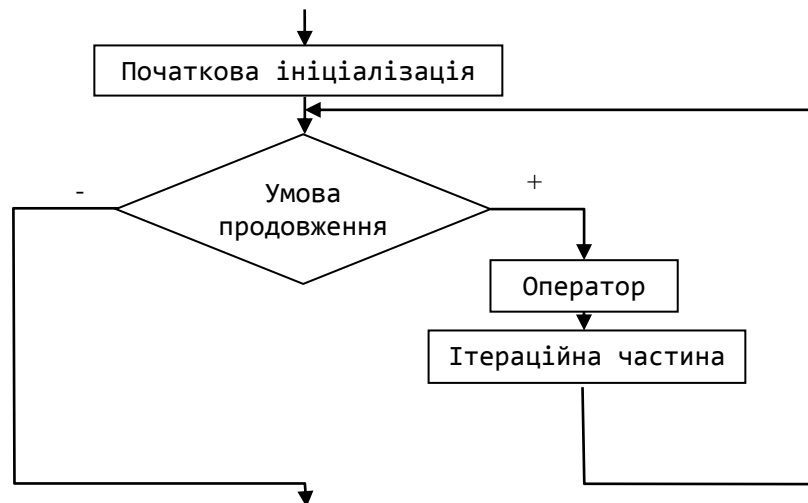
Програмна структура	Аналог на мові блок-схем	Приклад. Знайти суму перших n натуральних чисел.
<pre>do { <оператор>; } while (<умова>)</pre>		<pre>int sum=0; int i=1; do { sum=sum+i++; } while(i<=n)</pre>

1.3. Оператор циклу for

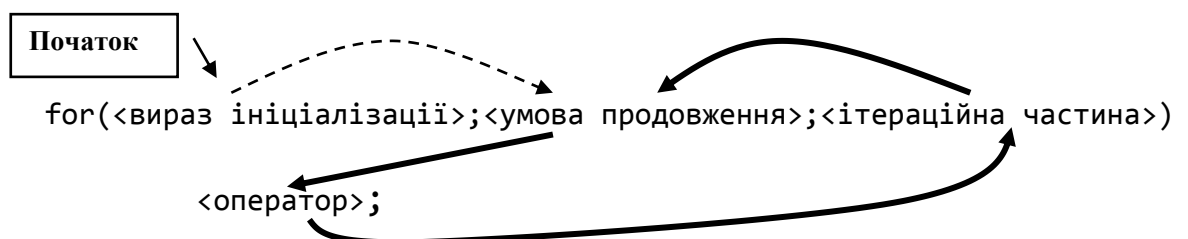
Загальний вигляд:

```
for (<вираз ініціалізації>;<умова продовження>;<ітераційна частина>)
    <оператор>;
```

Аналог на мові блок-схем:



Схематичне зображення виконання оператора:



Оператор `for` працює у відповідності до наступного алгоритму:

1. Обчислюється вираз ініціалізації. У цій частині допустима ініціалізація декількох лічильників циклу.
2. Перевіряється умова продовження. Якщо умова невірна то робота циклу завершується і передається управління наступному оператору.
3. Якщо умова істинна, виконується тіло даного оператора.
4. Виконується приріст одного або декількох лічильників циклу (або виконується довільна інша операція).
5. Здійснюється перехід до кроку 2.

Приклад. Знайти суму перших n натуральних чисел.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int n,sum=0;
            Console.WriteLine("n=");
            n=int.Parse(Console.ReadLine());
            for (int i = 1; i <= n; i++)
                sum = sum + i;
            Console.ReadKey();
        }
    }
}
```

Проілюструємо випадок, коли частина ініціалізації та ітераційна частина містять декілька виразів, розділених комою.

Приклад. Обчислити значення виразу

$$\frac{1}{0,3} + \frac{2}{0,4} + \dots + \frac{n}{0,3 + 0,1 * (n - 1)}.$$

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
```

```

class Program
{
    static void Main(string[] args)
    {
        int n;
        Console.WriteLine("n=");
        n=int.Parse(Console.ReadLine());
        double d;
        int c;
        double sum=0;

        for (c = 1, d=0.3; c <= n; c++, d+=0.1)
            sum +=c/d;

        Console.WriteLine("sum={0:f3}",sum);
        Console.ReadKey();
    }
}

```

Кожна з частин оператора циклу може бути відсутньою, але розділові знаки “;” є обов’язковими. Так нескінчений цикл може бути задано так:

```

for ( ; ; )
    <оператор>;

```

2. Оператори break та continue

Оператор continue може бути використаний у будь-якому із циклів у випадку, коли немає потреби виконувати до кінця усі оператори тіла циклу поточної ітерації, а необхідно одразу перейти до наступної ітерації.

Приклад. Знайти добуток непарних натуральних чисел, що менші за K.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int K;
            Console.Write("K=");
            K=int.Parse(Console.ReadLine());
            int mult=1;

```

```

        for (int i = 1; i <= K; i++)
        {
            if ((i % 2) == 0) continue;
            mult *= i;
        }

        Console.ReadKey();
    }
}

```

Оператор **break** у циклах використовують для негайного виходу з того циклу, в тілі якого він знаходиться.

Приклад. Знайти найменше значення факторіалу натурального числа, що перевищує число K.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int K;
            Console.Write("K=");
            K = int.Parse(Console.ReadLine());
            int fakt = 1;
            for (int i = 1; ; i++)
            {
                fakt *= i;
                if (fakt > K) break;
            }
            Console.WriteLine("fakt={0}", fakt);
            Console.ReadKey();
        }
    }
}

```

Приклад. Вивести на екран усі натуральні двоцифрові числа, у яких друга цифра не перевищує першу.

При розв'язанні цього завдання використаємо оператор **break**, який здійснює вихід тільки із вкладеного циклу (робота зовнішнього циклу продовжується).

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 1; i <= 9; i++)
            {
                for (int j = 1; j <= 9; j++)
                {
                    if (j > i) break;
                    Console.WriteLine("{0}{1}", i, j);
                }
            }
            Console.ReadKey();
        }
    }
}

```

Питання для самоконтролю

1. У яких випадках застосовують оператори циклу?
2. Що таке тіло циклу?
3. Що називають ітерацією циклу?
4. Який загальний вигляд оператора циклу *while*?
5. Який загальний вигляд оператора циклу *do-while*?
6. Які відмінності у роботі операторів циклу *while* та *do-while*?
7. Який загальний вигляд оператора *for*?
8. За яким алгоритмом виконується оператор *for*?
9. Яке призначення оператора *break*?
10. Яке призначення оператора *continue*?