Операторы языка C#

Блок или составной оператор. Пустой оператор

С помощью фигурных скобок несколько операторов языка, возможно перемежаемых объявлениями, можно объединить в единую синтаксическую конструкцию, называемую **блоком** или **составным оператором**:

```
{
    оператор_1
    …
    оператор_N
}
```

Пустой оператор – это пусто, завершаемое точкой с запятой. Иногда полезно рассматривать отсутствие операторов как существующий пустой оператор. Синтаксически допустимо ставить лишние точки с запятой, полагая, что вставляются пустые операторы. Например, синтаксически допустима следующая конструкция:

```
for (int j=1; j<5; j++)
    {;;;};
```

Эта конструкция может рассматриваться как задержка по времени, работа на холостом ходе.

Операторы выбора. Оператор if

if(выражение_1) оператор_1

else if(выражение_2) оператор_2

…

else if(выражение_K) оператор_K

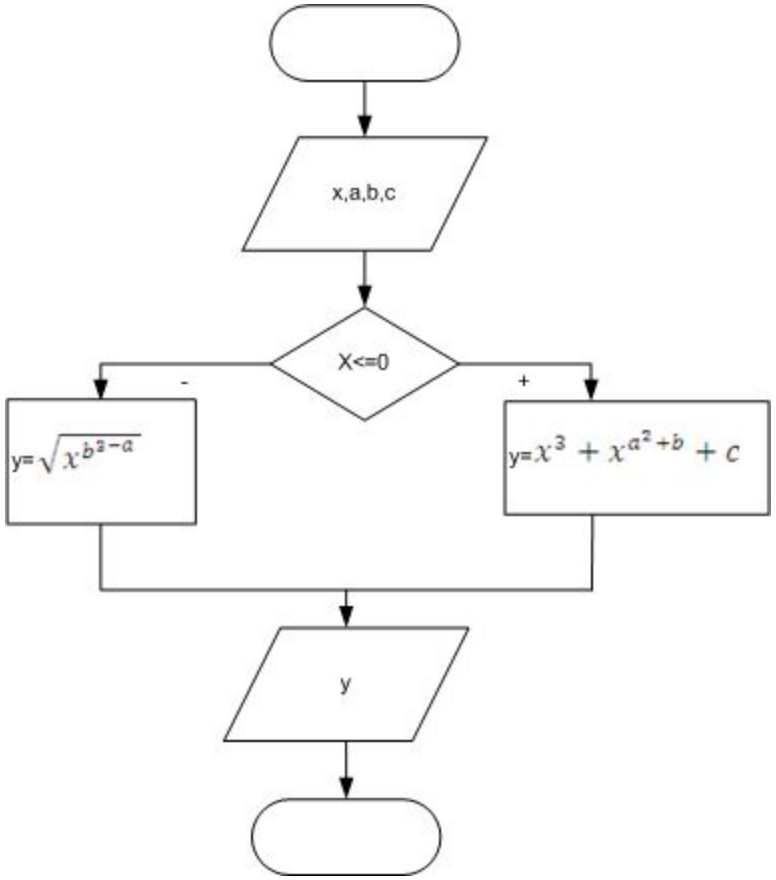
else оператор_N

Пример. скласти програму мовою C#для обчислення $y = \begin{cases} x^3 + x^{a^2+b} + c, x \le 0 \\ \sqrt{x^{b^3-a}}, \text{иначе} \end{cases}$

1. Постановка задачи

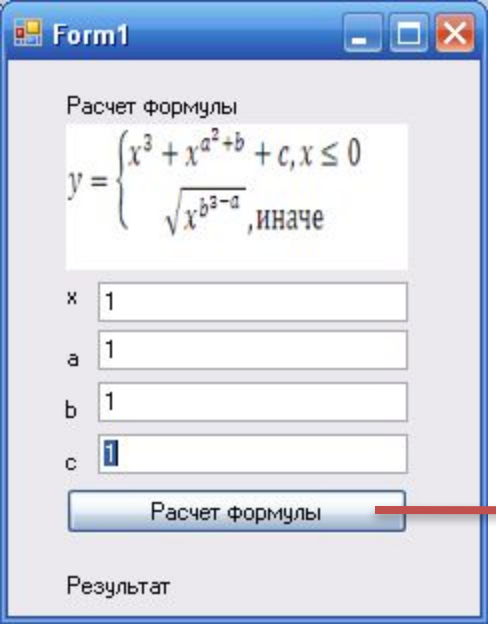
Вхід	Дії	Вихід
x,a,b,c- дійсні	1. введення x,a,b,c, 2. Якщо x<=0 то y= $x^3 + x^{a^2+b} + c$ Інакше y= $\sqrt{x^{b^3-a}}$	y - дійсне

2. Алгоритм



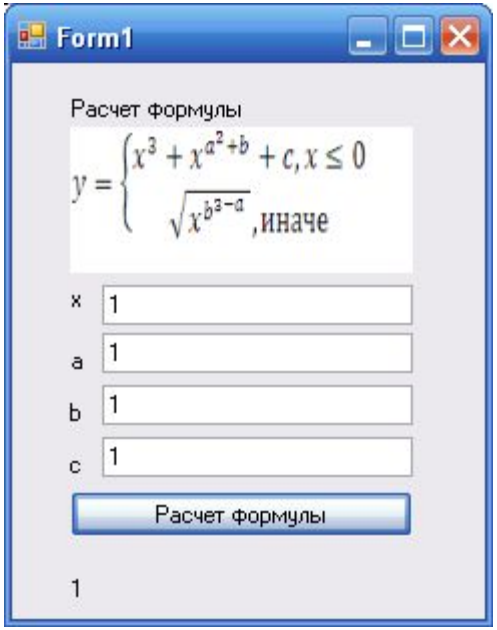
3. Тестовый пример

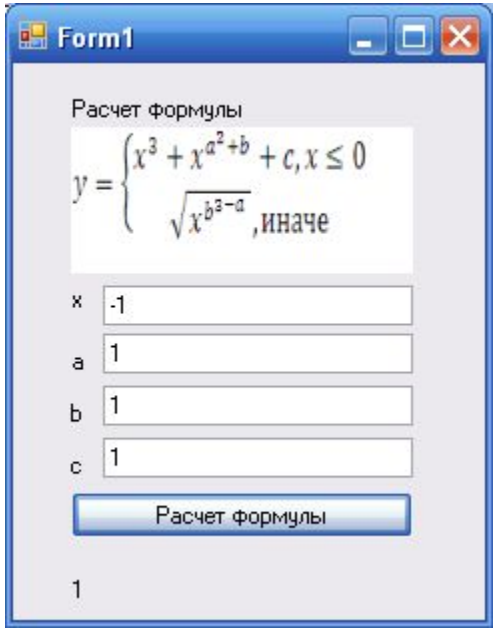
1. a=1 b=1 c=1 x=-1 y=1
2. a=1 b=1c=1 x=1 y=1




```
private void button1_Click(object sender, EventArgs e)
{
    double x, a, b, c,y;
    if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" && textBox4.Text != "")
    {
        x = Convert.ToDouble(textBox1.Text);
        a = Convert.ToDouble(textBox2.Text);
        b = Convert.ToDouble(textBox3.Text);
        c = Convert.ToDouble(textBox4.Text);
        if (x <= 0)
            y = x * x * x + Math.Pow(x, a * a + b) + c;
        else
            y = Math.Sqrt(Math.Pow(x, b * b * b - a));
        label6.Text = y.ToString();
    }
    else
        MessageBox.Show("Введите значения");

}
```





Оператор switch

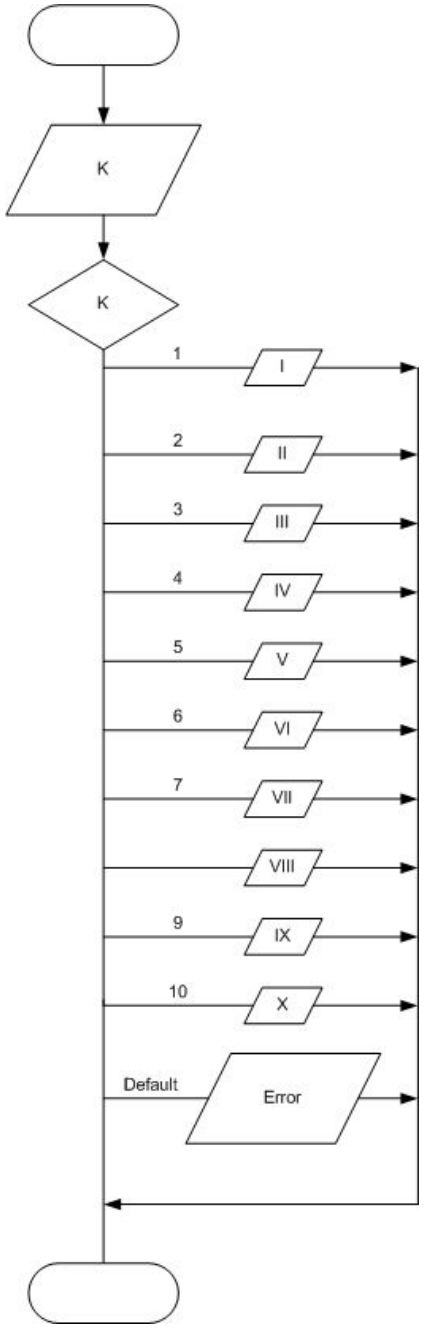
Частным, но важным случаем выбора из нескольких вариантов является ситуация, при которой выбор варианта определяется значениями некоторого выражения. Соответствующий оператор C#, наследованный от C++, но с небольшими изменениями в синтаксисе, называется оператором switch. Вот его синтаксис:

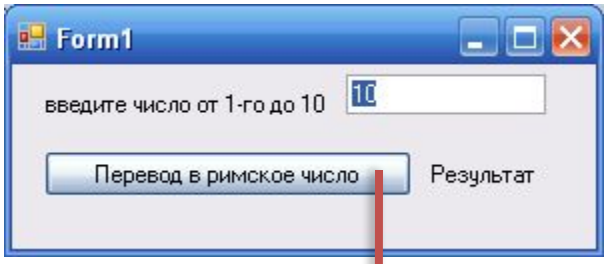
switch(выражение)
{
    case константное_выражение_1: [операторы_1 оператор_перехода_1]
    …
    case константное_выражение_K: [операторы_K оператор_перехода_K]
    [**default**: операторы_N оператор_перехода_N]
}

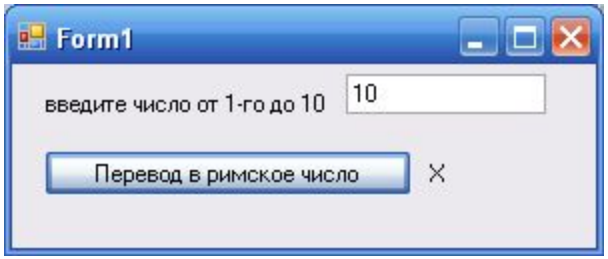
Ветвь default может отсутствовать. Заметьте, по синтаксису допустимо, чтобы после двоеточия следовала пустая последовательность операторов, а не последовательность, заканчивающаяся оператором перехода. Константные выражения в case должны иметь тот же тип, что и switch-выражение.

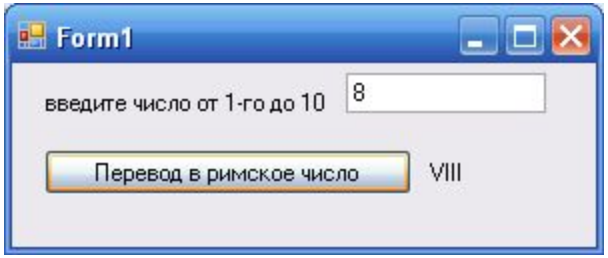
Пример. Составить программу, которая реализовала бы следующие действия: по введенному числу К (до 10) выдавала бы соответствующую ей римскую цифру.

Вхід	Дія	Вихід
К - ціле	Якщо К=	
	1	I
	2	II
	3	III
	4	IV
	5	V
	6	VI
	7	VII
	8	VIII
	9	IX
	10	X










```
private void button1_Click(object sender, EventArgs e)
{
    int k;
    string r;
    if (textBox1.Text != "")
    {
        k = Convert.ToInt16(textBox1.Text);
        switch (k)
        {
            case 1: { r = "I"; break; }
            case 2: { r = "II"; break; }
            case 3: { r = "III"; break; }
            case 4: { r = "IV"; break; }
            case 5: { r = "V"; break; }
            case 6: { r = "VI"; break; }
            case 7: { r= "VII"; break; }
            case 8: { r = "VIII"; break; }
            case 9: { r = "IX"; break; }
            case 10: { r = "X"; break; }
            default: { r = "Ошибка"; break; }
        }
        label2.Text = r;
    }
    else
        MessageBox.Show("введите данные");
}
```

Операторы перехода

Оператор goto

goto [метка|**case** константное_выражение|**default**];

Все операторы языка C# могут иметь метку – уникальный идентификатор, предшествующий оператору и отделенный от него символом двоеточия. Передача управления помеченному оператору – это классическое использование оператора goto. Два других способа использования goto – это передача управления в case или default-ветвь – используются в операторе switch.

Операторы break и continue

В структурном программировании признаются полезными «переходы вперед» (но не назад), позволяющие при выполнении некоторого условия выйти из цикла, оператора выбора, из блока. Для этой цели можно использовать оператор goto, но лучше использовать специально предназначенные для этих целей операторы break и continue.

Оператор break может стоять в теле цикла или завершать case-ветвь в операторе switch. Пример его использования в операторе switch уже демонстрировался. При выполнении оператора break в теле цикла завершается выполнение самого внутреннего цикла.

Оператор return

Еще одним оператором, относящимся к группе операторов перехода, является оператор return, позволяющий завершить выполнение процедуры или функции. Его синтаксис:

return [выражение];

Для функций его присутствие и аргумент обязательны, поскольку выражение в операторе return задает значение, возвращаемое функцией.

Операторы цикла

Оператор for

Наследованный от C++ весьма удобный оператор цикла for обобщает известную конструкцию цикла типа арифметической прогрессии. Его синтаксис:

for(инициализаторы; условие; список_выражений) оператор

Инициализаторы задают начальное значение одной или нескольких переменных, часто называемых счетчиками или просто переменными цикла.

Условие задает условие окончания цикла, соответствующее выражение при вычислении должно получать значение true или false.

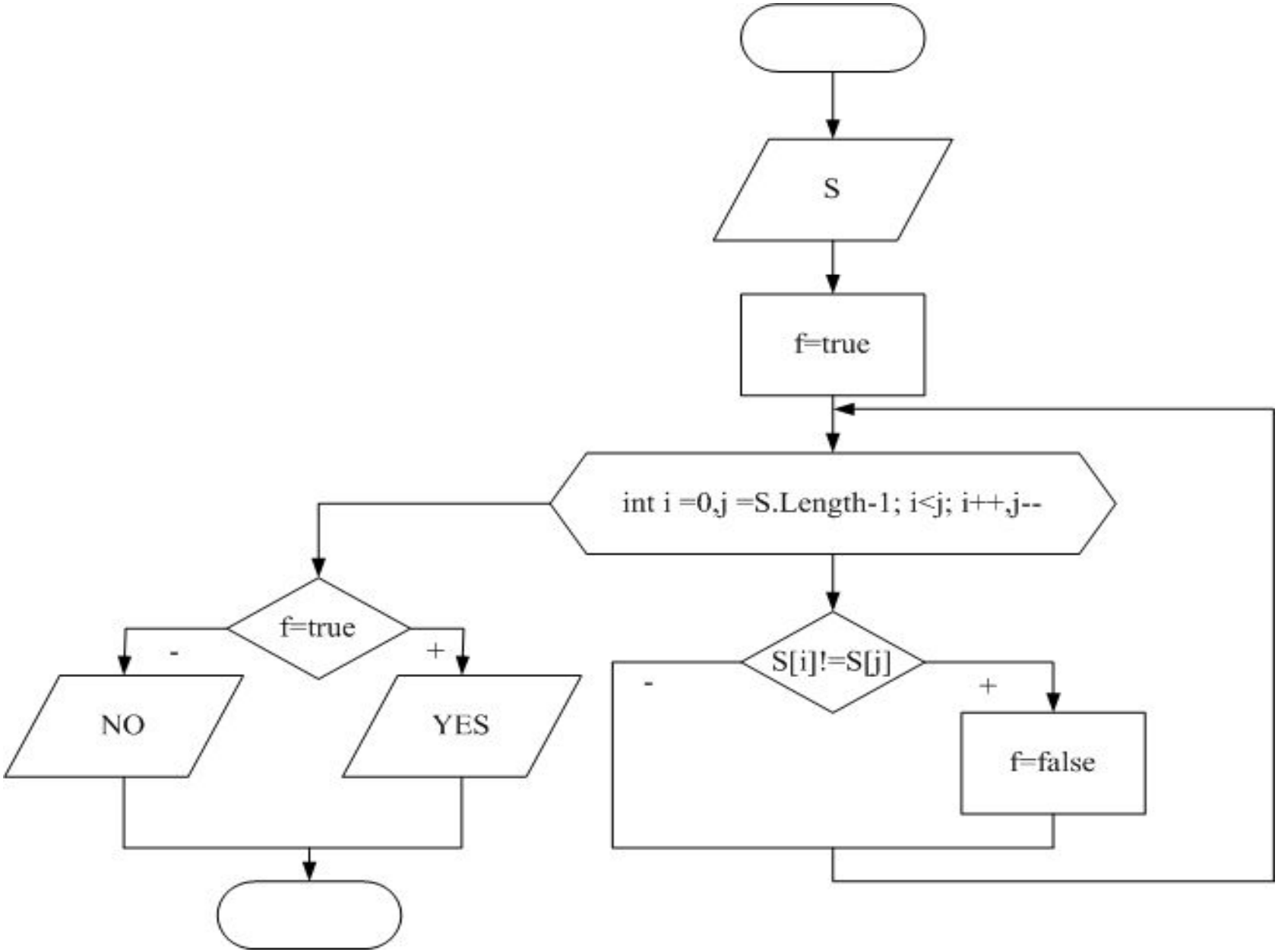
Список выражений, записанный через запятую, показывает, как меняются счетчики цикла на каждом шаге выполнения

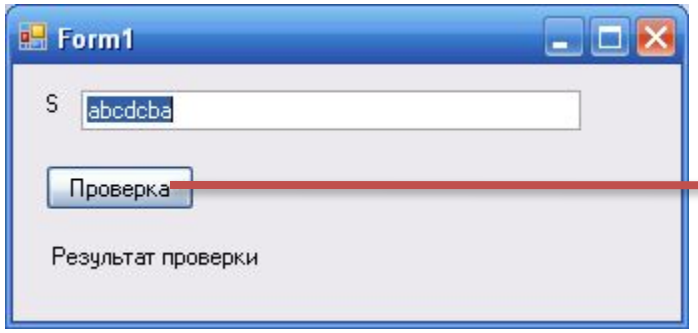
Пример. Составить программу, которая проверяет является ли введенная строка палиндромом

Вхід	Дії	Вихід
S - рядок	1. Введення S 2. i=0 3. f=true; 4. j=S.length-1 5. i<j 5.1 Якщо S[i]!=S[j] тоді f=false 5.2 i++ 5.3j— 6. Якщо f=true Інакше	 YES No

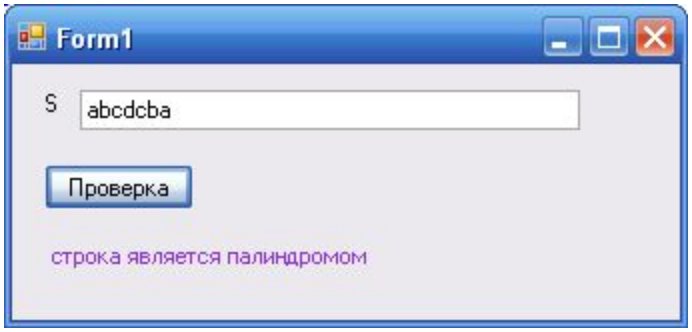
3. Тестовый пример

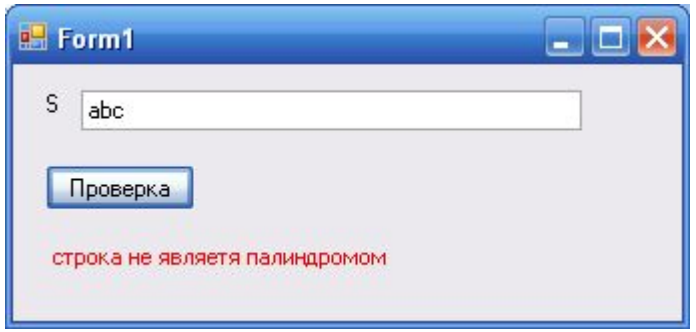
1. abcdcba - YES
2. abc - NO


S
f=true
int i =0,j =S.Length-1; i<j; i++,j--
S[i]!=S[j]
-
+
f=false
f=true
-
+
NO
YES




```
private void button1_Click(object sender, EventArgs e)
{
    string S;
    bool f=true;
    S = textBox1.Text;
    for (int i = 0, j = S.Length - 1; i < j; i++, j--)
    {
        if (S[i] != S[j]) f = false;
    }
    if (f == true)
    {
        label2.ForeColor = Color.BlueViolet;
        label2.Text = "строка является палиндромом";
     }
    else
    {
        label2.ForeColor = Color.Red;
        label2.Text = "строка не являетя палиндромом";
    }
}
```





Циклы While

Цикл while(выражение)

является универсальным видом цикла, включаемым во все языки программирования. Тело цикла выполняется до тех пор, пока остается истинным выражение while. В языке C# у этого вида цикла две модификации – с проверкой условия в начале цикла и в конце цикла. Первая модификация имеет следующий синтаксис:

while(выражение)

оператор

Эта модификация цикла соответствует стратегии: «вначале проверь, а потом делай». В результате проверки может оказаться, что и делать ничего не нужно. Тело такого цикла может ни разу не выполняться. Конечно же, возможно и зацикливание. В нормальной ситуации каждое выполнение тела цикла – это очередной шаг к завершению цикла.

Цикл, проверяющий условие завершения в конце, соответствует стратегии: «вначале делай, а потом проверь». Тело такого цикла выполняется по меньшей мере один раз.

do

оператор

while(выражение);

Создать приложение для табулирования и вывода на экран значения функции, также построить график функции:

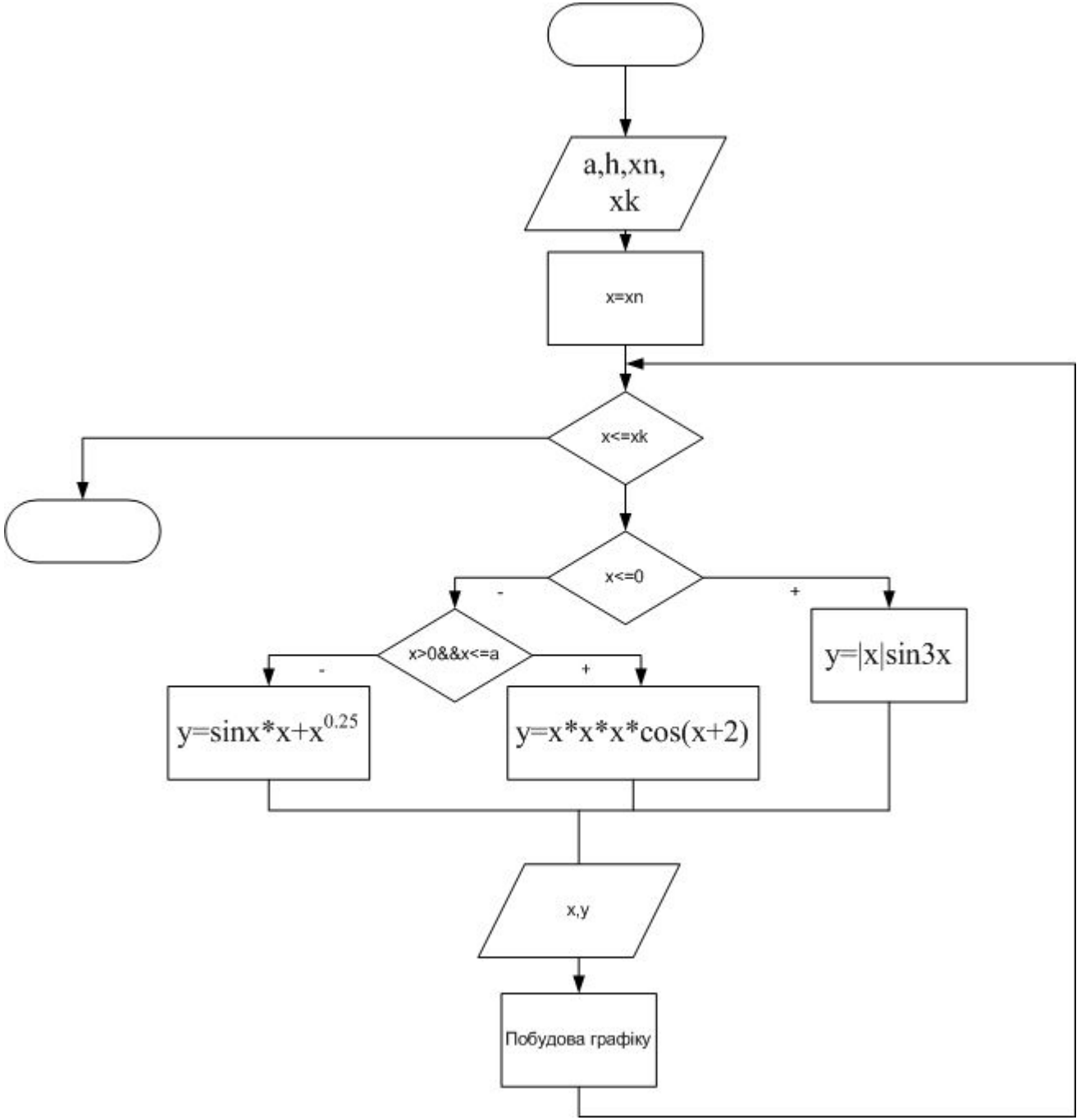
$$y = f(x) = \begin{cases} f_1(x), \text{если} & x \le 0 \\ f_2(x), \text{если} & 0 < x \le a \\ f_3(x), \text{если} & x > a \end{cases}$$

Выражения для функции $f_1(x)$,f2(x) и f3(x) выбрать из таблицы. В форме предусмотреть поля для ввода значения параметра a и переменной x, вывода результата вычисления y, а также командные кнопки для осуществления расчета.

30	$\lvert x\rvert \sin(3x)$	$x^3 \cos(x+2)$	$\sin x^2 + x^{0.25}$

1. Постановка задачи

Вхід	Дія	Вихід
a,h,xn,xk - дійсні	1. введеня початкових значень 2. x=xn 3. Доки x<=xk 4.1 Якщо x<=0 y=|x|sin3x Інакше 4.2 Якщо x>0 та x<=a y=x*x*x*cos(x+2) Інакше y=sinx*x+ $x^{0.25}$ 4.3 Виведення значень 4.4 Побудова графіку 4.5 x=x+h	y-дійсне


a,h,xn,
xk
x=xn
x<=xk
x<=0
-
+
x>0&&x<=a
-
+
y=|x|sin3x
y=sinx*x+x^0.25
y=x*x*x*cos(x+2)
x,y
Побудова графіку

Form1

Табулирование и построение графика функции на заданном интервале

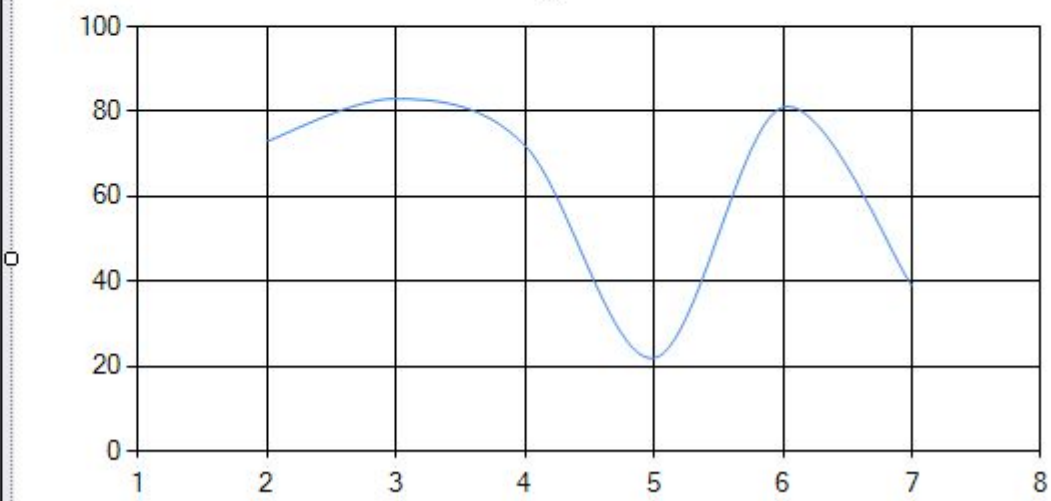
$$y = f(x) = \begin{cases} |x|\sin(3x), x \le 0 \\ x^3\cos(x+2), 0 < x \le a \\ \sin x^2 + x^{0.25}, x > a \end{cases}$$

Ввод данных

xn	-5
xk	5
h	0,5
a	3

Расчет и построение графика

x	y



```
private void button1_Click(object sender, EventArgs e)
{
    double x, xn, xk, h, a, y;
    try
    {
        xn = Convert.ToDouble(textBox1.Text);
        xk = Convert.ToDouble(textBox2.Text);
        h = Convert.ToDouble(textBox3.Text);
        a = Convert.ToDouble(textBox4.Text);
        x = xn;
        dg.Rows.Clear();
        chart1.Series[0].Points.Clear();
        while (x <= xk)
        {
            if (x <= 0)
                y = Math.Abs(x) * Math.Sin(3 * x);
            else
                if (x > 0 && x <= a)
                    y = x * x * x * Math.Cos(x + 2);
                else
                    y = Math.Sin(x * x) + Math.Pow(x, 0.25);
            dg.Rows.Add(x.ToString(), y.ToString());
            chart1.Series[0].Points.AddXY(x, y);
            x = x + h;

        }
    }
    catch
    {
        MessageBox.Show("Ошибка ввода");
    }
}
```

Form1

Табулирование и построение графика функции на заданном интервале

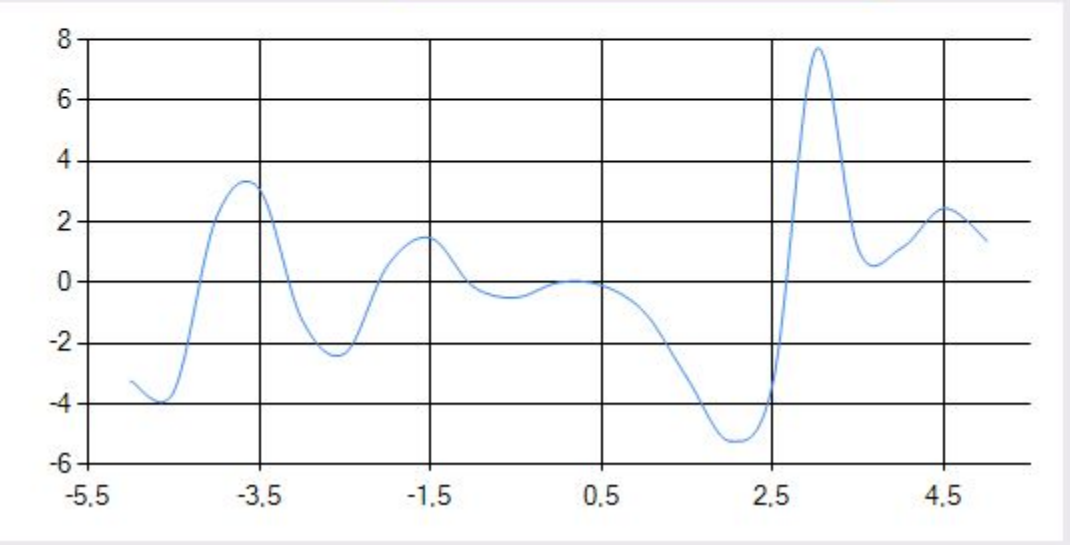
$$y = f(x) = \begin{cases} |x|\sin(3x), x \le 0 \\ x^3 \cos(x+2), 0 < x \le a \\ \sin x^2 + x^{0.25}, x > a \end{cases}$$

Ввод данных

xn	-5
xk	5
h	0,5
a	3

Расчет и построение графика

	x	y
▶	-5	-3,2514392007...
	-4,5	-3,6170299194...
	-4	2,1462916720...
	-3,5	3,0789351599...
	-3	-1,2363554557...
	-2,5	-2,3449999419...
	-2	0,5588309963...
	-1,5	1,4662951764...
	-1	-0,1411200080...
	-0,5	-0,4987474933...
	0	0



Цикл foreach

Новым видом цикла, не унаследованным от C++, является цикл foreach, удобный при работе с массивами, коллекциями и другими подобными контейнерами данных. Его синтаксис:

foreach(тип идентификатор **in** контейнер) оператор

Цикл работает в полном соответствии со своим названием – тело цикла выполняется для каждого элемента в контейнере. Тип идентификатора должен быть согласован с типом элементов, хранящихся в контейнере данных. Предполагается также, что элементы контейнера (массива, коллекции) упорядочены. На каждом шаге цикла идентификатор, задающий текущий элемент контейнера, получает значение очередного элемента в соответствии с порядком, установленном на элементах контейнера. С этим текущим элементом и выполняется тело цикла. Тело цикла выполняется столько раз, сколько элементов находится в контейнере. Цикл заканчивается, когда полностью перебраны все элементы контейнера.