Университет машиностроения

Кафедра «Автоматика и процессы управления»

Дисциплина

Информационные технологии

2 семестр

Тема 15

Структура языка C#

Управляющие конструкции языка высокого уровня

Реализуют логику выполнения программы:

- следование
- ветвление
- цикл
- передача управления

Блок (составной оператор)

Блок — последовательность операторов, заключенная в операторные скобки:

begin end – в Паскале

{ } – в С-подобных языках

Блок воспринимается компилятором как один оператор и может использоваться **всюду, где синтаксис требует одного оператора, а алгоритм — нескольких**.

Блок может содержать один оператор или быть пустым.

Оператор (инструкция) «выражение»

Любое выражение, завершающееся точкой с запятой, рассматривается как оператор, выполнение которого заключается в вычислении выражения.

```
i++;              // выполняется операция инкремента
a *= b + c;       // выполняется умножение с присваиванием
fun( i, k );      // выполняется вызов функции
```

Пустой оператор

пустой оператор **;**

используется, когда по синтаксису оператор требуется, а по смыслу — нет:

```
while ( true );
```

Это цикл, состоящий из пустого оператора (бесконечный)

```
;;;
```

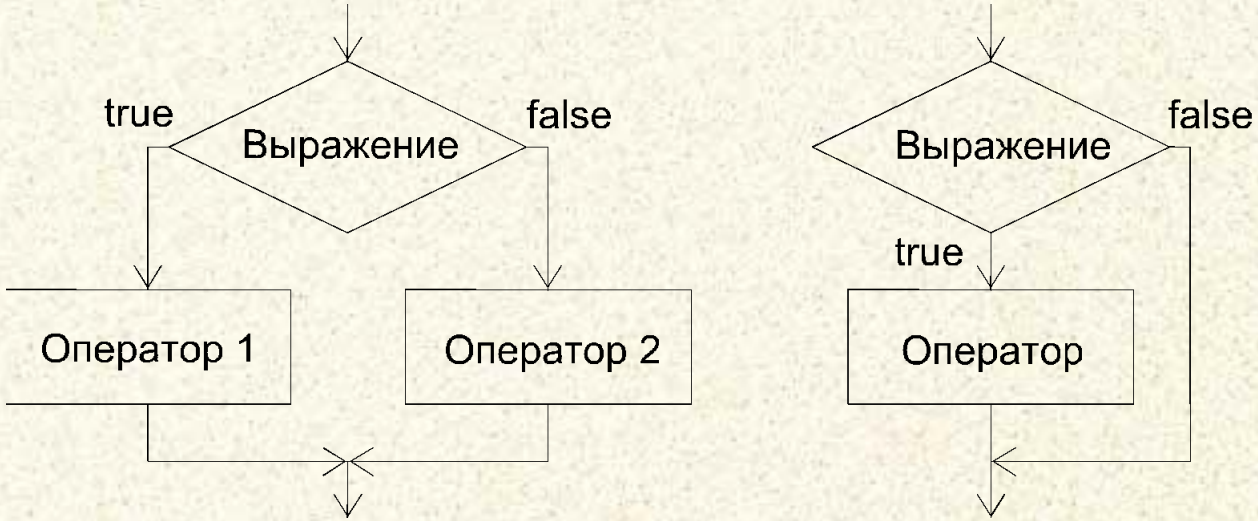
Три пустых оператора

Операторы ветвления:

- развилка (if)
- переключатель (switch)

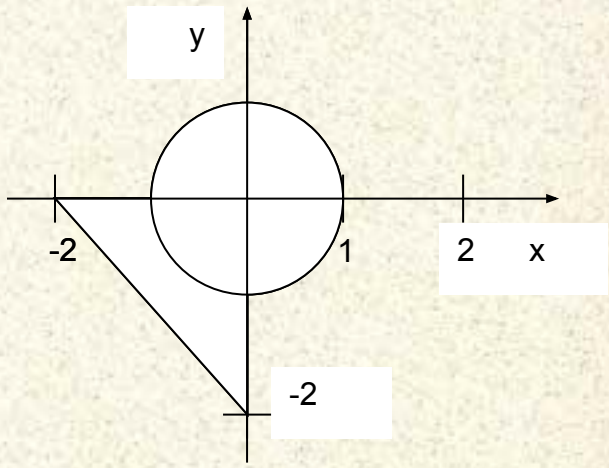
Условный оператор if

if (выражение) оператор_1;
[else оператор_2;]




```
if ( a < 0 ) b = 1;
if ( a < b && (a > d || a == 0))   ++b;
else {    b *= a; a = 0; }
if ( a < b ) if ( a < c ) m = a;
      else          m = c;
else      if ( b < c ) m = b;
      else          m = c;
```

Пример

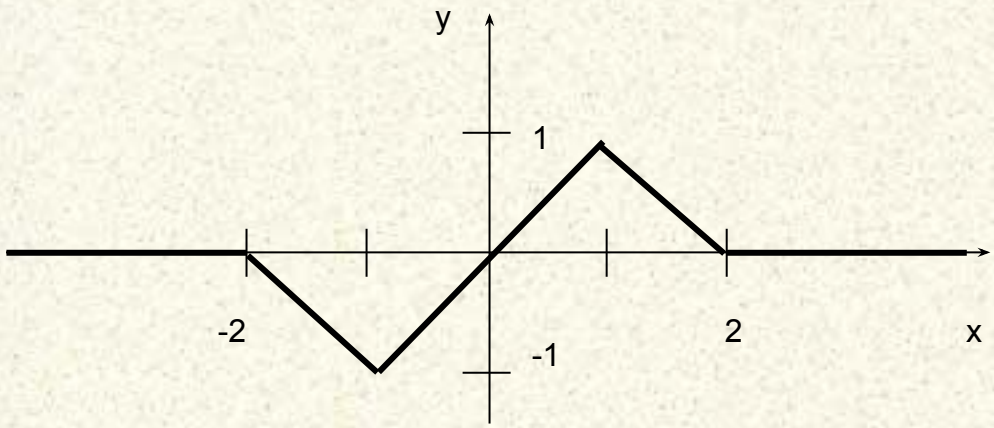



```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            Console.WriteLine( "Введите координату x" );
            double x = Convert.ToDouble(Console.ReadLine() );

            Console.WriteLine( "Введите координату y" );
            double y = double.Parse(Console.ReadLine() );

                        if  ( x * x + y * y <= 1 ||
                                x <= 0 && y <= 0 && y >= - x – 2 )
                            Console.WriteLine( " Точка попадает в область " );
                        else
                            Console.WriteLine( " Точка не попадает в область " );
}}}
```

Пример 2



$$y = \begin{cases} 0, & x < -2 \\ -x - 2, & -2 \le x < -1 \\ x, & -1 \le x < 1 \\ -x + 2, & 1 \le x < 2 \\ 0, & x \ge 2 \end{cases}$$

```
if ( x < -2 )                y = 0;
if ( x >= -2 && x < -1 ) y = -x - 2;
if ( x >= -1 && x <  1 ) y = x;
if ( x >=  1 && x <  2 ) y = -x + 2;
if ( x >=  2 )               y = 0;
```

```
if       ( x <= -2 )  y = 0;
else if ( x < -1 )     y = -x - 2;
  else if ( x <  1 )     y = x;
    else if ( x <  2 )     y = -x + 2;
      else                   y = 0;
```

```
y = 0;
if ( x > -2 ) y = -x - 2;
if ( x > -1 ) y = x;
if ( x >  1 ) y = -x + 2;
if ( x >  2 ) y = 0;
```

Проверка вещественных величин на равенство

Из-за погрешности представления вещественных значений в памяти следует ее избегать, вместо этого лучше сравнивать модуль разности с некоторым малым числом.

```
float a, b; …
if  ( a == b ) …                    // не рекомендуется!
if  ( Math.Abs(a - b) < 1e-6 ) …    // надежно!
```

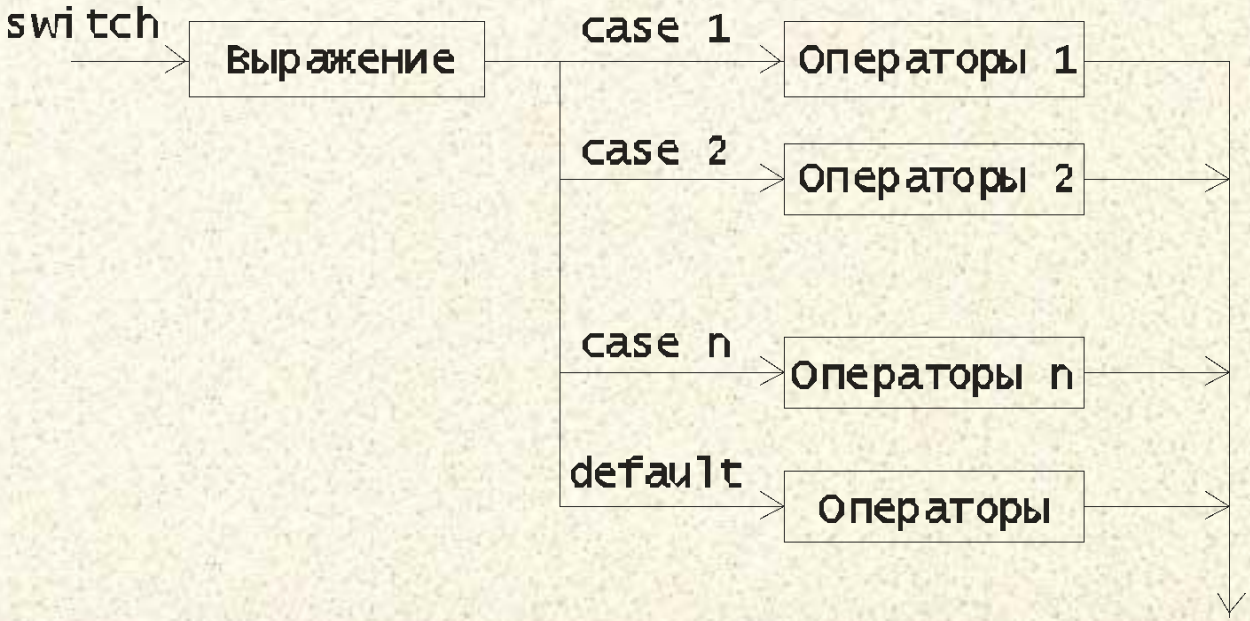
Значение величины, с которой сравнивается модуль разности, следует выбирать в зависимости от решаемой задачи и точности участвующих в выражении переменных.

Снизу эта величина ограничена определенной в классах Single и Double константой Epsilon. Это минимально возможное значение переменной такое, что

```
1.0 + Epsilon != 1.0
```

Оператор выбора switch

```
switch ( выражение ) {
    case константное_выражение_1: [ список_операторов_1 ] ; break;
    case константное_выражение_2: [ список_операторов_2 ]
    …
    case константное_выражение_n: [ список_операторов_n ] : break;
    [ default: операторы ]
                                                    }
```

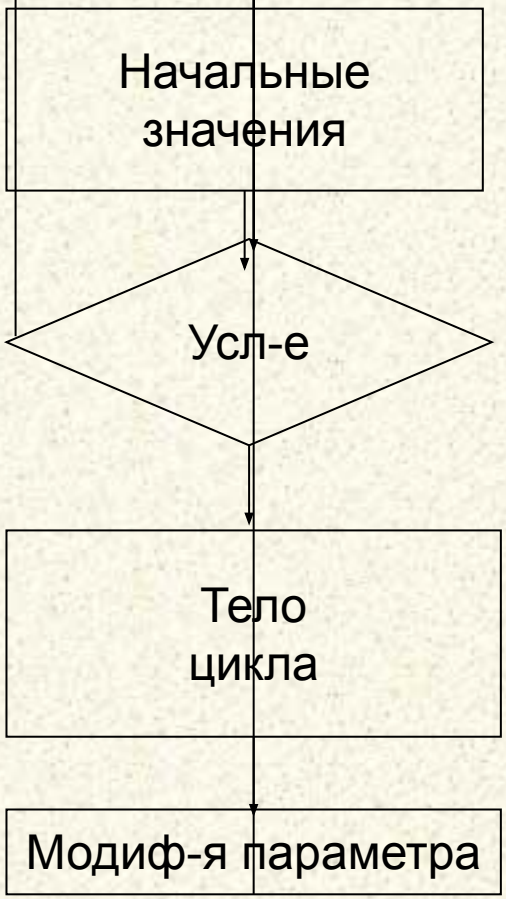


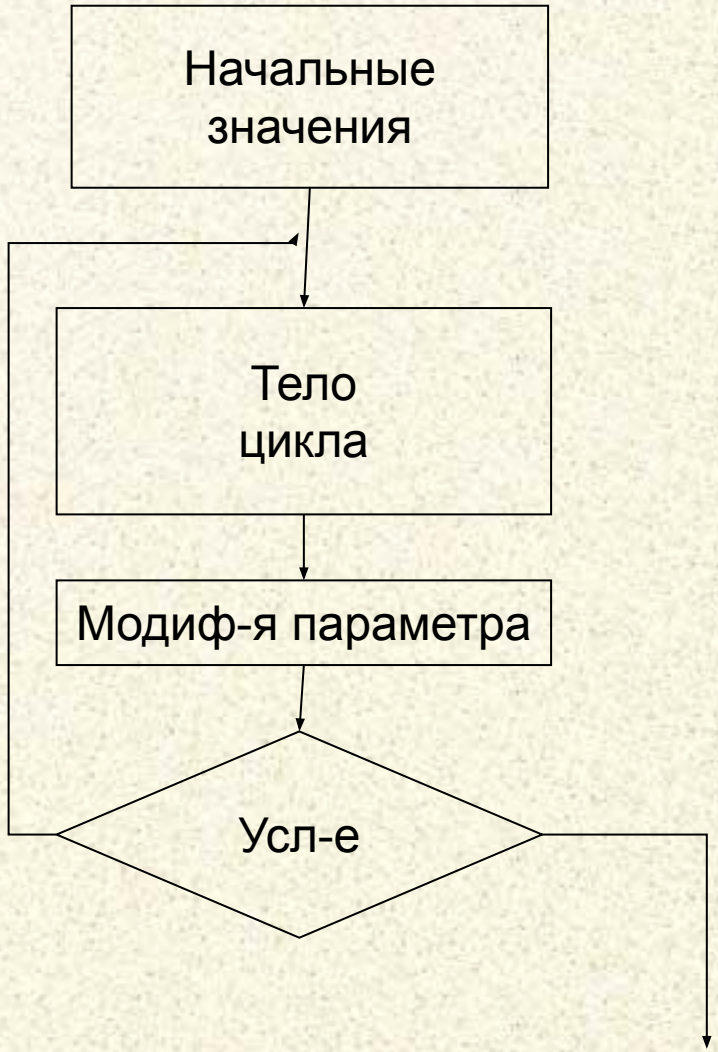
Пример: Калькулятор на четыре действия

```
using System; namespace ConsoleApplication1
{   class Class1    {   static void Main()       {
        Console.WriteLine( "Введите 1й операнд:" );
          double a = double.Parse(Console.ReadLine());
          Console.WriteLine( "Введите знак" );
          char op = (char)Console.Read(); Console.ReadLine();
          Console.WriteLine( "Введите 2й операнд:" );
          double b = double.Parse(Console.ReadLine());
        double res = 0;
        bool ok = true;
        switch (op)
           {  case '+' : res = a + b; break;
              case '-'  : res = a - b;  break;
              case '*' : res = a * b; break;
              case '/' : res = a / b; break;
              default  : ok = false; break;
           }
       if (ok) Console.WriteLine( "Результат: " + res );
        else    Console.WriteLine( "Недопустимая операция" );
}}}
```

Операторы цикла

Структура оператора цикла

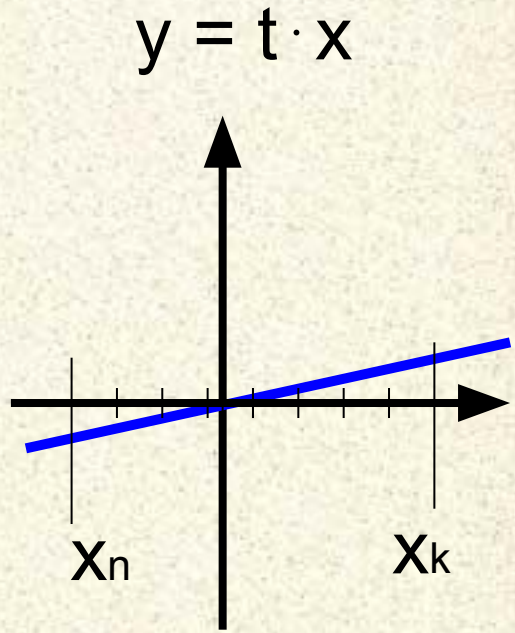




Цикл с предусловием

while (выражение) оператор

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            double Xn = -2, Xk = 12, dX = 2, t = 2, y;
            Console.WriteLine( "|     x     |     y     |" );
            double x = Xn;
            while ( x <= Xk )
            {
                y = t * x;
                Console.WriteLine( "| {0,9} | {1,9} |", x, y );
                x += dX;
            }
        }
    }
}
```



Цикл с постусловием

do оператор
while
выражение;

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            char answer;
            do
            {
                Console.WriteLine( "Купи слоника, а?" );
                answer = (char) Console.Read();
                Console.ReadLine();
            } while ( answer != 'y' );
        }
    }
}
```

Цикл с параметром

for (инициализация; выражение; модификации)
оператор;

```
int s = 0;
for ( int i = 1; i <= 100; i++ ) s += i;
```

Пример цикла с параметром

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            double Xn = -2, Xk = 12, dX = 2, t = 2, y;
            Console.WriteLine( "|     x     |     y     |";
            for ( double x = Xn; x <= Xk; x += dX )
            {
                y = t * x;
                Console.WriteLine( "| {0,9} | {1,9} |", x, y );
            }
        }
    }
}
```

Рекомендации по написанию циклов

- не забывать о том, что если в теле циклов while и for требуется выполнить более одного оператора, нужно заключать их в блок;
- убедиться, что всем переменным, встречающимся в правой части операторов присваивания в теле цикла, до этого присвоены значения, а также возможно ли выполнение других операторов;
- проверить, изменяется ли в теле цикла хотя бы одна переменная, входящая в условие продолжения цикла;
- предусматривать аварийный выход из итеративного цикла по достижению некоторого предельно допустимого количества итераций.

Далее…

- Основы ООП
- Событийное программирование
- Работа с памятью
- Визуальное проектирование и элементы управления
- Работа с файлами
- Работа с реляционными СУБД