**УО «МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ А.А. КУЛЕШОВА»**
**СОЦИАЛЬНО-ГУМАНИТАРНЫЙ КОЛЛЕДЖ**

**Дисциплина**
**«Конструирование программ и языки программирования»**

# Разработка программ с использованием операторов ветвления

(2 часа)

Методические рекомендации к лабораторной работе № 3

Могилев 2018

Понятие «Операторы ветвления». Методические указания по лабораторной работе № 3 по дисциплине «Конструирование программ и языки программирования». Для учащихся 3 курса очной формы обучения специальности 2–40 01 01 «Программное обеспечение информационных технологий.

**Оглавление**

## 1 Цель работы

выработать умение разрабатывать программы с использованием операторов ветвления.

## 2 Ход работы

1. Изучение теоретического материала.
2. Выполнение практических индивидуальных заданий по вариантам (вариант уточняйте у преподавателя).
3. Оформление отчета.
   3.1. Отчет оформляется индивидуально каждым студентом. Отчет должен содержать задание, алгоритм и листинг программы.
   3.2. Отчет по лабораторной работе выполняется на листах формата А4. В состав отчета входят:
      1) титульный лист;
      2) цель работы;
      3) текст индивидуального задания;
      4) выполнение индивидуального задания.
4. Контрольные вопросы.

## 3 Краткие теоретические сведения

Как известно, из предыдущих лабораторных работ, все программы состоят из последовательности операторов, которые обычно выполняются поочерёдно в том порядке, в каком они записаны в программе. Однако часто возникает необходимость изменить очерёдность выполнения операторов, т.е. пропустить или наоборот выполнить какую-то группу операторов в зависимости от выполнения или невыполнения некоторых заданных условий. Кроме того иногда необходимо повторить группу операторов несколько раз, то есть организовать цикл. Для выполнения этих задач служат управляющие операторы. Управляющие операторы подразделяются на операторы принятия решения, к ним относятся операторы условного и безусловного переходов, и операторы для организации циклов, которые будут рассмотрены в следующей лабораторной работе.

Одной из важнейших возможностей компьютерного процессора является возможность принятия решения. Под этим выражением подразумевается, что процессор может направить поток выполнения запрограммированных команд по тому или иному пути в зависимости от того истинно или ложно некоторое заданное условие. Любой язык программирования обеспечивает возможность принятия решения. В алгоритмическом языке C#, как и во многих других, основой для такой возможности является оператор условного перехода if, который действует в C# практически также, как и оператор IF в любом другом языке программирования./

### 3.1 Оператор условного перехода if и его конструкции

Оператор условного перехода if (если), как было уже сказано, предназначен для выбора одного из возможных вариантов исполнения операторов программы, поэтому его и называют оператором принятия решения. Существует несколько разновидностей конструкций этого оператора. Рассмотрим их последовательно по мере усложнения.

```
if (условие выбора)
      {
      // Записанные в скобках операторы (оператор)
      // будут выполняться, если условие выбора истинно
      }
      // Записанные далее операторы будут выполняться
      // в любом случае, независимо от условия выбора.
```

В качестве условия выбора используется значение логического выражения. При выполнении этой конструкции вначале вычисляется значение логического выражения, записанного в скобках. Результат вычисления имеет тип `boolean`. Если вычисленное выражение имеет значение `true (истина)`, то выполняются операторы в фигурных скобках и все следующие за ними. Если получено значение `false (ложь)` то операторы в фигурных скобках пропускаются и выполняются только операторы следующие за ними.

**Пример 1.**

```
// Сохраняем наибольшее значение из двух, a и b,
// в переменной max max = b;
if (a > b)
```

```
{
max = a;
}
```

В данном фрагменте программы изначально предполагается, что наибольшее значение имеет переменная b, и она присваивается переменной `max`. Если это ни так, то переменной `max` присваивается значение переменной a.

Операторы, записываемые в фигурных скобках можно размещать в одной строке, как в следующем примере.

**Пример 2.**
```
max = b;
if (a > b){max = a;}
```

При записи в фигурных скобках нескольких операторов в конце каждого из них ставиться точка с запятой. Если в фигурных скобках записывается один оператор, то фигурные скобки можно опустить (см. пример 3).

**Пример 3.**
```
max = b;
if (a > b) max = a;
```

### 3.1.1Конструкция if-else (если – иначе)

Данную конструкцию целесообразно использовать, когда необходимо задать выполнение одного из двух блоков операторов (или одного из двух операторов), в зависимости от результата проверки условия выбора. Конструкция имеет следующий вид записи.

```
If (условия выбора)
{// Если условие выбора истинно, то будут выполняться
// оператор или операторы блока 1.}
else
{// В противном случае (иначе), если условие выбора ложно, то будут
выполняться оператор или операторы блока 2.}
// Записанные далее операторы будут выполняться
// в любом случае, независимо от условия выбора.
```

Если результатом проверки условия выбора является значение true (истина), то будут выполнены операторы блока 1. Далее будет выполняться первый оператор, следующий за последней фигурной скобкой. Операторы блока 2 выполняться не будут. Если проверка условия выбора даст результат false (ложь), то операторы блока 1 будут пропущены, и будут выполнены операторы блока 2. Далее будет выполняться первый оператор, следующий за последней фигурной скобкой. Каждый из указанных блоков может состоять из одного оператора, тогда фигурные скобки могут быть опущены.

**Пример 4.**
```
If (a > b)
{
max = a; // Эти операторы будут выполняться,
```

```
min = b; // если условие выбора a>b истинно.
}
else
{
max = b; // Эти операторы будут выполняться,
min = a; // если условие выбора a > b ложно.
}
// Записанные далее операторы будут выполняться
// в любом случае, независимо от условия выбора.
```

Если `a > b`, то переменной `max` будет присвоено значении a, переменной `min` – значение b, в противном случае наоборот переменной `max` присваивается значение b, а переменной `min` – значение a. Рассмотренную конструкцию допускается записывать в одной строке, как в следующем примере 5.

**Пример 5.**

```
if (a > b)
{
max = a; min = b;
}
else
{
max = b; min = a;
}
```

Если в фигурных скобках записано по одному оператору, то скобки можно опустить, как в примере 6.

Пример 6.

```
if (a > b) max = a;
else max = b;
```

### 3.1.2 Вложенные конструкции оператора if

В том случае, когда определённый блок операторов (или один оператор) нужно выполнить после проверки ни одного, а нескольких условий, то используют несколько конструкций оператора `if`. Операторы `if`, находящиеся внутри другого оператора `if`, называются вложенными конструкциями оператора `if`.

```
if (Условие 1 выбора)
    {
    // Если условие 1 выбора истинно будут выполняться, // записанные в
    скобках операторы блока 1.
    }
    else
    // В противном случаи будет выполняться
```

```
// вложенная конструкция оператора if.
{
// Начало вложенной конструкции оператора if.
if (условие 2 выбора)
{
// Если условие 2 выбора истинно будут
// выполняться, записанные здесь в скобках
// операторы блока 2.
}
Else
{
// В противном случае,
// если условие 2 выбора ложно будут
// выполняться, записанные здесь в скобках
// операторы блока 3.
}
}// Конец вложенной конструкции оператора if
```

Если `условие 1` выбора истинно, то выполняются операторы `блока 1`, и далее первый оператор, который следует за последней фигурной скобкой, концом вложенной конструкции оператора `if`. В противном случае выполняется вложенная конструкция оператора `if`. Если `условие 2` выбора вложенного оператора `if` истинно, то выполняются записанные в фигурных скобках операторы `блока 2`, и далее первый оператор, который следует за последней фигурной скобкой, концом вложенной конструкции оператора `if`. В противном случае, если `условие 2` выбора ложно выполняются операторы `блока 3`.

**Пример 7.**

```
if (x < -1)
{ n = 1; } else {
// Начало вложенной конструкции if.
if (x > 1)
{ n = 2; } else
{ n = 0; }
// Конец вложенной конструкции if.
}
```

**Пример 8.**

```
if (x < -1)
{ n = 1; } else {
if (x > 1) { n = 2; } else { n = 0; }
}
```

Поскольку фигурные, операторные скобки являются обязательными только в случае записи в них нескольких операторов, поэтому в данном случае они могут быть отпущены.

**Пример 9.**

if (x < -1) n = 1; else if (x > 1) n = 2; else n = 0;

### 3.2 Операторы логического сравнения

Эти операторы называются логическими сравнениями (logical comparisons), поскольку они возвращают результат сравнения в виде значения `true (истина)` или `false (ложь)` имеющие тип `bool`. Для записи и проверки условия равенства двух выражений, в алгоритмическом язык C# используется символ ==. Аналогично: символ > используется для проверки условия «больше»; символ < для проверки условия «меньше»; >= – «больше или равно»; <= – «меньше или равно»; `!=` «не равно». Например: `a!=b`, означает, что оператор логического сравнения `!=` возвращает значение `true`, если a не равно b.

Для переменных типа bool используются специальные логические операторы:

| Оператор | Описание |
|---|---|
| & | конъюнкция (логическое и, and), используется для логического объединения двух выражений; |
| \| | дизъюнкция (логическое или, or), используется, чтобы убедиться в том, что хотя бы одно из выражений true, истинно; |
| ! | отрицание (логическое не, not), возвращает обратное логическое выражение; |
| ^ | исключение (логическое исключающее или), используется для того, чтобы убедиться в том, что одно из двух выражений true, истинно. |

Операторы &, | и ^ обычно используются с целыми типами данных, а также могут применяться к типу данных bool.

Кроме того могут применяться операторы && и ||, которые отличаются от своих односимвольных версий тем, что выполняют ускоренные вычисления. Например в выражении a && b, b вычисляется лишь в том случае, если a равно true, истинно. В выражении a || b, b вычисляется в том случае, если a равно false, ложно.

```
Пример 10. if (x > -1 && x < 1)
//В условии оператора if записано обычное алгебраическое //неравенство
-1 < x < 1.
```

```
Пример 11. if (x < -1 || x > 1)
//В условии оператора if записаны алгебраические неравен//ства x < -1
либо x > 1.
```

### 3.3 Оператор проверки

Оператор проверки (тернарный оператор) выбирает одно из двух выражений в зависимости от проверки значения логического условия. Его синтаксис:

```
Имя переменной = (условие выбора)?Значение1:значение2
```

```
Пример 12. int value = (x < 25) ? 5:15
```

В этом примере сначала вычисляется выражение x < 25 являющееся условием выбора. Если оно равно `true`, то переменной `value` будет присвоено значение равное 5, в противном случае – равное 15.

### 3.4 Оператор безусловного перехода goto

Оператор безусловного перехода `goto` (перейти к) осуществляет переход, без проверки каких-либо условий, к оператору, обозначенному соответствующей меткой. Синтаксис этого оператора выглядит следующим образом:

```
метка: оператор
…
goto метка
…
```

где метка – метка. Это любой допустимый идентификатор C#, который помещается слева от оператора, которому надо передать управление выполнением программы и отделяется от него двоеточием. Причём метка может ставиться у оператора расположенного как до оператора `go-to`, так и после него. В случае если оператор `goto` используется самостоятельно, без каких либо конструкций, то первый оператор, следующий за оператором `goto,` должен иметь свою метку, иначе он не будет выполнен в процессе работы программы. Обычно оператор `goto` используется совместно с оператором условного перехода `if`, и используется в программах редко, т. к. есть более эффективные операторы.

### 3.5 Конструкция switch (переключатель)

Этот оператор позволяет сделать выбор среди нескольких альтернативных вариантов дальнейшего выполнения программы. Несмотря на то, что это может быть организовано с помощью последовательной записи вложенных операторов if, во многих случаях более эффективным оказывается применение оператора switch. Ниже приведена общая форма оператора.

```
switch (выражение)
{
case константа 1:
последовательность операторов блока 1 break;
case константа 2:
последовательность операторов блока 2 break;
default
последовательность операторов блока n break;
}
```

Этот оператор работает следующим образом. Значение выражения последовательно сравнивается с константами. Как только будет обнаружено совпадение, выполняется оператор или последовательность операторов, связанных с этим совпадением, до оператора break. Оператор break передаёт управление оператору, следующему за конструкцией switch. Если совпадений нет, то выполняется последовательность операторов, следующая после оператора default. Эта ветвь не является обязательной.

При использовании конструкции switch действуют следующие правила:

- выражение в конструкции switch должно быть целочисленного типа (char, byte, short или int) перечислимого типа или же типа строкового;
- нельзя использовать числа с плавающей точкой;
- константы оператора case должны иметь тот же тип, что и выражение в конструкции

switch;

- в одном операторе switch не допускается наличие двух одинаковых по значению констант;
- допускается использовать одну и ту же последовательность операторов, в этом случае оператор break не записывается.

**Пример 13.**

```
using System;

namespace Test
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
        m1: Console.WriteLine("Введите целое число");
            int a = int.Parse((Console.ReadLine()));
            switch (a)
            {
                case 1:
                    n = 10; break;
                case 2:
                    n = 15; break;
                case 3:
                    n = 20; break;
                default: n = 0; break;
            }
            Console.WriteLine("a = " + a + " n = " + n);
            if (a != 0) goto m1;
            Console.Read();
        }
    }
}
```

В данном примере в программу вводится и присваивается переменной a, любое целое число. С помощью конструкции `switch` происходит анализ. Если переменная a имеет значение равное 1, переменной n присваивается значение 10 и далее следует вывод этих переменных. Если a имеет значение равное 2 или 3, то переменной n присваивается значение 20 и далее вывод этих переменных. Во всех остальных случаях переменной n присваивается значение 0. Программа продолжает работать до тех пор, пока переменой a, не будет задано значение 0.

Один оператор `switch` может быть частью последовательности другого внешнего оператора `switch`. Такой оператор называется вложенным. Константы внешнего и внутреннего операторов `switch` могут содержать общие значения, не вызывая каких либо конфликтов.

В операторе `switch` отсутствует возможность задания диапазона выбора, что является его недостатком. Например, в языке программирования Visual Basic в аналогичном операторе задание диапазона выбора допускается.

## 4 Задания

**Задание 1.** Составить программу для решения поставленной задачи двумя способами: используя: 1) команды case 2) команды if. Данные сформировать таким образом, что бы выбор был из 3-5 альтернатив.

1. По номеру студента в списке группы вывести его фамилию.
2. По номеру поезда вывести название пункта назначения.
3. По номеру для недели напечатать его название.
4. По номеру троллейбуса вывести название конечной остановки.
5. По номеру месяца вывести название поры года.
6. По номеру студента в списке группы вывести его имя.
7. По номеру месяца вывести номер квартала.
8. По номеру маршрута автобуса вывести количество остановок его маршрута.
9. По телефонному коду города вывести название города.
10. По номеру дня недели вывести количество пар в этот день.
11. По номеру месяца выводить количество дней в этом месяце.
12. По числу из диапазона 0-5 напечатать его написание на английском языке.
13. По номеру поезда вывести время его отправления.
14. По названию реки вывести ее длину.
15. По номеру квартиры в доме вывести количество жильцов.

**Задание 2.** Выполнить по вариантам задание с использованием оператора ветвления if .. else.

1. Даны три действительные числа A, B, C. Те из них, которые принадлежат интервалу [2.0,7.2] заменить нулем, а те, которые принадлежат интервалу [-2.0,-0.5] удвоить, остальные оставить без изменения.
2. Если сумма трех различных чисел A, B, C равна 2, то наибольшее из этих чисел заменить наименьшим, если сумма больше 20 , то все числа возвести в квадрат, в противном случае наибольшее из этих трех чисел заменить числом 34.
3. Если сумма трех попарно различных чисел x, y, z меньше единицы, то наименьшее из этих трех чисел заменить полусуммой двух других, в противном случае заменить меньшее из x и y полусуммой двух оставшихся значений.
4. Даны действительные числа A и B. Если A и B неотрицательны и их сумма меньше 15, то оба числа заменить их произведением, если хотя бы одно из них меньше -3, то оба числа удвоить, в противном случае возвести в квадрат большее из них, а меньшее заменить нулем.
5. Если произведение трех попарно различных действительных чисел x, y, z - число отрицательное, то наименьшее из этих трех чисел заменить произведением двух остальных, в противном случае заменить наибольшее из них их среднеарифметическим.
6. Даны действительные числа p, q, r. Если p<q<r , то все числа заменить абсолютным значением минимального из них, если p<q<r, то те из них, которые больше 2, заменить нулями, в противном случае оставить числа без изменения.
7. Даны действительные числа x, y , z. Если $x \le z \le y$, то каждое число заменить наибольшим из них, если $x^3 z^3 y$, то числа оставить без изменения, в противном случае все числа заменить их квадратами.

8. Даны действительные числа X и Y. Если X и Y положительны, но их произведение меньше 100, то оба числа заменить нулями, если хотя бы одно из них отрицательно, то заменить их абсолютными значениями, в противном случае минимальное число оставить без изменения, а максимальное возвести в квадрат.
9. Даны действительные числа X и Y. Если X и Y отрицательны, то каждое значение заменить их произведением; если отрицательно только одно из них, то оба значения увеличить на 0,5; если оба значения неотрицательны и их сумма меньше 10, то меньшее из чисел заменить квадратом, в противном случае оставить без изменения.
10. Даны действительные числа a, в, c. Если сумма минимального и максимального из них меньше 5, то максимальное число заменить нулем, в противном случае заменить минимальное число произведением двух остальных.
11. Даны два действительных числа. Если оба они положительны, но минимальное из них не превышает 1, то заменить их нулями. Если минимальное число больше 1, но не превышает 10, то заменить только минимальное нулем, в противном случае поменять их местами.
12. Если сумма трех различных чисел A, B, C равна 2, то наибольшее из этих чисел заменить наименьшим, если сумма больше 20 , то все числа возвести в квадрат, в противном случае наибольшее из этих трех чисел заменить числом 34.
13. Даны действительные числа X и Y. Если X и Y положительны, но их произведение меньше 60, то оба числа заменить нулями, если хотя бы одно из них отрицательно, то заменить их -1, в противном случае минимальное число оставить без изменения, а максимальное возвести в куб.
14. Если произведение трех попарно различных действительных чисел x, y, z - число отрицательное, то наименьшее из этих трех чисел заменить суммой двух остальных, в противном случае заменить наибольшее из них среднеарифметическим двух других чисел.
15. Даны действительные числа Q и Z. Если Q и Z неотрицательны и их сумма больше 15, то оба числа заменить их произведением, если хотя бы одно из них меньше -1, то оба числа утроить, в противном случае возвести в квадрат большее из них, а меньшее заменить нулем.

**Задание 3.** Выполнить по вариантам задание с использованием оператора ветвления switch .. case.

1. Дано целое число R. Пусть k - остаток от деления $(5R + 2)^2$ на 4. Вычислить u , используя SWITCH-CASE.

$$u = \begin{cases} \operatorname{tg} R + \sin R^3 \text{ , при k=0} \\ R^2 + \sqrt[5]{R+1} \text{ , при k=1} \\ e^R + 2{,}5\,(R - 3) \text{ , при k=2 или 3} \end{cases}$$

2. Используя оператор SWITCH-CASE составить программу вычисления значения функции y при $X \in [1;5]$.

$$y = \begin{cases} A + BX + CX^2, \text{ при } 1 <= X < 2 \\ (a * \sin(X-B))^2, \text{ при } 2 <= X < 3 \\ \sqrt{|a + BX^3|} + C, \text{ при } 3 <= X < 4 \\ a * \ln|B + C/2X|, \text{ при } 4 <= X < 5 \end{cases}$$

3. Используя оператор SWITCH-CASE составить программу вычисления функции y при $x \in [0;50]$.

$$y = \begin{cases} 2{,}3\,x + \sqrt{\ln x / \pi}, \text{ при } 0 <= x < 10 \\ (x+2)^6 + x/(x+7), \text{ при } 10 <= x < 20 \\ 60 - x + x^2, \text{ при } 20 <= x < 30 \\ e^{\sin x} + \sqrt{|x-1|}, \text{ при } 30 <= x < 40 \\ 25\,x^3 - 7, \text{ при } 40 <= x < 50 \end{cases}$$

4. Используя оператор SWITCH-CASE составить программу вычисления функции y при $x \in [0;20]$.

$$y = \begin{cases} \sqrt{(x+1)^2 + 5}, \text{ если } 0 <= x < 5 \\ \ln x/2, \text{ если } 5 <= x < 10 \\ \operatorname{tg} x, \text{ если } 10 <= x < 15 \\ \sqrt{x + \pi/2}, \text{ если } 15 <= x < 20 \end{cases}$$

5. Дано действительное число R. Пусть k – округленное значение $\sin^2 R$. Вычислить z, используя оператор SWITCH-CASE.

$$z = \begin{cases} R + |\sin R/3|, \text{ если k кратно 5} \\ \operatorname{tg}(R+2), \text{ если при делении k на 5 остаток равен 1} \\ e^{R-1} + \operatorname{arctg} R, \text{ если при делении k на 5 остаток равен 2 или 4} \\ \ln(R+5)/3{,}7, \text{ в остальных случаях} \end{cases}$$

6. Дано действительное число R. Пусть k – округленное значение $R \sin^3 R$. Вычислить z, используя оператор SWITCH-CASE.

$$z = \begin{cases} R + |\sin R/3|, \text{ если k кратно 5} \\ \operatorname{tg}(R+2), \text{ если при делении k на 5 остаток равен 1} \\ e^{R-1} + \operatorname{arctg} R, \text{ если при делении k на 5 остаток равен 2 или 4} \\ \ln(R+5)/3{,}7, \text{ в остальных случаях} \end{cases}$$

7. Используя оператор SWITCH-CASE написать программу для вычисления значения функции y при $k \in [0;20]$.

$$y=\begin{cases}\text{Sin k / 3} + \sqrt[5]{(k+1)}\ \text{, при } 0 <= k < 5\\ \text{tg k}^2 + \sqrt{(k+1)}\ \text{, при } 5 <= k < 10\\ \text{arctg}^2\,(k+1)\ \text{, при } 10 <= k < 15\\ e^{(k+1)/10}\ \text{, при } 15 <= k < 20\end{cases}$$

8. Используя оператор SWITCH-CASE составить программу вычисления функции y при $x\in[0;8]$.

$$y=\begin{cases}\text{Sin}^3 x + \sqrt{x/2}\ \text{, при } 0 <= x < 2\\ |\ x - 5\ |\ /\ \text{tg x}\ \text{, при } 2 <= x < 4\\ \text{sin (x + 3) * ln x}\ \text{, при } 4 <= x < 6\\ e^{3x} + |\ \text{tg x}\ |\ \text{, при } 6 <= x < 8\end{cases}$$

9. Дано целое четырехзначное число k. Пусть M – сумма первой и четвертой цифр числа k. Вычислить y , используя оператор SWITCH-CASE.

$$y=\begin{cases}\text{tg}^2\ k + \sin^2 k/2{,}7\ \text{, при M = 3 или 5}\\ \ln|\ k - 6{,}3\ |\ \text{, при M = 7 или 8 или 9}\\ e^{|k-0{,}2k^2|}\ \text{, при M = 10}\\ k^2 + 2{,}6\ k + 3{,}7\ \text{, при M = 12 или 13}\\ k*\sin k^2 - 6k\ \text{, в остальных случаях}\end{cases}$$

10. Даны три положительных числа a, b, c. Пусть k – количество десятков в числе $R=a^2+b^2+c^2$ . Используя оператор SWITCH-CASE, составить программу для вычисления y.

$$y=\begin{cases}(a + b)^4 / c + \text{tg a/b}\ \text{, при k = 1 или 7}\\ |\ e^a + b^2\ |\ \text{, при k = 2 или 3 или 4}\\ \sin (a - \pi/2) - 3\ \text{, при k = 5}\\ (a - \pi) / 25b - a/c\ \text{, в остальных случаях}\end{cases}$$

a.

11. Дано действительное число R. Пусть k – округленное значение $R \sin^3 R$. Вычислить z, используя оператор SWITCH-CASE.

$$z=\begin{cases}R + |\ \sin R/3\ |\ \text{, если k кратно 5}\\ \text{tg (R + 2)}\ \text{, если при делении k на 5 остаток равен 1}\\ e^{R-1} + \text{arctg R}\ \text{, если при делении k на 5 остаток равен 2 или 4}\\ \ln (R + 5) / 3{,}7\ \text{, в остальных случаях}\end{cases}$$

12. Дано целое число L. Определить W, используя оператор SWITCH-CASE.

$$W=\begin{cases}\pi L^2 + \sqrt{L}\ \text{, если L кратно 7}\\ L^5 / (L + 2)\ \text{, если остаток от деления L на 7 равен 2 или3}\\ e^{L \sin L} / (1 + L*\ln L)\ \text{, если остаток от деления L на 7 равен 5}\\ 75{,}3\ \text{, в прочих случаях}\end{cases}$$

13. Даны три положительных числа a, b, c. Пусть k – количество десятков в числе $R=a^3+b^3+c^3$ . Используя оператор SWITCH-CASE, составить программу для вычисле-

ния y.

$$y = \begin{cases} (a + b) \ / c + tg\ a/b\ , \text{при } k = 1 \text{ или } 7 \\ | e + b |\ , \text{при } k = 2 \text{ или } 3 \\ \sin(a - \pi/2)\ , \text{при } k = 5 \\ (a - \pi) / 25b - a/c\ , \text{в остальных случаях} \end{cases}$$

14. Используя оператор SWITCH-CASE составить программу вычисления функции y при x∈[0;9].

$$y = \begin{cases} \mathrm{Sin}^2 x + \sqrt{x/2}\ , \text{при } 0 <= x < 2 \\ | x + 5 | \ / \ tg\ x\ , \text{при } 2 <= x < 4 \\ \sin(x + 3) * \ln x\ , \text{при } 4 <= x < 6 \\ e^3 + | \ tg\ x |\ , \text{при } 6 <= x < 9 \end{cases}$$

15. Используя оператор SWITCH-CASE написать программу для вычисления значения функции y при k∈[0;15].

$$y = \begin{cases} \mathrm{Sin}\ k / 3 + \sqrt[5]{(k+1)}\ , \text{при } 0 <= k < 5 \\ tg\ k^2 + \sqrt{(k+1)}\ , \text{при } 5 <= k < 10 \\ \mathrm{arctg}^2\ (k + 1)\ , \text{при } 10 <= k < 15 \end{cases}$$

## 5 Контрольные вопросы

1. Какие операторы используются для программирования разветвлений? В чем их разница?
2. Как выполняется оператор if ?
3. Какие существуют операторы логического сравнения в C#?
4. Какой безусловный оператор существует в C#?
5. Что такое тернарный оператор?
6. Как выполняется оператор switch?