

Лабораторная работа №5

**ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ
АВТОМОБИЛЬНО-ДОРОЖНЫЙ УНИВЕРСИТЕТ**

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И МЕХАТРОНИКИ

Кафедра информационных технологий и мехатроники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

**по проведению лабораторных работ по дисциплине «Алгоритмизация и
программирование»**

для студентов специальности 6.050101 «Компьютерные науки»

Разработчик - доцент кафедры информационных технологий и мехатроники
кандидат технических наук, старший научный сотрудник
Тимонин Владимир Алексеевич

Харків 2015

Лабораторная работа №5

Исследование возможностей интегрированной среды разработки Visual C# для создания приложений разветвляющейся структуры.

Цель работы – исследовать возможности интегрированной среды разработки Visual Studio 2010 и получить практические навыки по созданию приложений разветвляющейся структуры.

1. Теоретические сведения

Решение ряда практических задач не ограничивается линейным алгоритмом, в котором все операторы выполняются последовательно, а предусматривает различные пути вычисления решения. Причем выбор того или иного пути определяется либо условиями задачи, либо результатами, полученными в процессе решения. Каждое из возможных направлений вычисления называется ветвью, в зависимости от выполнения не которого условия вычислительный процесс может идти по одной или другой ветви. Алгоритм такого вычислительного процесса называется разветвляющимся алгоритмом. Количество ветвей в общем случае может быть больше двух.

Для того, чтобы в зависимости от исходных данных обеспечить выполнение разных последовательностей операторов, применяются операторы ветвления **if** и **switch**. Оператор **if** обеспечивает передачу управления на одну из двух ветвей вычислений, а оператор **switch** — на одну из произвольного числа ветвей.

1.1. Условный оператор **if**

Условный оператор **if** используется для разветвления процесса вычислений на два направления (структурная схема представлена на рис. 1).

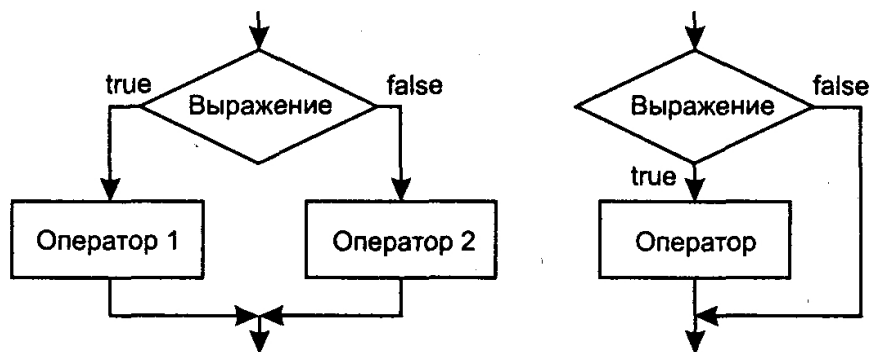


Рис. 1. Структурная схема условного оператора **if**

Формат оператора (полная форма) имеет вид:

```
if ( выражение )  
  оператор 1;  
  else оператор 2;
```

Сначала вычисляется выражение, представляющее собой арифметические, логические операции и операции сравнения. Если оно не равно нулю (имеет значение **true**), выполняется первый оператор, иначе — второй. После этого управление передается на оператор, следующий за условным.

Одна из ветвей может отсутствовать, логичнее опускать вторую ветвь вместе с ключевым словом **else**, в этом случае формат оператора (сокращенная форма) имеет вид:

```
if ( выражение )  
  оператор 1;
```

Если в какой-либо ветви требуется выполнить несколько операторов, их необходимо заключить в блок, иначе компилятор не сможет понять, где заканчивается ветвление. Блок может содержать любые операторы, в том числе описания и другие условные операторы (но не может состоять из одних описаний). Необходимо учитывать, что переменная, описанная в блоке, вне блока не существует.

```
if ( выражение )
{
    оператор 1;
    ...
    оператор k;
}
else
{
    оператор k+1;
    ...
    оператор N;
}
```

Когда в соответствии с реализуемым алгоритмом приходится делать многовариантный выбор, то применяют конструкцию вида:

```
if ( выражение )
{ блок }
else if ( выражение )
{ блок }
else if ( выражение )
{ блок }
else { блок }
```

Данная конструкция последовательно вычисляет выражение в каждой строке. Если выражение истинно, то выполняется тело (т.е. блок операторов) и происходит выход из конструкции на выполнение следующего за ней оператора. Если выражение ложно, то начинает вычисляться выражение в следующей строке. Последняя часть конструкции (**else {блок}**) не обязательна.

Примеры условного оператора **if**:

```
if (b>a) max=b; else max=a; /* полная форма. Конструкции, подобные в этом
операторе, проще и нагляднее записывать в виде
условной операции: max = (b>a) ? b : a; */

if (a<0) b=1; // сокращенная форма
if (a<b && (a>d || a==0)) b++; else {b*=a; a=0} /* Если требуется проверить
несколько условий, их объединяют знаками логических
операций. Выражение будет истинно в том случае, если
выполнится одновременно условие a<b и одно из
условий в скобках. Если опустить внутренние скобки,
будет выполнено сначала логическое И, а потом - ИЛИ.*/

if (a<b) { if (a<c) m=a; else m=c; } else { if (b<c) m=b; else m=c; } /* Оператор
вычисляет наименьшее значение из трех переменных.
Фигурные скобки в данном случае не обязательны, так как
компилятор относит часть else к ближайшему if.*/

if (a++) b++; /* в качестве выражения не обязательно используются операции
отношения. */
```



Пример 1. В качестве примера создадим консольное приложение, которое по введенному значению аргумента вычисляет значение функции, заданной в виде графика (рис. 2).

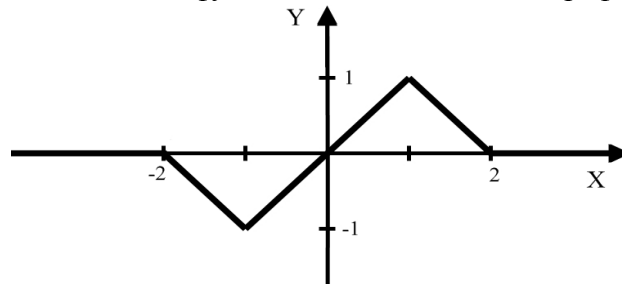


Рис. 2. График функции

1. Анализ задачи. Исходные данные: x – вещественное значение аргумента, результату y – вещественное значение функции. Ввод значения x осуществляется с клавиатуры.

Так как ввод и вывод осуществляется в виде строк, необходима переменная для хранения вводимых и выводимых значений – введем промежуточную переменную **Str**.

Вычислительный процесс – разветвляющийся.

2. Разработка алгоритма. В алгоритме необходимо выполнить следующие действия:

1. ввод значения аргумента x ;
2. определение, какому интервалу принадлежит значение x ;
3. вычисление значения функции y ;
4. вывод результата на печать

Алгоритм представляет собой последовательность функциональных блоков: объявления переменных, формирования заголовка приложения, ввода значения аргумента x с выдачей соответствующего сообщения и преобразованием его в число и присвоения начального значения функции y ; проверки попадания значения x в соответствующий интервал и вычисления значения функции y ; преобразования результата в строковую переменную, вывода результата, ожидания нажатия клавиши <Enter>. Схема алгоритма приведена на рис. 3.

3. Разработка программного кода. Программный код решения задачи представлен на рис. 4. В теле главной функции **Main ()** объявляются переменные, используемые в программе: x , y – переменные вещественного типа, переменная **Str** – переменная строкового типа.

Оператор 15 определяет заголовок приложения. Оператор 16 выводит сообщение о необходимом действии пользователя. Оператор 17 записывает значение аргумента x в виде строки в переменную **Str**. Оператор 18 преобразует строковую переменную в вещественное число.

Операторы 20 - 24 осуществляют проверку, к какому интервалу принадлежит значение аргумента x . Если значение x принадлежит соответствующему интервалу, то вычисляется значение функции y . Оператор 25 преобразует результат вычисления (вещественное число) в строковую переменную **Str**.

Операторы 26, 27 выдают результат на экран монитора

Один из вариантов результата решения задачи представлен на рис. 5.

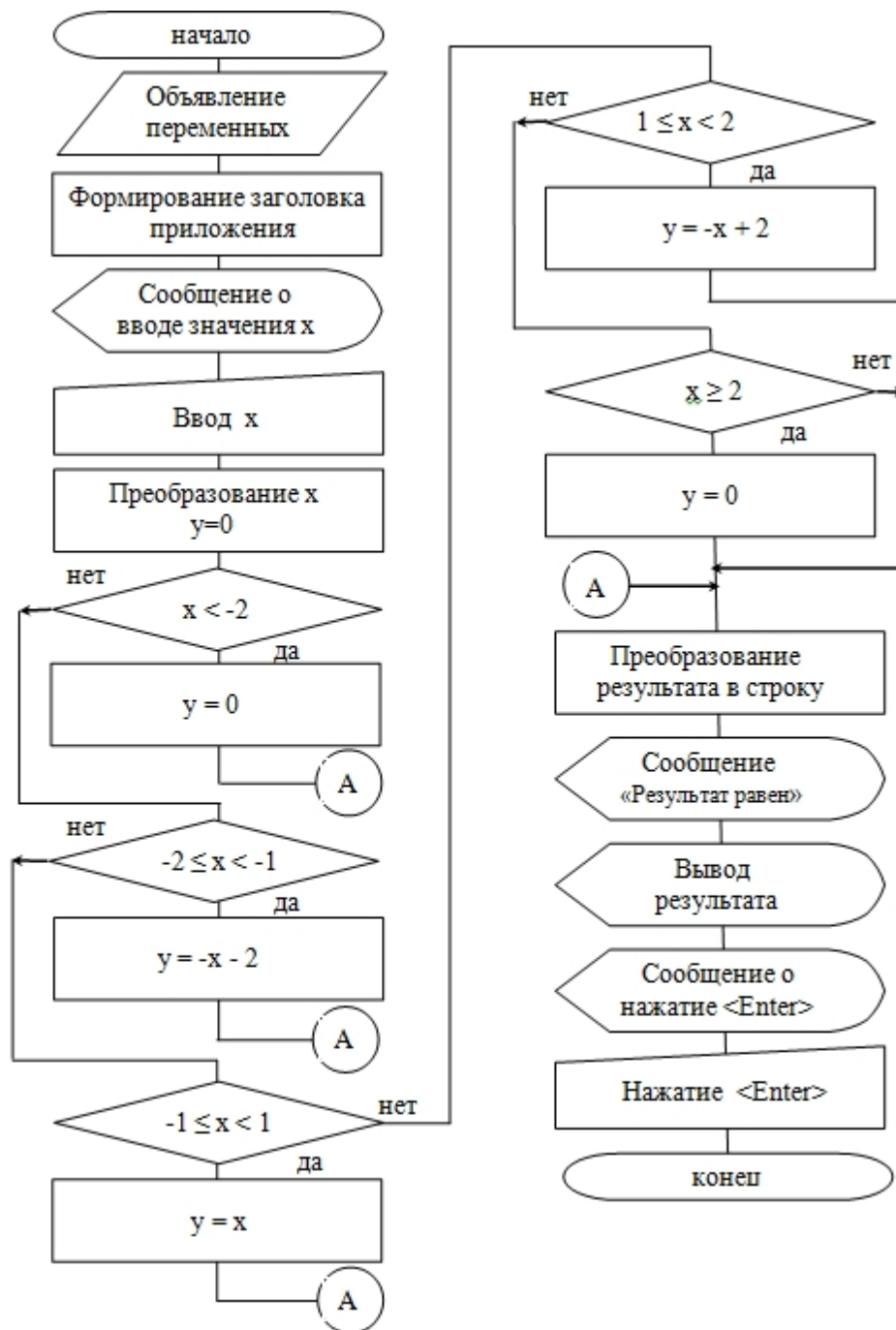


Рис. 3. Блок-схема алгоритма

```

Program.cs X
Primer1.Program Main()
10 static void Main()
11 {
12     double x; // имя переменной, в которой хранится аргумент x
13     double y; // имя переменной, в которой хранится результат
14     String Str; // имя переменной, в которой хранится совокупность символов
15     Console.Title = "Решение уравнения";
16     Console.Write("Введите значение аргумента x ");
17     Str = Console.ReadLine();
18     x = Convert.ToDouble(Str);
19     y = 0;
20     if (x < -2) y = 0;
21     if (x >= -2 && x < -1) y = -x - 2;
22     if (x >= -1 && x < 1) y = x;
23     if (x >= 1 && x < 2) y = -x + 2;
24     if (x >= 2) y = 0;
25     Str = y.ToString("f2");
26     Console.Write("Результат равен ");
27     Console.WriteLine(Str);
28     Console.Write("Для завершения работы приложения нажмите <Enter>");
29     Console.Read();
30 }

```

Рис. 4. Программный код решения уравнения

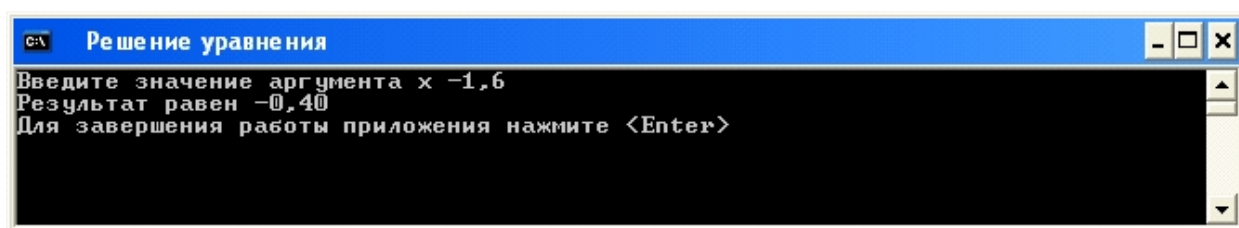


Рис. 5. Результат решения уравнения

1.2. Оператор switch

Оператор **switch** (переключатель) предназначен для разветвления процесса вычислений на несколько направлений (структурная схема представлена на рис. 6).

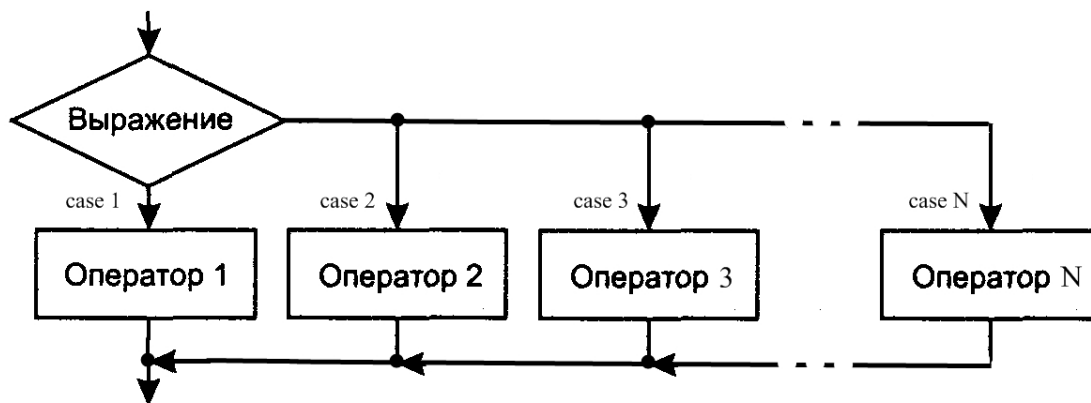


Рис. 6. Структурная схема условного оператора switch

Формат оператора имеет вид:

```

switch ( выражение )
{
    case <значение 1>: оператор 1; break;
    case <значение 2>: оператор 2; break;
    case <значение 3>: оператор 3; break;
}


```

```

        . . .
        case <значение N>: оператор N; break;
        default: оператор N+1; break;
    }

```

Выполнение оператора начинается с вычисления выражения **switch** (оно должно быть целочисленным), а затем программа последовательно сравнивает значение выражения со значениями **case** и, если находит совпадение, выполняет соответствующий оператор. Если ничего не найдено, то будет выполнен код который идет после ключевого слова **default**. Выход из переключателя обычно выполняется с помощью операторов **break** или **return**.


 **Пример 2.** Нижеприведенный код консольного приложения, которое определяет значение целого числа, вводимого с клавиатуры, демонстрирует применение оператора **switch**.


```

static void Main()
{
    int i;
    Console.Write("Введите значение целого числа от 0 до 9\n");
    i = Convert.ToInt16(Console.ReadLine());
    switch (i)
    {
        case 0: Console.WriteLine("Число равно нулю.");
                break;
        case 1: Console.WriteLine("Число равно единице.");
                break;
        case 2: Console.WriteLine("Число равно двум.");
                break;
        case 3: Console.WriteLine("Число равно трем.");
                break;
        case 4: Console.WriteLine("Число равно четырем.");
                break;
        default: Console.WriteLine("Число равно или больше пяти.");
                break;
    }
    Console.Write("Для завершения работы нажмите клавишу <Enter>");
    Console.Read();
}

```

2. Рабочее задание

 **Задание 1.** Руководствуясь теоретическим материалом раздела 1 изучить возможности языка C# по созданию приложений разветвляющейся структуры и выполнить практически все примеры, описанные в этом разделе.

 **Задание 2.** Разработать приложение с заголовком «Оператор if», которое выводит на экран значение функции $y = f(x)$, вычисляемой по формуле:

$$y = \begin{cases} ax^2 + bx + c, & \text{если } x < 0 \text{ и } b \neq 0 \\ \frac{x-a}{x-c}, & \text{если } x > 0 \text{ и } b = 0 \\ \frac{x}{c}, & \text{в остальных случаях} \end{cases}$$

где a, b, c, x – переменные величины, значения которых вводятся с клавиатуры

В отчете представить блок-схему, исходный код и результаты выполнения приложения



Задание 3. Разработать приложение с заголовком «Оператор switch», с помощью которого можно выполнять арифметические (+, -, *, /, %) и логические операции (&, |, ^, >>, <<, ~). Значения переменных и знак операции вводится с клавиатуры. При разработке приложения использовать оператор **switch**.

3. Контрольные вопросы

1. Дайте определение и характеристику разветвляющегося алгоритма.
2. Перечислите основные операторы, реализующие разветвления в алгоритмах, и их предназначение.
3. Дайте полную характеристику оператору **if** (назначение, синтаксис и примеры).
4. Дайте полную характеристику оператору **switch** (назначение, синтаксис и примеры).

Литература

1. Голощапов А.Л. Microsoft Visual Studio 2010. – СПб.:БХВ-Петербург, 2011. – 544 с.: ил.
2. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ. - М.: «Русская Редакция», 2002.- 576 с.: ил.
3. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ. - М.: «Русская Редакция», 2002.- 624 с.: ил.
4. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. Пер. с англ. - М.: Издательский дом "Вильямс", 2011. — 1392 с.: ил.
5. Фленов М.Е. Библия C#. - СПб.: БХВ-Петербург, 2011. – 560с.: ил.
6. Шилдт Г. C# Учебный курс. – СПб.: Питер, Издательская группа BHV, 2003. – 512 с.: ил.