Лабораторная работа №7
Исследование возможностей интегрированной среды разработки Visual C# для создания приложений циклической структуры (оператор for).

Цель работы – исследовать возможности интегрированной среды разработки Visual Studio 2010 и получить практические навыки по созданию приложений циклической структуры с использованием операторов for.

1. Теоретические сведения

Оператор **for** используется в тех случаях, когда количество повторений тела цикла заранее известно. Цикл с параметром имеет следующий формат:

for (инициализация; выражение; модификации) оператор;

Инициализация используется для объявления и присвоения начальных значений величинам, используемым в цикле. В этой части можно записать несколько операторов, разделенных запятой (операцией «последовательное выполнение»), например, так:

```
for (int i=0, j=0; …; …) оператор;
```

или

```
int k, n;
for (k=1, n=1; …; …) оператор;
```

Областью действия переменных, объявленных в части инициализации цикла, является цикл. Инициализация выполняется один раз в начале исполнения цикла.

Выражение определяет условие выполнения цикла. Выполнение цикла осуществляется до тех пор, пока условное выражение принимает значение истина, т.е. если результат проверки условия, приведенный к типу **bool**, равен **true**, цикл выполняется, в противном случае, осуществляется выход из цикла. Цикл с параметром реализован как цикл с предусловием.

Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметров цикла. В части модификаций можно записать несколько операторов через запятую, например, так:

```
for ( …; …; i++, k=k+2) оператор;
```

Простой или составной оператор представляет собой тело цикла.

Любая из частей оператора **for** может быть опущена (но точки с запятой надо оставить на своих местах **!!!**).

Например, оператор, вычисляющий сумму чисел от 1 до 100:

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

1.1. Разработка простого приложения циклической структуры

Пример 1. В качестве примера создадим консольное приложение, которое вычисляет значение функции $y=a-4\cos^3(x)$ с заданным шагом Δx в заданном диапазоне аргумента x.

1. Анализ задачи. Исходные данные: **a** – целочисленная переменная, $\mathbf{x_n}$, $\mathbf{x_k}$ – вещественные значения угла в градусах, определяющие начальное и конечное значение диапазона, **Δx** – вещественное значение приращения угла, результат **y** – вещественное значение функции. Ввод значений **a,** $\mathbf{x_n}$, $\mathbf{x_k}$, **Δx** 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 – циклический.

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

1. ввод значения переменной a;

2. ввод значений диапазона изменения угла и шаг приращения угла x;
3. установка начального значения угла x;
4. проверка, находится выбранное значение угла x в заданном интервале x;
5. вычисление значения функции y;
6. вывод результата на печать;
7. модификация значения угла.

Алгоритм представляет собой последовательность функциональных блоков: формирования заголовка приложения; ввода значений переменной **a,** диапазона изменения угла **x** (**x_n, x_k**), шага приращения угла (**dx**) с выдачей соответствующего сообщения и преобразованием введенного значения в число; присвоения начального значения угла **x**, проверки условия выражения (**x <= x_k**) и модификация параметра цикла; вычисления значения функции **y,** преобразования результата в строковую переменную и 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: **a** – переменная целого типа, **x_n, x_k, dx, x, y** – переменные вещественного типа, переменная **Str** – переменная строкового типа.

Оператор 18 определяет заголовок приложения. Оператор 19 выводит сообщение о необходимом действии пользователя. Оператор 20 записывает значение аргумента **a** в виде строки в переменную **Str**. Оператор 21 преобразует строковую переменную в целое число.

Операторы 22 – 24, 25 – 27, 28 - 30 обеспечивают ввод значений **x_n, x_k** и **dx** и преобразование их в вещественные числа.

Оператор 31 осуществляет начальную установку параметра цикла **x**,осуществляет проверку, принадлежит значение аргумента **x** заданному диапазону. Если значение **x** принадлежит соответствующему интервалу (не больше **x_k**), то выполняется тело цикла (операторы 33 – 35). Оператор 33 вычисляет значение функции **y**. Оператор 34 преобразует результат вычисления (вещественное число) в строковую переменную **Str**. Оператор 35 выдаёт результат на экран монитора. После чего управление передаётся оператору 31, который модифицирует значение параметра **x** и осуществляет проверку на принадлежность **x** заданному диапазону.

Если в результате проверки (оператор 31) начальное или модифицированное значение угла **x** не принадлежит интервалу, т.е. **x > x_k,** управление передается оператору 37.

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

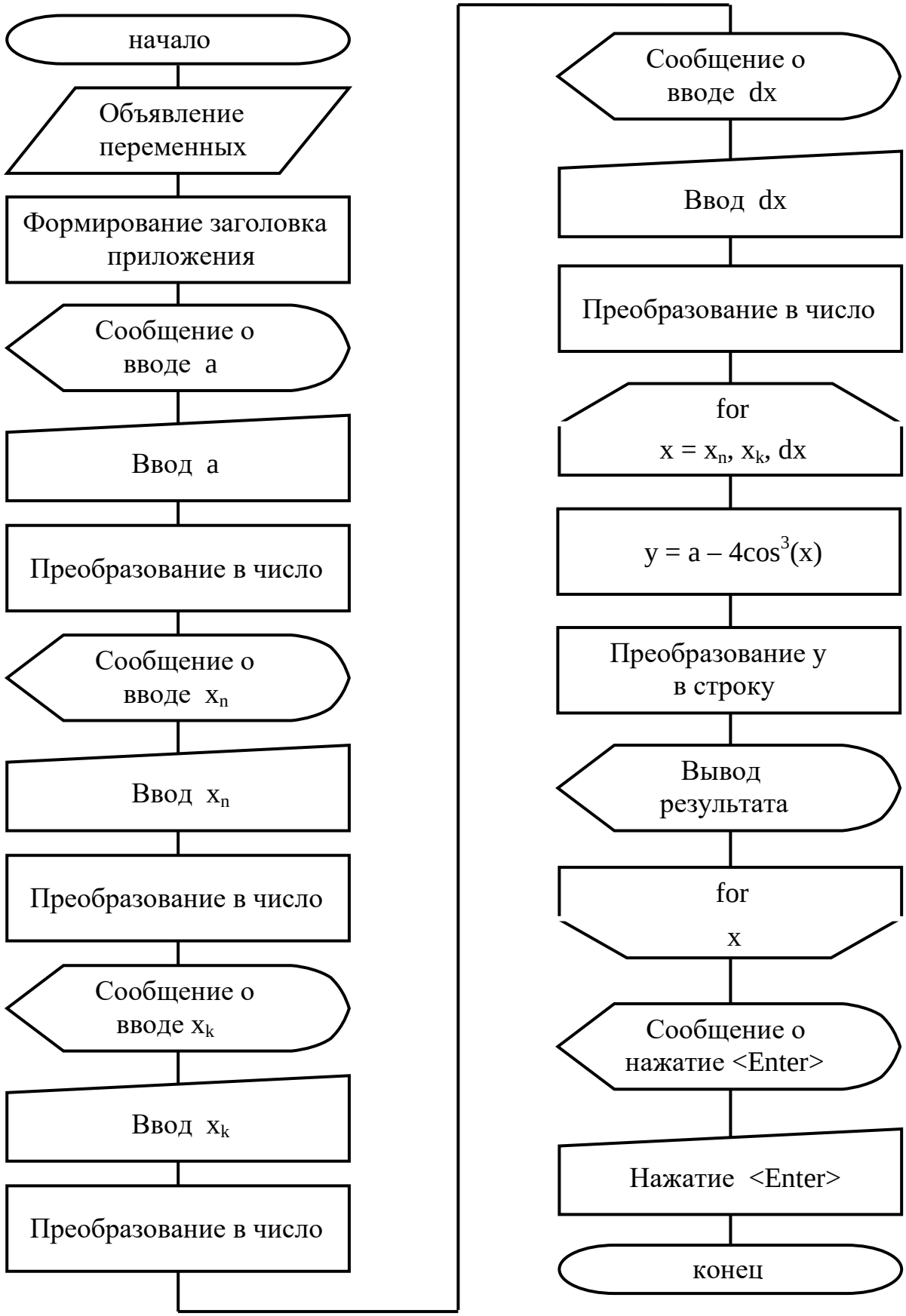

Рис. 3. Блок-схема алгоритма

Program.cs

Primer1.Program — Main()

```
10        static void Main()
11        {
12            int a; // имя переменной, в которой хранится значение постоянной a
13            double xn, xk, dx; // имена переменных, в которых хранятся начальное,
14                               // конечное значения диапазона и величина приращения угла x
15            double x; // имя переменной, в которой хранится текущее значение угла x
16            double y; // имя переменной, в которой хранится результат
17            String Str; // имя переменной, в которой хранится совокупность символов
18            Console.Title = "   Решение уравнения";
19            Console.Write("Введите значение переменной a ");
20            Str = Console.ReadLine();
21            a = Convert.ToInt32(Str);
22            Console.Write("Введите начальное значение диапазона в градусах ");
23            Str = Console.ReadLine();
24            xn = Convert.ToDouble(Str);
25            Console.Write("Введите конечное значение диапазона в градусах ");
26            Str = Console.ReadLine();
27            xk = Convert.ToDouble(Str);
28            Console.Write("Введите значение шага приращения угла в градусах ");
29            Str = Console.ReadLine();
30            dx = Convert.ToDouble(Str);
31            for (x=xn; x<=xk; x=x+dx)
32                {
33                 y = a - 4*Math.Pow(Math.Cos(x*Math.PI/180), 3);
34                 Str = y.ToString("f2");
35                 Console.WriteLine("При значении x = {0} y = {1}", x, Str);
36                }
37            Console.Write("Для завершения работы приложения нажмите <Enter>");
38            Console.Read();
39        }
```

100 %

Рис. 4. Программный код решения уравнения

```
Решение уравнения
Введите значение переменной a 2
Введите начальное значение диапазона в градусах 0
Введите конечное значение диапазона в градусах 360
Введите значение шага приращения угла в градусах 10
При значении x = 0 y = -2,00
При значении x = 10 y = -1,82
При значении x = 20 y = -1,32
При значении x = 30 y = -0,60
При значении x = 40 y = 0,20
При значении x = 50 y = 0,94
При значении x = 60 y = 1,50
При значении x = 70 y = 1,84
При значении x = 80 y = 1,98
При значении x = 90 y = 2,00
При значении x = 100 y = 2,02
При значении x = 110 y = 2,16
При значении x = 120 y = 2,50
При значении x = 130 y = 3,06
При значении x = 140 y = 3,80
При значении x = 150 y = 4,60
При значении x = 160 y = 5,32
При значении x = 170 y = 5,82
При значении x = 180 y = 6,00
При значении x = 190 y = 5,82
При значении x = 200 y = 5,32
При значении x = 210 y = 4,60
При значении x = 220 y = 3,80
При значении x = 230 y = 3,06
При значении x = 240 y = 2,50
При значении x = 250 y = 2,16
При значении x = 260 y = 2,02
При значении x = 270 y = 2,00
При значении x = 280 y = 1,98
При значении x = 290 y = 1,84
При значении x = 300 y = 1,50
При значении x = 310 y = 0,94
При значении x = 320 y = 0,20
При значении x = 330 y = -0,60
При значении x = 340 y = -1,32
При значении x = 350 y = -1,82
При значении x = 360 y = -2,00
Для завершения работы приложения нажмите <Enter>_
```

Рис. 5. Результат решения уравнения

1.2. Разработка приложения с вложенными циклами

Любой цикл может содержать внутри себя один или несколько других циклов. Такая структура называется ***вложенными циклами***. Охватывающие циклы называются ***внешними***, охватываемые — ***внутренними***. К алгоритмам с вложенными циклами приводят задачи табулирования функции нескольких переменных, вычисления кратных сумм и произведений и др.

Пример 2. В качестве примера создадим консольное приложение, с помощью которого вычисляется радиус окружности с центром в начале координат, представляемый уравнением $\mathbf{R^2 = x^2 + y^2}$. Аргумент **x** изменяется от $\mathbf{x_n}$ до $\mathbf{x_k}$ с шагом **dx**, аргумент **y** изменяется от $\mathbf{y_n}$ до $\mathbf{y_k}$ с шагом **dy**.

Для определения значений функции z при всех значениях аргументов **x** и **y** необходимо процесс вычислений организовать следующим образом. Зафиксировать начальное значение одного из аргументов, например **x**, и при этом значении организовать цикл изменений другой переменной **y** и вычислений соответствующих значений функции **z**. После выполнения этого цикла изменить на шаг значение переменной **x** и вновь перейти к полному циклу изменений переменной **y**. Таким образом, область цикла с параметром **y** будет вложена внутрь области цикла с параметром **x**. Схема алгоритма решения задачи приведена на рис. 6.

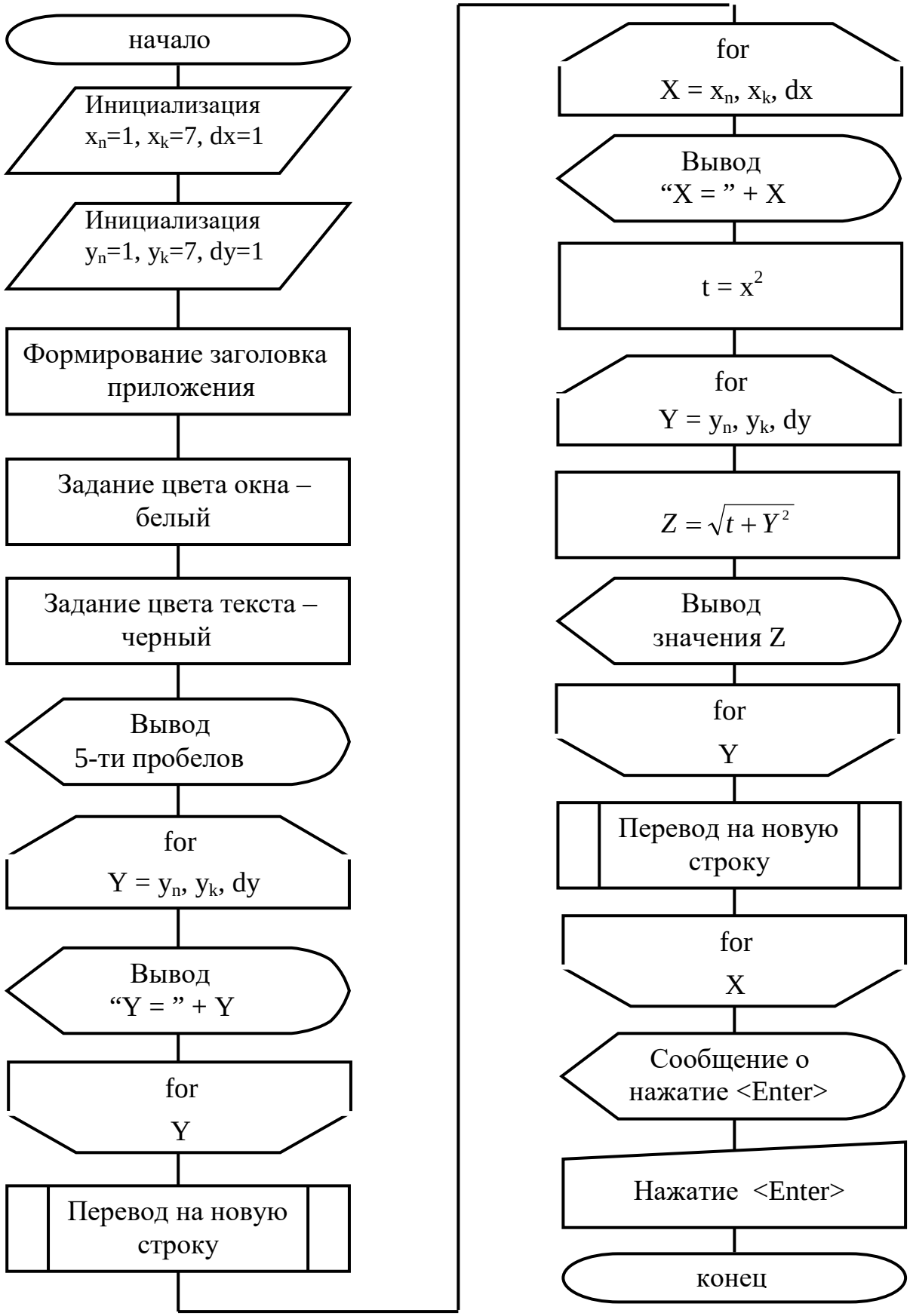

Рис. 6. Блок-схема алгоритма

Для сокращения объема вычислений величину $\mathbf{x}^2$ целесообразно определить вне цикла по **y**, так как значение ее не зависит от параметра **y**. Использование вспомогательной переменной **t** для вычисления значения $\mathbf{x}^2$ в данном примере иллюстрирует общее правило организации циклов: необходимо выносить из цикла все вычисления, которые не зависят от параметра этого цикла.

```
static void Main()
      {
       double X, xn=1, xk=7, dx=1;
       double Y, yn=1, yk=7, dy=1;
       double Z, t;
       Console.Title = "   Радиус окружности"; // заголовок приложения
       Console.BackgroundColor = ConsoleColor.White; // цвет окна - белый
       Console.Clear();
       Console.ForegroundColor = ConsoleColor.Black; // цвет текста - черный
       Console.Write("     ");
       for (Y = yn; Y <= yk; Y += dy) // заголовок таблицы: Y=1 Y=2 и т.д.
          {
           Console.Write("   Y=" + Y);
          }
       Console.Write('\n');
       for (X = xn; X <= xk; X += dx) // организация внешнего цикла по X
          {
           Console.Write("  X="+X);
           t = X * X;
           for (Y = yn; Y <= yk; Y += dy) // организация внутреннего цикла по Y
              {
               Z = Math.Sqrt(t + Y * Y); // радиус окружности
               Console.Write("  " + Z.ToString("f"));
              }
           Console.Write('\n'); // перевод на новую строку
          }
       Console.Write("Для завершения работы приложения нажмите <Enter>");
       Console.Read();
      }
```

Результат выполнения приложения «Радиус окружности», представлен на рис. 7.

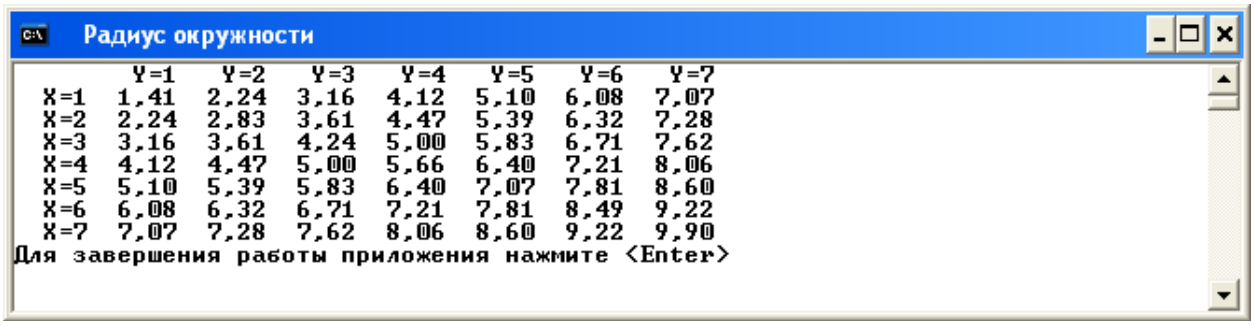

Рис. 7. Результат выполнения приложения «Радиус окружности»

2. Рабочее задание

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

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

$$W = a - \frac{1}{b}\ln\left|\frac{1+\cos(x)}{1-\cos(x)}\right| ,$$

где a, b – целочисленные переменные, x – значение угла в градусах, изменяющееся в интервале $[x_n, x_k]$ с шагом dx. Значения переменных a, b, x_n, x_k, dx вводятся с клавиатуры.

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

Задание 3. Разработать приложение с заголовком «Вложенный цикл», которое выводит на экран значение функции y = f(x), вычисляемой по формуле:

$$L = \sum_{k=0}^{5}\left(\frac{1+e^{-kx}}{2^k} + \begin{cases}\sqrt{1+x^k} & , если\ x > 0\\ \sqrt{\left|1+x^k\right|} & , если\ x <= 0\end{cases}\right),$$

где x – переменная, изменяющаяся в интервале $[x_n, x_k]$ с шагом dx. Значения переменных x_n, x_k, dx вводятся с клавиатуры.

3. Контрольные вопросы

1. В каких случаях используется оператор for?
2. Перечислите составные части оператора for.
3. Для чего используется в операторе for инициализация?
4. Для чего используется в операторе for выражение?
5. Для чего используется в операторе for модификация?
6. Опишите механизм действия оператора for.

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.