Лабораторная работа №8
Исследование возможностей интегрированной среды разработки Visual C# для создания приложений, использующих операторы управления break, continue, return, goto.

Цель работы – исследовать возможности интегрированной среды разработки Visual Studio 2010 и получить практические навыки по созданию приложений, использующие операторы управления break, continue, return, goto.

1. Теоретические сведения

В C# есть четыре оператора, изменяющих естественный порядок выполнения вычислений:

- оператор выхода из цикла **break**;
- оператор перехода к следующей итерации цикла **continue**;
- оператор возврата из функции **goto**;
- оператор безусловного перехода **return**.

1.1. Использование оператора break

Оператор **break** используется внутри операторов цикла и условных операторов (if, switch) для обеспечения перехода в точку программы, находящуюся непосредственно за оператором, внутри которого находится **break**. С помощью оператора **break** можно организовать немедленный выход из цикла, опустив выполнение кода, оставшегося в его теле, и проверку условного выражения. При обнаружении внутри цикла оператора **break** цикл завершается, а управление передается инструкции, следующей после цикла.

Пример 1. Нижеприведенный пример демонстрирует использование оператора **break** для “досрочного” прекращает перебор значений параметра цикла, когда его значение становится положительным. Результат решения представлен на рис. 1.

```
static void Main()
      {
        int a; // имя переменной, в которой хранится начальное значение a
        int b; // имя переменной, в которой хранится конечное значение b
        String Str;
        Console.Title = "   Перебор значений";
        Console.Write("Введите начальное значение диапазона a ");
        Str = Console.ReadLine();
        a = Convert.ToInt16(Str);
        Console.Write("Введите конечное значение диапазона b ");
        Str = Console.ReadLine();
        b = Convert.ToInt16(Str);
        for (int i = a; i < b; i++)
          {
           if (i > 0) break;
           Console.Write(i + " ");
          }
        Console.Write("Перебор завершен!\n");
        Console.Write("Для завершения работы приложения нажмите <Enter>");
        Console.Read();
      }
```

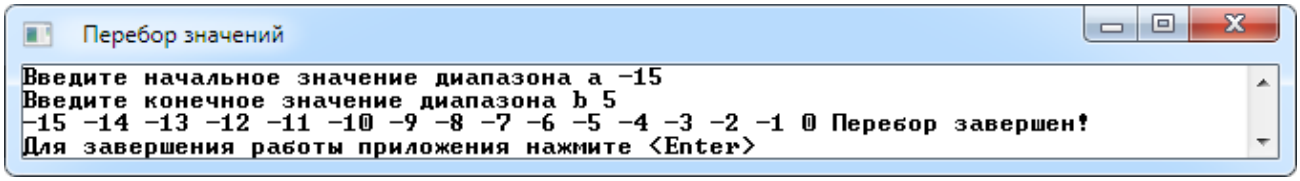

Рис.1. Результат выполнения приложения

1.2. Использование оператора continue

Помимо средства “досрочного” выхода из цикла, существует средство “досрочного” выхода из текущей его итерации. Этим средством является оператор **continue**, который принудительно выполняет переход к следующей итерации, пропуская все операции, оставшиеся до конца тела цикла.

Пример 2. Нижеприведенный пример демонстрирует использование оператора **continue** для “ускоренного” поиск четных чисел в заданном диапазоне. Результат решения представлен на рис. 2.

```
static void Main()
    {
      int a; // имя переменной, в которой хранится начальное значение a
      int b; // имя переменной, в которой хранится конечное значение b
      String Str;
      Console.Title = "   Поиск четных чисел";
      Console.Write("Введите начальное значение диапазона a ");
      Str = Console.ReadLine();
      a = Convert.ToInt16(Str);
      Console.Write("Введите конечное значение диапазона b ");
      Str = Console.ReadLine();
      b = Convert.ToInt16(Str);
      for (int i = a; i <= b; i++)
        {
         if ((i % 2)!=0) continue;
         Console.Write(i + " ");
        }
      Console.Write("Поиск завершен!\n");
      Console.Write("Для завершения работы приложения нажмите <Enter>");
      Console.Read();
    }
```

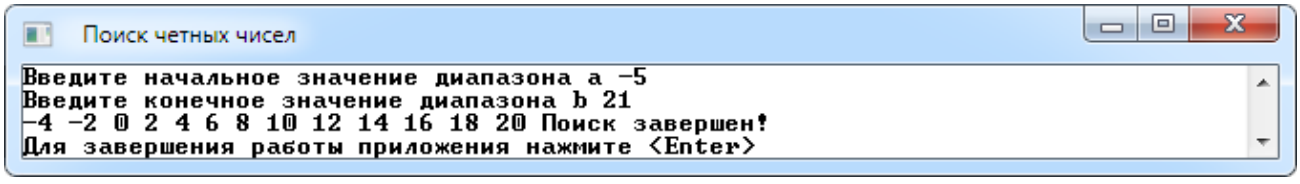

Рис.2. Результат выполнения приложения

1.3. Использование оператора goto

Оператор **goto** – это оператор безусловного перехода имеет формат:

goto <метка>;

Оператор **goto** требует наличие в программе метки. Метка – это действительный в C# идентификатор, за которым поставлено двоеточие. Метка должна находится в одном методе с оператором **goto**, который ссылается на эту метку. Оператор с меткой имеет формат:

<метка>: оператор;

Использование оператора безусловного перехода оправдано в двух случаях:

- принудительный выход вниз по тексту программы из нескольких вложенных циклов или переключателей;
- переход из нескольких мест метода (функции) в одно (например, если перед выходом из метода необходимо всегда выполнять какие-либо действия).

Не следует передавать управления внутрь операторов **if, switch** и циклов. Нельзя переходить внутрь блоков, содержащих инициализацию переменных, на операторы, расположенные после неё.

Пример 3. Нижеприведенный пример демонстрирует использование оператора **goto** для выхода из вложенных циклов. Результат выполнения приложения представлен на рис. 3.

```
static void Main()
    {
      Console.Title = "   Демонстрация работы оператора goto в циклах";
      int i=0, j=0, k=0;
      for (i = 0; i < 10; i++)
       {
        for (j = 0; j < 10; j++)
          {
           for (k = 0; k < 10; k++)
            {
              Console.Write("i, j, k:  " + i + " " + j + " " + k + "\n");
              if (k == 3) goto stop;
            }
          }
       }
    stop: Console.Write("Цикл прерван при i, j, k:  "+i+" "+j +" "+k+"\n");
      Console.Write("Для завершения работы приложения нажмите <Enter>");
      Console.Read();
    }
```

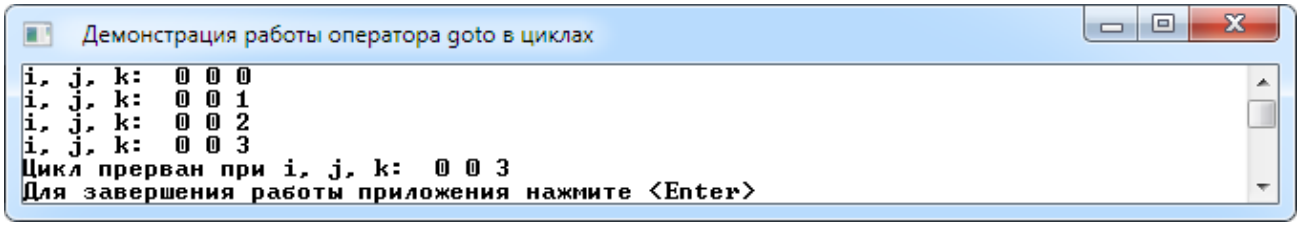

Рис.3. Результат выполнения приложения

Пример 4. Оператор **goto** можно использовать для перехода к **case-** или **default-**ветви внутри оператора **switch**. Нижеприведенный пример демонстрирует использование операторов **goto** и **switch**. Результат выполнения приложения представлен на рис. 4.

```
static void Main()
      {
        Console.Title = "   Демонстрация работы операторов goto и switch";
        for (int i = 1; i < 5; i++)
          {
           Console.Write("Значение i равно " + i + "\n");
           switch(i)
            {
              case 1: Console.Write("         работает ветвь 1\n");
                  goto case 3;
              case 2: Console.Write("         работает ветвь 2\n");
                  goto case 1;
              case 3: Console.Write("         работает ветвь 3\n");
                  goto default;
              default: Console.Write("         работает ветвь default\n");
                  break;
            }
           Console.Write("\n");
          }
        Console.Write("Для завершения работы приложения нажмите <Enter>");
        Console.Read();
      }
```

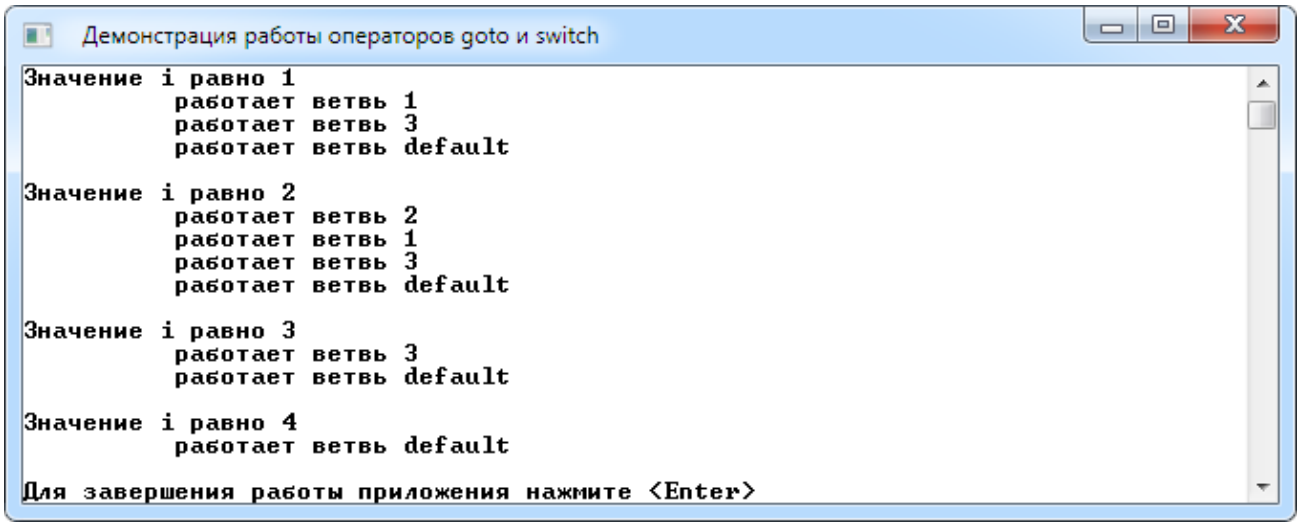

Рис.4. Результат выполнения приложения

1.4. Использование оператора return

Оператор возврата из метода **return** завершает выполнение метода (функции) и передает управление в точку его вызова. Вид оператора:

return [выражение];

Оператор можно использовать для возвращения значения, определенного выражением. Если тип возвращаемого значения описан как **void**, выражение должно отсутствовать.

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 «Наименьший делитель», которое находит наименьший делитель заданного числа (значение числа вводится с клавиатуры). При разработке приложения использовать оператор **break**.

Задание 3. Разработать приложение с заголовком «Действительные значения», с помощью которого на экран выводятся действительные значения функции

$$y = \frac{1}{\sqrt{a + \sin(x)}},$$

где x – переменная, изменяющаяся в интервале $[x_n, x_k]$ с шагом dx. Значение переменной a вводится с клавиатуры (значение a выбирается в интервале (-1, 1)). Если значение функции y – мнимое число, выдаётся соответствующее сообщение. Значения переменных x_n, x_k, dx вводятся с клавиатуры. При разработке приложения использовать оператор **continue**.

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

Задание 4. Разработать приложение с заголовком «Минимальное значение функции», позволяющее найти значение аргумента x, для которого значение функции **y = cos(x+b)** в интервале от **- π/2 до 3π/2** минимально. Значение переменной b вводится с клавиатуры (значение b выбирается в интервале [-1, 3]). При разработке приложения использовать оператор **goto**.

3. Контрольные вопросы

1. Перечислите операторы, с помощью которых можно изменять естественный порядок следования.
2. Дайте полную характеристику оператору **break** (назначение, синтаксис и примеры).
3. Дайте полную характеристику оператору **continue** (назначение, синтаксис и примеры).
4. Дайте полную характеристику оператору **goto** (назначение, синтаксис и примеры).
5. Дайте полную характеристику оператору **return** (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.