Лабораторная работа №12
Исследование возможностей интегрированной среды разработки Visual C# для создания приложений по обработке строк.

Цель работы – исследовать возможности интегрированной среды разработки Visual Studio 2010 и получить практические навыки по созданию приложений, обрабатывающие строки.

1. Теоретические сведения

Одним из самых важных типов данных C# является тип string. Он предназначен для определения и поддержки символьных строк. Так как в C# строки являются объектами (во других языках программирования строка представляет собой массив символов), то тип string — это ссылочный тип. При создании строкового литерала создается объект класса string. Например, в операторе

Console.WriteLine("Это текст");

строка "Это текст" в C# автоматически превращена в объект класса string.

1.1. Создание строк

Самый простой способ создать объект типа string — использовать строковый литерал, например, оператор:

string str = "Пример объявления строки с инициализацией";

объявляет переменную **str** ссылочной переменной типа string, которой присваивается ссылка на строковый литерал. В данном случае переменная **str** инициализируется последовательностью символов "Пример объявления строки с инициализацией".

Можно также создать string-объект из массива типа char, например

char[] arraych = {‘M’, ‘a’, ‘c’, ‘c’, ‘и’, ‘в’, ‘ ’, ‘c’, ‘и’, ‘м’, ‘в’, ’o’, ‘л’, ‘o’, ‘в’} ;

string str = new string(arraych);

После создания string-объект можно использовать везде, где разрешается использование строки символов, заключенной в кавычки. Например, string-объект можно использовать в качестве аргумента метода WriteLine(), как показано в следующем примере. Результат выполнения представлен на рис. 1.

```
char[] arraych = {'M', 'a', 'c', 'c', 'и', 'в', ' ', 'c', 'и', 'м', 'в', 'o', 'л', 'o', 'в'} ;
string str1 = new string(arraych);
string str2 = "Еще один объект типа string";
Console.WriteLine(str1) ;
Console.WriteLine(str2);
```

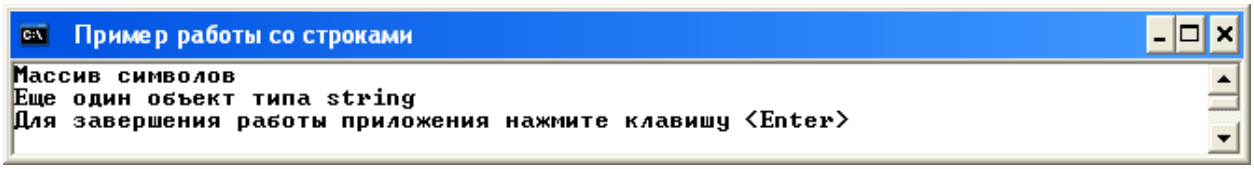

Рис. 1. Результат выполнения «Пример»

1.2. Работа со строками

Содержимое string-объектов неизменно, то есть последовательность символов, составляющих строку, изменить нельзя. Если требуется строка, которая должна представлять собой "модификацию" уже существующей строки, то необходимо создать

новую строку, которая содержит желаемые изменения. Класс string содержит ряд методов (некоторые из них показаны в табл. 1), которые предназначены для обработки строк.

Таблица 1. Методы обработки строк

Метод	Описание
static string Copy (string str)	Возвращает копию строки **str**
int CompareTo (string str)	Возвращает отрицательное значение, если вызывающая строка меньше строки **str**, положительное значение, если вызывающая строка больше строки **str** , и нуль, если сравниваемые строки равны
int IndexOf (string str)	Выполняет в вызывающей строке поиск подстроки, заданной параметром **str**. Возвращает индекс первого вхождения искомой подстроки или - 1 , если она не будет обнаружена
int LastIndexOf (string str)	Выполняет в вызывающей строке поиск подстроки, заданной параметром **str**. Возвращает индекс последнего вхождения искомой подстроки или - 1 , если она не будет обнаружена
string ToLower ()	Возвращает строчную версию вызывающей строки
string ToUpper()	Возвращает прописную версию вызывающей строки
string Substring(int start, int len)	Извлекает подстроку из string-объекта. Подстрока начинается с указанной позиции знака **start** и имеет указанную длину **len.**

Тип string также включает свойство Length, которое содержит длину строки. Чтобы получить значение отдельного символа строки, достаточно использовать индекс, например:

```
string str = "test";
Console.WriteLine(str[0]);
```

При выполнении этого фрагмента программы на экран будет выведен символ **t** (первый символ слова **"test"**).

Как и у массивов, индексация строк начинается с нуля. С помощью индекса нельзя присвоить символу внутри строки новое значение, индекс можно использовать только для получения символа.

Чтобы узнать, равны ли две строки, необходимо использовать оператор "==". В этом случае проверяется равенство содержимого двух строк. Тоже справедливо и в отношении оператора "**!** =". Что касается остальных операторов отношения (например, ">" или ">="), то они сравнивают ссылки так же, как и объекты других типов.

Пример 1. Нижеприведенный код приложения «Пример работы со строками» демонстрирует выполнение операций над строками. Результат выполнения приложения представлен на рис. 2.

```
static void Main(string[] args)
    {
      Console.Title = "  Пример работы со строками";
      Console.BackgroundColor = ConsoleColor.White;
      Console.Clear();
      Console.ForegroundColor = ConsoleColor.Black;
      string strl = "Первая строка";
```

```
  string str2 = string.Copy(strl);
  string str3 = "Третья строка";
  string strUp, strLow;
  int result, idx;
  Console.WriteLine("  strl: " + strl);
  Console.WriteLine("  Длина строки strl: " + strl.Length);
  strLow = strl.ToLower(); // Создаем строчную версию строки strl
  strUp = strl.ToUpper();   // Создаем прописную версию строки strl
  Console.WriteLine("  Строчная версия строки strl: " + strLow);
  Console.WriteLine("  Прописная версия строки strl: " + strUp);
  Console.WriteLine();
  Console.Write("  Отображаем strl посимвольно: ");
  for (int i=0; i < strl.Length; i++)
    Console.Write(strl[i]);  // Отображаем strl в символьном режиме
  Console.WriteLine("\n");
  // Сравниваем строки
  if (strl == str2) Console.WriteLine("  strl == str2");
   else Console.WriteLine("  strl != str2");
  if (strl == str3) Console.WriteLine("  strl == str3") ;
   else Console.WriteLine("  strl != str3");
  result = strl.CompareTo(str3);
  if (result == 0) Console.WriteLine("  strl и str3 равны.");
   else if (result < 0) Console.WriteLine("  strl меньше, чем str3");
         else Console.WriteLine("  strl больше, чем str3");
  Console.WriteLine();
  str2 = "Один Два Три Один"; // Присваиваем str2 новую строку
  // Поиск строк
  idx = str2.IndexOf("Один");
  Console.WriteLine("  Индекс первого вхождения подстроки Один: " + idx);
  idx = str2.LastIndexOf("Один");
  Console.WriteLine("  Индекс последнего вхождения подстроки Один: " +
                              idx);
  Console.WriteLine("\n  str2: " + str2);
  string str4 = str2.Substring(5, 7);
  Console.WriteLine("  Извлеченная строка из str2:  " + str4);
  Console.Write("\nДля завершения работы приложения нажмите
                        клавишу <Enter>");
  Console.Read();
 }
```

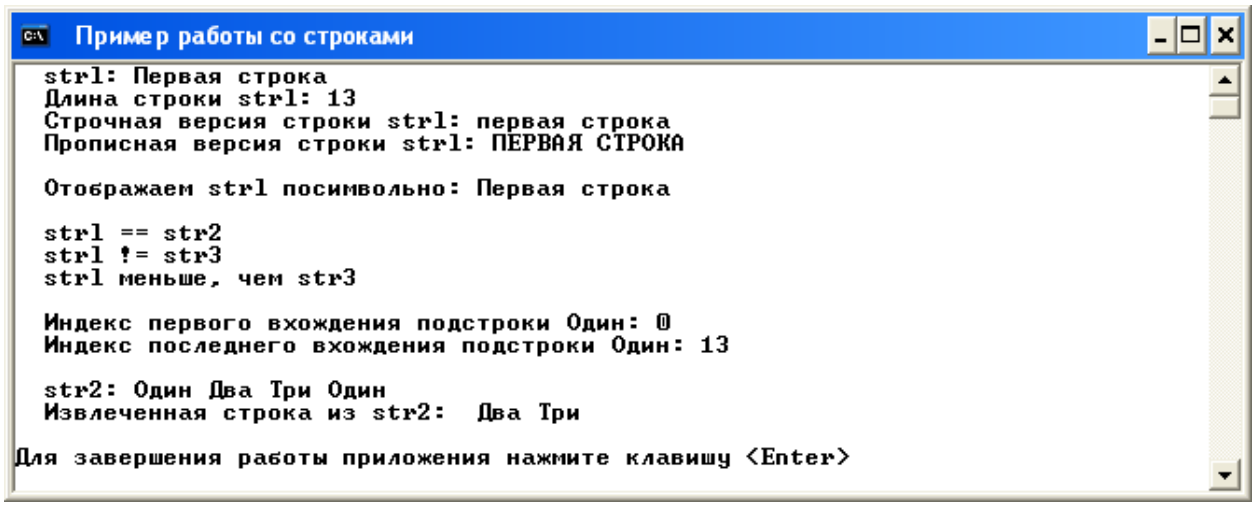

Рис. 2. Результат выполнения «Пример работы со строками»

С помощью оператора "+" можно конкатенировать (объединить) несколько строк, например, при выполнении нижеприведенного кода

```
string str1 = "Один";
string str2 = "Два";
string str3 = "Три";
string str4 = str1 + str2 + str3;
```

переменная **str4** инициализируется строкой **"ОдинДваТри"**.

1.3. Массивы строк

Подобно другим типам данных, строки могут быть собраны в массивы.

Пример 2. Нижеприведенный код приложения «Работа с массивами строк» демонстрирует работу с массивами строк. Результат выполнения приложения представлен на рис. 3.

```
static void Main(string[] args)
    {
      Console.Title = "  Работа с массивом строк";
      Console.BackgroundColor = ConsoleColor.White;
      Console.Clear();
      Console.ForegroundColor = ConsoleColor.Black;
      string[] str = { "Это", "очень", "простой", "текст!" };
      Console.WriteLine("  Исходный массив: ") ;
      for (int i=0; i < str.Length; i++)
        Console.Write(str[i] + " " ) ;
      Console.WriteLine("\n");
      str[1] = "тоже простой,";    // Изменяем 2 строку
      str[2] = "но измененный"; // Изменяем 3 строку
      Console.WriteLine("  Модифицированный массив: ") ;
      for (int i = 0; i < str.Length; i++ )
        Console.Write(str[i] + " ");
      Console.Write("\n\nДля завершения работы приложения нажмите
                     клавишу <Enter>");
      Console.Read();
    }
```

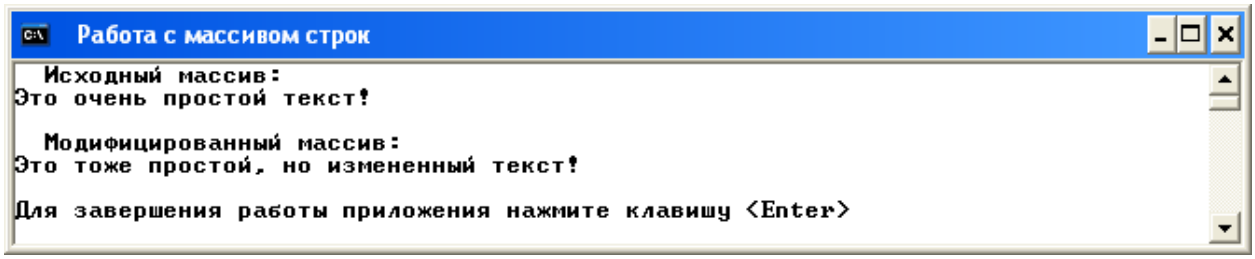

Рис. 3. Результат выполнения приложения «Работа с массивами строк»

1.4. Использование строк в switch-операторах

Для управления switch-операторами можно использовать string-объекты, причем это единственный тип, кроме типа int, который там допускается. Эта возможность в некоторых случаях облегчает обработку.

Пример 3. Нижеприведенный фрагмент кода приложения «Строки в switch-операторах» демонстрирует работу строк в switch-операторах для отображения цифрового эквивалента слов "один", "два" и "три":

```
string[] strs = { "один", "два", "три", "два", "один" };
        foreach (string s in strs)
        {
           switch (s)
           {
              case "один": Console.Write(1); break;
              case "два": Console.Write(2); break;
              case "три": Console.Write(3); break;
           }
        }
```

2. Рабочее задание

Задание 1. Разработать приложение с заголовком «Задание №1», с помощью которого целочисленное значение, вводимое с клавиатуры, можно отобразить с помощью слов. Например:

«Исходное число: 26404»

«Результат: два, шесть, четыре, нуль, четыре».

Задание 2. Разработать приложение с заголовком «Задание №2», с помощью которого можно извлечь из строки, вводимой с клавиатуры, слова, начинающиеся с гласной буквы. Слова, извлеченные из строки, разместить в массив слов и вывести на экран монитора.

Задание 3. Разработать приложение с заголовком «Задание №3», с помощью которого можно подсчитать количество символов в строке. Подсчет количества символов в строке оформить в виде отдельного метода, параметрами которого являются строка и массив значений количества символов. Исходные данные (строка) вводятся с клавиатуры. Результат вычислений выводятся на экран монитора из метода **Main()**.

3. Контрольные вопросы

1. Что такое строка в C#?
2. Какие существуют способы создания строк?

3. Перечислите основные методы работы со строками.
4. Что такое массив строк и как он описывается?
5. Как используются строки в операторе switch?

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. Онуфрей Ю.Є., Подоляка О.О. Методичні вказівки до лабораторних робіт «Програмування в середовищі С (С++)». – Харків, ХНАДУ, 2006. – 108с.