ПРАКТИЧЕСКАЯ РАБОТА № 5. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРОК

Цель практической работы: изучить правила работы с компонентом ListBox. Написать программу для работы со строками.

5.1. Тип данных string

Для хранения строк в языке C# используется тип string. Так, чтобы объявить (и, как правило, сразу инициализировать) строковую переменную, можно написать следующий код:

```
string a = "Текст";
string b = "строки";
```

Над строками можно выполнять операцию сложения – в этом случае текст одной строки будет добавлен к тексту другой:

```
string c = a + " " + b; // Результат: Текст строки
```

Тип string на самом деле является псевдонимом для класса String, с помощью которого над строками можно выполнять ряд более сложных операций. Например, метод IndexOf может осуществлять поиск подстроки в строке, а метод Substring возвращает часть строки указанной длины, начиная с указанной позиции:

```
string a = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int index = a.IndexOf("OP");  // Результат: 14 (счёт с 0)
string b = a.Substring(3, 5); // Результат: DEFGH
```

Если требуется добавить в строку специальные символы, это можно сделать с помощью escape-последовательностей, начинающихся с обратного слэша:

Escape-последовательность	Действие
\"	Кавычка
\\	Обратная косая черта
\n	Новая строка
\r	Возврат каретки
\t	Горизонтальная табуляция

5.2. Компонент ListBox

Компонент **ListBox** представляет собой список, элементы которого выбираются при помощи клавиатуры или мыши. Список элементов задается свойством **Items**. Items – это элемент, который имеет свои свойства и свои методы. Методы **Add**, **RemoveAt** и **Insert** используются для добавления, удаления и вставки элементов.

Объект **Items** хранит объекты, находящиеся в списке. Объект может быть любым классом – данные класса преобразуются для отображения в строковое представление методом ToString. В нашем случае в качестве объекта будут выступать строки. Однако, поскольку объект Items хранит объекты, приведённые к типу object, перед использованием необходимо привести их обратно к изначальному типу, в нашем случае string:

```
string a = (string)listBox1.Items[0];
```

Для определения номера выделенного элемента используется свойство **SelectedIndex**.

5.3. Порядок выполнения индивидуального задания

Задание: Написать программу подсчета числа слов в произвольной строке. В качестве разделителя может быть любое число пробелов. Для ввода строк использовать **ListBox**. Строки вводятся на этапе проектирования формы, используя окно свойств. Вывод результата организовать в метку **Label**.

Панель диалога будет иметь вид:

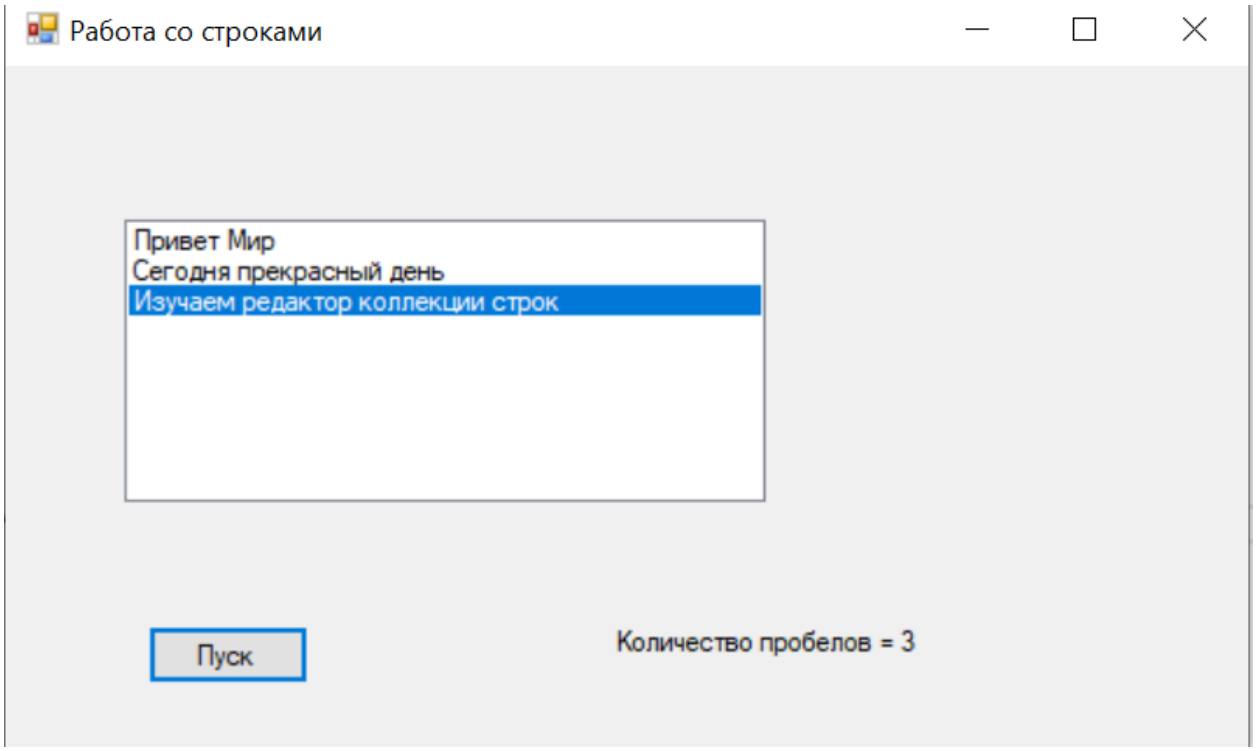

Рис. 5.1. Окно программы обработки строк

Текст обработчика нажатия кнопки «Пуск» приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
{
   // Получаем номер выделенной строки
   int index = listBox1.SelectedIndex;
   // Считываем строку в перменную str
   string str = (string)listBox1.Items[index];
   // Узнаем количество символов в строке
   int len = str.Length;
   // Считаем, что количество пробелов равно 0
   int count = 0;
   // Устанавливаем счетчик символов в 0
   int i = 0;
   //Организуем цикл перебора всех символов в строке
   while (i < len - 1)
   {
      // Если нашли пробел, то увеличиваем
      // счетчик пробелов на 1
      if (str[i] == ' ')
         count++;
      i++;
   }
   label1.Text = "Количество пробелов = " +
      count.ToString();
}
```

5.4. Индивидуальные задания

Во всех заданиях исходные данные вводить с помощью **ListBox**. Строки вводятся на этапе проектирования формы, используя окно свойств. Вывод результата организовать в метку **Label**.

1. Подсчитать, сколько раз среди символов заданной строки встречается буква «а».
2. Найти долю пробелов в строке *A*.
3. Заменить все буквы «а» на буквы «б» в заданной строке.
4. Из заданной строки получить новую, повторив каждый символ дважды.
5. Дано слово. Вывести слово, содержащее те же символы, но расположенные в обратном порядке.
6. Проверить, является ли заданное слово палиндромом.
7. Строка *X* состоит из нескольких предложений, каждое из которых кончается точкой, восклицательным или вопросительным знаком. Определить количество предложений в строке *X*.

8. Проверить правильность расстановки скобок в формуле. Расстановку считать правильной, если число открывающих скобок равно числу закрывающих скобок.
9. Проверить правильность расстановки скобок в формуле. Учитывать порядок скобок.
10. В заданной строке подсчитать количество букв латинского алфавита.
11. Подсчитать количество цифр в заданной строке.
12. Из заданной строки получить новую, удалив из нее все пробелы.
13. Из заданной строки получить новую, удалив все буквы латинского алфавита.
14. Подсчитать, сколько раз встречается в тексте заданный фрагмент.
15. Проверить, является ли частью данного слова слово «сок». Ответ должен быть «да» или «нет».