ЛАБОРАТОРНАЯ РАБОТА № 1. ПРОГРАММИРОВАНИЕ МЕТОДОВ С ИСПОЛЬЗОВАНИЕМ СТРОК

Цель лабораторной работы: изучить методы в языке C#, правила работы с символьными данными и с компонентом ListBox. Написать программу для работы со строками.

1.1. Методы

Метод – это элемент класса, который содержит программный код. Метод имеет следующую структуру:

```
[атрибуты] [спецификторы] тип имя ([параметры])
{
    Тело метода;
}
```

Атрибуты – это особые указания компилятору на свойства метода. Атрибуты используются редко.

Спецификаторы – это ключевые слова, предназначенные для разных целей, например:

- Определяющие доступность метода для других классов:
  - **`private`** – метод будет доступен только внутри этого класса
  - **`protected`** – метод будет доступен также дочерним классам
  - **`public`** – метод будет доступен любому другому классу, который может получить доступ к данному классу
- Указывающие доступность метода без создания класса
- Задающие тип

Тип определяет результат, который возвращает метод: это может быть любой тип, доступный в C#, а также ключевое слово `void`, если результат не требуется.

Имя метода – это идентификатор, который будет использоваться для вызова метода. К идентификатору применяются те же требования, что и к именам переменных: он может состоять из букв, цифр и знака подчёркивания, но не может начинаться с цифры.

Параметры – это список переменных, которые можно передавать в метод при вызове. Каждый параметр состоит из типа и названия переменной. Параметры разделяются запятой.

Тело метода – это обычный программный код, за исключением того, что он не может содержать определения других методов, классов, пространств имён и т. д. Если метод должен возвращать какой-то результат, то обязательно в конце должно присутствовать ключевое слово return с возвращаемым значением. Если возвращение результатов не нужно, то использование ключевого слова return не обязательно, хотя и допускается.

Пример метода, вычисляющего выражение:

```
public double Calc(double a, double b, double c)
{
    if (a > b)
        return Math.Sin(a) * Math.Cos(b);
    else
    {
        double k = Math.Tan(a * b);
        return k * Math.Exp(c / k);
    }
}
```

1.2. Перегрузка методов

Язык C# позволяет создавать несколько методов с одинаковыми именами, но разными параметрами. Компилятор автоматически подберёт наиболее подходящий метод при построении программы. Например, можно написать два отдельных метода возведения числа в степень: для целых чисел будет применяться один алгоритм, а для вещественных – другой:

```
/// <summary>
/// Вычисление X в степени Y для целых чисел
/// </summary>
private int Pow(int X, int Y)
{
    int b = 1;
    while (Y != 0)
        if (Y % 2 == 0)
        {
            Y /= 2;
            X *= X;
        }
        else
        {
            Y--;
            b *= X;
        }
    return b;
}

/// <summary>
/// Вычисление X в степени Y для вещественных чисел
/// </summary>
private double Pow(double X, double Y)
```

```
{
    if (X != 0)
        return Math.Exp(Y * Math.Log(Math.Abs(X)));
    else if (Y == 0)
        return 1;
    else
        return 0;
}
```

Вызывается такой код одинаково, разница лишь в параметрах – в первом случае компилятор вызовет метод Pow с целочисленными параметрами, а во втором – с вещественными:

```
Pow(3, 17);
Pow(3.0, 17.0);
```

1.3. Параметры по умолчанию

Язык C# начиная с версии 4.0 (Visual Studio 2010) позволяет задавать некоторым параметрам значения по умолчанию – так, чтобы при вызове метода можно было опускать часть параметров. Для этого при реализации метода нужным параметрам следует присвоить значение прямо в списке параметров:

```
private void GetData(int Number, int Optional = 5)
{
    Console.WriteLine("Number: {0}", Number);
    Console.WriteLine("Optional: {0}", Optional);
}
```

В этом случае вызывать метод можно следующим образом:

```
GetData(10, 20);
GetData(10);
```

В первом случае параметр Optional будет равен 20, так как он явно задан, а во втором будет равен 5, т.к. явно он не задан и компилятор берёт значение по умолчанию.

Параметры по умолчанию можно ставить только в правой части списка параметров, например, такая сигнатура метода компилятором принята не будет:

```
private void GetData(int Optional = 5, int Number)
```

1.4. Передача параметров по значению и по ссылке

Когда параметры передаются в метод обычным образом (без дополнительных ключевых слов `ref` и `out`), любые изменения параметров внутри метода не влияют на его значение в основной программе. Предположим, у нас есть следующий метод:

```
private void Calc(int Number)
{
    Number = 10;
}
```

Видно, что внутри метода происходит изменение переменной Number, которая была передана как параметр. Попробуем вызвать метод:

```
int n = 1;
Calc(n);
Console.WriteLine(n);
```

На экране появится число 1, то есть, не смотря на изменение переменной в методе Calc, значение переменной в главной программе не изменилось. Это связано с тем, что при вызове метода создаётся *копия* переданной переменной, именно её изменяет метод. При завершении метода значение копий теряется. Такой способ передачи параметра называется *передачей по значению*.

Чтобы метод мог изменять переданную ему переменную, её следует передавать с ключевым словом ref – оно должно быть как в сигнатуре метода, так и при вызове:

```
private void Calc(ref int Number)
{
    Number = 10;
}
int n = 1;
Calc(ref n);
Console.WriteLine(n);
```

В этом случае на экране появится число 10: изменение значения в методе сказалось и на главной программе. Такая передача метода называется *передачей по ссылке*, т.е. передаётся уже не копия, а ссылка на реальную переменную в памяти.

Если метод использует переменные по ссылке только для возврата значений и ему не важно что в них было изначально, то можно не инициализировать такие переменные, а передавать их с ключевым словом

out. Компилятор понимает, что начальное значение переменной не важно и не ругается на отсутствие инициализации:

```
private void Calc(out int Number)
{
    Number = 10;
}
int n; // Ничего не присваиваем!
Calc(out n);
```

1.5. Тип данных `string`

Для хранения строк в языке C# используется тип `string`. Для того, 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

1.6. 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**.

1.7. 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:

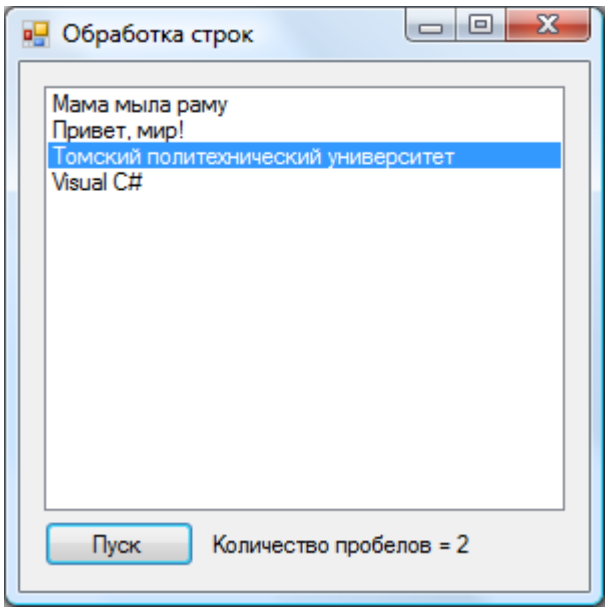

Рис. 1.1. Окно программы обработки строк

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();
}
```

1.8. Индивидуальные задания

Во всех заданиях исходные данные вводить с помощью **ListBox**. Вывод результата организовать в метку **Label**. Разработать метод класса **Form**, реализующий задание.

1. Дана строка, состоящая из групп нулей и единиц. Посчитать количество нулей и единиц.
2. Посчитать в строке количество слов.
3. Найти количество знаков препинания в исходной строке.
4. Дана строка символов. Вывести цифры, содержащиеся в строке.
5. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести количество четных чисел в этой строке.
6. Поменять местами символы с четными и нечетными номерами в заданной строке.
7. Дана строка символов. Вывести количество строчных русских букв, входящих в эту строку.
8. Дана строка символов. Вывести на экран только строчные русские буквы, входящие в эту строку.

9. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. В каждом слове заменить первую букву на прописную.

10. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Удалить первую букву в каждом слове.

11. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами *i*- и *j*-ю буквы в каждом слове. Для ввода *i* и *j* на форме добавить свои поля ввода.

12. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами первую и последнюю буквы каждого слова.

13. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Заменить все буквы латинского алфавита на знак ‘+’.

14. Дана строка символов, содержащая некоторый текст на русском языке. Заменить все большие буквы буквы ‘А’ на символ ‘*’.

15. Дана строка символов, содержащая некоторый текст. Проверить, является ли данный текст палиндромом, т.е. читается ли он слева направо так же, как и справа налево (например, «А роза упала на лапу азорА»).

16. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Сформировать новую строку, состоящую из чисел длин слов в исходной строке.

17. Дана строка символов. Сформировать из нее строку, в которой символы следуют в порядке возрастания их кодов.

18. Дана строка, содержащая слова, разделенные одиночными пробелами. Сформировать строку, в которой количество пробелов перед каждым словом будет равно его длине.

19. Дана строка, содержащая слова, разделенные пробелами. Вывести слова в порядке возрастания их длины.

20. Дана строка слов, разделенных группами пробелов. Вывести последовательность пар чисел «длина слова – длина группы пробелов».

21. Дана строка. Подсчитать и вывести частоту появления символов.

22. Дана строка, определить наибольшую длину последовательности, включающей заданные символы.

23. Найти и вывести все вещественные числа, содержащиеся в заданной строке.

24. Все подпоследовательности символов заданной строки, коды которых возрастают на единицу, заменить на тройки:
«первый символ» «–» «последний символ».

25. Дана строка, содержащая слова, разделенные пробелами. Вывести все слова, начинающиеся и заканчивающиеся одним и тем же символом без учета регистра.