МЕТОДЫ

**Цель лабораторной работы**: научиться работать с методами, написать программу с использованием методов.

**Задание:** написать метод, который подсчитывает количество знаков препинания в переданной строке. С его помощью обработать пять разных строк и отобразить результаты на экране.

**Теория:**

**Общие понятия**

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

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

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

1. 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; 

2. указывающие доступность метода без создания класса; 

3. 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 подчеркивания, 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 имен 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.

**Перегрузка методов**

Язык C# позволяет создавать несколько методов с одинаковыми именами, но разными параметрами. Компилятор автоматически подберет 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.

**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)

{

MessageBox.Show("Number: {0}", Number);

MessageBox.Show("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 берет 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)

**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);  MessageBox.Show(n.ToString());

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

Чтобы метод мог изменять переданную ему переменную, ее 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);  MessageBox.Show(n.ToString());

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. передается уже не копия, а ссылка на реальную переменную в памяти.

Если метод использует переменные по ссылке только для возврата значений и не имеет значения, 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);

**Код программы:**

using System;

using System.Windows.Forms;

namespace PP\_LR4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public string Znaki (string stroka)

{

int kol = 0;

for(int i=0; i<stroka.Length; i++)

{

switch(stroka[i])

{

case '.':

case ',':

case ':':

case ';':

case '-':

case '!':

case '?':

{

kol++;

break;

}

default: break;

}

}

stroka = kol.ToString();

return stroka;

}

private void button1\_Click(object sender, EventArgs e)

{

listBox1.Items.Clear();

listBox1.Items.Add("В первой строке " + Znaki(textBox1.Text) + " знаков препинания.");

listBox1.Items.Add("Во второй строке " + Znaki(textBox2.Text) + " знаков препинания.");

listBox1.Items.Add("В третей строке " + Znaki(textBox3.Text) + " знаков препинания.");

listBox1.Items.Add("В четвертой строке " + Znaki(textBox4.Text) + " знаков препинания.");

listBox1.Items.Add("В пятой строке " + Znaki(textBox5.Text) + " знаков препинания.");

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

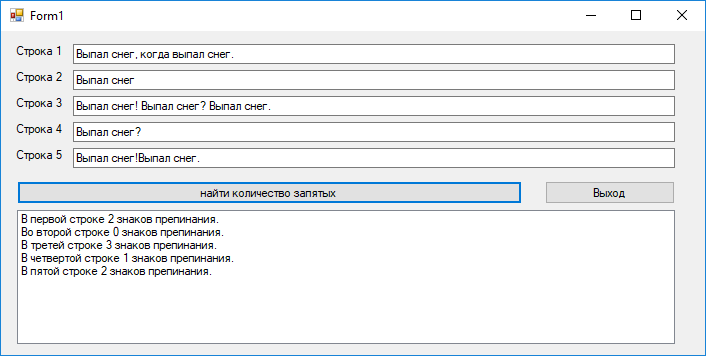
****

Рис.1 Работа программы

**Вывод:**

Научились работать с методами, написали программу с использованием методов.