ПРАКТИЧЕСКАЯ РАБОТА № 10

(4 академических часа)

Тема: Модульное программирование. Методы в C#.

Цель задания:

Получение практических навыков по разработке методов.

Теоретическая часть.

Методы содержат собой набор операторов, которые выполняют определенные действия. Общее определение методов выглядит следующим образом:

```
[модификаторы] тип_возвращаемого_значения название_метода ([параметры])
{
   // тело метода
}
```

Модификаторы и параметры необязательны.

По умолчанию консольная программа на языке C# должна содержать как минимум один метод - метод Main, который является точкой входа в приложение:

```
static void Main(string[] args)
{

}
```

Ключевое слово static является модификатором. Далее идет тип возвращаемого значения. В данном случае ключевое слово void указывает на то, что метод ничего не возвращает. Далее идет название метода - Main и в скобках параметры - string[] args. И в фигурные скобки заключено тело метода - все действия, которые он выполняет. В данном случае метод Main пуст, он не содержит никаких операторов и по сути ничего не выполняет.

Определим еще пару методов:

```
using System;
namespace HelloApp
{
   class Program
   {
      static void Main(string[] args)
      {

      }

      static void SayHello()
      {
         Console.WriteLine("Hello");
      }
      static void SayGoodbye()
      {
         Console.WriteLine("GoodBye");
      }
   }
}
```

Оба метода определены в рамках класса Program, они имеют модификатор static, а в качестве возвращаемого типа для них определен тип void. То есть данные методы ничего

не возвращают, просто производят некоторые действия. И также оба метода не имеют никаких параметров, поэтому после названия метода указаны пустые скобки.

Чтобы использовать методы SayHello и SayGoodbye в программе, необходимо вызвать их в методе Main. Для вызова метода указывается его имя, после которого в скобках идут значения для его параметров (если метод принимает параметры):

```
название_метода (значения_для_параметров_метода);
```

```
...
    static void Main(string[] args)
    {
       SayHello();
       SayGoodbye();

       Console.ReadKey();
    }
...
```

Метод может возвращать значение, какой-либо результат. Если метод имеет любой другой тип, отличный от void, то такой метод обязан вернуть значение этого типа. Для этого применяется оператор *return*, после которого идет возвращаемое значение:

```
return возвращаемое значение;
```

Например:

```
static string GetHello()
{
   return "Hello";
}
```

После оператора return также можно указывать сложные выражения, которые возвращают определенный результат.

```
static int GetSum()
{
   int x = 2;
   int y = 3;
   return x + y;
}
```

Оператор return не только возвращает значение, но и производит выход из метода.

Пример.

Необходимо **с**оздать метод *Sum()*, который принимает два целочисленных аргумента и суммирует их.

1. Запустите Visual Studio.
2. Создайте консольное приложение.
3. В главной функции запросите пользователя ввести два числа:

```
...
Console.Writeline( “Введите два числа”);
int a = Convert.ToInt32(Console.Readline());
int b = Convert.ToInt32(Console.Readline());
...
```

4. Поместите курсор после закрывающей фигурной скобки функции Main() и нажмите клавишу enter, т.к. нельзя помещать методы/функции внутри другой функции.
5. Объявите новый метод Sum(), который будет использоваться для суммы переданных в него значений (метод не возвращает никакого значения основной программе, поэтому необходимо использовать ключевое слово *void*):

```
...
static void Sum (int first, int second);
{
int Sum = first + second;
Console.WriteLine($ “Сумма {first}  + {second} = {Sum} “);
}
...
```

6. Вызов метода:

```
static void Main(string[] args)
      {
         ...
         Sum(a, b);
      }
```

7. Изменим этот метод так, чтобы он возвращал результат вызывающему методу (основному методу, откуда он был вызван):

```
...
static int Sum (int first, int second);
{
int Sum = first + second;
return Sum;
...
```

Необходимо изменить и способ вызова метода: объявите целочисленную переменную (в функции Main()), чтобы получить возвращаемое значение:

```
static void Main(string[] args)
      {
         ...
        int Result=Sum(a, b);
       Console.WriteLine($ “Сумма {first}  + {second} = {Result} “);

      }
```

Задание.

Разработать консольное приложение, осуществляющее работу со строковыми данными. В программе должно быть меню вида:

```
Выберите пункт меню:
1 – Ввод данных
2 – Просмотр данных
3 – Обработка
4 – Выход
```

В программе при выборе пункта меню загружается соответствующий метод.
В метод обработки необходимо передать в качестве параметров исходную строку. Метод просмотра в зависимости от передаваемых входных параметров реализует либо просмотр исходной строки, либо результат преобразования строки. Выход из программы осуществляется при выборе соответствующего пункта меню.

Варианты заданий:

1. Дана символьная строка. Заменить все символы '!' точками, кроме первого и вывести полученную строку.
2. Дана символьная строка. Определить, есть ли в данной строке два любых одинаковых символа, и вывести соответствующее сообщение.
3. Дана символьная строка и слово, состоящее из четырех символов. Определить, есть ли в данной строке все буквы данного слова.
4. Дана символьная строка. Получить новую строку, взяв из данной все символы до первого двоеточия и после последнего. Если двоеточие отсутствует или встречается в строке только один раз, то вывести соответствующее сообщение.
5. Дана символьная строка. Получить новую строку, взяв из данной все символы, находящиеся между первой открывающейся скобкой и последней закрывающейся (если какие-либо скобки отсутствует, то вывести соответствующее сообщение).
6. Дана символьная строка. Заменить все последовательности символов 'on' на 'online' и вывести новую строку (если искомой последовательности в строке нет, то вывести соответствующее сообщение).
7. Дана символьная строка. Слово - последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить количество слов в данной строке.
8. Дана символьная строка. Слово - последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить длину самого короткого слова.
9. Дана символьная строка. Слово - последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить длину самого короткого слова.
10. Дана символьная строка. Слово - последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить количество слов заданной длины.
11. Дана символьная строка. Слово - последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить количество и вывести все самые длинные слова.
12. Дана символьная строка и натуральное число N. Слово - последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить длину слова, стоящего на N-ом месте и вывести все слова, состоящие из такого же количества символов, что и найденное слово. Если N больше количества слов в предложении, то вывести соответствующее сообщение.
13. Дана символьная строка и символ. Слово - последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить количество слов в строке, оканчивающихся на заданный символ.
14. Дана строка символов. Определить количество букв 'o' между самой левой открывающейся скобкой и самой правой закрывающейся скобкой (если какие-либо скобки отсутствует, то вывести соответствующее сообщение).
15. Дана символьная строка. Подсчитать наибольшее количество букв 'a', идущих в ней подряд.

Контрольные вопросы

1. Для каких целей предназначены спецификаторы в структуре описания метода?
2. Какие требования предъявляются к имени метода?
3. Какое отличие между параметрами-переменными и параметрами-значениями?
4. Назначение оператора Return.
5. Формальные параметры и фактические параметры: назначение и различия.