

**УО «МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ А.А. КУЛЕШОВА»  
СОЦИАЛЬНО-ГУМАНИТАРНЫЙ КОЛЛЕДЖ**



**Дисциплина  
«Конструирование программ и языки программирования»**

**Разработка программ с использованием символов и строк**

Методические рекомендации к лабораторной работе № 7  
(4 часа)

Могилев 2018

Понятия «Символ», «Строка». Методические указания по лабораторной работе № 7 по дисциплине «Конструирование программ и языки программирования». Для учащихся 3 курса очной формы обучения специальности 2–40 01 01 «Программное обеспечение информационных технологий».

## Оглавление

1 Цель работы .....	4
2 Ход работы.....	5
3 Краткие теоретические сведения .....	6
3.1 Спецсимволы .....	6
3.2 Методы (функции) класса String для работы со строками в Си-шарп.....	6
3.2.1. Как проверить, пуста ли строка? .....	6
3.2.2. Как проверить, является ли одна строка "больше" другой? .....	7
3.2.3. Как перевести всю строку в верхний/нижний регистр? .....	7
3.2.4. Как проверить, содержит ли строка подстроку? .....	8
3.2.5. Как найти индекс первого символа подстроки, которую содержит строка? .....	8
3.2.6. Как узнать, начинается/заканчивается ли строка указанной подстрокой? .....	8
3.2.7. Как вставить подстроку в строку, начиная с указанной позиции? .....	9
3.2.8. Как обрезать строку, начиная с указанной позиции? .....	9
3.2.9. Как получить подстроку из строки, начиная с указанной позиции? .....	9
3.2.10. Как заменить в строке все подстроки указанной новой подстрокой? .....	10
3.2.11. Как преобразовать строку в массив символов? .....	10
3.2.12. Как разбить строку по указанному символу на массив подстрок? .....	10
3.2.13. Как соединить строку по указанному символу на массив подстрок? .....	10
3.2.14. Неизменяемые строки.....	11
3.3 Класс StringBuilder .....	11
3.3.1 Объявление строк. Конструкторы класса StringBuilder.....	11
3.3.2 Операции над строками.....	12
3.3.3 Основные методы и свойства.....	13
3.3.4 Емкость буфера .....	14
4 Задания .....	15
5 Контрольные вопросы .....	18

## **1 Цель работы**

выработать умение разрабатывать программы с использованием символов и строк.

## **2 Ход работы**

1. Изучение теоретического материала.
2. Выполнение практических индивидуальных заданий по вариантам (вариант уточняйте у преподавателя).
3. Оформление отчета.
  - 3.1. Отчет оформляется индивидуально каждым студентом. Отчет должен содержать задание, алгоритм и листинг программы.
  - 3.2. Отчет по лабораторной работе выполняется на листах формата А4. В состав отчета входят:
    - 1) титульный лист;
    - 2) цель работы;
    - 3) текст индивидуального задания;
    - 4) выполнение индивидуального задания.
4. Контрольные вопросы.

### 3 Краткие теоретические сведения

Строки в Си-шарп – это объекты класса `String`, значением которых является текст. Для работы со строками в этом классе определено множество методов (функций) и в этом уроке мы рассмотрим некоторые из них.

Чтобы использовать строку, ее нужно сначала создать – присвоить какое-либо значение, иначе мы получим ошибку: "Использование локальной переменной "[имя переменной]", которой не присвоено значение". Объявим простую строку и выведем ее на экран:

```
static void Main(string[] args)
{
    string s = "Hello, World!";
    Console.WriteLine(s);
}
static void Main(string[] args)
{
    string s;
    Console.WriteLine(s); // ошибка, строка не создана
}
```

Для объединения (конкатенации) строк используется оператор "+".

```
string s = "Hello," + " World!";
```

Оператор "[ ]" используется для доступа (только чтение) к символу строки по индексу:

```
string s = "Hello, World!";
char c = s[1]; // 'e'
```

Свойство `Length` возвращает длину строки.

#### 3.1 Спецсимволы

Символ "\" является служебным, поэтому, чтобы использовать символ обратного слэша необходимо указывать его дважды "\\".

Символ табуляции – "\t"

Символ перевода строки – "\r\n"

Двойные кавычки – "\""

#### 3.2 Методы (функции) класса `String` для работы со строками в Си-шарп

##### 3.2.1. Как проверить, пуста ли строка?

Метод `IsNullOrEmpty()` возвращает `True`, если значение строки равно `null`, либо когда она пуста (значение равно `""`):

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

```

string s1 = null, s2 = "", s3 = "Hello";
String.IsNullOrEmpty(s1); // True
String.IsNullOrEmpty(s2); // True
String.IsNullOrEmpty(s3); // False
}

```

Метод `IsNullOrWhiteSpace()` работает как и метод `IsNullOrEmpty()`, только возвращает `True` еще и тогда, когда строка представляет собой набор символов пробела и/или табуляции ("`\t`"):

```

static void Main(string[] args)
{
    string s1 = null, s2 = "\t", s3 = " ", s4 = "Hello";
    String.IsNullOrEmpty(s1); // True
    String.IsNullOrEmpty(s2); // True
    String.IsNullOrEmpty(s3); // True
    String.IsNullOrEmpty(s4); // False
}

```

### 3.2.2. Как проверить, является ли одна строка "больше" другой?

Для сравнения строк используется метод `Compare()`. Суть сравнения строк состоит в том, что проверяется их отношение относительно алфавита. Строка "a" "меньше" строки "b", "bb" "больше" строки "ba". Если обе строки равны – метод возвращает "0", если первая строка меньше второй – "-1", если первая больше второй – "1":

```

static void Main(string[] args)
{
    String.Compare("a", "b"); // возвращает -1
    String.Compare("a", "a"); // возвращает 0
    String.Compare("b", "a"); // возвращает 1
    String.Compare("ab", "abc"); // возвращает -1
    String.Compare("Romania", "Russia"); // возвращает -1
    String.Compare("Rwanda", "Russia"); // возвращает 1
    String.Compare("Rwanda", "Romania"); // возвращает 1
}

```

Чтобы игнорировать регистр букв, в метод нужно передать, как третий аргумент `true`.

```

String.Compare("ab", "Ab"); // возвращает -1
String.Compare("ab", "Ab", true); // возвращает 0

```

### 3.2.3. Как перевести всю строку в верхний/нижний регистр?

Для этого используются методы `ToUpper()` и `ToLower()`:

```

static void Main(string[] args)
{

```

```

string s = "Hello, World";
Console.WriteLine(s.ToUpper()); // выводит "HELLO, WORLD"
Console.WriteLine(s.ToLower()); // выводит "hello, world"
Console.ReadLine();
}

```

### 3.2.4. Как проверить, содержит ли строка подстроку?

Для проверки содержания подстроки строкой используется метод `Contains()`. Данный метод принимает один аргумент – подстроку. Возвращает `True`, если строка содержит подстроку, в противном случае – `False`. Пример:

```

static void Main(string[] args)
{
    string s = "Hello, World";

    if (s.Contains("Hello"))
        Console.WriteLine("Содержит");
    Console.ReadLine();
}

```

Данная программа выводит слово "Содержит", так как "Hello, World" содержит подстроку "Hello".

### 3.2.5. Как найти индекс первого символа подстроки, которую содержит строка?

Метод `IndexOf()` возвращает индекс первого символа подстроки, которую содержит строка. Данный метод принимает один аргумент – подстроку. Если строка не содержит подстроки, метод возвращает `-1`. Пример:

```

static void Main(string[] args)
{
    string s = "Hello, World";
    Console.WriteLine(s.IndexOf("H")); // 0
    Console.WriteLine(s.IndexOf("World")); // 7
    Console.WriteLine(s.IndexOf("Zoo")); // -1
    Console.ReadLine();
}

```

### 3.2.6. Как узнать, начинается/заканчивается ли строка указанной подстрокой?

Для этого используются соответственно методы `StartsWith()` и `EndsWith()`, которые возвращают логическое значение. Пример:

```

static void Main(string[] args)
{
    string s = "Hello, World";
    Console.WriteLine(s.StartsWith("Hello")); // True
    Console.WriteLine(s.StartsWith("World")); // False
}

```



```

Console.WriteLine(s.EndsWith("World")); // True
Console.ReadLine();
}

```

### 3.2.7. Как вставить подстроку в строку, начиная с указанной позиции?

Метод `Insert()` используется для вставки подстроки в строку, начиная с указанной позиции. Данный метод принимает два аргумента – позиция и подстрока. Пример:

```

static void Main(string[] args)
{
    string s = "Hello World";
    Console.WriteLine(s.Insert(5, ",")); // вставляет запятую на 5 позицию
    Console.ReadLine();
}

```

### 3.2.8. Как обрезать строку, начиная с указанной позиции?

Метод `Remove()` принимает один аргумент – позиция, начиная с которой обрезается строка:

```

static void Main(string[] args)
{
    string s = "Hello, World";
    Console.WriteLine(s.Remove(5)); //удаляем все символы, начиная с
    //5 позиции, на экран выведется "Hello"
    Console.ReadLine();
}

```

В метод `Remove()` можно передать и второй аргумент – количество обрезаемых символов. `Remove(3, 5)` – удалит из строки пять символов начиная с 3-го.

### 3.2.9. Как получить подстроку из строки, начиная с указанной позиции?

Для этого используется метод `Substring()`. Он принимает один аргумент – позиция, с которой будет начинаться новая подстрока:

```

static void Main(string[] args)
{
    string s = "Hello, World";
    Console.WriteLine(s.Substring(7)); // получаем строку начиная с
    //7 позиции, выведет "World"
    Console.ReadLine();
}

```

В метод `Substring()`, как в метод `Remove()` можно передать и второй аргумент – длина подстроки. `Substring(3, 5)` – возвратит подстроку длиной в 5 символов начиная с 3-й позиции строки.

### 3.2.10. Как заменить в строке все подстроки указанной новой подстрокой?

Метод `Replace()` принимает два аргумента – подстрока, которую нужно заменить и новая подстрока, на которую будет заменена первая:

```
static void Main(string[] args)
{
    string s = "Hello, World, Hello";
    Console.WriteLine(s.Replace("Hello", "World"));
    //выведет "World, World, World"
    Console.ReadLine();
}
```

### 3.2.11. Как преобразовать строку в массив символов?

Метод `ToCharArray()` возвращает массив символов указанной строки:

```
static void Main(string[] args)
{
    string s = "Hello, World";
    char[] array = s.ToCharArray(); // элементы массива - 'H', 'e', 'l', 'l'...
}
```

### 3.2.12. Как разбить строку по указанному символу на массив подстрок?

Метод `Split()` принимает один аргумент – символ, по которому будет разбита строка. Возвращает массив строк. Пример:

```
static void Main(string[] args)
{
    string s = "Arsenal,Milan,Real Madrid,Barcelona";
    string[] array = s.Split(',');
    // элементы массива - "Arsenal", "Milan", "Real Madrid", "Barcelona"
}
```

### 3.2.13 Как соединить строку по указанному символу на массив подстрок?

Сцепляет указанные элементы массива строк, помещая между ними заданный разделитель.

```
public static string Join (string separator, string[] value, int
startIndex, int count);
```

**separator**     [String](#) Строка для использования в качестве разделителя. separator включается в возвращаемую строку, только если в value более одного элемента.

**value**           [String\[\]](#) Массив, содержащий элементы, которые требуется сцепить.

**startIndex**     [Int32](#) Первый используемый элемент массива value.

**count**           [Int32](#) Число используемых элементов массива value.

Пример:

```
// Sample for String.Join(String, String[], int int)
using System;

class Sample {
    public static void Main() {
        String[] val = {"apple", "orange", "grape", "pear"};
        String sep = ", ";
        String result;

        Console.WriteLine("sep = '{0}'", sep);
        Console.WriteLine("val[] = {{ '{0}' '{1}' '{2}' '{3}' }}", val[0],
val[1], val[2], val[3]);
        result = String.Join(sep, val, 1, 2);
        Console.WriteLine("String.Join(sep, val, 1, 2) = '{0}'", result);
    }
}
/*
This example produces the following results:
sep = ', '
val[] = {'apple' 'orange' 'grape' 'pear'}
String.Join(sep, val, 1, 2) = 'orange, grape'
*/
```

### 3.2.14. Неизменяемые строки

Стоит знать, что объекты класса `String` представляют собой неизменяемые (`Immutable`) последовательности символов `Unicode`. Когда вы используете любой метод по изменению строки (например `Replace()`), он возвращает новую измененную копию строки, исходные же строки остаются неизменными. Так сделано потому, что операция создания новой строки гораздо менее затратна, чем операции копирования и сравнения, что повышает скорость работы программы. В Си-шарп также есть класс `StringBuilder`, который позволяет изменять строки.

## 3.3 Класс `StringBuilder`

Класс `StringBuilder` - строитель строк. Класс `string` не позволяет изменять существующие объекты. Строковый класс `StringBuilder` позволяет компенсировать этот недостаток. Этот класс принадлежит к изменяемым классам и его можно найти в пространстве имен `System.Text`.

### 3.3.1 Объявление строк. Конструкторы класса `StringBuilder`

Объекты этого класса объявляются с явным вызовом конструктора класса. Поскольку специальных констант этого типа не существует, то вызов конструктора для инициализации объекта просто необходим. Конструктор класса перегружен, и наряду с конструктором без параметров, создающим пустую строку, имеется набор конструкторов, которым можно передать две группы параметров. Первая группа позволяет задать строку или подстроку, значением которой будет инициализироваться создаваемый объект класса `StringBuilder`. Вторая группа параметров позволяет задать емкость объекта - объем памяти, отводимой данному экземпляру класса `StringBuilder`. Каждая из этих групп не является обязательной и может быть опущена. Примером может служить кон-

структор без параметров, который создает объект, инициализированный пустой строкой, и с некоторой емкостью, заданной по умолчанию, значение которой зависит от реализации. Пример синтаксиса трех конструкторов:

- `public StringBuilder (string str, int cap)`. Параметр `str` задает строку инициализации, `cap` - емкость объекта;
- `public StringBuilder (int curcap, int maxcap)`. Параметры `curcap` и `maxcap` задают начальную и максимальную емкость объекта;
- `public StringBuilder (string str, int start, int len, int cap)`. Параметры `str`, `start`, `len` задают строку инициализации, `cap` - емкость объекта.

Обратите внимание, что для этого класса нельзя использовать простое присваивание:

```
StringBuilder s="abc"; //Неправильно!
```

В этом случае необходимо записать так:

```
StringBuilder s=new StringBuilder("abc"); //Правильно
```

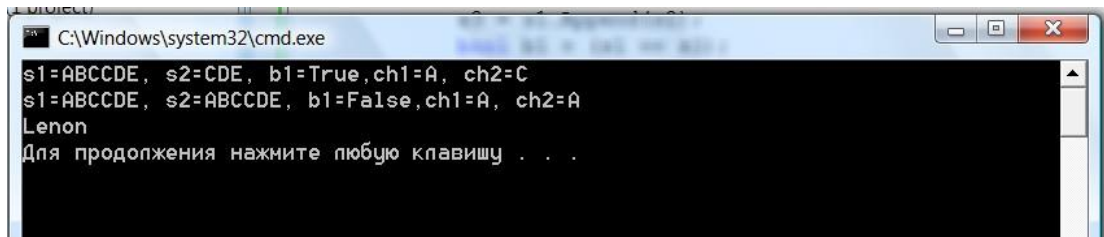
### 3.3.2 Операции над строками

Над строками этого класса определены практически те же операции с той же семантикой, что и над строками класса `String`, однако, операция конкатенации (+) не определена над строками класса `StringBuilder`, ее роль играет метод `Append`, дописывающий новую строку в хвост уже существующей.

Со строкой этого класса можно работать как с массивом, но, в отличие от класса `String`, здесь уже допускается не только чтение отдельного символа, но и его изменение.

```
using System;
using System.Text;

namespace String3
{
    class Program
    {
        static void Main(string[] args)
        {
            //Строки класса StringBuilder операции над строками
            StringBuilder s1 = new StringBuilder("ABC"),
            s2 = new StringBuilder("CDE");
            StringBuilder s3 = new StringBuilder();
            //s3= s1+s2;
            s3 = s1.Append(s2);
            bool b1 = (s1 == s3);
            char ch1 = s1[0], ch2 = s2[0];
            Console.WriteLine("s1={0}, s2={1}, b1={2}, " +
                "ch1={3}, ch2={4}", s1, s2, b1, ch1, ch2);
            s2 = s1;
            b1 = (s1 != s2);
            ch2 = s2[0];
            Console.WriteLine("s1={0}, s2={1}, b1={2}, " +
                "ch1={3}, ch2={4}", s1, s2, b1, ch1, ch2);
            StringBuilder s = new StringBuilder("Zenon");
            s[0] = 'L';
            Console.WriteLine(s);
        }
    }
}
```



Результаты работы программы

### 3.3.3 Основные методы и свойства

У класса есть основные методы, позволяющие выполнять такие операции над строкой как вставка, удаление и замена подстрок, но нет методов, подобных поиску вхождения, которые можно выполнять над обычными строками. Технология работы обычно такова: конструируется строка класса `StringBuilder`; выполняются операции, требующие изменение значения; полученная строка преобразуется в строку класса `String`; над этой строкой выполняются операции, не требующие изменения значения строки. У класса `StringBuilder` нет статических методов. Все его методы - динамические.

**public StringBuilder Append (<объект>).** К строке, вызвавшей метод, присоединяется строка, полученная из объекта, который передан методу в качестве параметра. Метод перегружен и может принимать на входе объекты всех простых типов, начиная от `char` и `bool` до `string` и `long`. Поскольку объекты всех этих типов имеют метод `ToString`, всегда есть возможность преобразовать объект в строку, которая и присоединяется к исходной строке. В качестве результата возвращается ссылка на объект, вызвавший метод. Поскольку возвращаемую ссылку ничему присваивать не нужно, то правильнее считать, что метод изменяет значение строки;

**public StringBuilder Insert (int location, <объект>).** Метод вставляет строку, полученную из объекта, в позицию, указанную параметром `location`. Метод `Append` является частным случаем метода `Insert`;

**public StringBuilder Remove (int start, int len).** Метод удаляет подстроку длины `len`, начинающуюся с позиции `start`;

**public StringBuilder Replace (string str1, string str2).** Все вхождения подстроки `str1` заменяются на строку `str2`;

**public StringBuilder AppendFormat (<строка форматов>, <объекты>).** Метод является комбинацией метода `Format` класса `String` и метода `Append`. Строка форматов, переданная методу, содержит только спецификации форматов. В соответствии с этими спецификациями находятся и форматируются объекты. Полученные в результате форматирования строки присоединяются в конец исходной строки.

Примеры использования основных свойств и методов класса `StringBuilder`:

**Свойство Length.** Возвращает длину строки:

```
int k=s.Length;
```

**Метод Append.** Прибавляет строку к существующей:

```
StringBuilder s1=new StringBuilder("Cogito ");
StringBuilder s2=new StringBuilder("ergo ");
s1.Append(s2);
s1.Append("sum");
System.Console.WriteLine(s1); //Напечатается "Cogito ergo sum"
                               //Я мыслю, значит я существую
```

**Метод Equals.** Служит для сравнения двух строк. Возвращает `true` или `false`. Пример использования:

```

if (s1.Equals(s2))
    System.Console.WriteLine("Строки равны");
else
    System.Console.WriteLine("Строки не равны");

```

**Метод Insert.** Вставляет символы в заданную позицию (нумерация идет с нуля):

```

StringBuilder s1=new StringBuilder("abcde");
s1.Insert(2, "xyz");
System.Console.WriteLine(s1); //Напечатается "abxyzcde"

```

**Метод Remove.** Удаляет символы из строки:

```

StringBuilder s1=new StringBuilder("abcde");
s1.Remove(1, 2);
System.Console.WriteLine(s1); //Напечатается "ade"

```

Первый параметр у Remove - это с какой позиции удаляем (нумерация с нуля), второй - сколько символов удаляем.

**Метод Replace.** Заменяет символы:

За исключением метода Remove, все рассмотренные методы являются перегруженными

```

StringBuilder s=new StringBuilder("abcdeabcde");
s.Replace("abc", "ZZZ");
System.Console.WriteLine(s); //Напечатается "ZZZdeZZZde"

```

### 3.3.4 Емкость буфера

Каждый экземпляр строки класса StringBuilder имеет буфер, в котором хранится строка. Объем буфера - его емкость - может меняться в процессе работы со строкой. Объекты класса имеют две характеристики емкости - текущую и максимальную. В процессе работы текущая емкость изменяется в пределах максимальной емкости. Если размер строки увеличивается, то соответственно автоматически растет и текущая емкость. Если же размер строки уменьшается, то емкость буфера остается на том же уровне. По этой причине иногда правильно уменьшать емкость. Следует помнить, что попытка уменьшить емкость до величины, меньшей длины строки, приведет к ошибке.

У класса StringBuilder имеется 2 свойства и один метод, позволяющие анализировать и управлять емкостными свойствами буфера. Этими характеристиками можно управлять также еще на этапе создания объекта, - для этого имеется соответствующий конструктор. Рассмотрим свойства и метод класса, связанные с емкостью буфера:

- свойство Capacity - возвращает или устанавливает текущую емкость буфера;
- свойство MaxCapacity - возвращает максимальную емкость буфера (максимальное количество символов, которые можно записать в объект типа StringBuilder).

```

System.Console.WriteLine(s.MaxCapacity);

```

Метод `int EnsureCapacity (int capacity)` - позволяет уменьшить емкость буфера. Метод пытается вначале установить емкость, заданную параметром `capacity`; если это значение меньше размера хранимой строки, то емкость устанавливается такой, чтобы гарантировать размещение строки. Это число и возвращается в качестве результата работы метода.

## 4 Задания

### 1. Выполнить задания по вариантам.

1. Дана строка символов. Преобразовать данную строку, удалив из нее каждую пару символов «» и повторив (вставив еще раз) каждую пару символов «. После преобразования полученную строку вывести на печать.

**Примечание.** Удалить символ – не значит заменить его пробелом, так как в данном случае пробел тоже символ.

2. Дана строка символов. Исключить из этой строки группы символов, расположенные между скобками [ , ]. Сами скобки тоже должны быть исключены. Предполагается, что внутри каждой пары скобок нет других скобок.
3. Задан одномерный массив, каждым элементом которого является строка символов, состоящая из одних цифр. Рассматривая каждую строку как число, определить сумму четных и нечетных значений элементов массива.
4. Задан одномерный массив, каждым элементом которого является строка символов, состоящая из одних цифр. Упорядочить элементы массива по возрастанию их числовых значений и вывести на экран. От максимального элемента отнять значение минимального и вывести разность на экран. Подсчитать среднее значение всех элементов.
5. Дана строка символов, состоящая из нулей, единиц и пробелов. Группы нулей и единиц, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Требуется найти самое длинное и самое короткое слово в строке и, рассматривая эти слова как числа, определить их сумму.
6. Дана строка символов, состоящая из нулей, единиц и пробелов. Группы нулей и единиц, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Требуется подсчитать количество слов в данной строке. Рассматривая слова как числа, определить количество слов, делящихся на 5 без остатка.
7. На складе хранится ряд деталей, наименования которых представляют некоторую последовательность символов (например, Д21А, Д52Н и т.д.). Подсчитать количество наименований деталей, которые начинаются с Д2. Все имеющиеся наименования вводить с клавиатуры; поскольку число деталей заранее не известно, заканчивать ввод данных следует вводом пустой строки (строки, не содержащей ни одного символа), т.е. нажатием клавиши <Enter> без ввода наименования детали.
8. Разработать программу, которая работает следующим образом. Пользователь вводит свою фамилию, год рождения и место рождения (город). Программа подсчитывает, сколько ему лет, формирует строку символов вида: **‘фамилия - количество\_лет - место\_рождения’** и запоминает ее. Следующий пользователь вводит аналогичные данные о себе и т.д. Ввод данных заканчивается вводом пустой строки. По окончании ввода программа выводит на экран сведения о пользователях,
9. Разработать программу, которая проверяла бы орфографию (правильность написания) слов в некоторой строке, сравнивая их со словами из словаря. Использовать при этом в качестве словаря одномерный массив слов (описать в программе в виде типизированной константы).

10. Дана строка символов, содержащая некоторый текст. Разработать программу, которая определяет, 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").
11. Составить программу, которая читает произвольные строки длиной 80 символов, разбивает их по словам (подстрока между двумя пробелами), находит максимальное слово и выводит его на печать. Ввод строк заканчивается вводом символа '!', который не является элементом строки.
12. Составить программу, которая читает построчно текст другой программы (ввести с клавиатуры) на языке C# (5 строк), обнаруживает комментарии и выводит их на печать.
13. Составить программу, которая читает произвольный текст (5 строк по 40 символов) и распечатывает в алфавитном порядке все латинские буквы, входящие в этот текст.
14. Разработать программу, которая предназначена для зашифровки текстов. Принять следующий тривиальный алгоритм шифрования. Все буквы А в исходном тексте заменяются на У, буквы П - на Ж, буквы О - на Ю и т.д. (по вашему усмотрению). Вывести на печать исходный текст и результат шифрования. Текст читать построчно (строка - 80 символов), хранить текст в массиве строк.
15. Составить программу, которая предназначена для ввода вещественных чисел с фиксированной точкой и проверки, является ли введенное число палиндромом. Палиндром принимает одно и то же значение при чтении его как справа налево, так и слева направо. Программа должна быть рассчитана на ввод десяти чисел.

## **2. Выполнить задания по вариантам.**

Ввести вашу фамилию, имя и отчество как строку символов. Определить длину строки и количество букв совпадающих со второй буквой фамилии. Используя методы класса `StringBuilder` вставить между каждой буквой фамилии и имени знак " – ". В отчестве заменить гласные строчные буквы на прописные. Вывести полученный результат, прибавив к нему комментарии - студент какой специальности и группы это выполнил. Выполнить задание в соответствии со своим вариантом.

1. Вывести имя и количество букв в третьем слове.
2. Определить количество букв «а» в фамилии.
3. Вывести первые буквы фамилии, имени и отчества с точками.
4. Вывести длину фамилии и имени.
5. Вывести фамилию и инициалы.
6. Вывести имя и количество букв в фамилии.
7. Определить количество букв «а» в имени.
8. Вывести самое длинное слово.
9. Удалить все буквы «а» и «о» из фамилии.
10. Вывести имя в столбец.
11. Проверить начинается ли хотя бы одно из слов с буквы «В»
12. Все буквы «и» имени продублировать.
13. Вывести фамилию и количество букв имени.
14. Вывести имя в обратном порядке.
15. Вывести фамилию в столбец.
16. Вывести имя, отчество и количество букв имени.
17. Вывести слово, которое имеет наименьшее количество букв.
18. Вывести фамилию, имя, отчество без пробелов. Сколько букв имеет имя.
19. Вывести длины трех слов.



20. Вывести имя и количество букв фамилии.
  21. Вывести имя, фамилию и суммарную длину слов.
  22. Каждую букву имени продублировать.
  23. Вывести фамилию в обратном порядке.
  24. Определить количество букв «а» и «о» в имени.
  25. Вывести третье слово и количество букв в фамилии.
3. Создать свой способ шифрования текста. Составить программу для введения текста, его шифрования и печати результатов.

## **5 Контрольные вопросы**

1. Как описываются строковые переменные.
2. Какая максимальная длина строки допустима в C#?
3. Какие операции допустимы над строковыми данными?
4. В чем отличие строковой переменной от массива символов?
5. Какие функции для работы со строками вы знаете?
6. В чем отличие классов String и StringBuilder?