Практическая работа №6
Класс Array. Строки. Класс String

1. Цель работы:

1) Получение практических навыков при работе со строками в C#.
2) Получение практических навыков при работе с функциями C#.
3) Получение практических навыков при создании диалоговых консольных приложений.

2. Теоретические сведения

2.1. Строковые и буквальные строковые литералы

Для представления текстовой информации в C# используются объекты класса string. Класс string представляет собой один из предопределенных типов языка C#. В .Net Framework этому типу соответствует класс System.String. Один из видов объектов класса string мы уже многократно применяли — это строковые константы, или строковые литералы.

Строковая константа, или строковый литерал, имеет две формы:

- обычный (регулярный) строковый литерал (regular-string-literal);
- буквальный строковый литерал (verbatim-string-literal).

Регулярный строковый литерал - это последовательность символов и эскейп-последовательностей, заключенная в кавычки (не в апострофы).

Обрабатывая регулярный строковый литерал, компилятор из его символов формирует строковый объект и при этом заменяет эскейп-последовательности соответствующими кодами (символов или управляющих кодов). Например, литералу

"\u004F\x4E\u0045\ttwo"

будет соответствовать строка, при выводе которой на экране текст появится в таком виде:

ONE two

Буквальный (дословный) строковый литерал начинается с префикса @, за которым в кавычках размещается последовательность символов. Символы такого литерала воспринимаются буквально, т.е. в такой строке не обрабатываются эскейп-последовательности, а каждый символ воспринимается как таковой. В результате выполнения оператора:

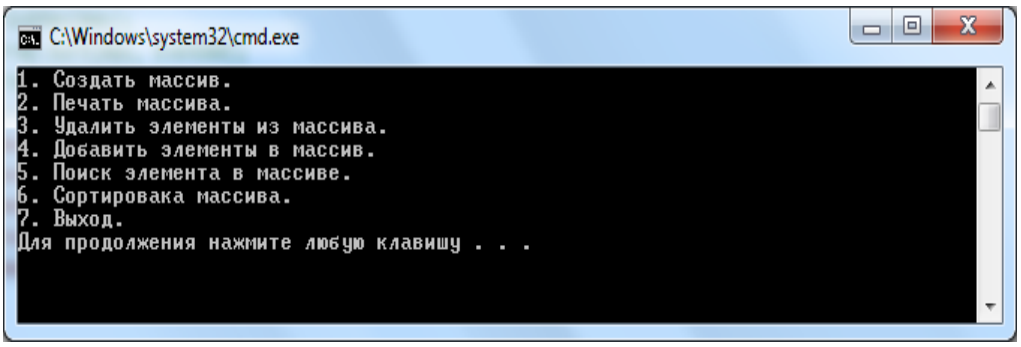
```
Console.WriteLine(@"\u004F\x4E\u0045\ttwo");
```

на экране появится \u004F\x4E\u0045\ttwo

Если в буквальном литерале необходимо поместить кавычку, то она изображается двумя рядом стоящими кавычками.

Буквальный литерал может быть размещен в коде программы на нескольких строках, и это размещение сохраняется при его выводе.

```
Console.WriteLine(@"1. Создать массив.
2. Печать массива.
3. Удалить элементы из массива.
4. Добавить элементы в массив.
5. Поиск элемента в массиве.
6. Сортировака массива.
7. Выход.");
```



2.2. Ссылки типа string

Каждый строковый литерал — это объект класса (типа) string.

```
string stroka;
```

Класс string является ссылочным типом. Кроме литералов, можно определить объекты класса string с использованием конструкторов. (Конструктор - специальный метод класса, предназначенный для инициализации объекта класса в процессе его создания.) Конструкторы класса string позволяют инициализировать объекты-строки несколькими способами.

```
string str1="Это строка 1";

char []charArr={'M','a','c','c','и','в'};
string str2=new string(charArr);

string str3=new string('S',5);

string str4 = new string(charArr, 4, 1);
```

Строковые объекты, как создаваемые с применением конструкторов, так и формируемые для представления строковых литералов, компилятор размещает в динамической памяти. Ссылки на строки размещаются в стеке. Размер строки при определении строкового объекта явно не указывается, он определяется автоматически при инициализации. **Ни размер строки, ни ее содержимое не могут изменяться после создания строки!!**

2.3. Операции над строками

Строки языка C# предназначены для хранения последовательностей символов, для каждого из которых отводится 2 байта, и они хранятся в кодировке Unicode (как данные типа char). В некотором смысле строка подобна одномерному массиву с элементами типа char. Элементы (символы строки) последовательно нумеруются, начиная с 0. Последний символ имеет номер на 1 меньше длины строки.

- **операция индексирования**:

`строка[индекс]`, индекс – целое число,>0.

Результат выражения с операцией индексирования - символ (значение типа **char),** размещенный в той позиции строки, номер которой соответствует индексному выражению. Если значение индекса меньше нуля, а также больше или равно длине строки, возникает исключительная ситуация (генерируется исключение).

- **Операция присваивания (=)** для строк выполняется не так как для массивов. Когда ссылке с типом массива присваивается значения ссылки на другой уже существующий массив, изменяет только значение ссылки. Массив, как объект, становится доступен для нескольких ссылок. Т.е. адрес первого элемента массива хранится в нескольких переменных ссылочного типа.

Операция присваивания для строк приводит к созданию нового экземпляра той строки, на которую ссылается выражение справа от знака операции =. Ранее существовавшая строка никак не ассоциируется с новой ссылкой

- **Операции сравнения** на равенство == и неравенство !=, применяемые к строкам, сравнивают последовательности символов в строках. (Для массивов сравниваются значения ссылок.)
- **Сцепление (конкатенацию) строк выполняет операция +.**

2.5. Методы и свойства класса String

- **int Length() –** свойство, позволяющее получить длину (количество символов) конкретной строки (объекта класса **string).**
- **int CompareTo()** - метод, который сравнивает две строки и возвращает целочисленное значение. Для двух строк **S1,** S2 результат положительный, если S1>S2, отрицательный, если **S1<S2,** и нулевой, если SI == S2. Сравнение строк выполняется лексикографически.
- **static string Concat()** - метод (их несколько) выполняет конкатенацию строк-параметров. Аргументов-строк может быть два, три или произвольное количество.
- **static string Copy()** — статический метод возвращает копию существующей строки.
- **static string Format()** - статический метод, формирующий строку на основе набора параметров.
- **int IndexOf()** — нестатический метод поиска в вызывающей строке подстроки, заданной параметром. Возвращает индекс или -1, если поиск неудачен. Поиск - с начала строки.
- **string Insert()** - нестатический метод для вставки строки-параметра в копию вызывающей строки с позиции, заданной дополнительным параметром.
- **static string Join()** - статический метод, объединяющий в одну строку строки массива-параметра. Первый параметр типа **string** задает разделитель, которым будут отделены друг от друга в результирующей строке элементы массива.
- **int LastIndexOf()** - нестатический метод поиска в вызывающей строке подстроки, заданной параметром. Возвращает индекс или -1, если поиск неудачен. Поиск с конца строки.
- **string Remove()** - удаляет символы из копии строки.
- **string Replace()** - заменяет символы в копии строки.
- **string [] Split()** - формирует массив строк из фрагментов вызывающей строки. Параметр типа **char** задает разделители, которыми в строке разделены фрагменты.
- **char [] ToCharArray()** — копирует символы вызывающем строки в массив типа **char[].**
- **string Trim()** - удаляет вхождение заданных символом (например, пробела) в начале и в конце строки.
- **string Substring()** — выделяет из строки подстроку. Параметры задают начало и длину выделяемой части строки.

2.6. Форматирование строк

При выводе, например, с помощью Console.Write(), значений базовых типов (например, int или double) они автоматически преобразуются в символьные строки. Если программиста не устраивает автоматически выбранный формат их внешнего представления, он может его изменить. Для этого можно воспользоваться статическим методом Format класса string или использовать так называемую строку форматирования в качестве первого параметра методов, поддерживающих форматирование, например, Console.Write() и Console.WriteLine (). В обоих случаях правила подготовки исходных данных для получения желаемого результата (новой строки) одинаковы.

```
static string Format (string form, params object[]ar);
```

- `string form` – строка форматирования, включает поля подстановок `{N[,W]:S[R]]},`

где N – номер аргумента,

W – ширина поля,

S – спецификатор формата,

R – спецификатор точности.

- `params object[]ar` – параметры, подставляемые вместо номера аргумента.

W - ширина поля в поле подстановки определяет количество позиций, выделяемых для изображения подставляемого значения. Если ширина поля не указана, то она определяется автоматически - минимально достаточной для изображения значения. Если ширина поля указана и превышает длину помещаемого в поле значения, то при положительной длине поля W значение выравнивается по правой границе. Если перед шириной поля W стоит минус, то выравнивание выполняется по левой границе поля.

Спецификатор формата S задает вид изображаемого значения.

C,c – валютный, R – количество десятичных разрядов.

D,d – целочисленный, R – минимальное количество цифр.

E,e – экспоненциальный, R – число разрядов после точки.

F,f – с фиксированной точкой, R – число разрядов после точки.

G,g – короткий из E или F.

X,x – шестнадцатеричный, R – минимальное число цифр.

2.7. Массивы строк

В массив помещаются не строки, а только ссылки на них, но при использовании массивов ссылок на строки не требуются никакие специальные операции для организации обращения к собственно строкам через ссылки на них. Поэтому в литературе, посвященной языку C#, зачастую говорят просто о массивах строк.

2.8. Неизменяемость объектов класса String

К символам объекта класса **string,** можно обращаться только для получения их значений. Например, для получения значения одного символа строки используется выражение с операцией индексирования []. Чтобы изменить строку можно воспользоваться следующим алгоритмом:

1. Переписать символы строки в массив с элементами типа char.
2. Выполнить преобразования в массиве с элементами типа char.
3. Создать новую строку, используя конструктор с параметром string(char[]).

2.9. Базовый класс System. Array

Каждый создаваемый массив получает большую часть функциональности от класса System. Array.

Некоторые члены класса System. Array:

Clear ()	Статический метод, который позволяет устанавливать для всего ряда элементов в массиве пустые значения (0 — для чисел, null — для объектных ссылок и false — для булевских выражений)
CopyTo ()	Метод, который позволяет копировать элементы из исходного массива в целевой
Copy ()	Статический метод, который позволяет «опировать заданный диапазон элементов одного массива в другой массив
Length	Свойство, которое возвращает информацию о количестве элементов в массиве
Rank	Свойство, которое возвращает информацию о количестве измерений в массиве

Reverse ()	Статическое свойство, которое представляет содержимое одномерного массива в обратном порядке
Sort ()	Статический метод, который позволяет сортировать одномерный массив
BinarySearch()	Статический метод, который находит номер элемента в упорядоченном одномерном массиве методом бинарного поиска
IndexOf()	Статический метод, который находит номер первого вхождения заданного элемента в одномерный массив
LastIndexOf()	Статический метод, который находит номер последнего вхождения заданного элемента в одномерный массив

3. Постановка задачи

Постановка задачи 1.

1. Создать динамический массив (одномерный, двумерный, рваный) из элементов заданного типа. При заполнении массива использовать 2 способа (ручной и с помощью ДСЧ).
2. Массив вывести на печать.
3. Выполнить операции с массивом, указанные в варианте, используя, по возможности, методы класса Array.
4. Результаты обработки вывести на печать.

Постановка задачи 2.

1. Ввести строку символов с клавиатуры. Строка состоит из слов, разделенных пробелами (пробелов может быть несколько) и знаками препинания (, ;:). В строке может быть несколько предложений, в конце каждого предложения стоит знак препинания (.!?).
2. Выполнить обработку строки в соответствии с вариантом.
3. Результаты обработки вывести на печать.

Постановка задачи для дополнительного заданчи:

1. Определить соответствует ли строка правилам, заданным в варианте.

4. Варианты

Варианты для задачи 1

№	Тип массива	Тип элементов	Операция
1	Одномерный	int	Отсортировать по убыванию только четные элементы массива, остальные элементы остаются на своих местах.
2	Двумерный	double	Удалить из массива первую строку, в которой есть хотя бы один элемент равный 0.
3	Рваный	char	Удалить из массива первую строку, в которой есть не менее 3 гласных букв.
4	Одномерный	char	Удалить из массива последнюю гласную букву.
5	Двумерный	int	Удалить из массива первый столбец, в котором встречается элемент,

			совпадающий с минимальным элементом массива.
6	Рваный	char	Удалить из массива последнюю строку, в которой есть не менее 3 символов цифр.
7	Одномерный	double	Удалить из массива все элементы кратные минимальному элементу массива.
8	Двумерный	char	Удалить из массива все строки, в которых нет цифр.
9	Рваный	int	Отсортировать строки массива по возрастанию, а затем переставить строки таким образом, чтобы их длины возрастали.
10	Одномерный	int	Найти сумму всех четных элементов, которые находятся между первым минимальным элементом и последним максимальным элементом массива. Минимальных и максимальных элементов в массиве может быть несколько.
11	Двумерный	double	Удалить все строки, в которых есть число, совпадающее с максимальным элементом.
12	Рваный	char	Отсортировать строки массива по убыванию кодов символов, а затем переставить строки таким образом, чтобы их длины возрастали.
13	Одномерный	char	Удалить из массива все цифры
14	Двумерный	double	Удалить первый столбец, в котором есть число, совпадающее с минимальным элементом.
15	Рваный	int	Удалить все строки, в которых есть не менее двух нулей.
16	Одномерный	double	Отсортировать по убыванию только отрицательные элементы массива, остальные элементы остаются на своих местах.
17	Двумерный	int	Удалить из массива первую строку, в которой больше одного элемента равного 0.
18	Рваный	char	Удалить из массива первую строку, в которой есть не менее 3 гласных букв.
19	Одномерный	char	Удалить из массива последнюю гласную букву.
20	Двумерный	int	Удалить из массива последний столбец, в котором встречается элемент, совпадающий с минимальным элементом массива.
21	Рваный	char	Удалить из массива первую строку, в которой есть не менее 3 символов цифр.
22	Одномерный	int	Удалить из массива все элементы кратные минимальному элементу

			массива.
23	Двумерный	char	Удалить из массива все строки, в которых нет цифр.
23	Рваный	double	Отсортировать строки массива по возрастанию, а затем переставить строки таким образом, чтобы их длины возрастали.
25	Одномерный	int	Найти сумму всех четных элементов, которые находятся между первым минимальным элементом и последним максимальным элементом массива. Минимальных и максимальных элементов массиве может быть несколько.

Варианты для задачи 2

№	Задание
1	Перевернуть каждое нечетное предложение.
2	Перевернуть каждое четное слово.
3	Определить есть ли в строке идентификаторы, если есть, то напечатать самый длинный идентификатор.
4	Поменять местами первое и последнее предложение в строке.
5	Поменять местами первое и последнее слово в строке.
6	Определить есть ли в строке ключевые слова C#. Если есть, то напечатать сколько раз встречается каждое слово.
7	Сдвинуть циклически влево каждое слово на количество символов равное номеру этого слова в строке.
8	Перевернуть каждое предложение, заканчивающееся символом ’!’.
9	Перевернуть каждое слово, номер которого совпадает с его длиной.
10	Определить есть ли в строке идентификаторы, если есть, то напечатать самый короткий идентификатор.
11	Удалить первое и последнее предложение в строке.
12	Удалить первое и последнее слово в строке.
13	Перевернуть все слова в предложении и отсортировать слова по убыванию.
14	Перевернуть все слова в предложении и отсортировать слова по убыванию длин слов.
15	Удалить из строки все слова, которые начинаются и заканчиваются на один и тот же символ.
16	Определить есть ли в строке ключевые слова C#. Если есть, то напечатать сколько раз встречается каждое слово.
17	Сдвинуть циклически влево каждое слово на количество символов равное номеру этого слова в строке.
18	Перевернуть каждое предложение, заканчивающееся символом ’!’.
19	Перевернуть каждое слово, номер которого совпадает с его длиной.
20	Определить есть ли в строке идентификаторы, если есть, то напечатать самый короткий идентификатор.
21	Удалить все предложения в строке, которые заканчиваются «!».
22	Удалить все слова в строке, которые начинаются с цифры .
23	Перевернуть все слова в предложении и отсортировать слова по убыванию.
24	Перевернуть все слова в предложении и отсортировать слова по убыванию длин

	слов.
25	Удалить из строки все слова, которые начинаются и заканчиваются на один и тот же символ.

5. Методические указания

1) Для организации взаимодействия с пользователем использовать текстовое меню.

2) Предусмотреть 2 способа формирования массивов: вручную (ввод значений с клавиатуры) и с помощью датчика случайных чисел.

3) Предусмотреть 2 способа ввода строк: с клавиатуры и из заранее сформированного массива строк.

4) Предусмотреть возникновение исключительных ситуаций при вводе символов вместо цифр числа.

5) При удалении элементов (строк, столбцов) предусмотреть ошибочные ситуации, т. е. ситуации, в которых будет выполняться попытка удаления элемента (строки, столбца) из пустого массива или количество удаляемых элементов будет превышать количество имеющихся элементов (строк, столбцов). В этом случае должно быть выведено сообщение об ошибке.

6) При попытке вывода пустого массива/строки должно выводиться сообщение о том, что массив/строка пустые.

7) Рекомендуется при отладке программы сначала полностью отладить выполнение одной задачи и только после этого переходить к следующей.

6. Требования к программе

1. Реализация основных функций задачи (создание, обработка в соответствии с вариантом, вывод полученных результатов).
2. Дополнительные функции (проверка правильности вводимых данных и т.д.)
3. Стилевое оформление программы.
4. Удобный интерфейс.
5. Использование разных типов функций (перегрузка, параметры по умолчанию, функции с переменным числом параметров, рекурсивные функции и т.п.).
6. Использование исключений.
7. Использование возможностей языка программирования, изучаемых самостоятельно.

7. Содержание отчета

1. Описание этапа анализа.
2. Описание этапа проектирования (описание функций и их интерфейсов).
3. Листинг программы.
4. Тесты с проверкой полноты по критериям черного и белого ящика.