Тематическое планирование по курсу «Основы программирования на языке C#»

Не гонитесь за рассмотрением всех тем, если дети не поняли что-то, стоит остановиться на теме еще одно занятие. Ссылка на учебник Герберт Шилдт C# Полное руководство 2011

1 семестр

№ занятия	Тема	Описание
1	Введение	Сбор ожиданий. Рассказать о курсе. Рассказать про программирование, роль языка C# в современном мире. Создание простого консольного и winform приложения. Структура и открытие/перенос компьютера на домашний компьютер.
2	Переменные и типы данных	Материал из учебника
3	Управляющие операторы: if	Материал из учебника
4	Управляющие операторы, switch и текстовые квесты	Материал из учебника
5	Методы	Материал из учебника
6	Цикл while	Материал из учебника
7	Турнир. Этап 1.	
8	Циклы, for	Материал из учебника
9	Одномерные массивы	Теория: Герберт Шилдт C# Полное руководство 2011 Стр.177 – 181 Свойство Length стр.189
10	Цикл foreach Многомерные массивы	Теория: foreach стр.194 - 198 Многомерные Герберт Шилдт C# Полное руководство 2011 Стр.182 – 185 (сильным группам 187)
11	Класс List	См.ниже после таблиц
12	Знакомство с Winforms	Калькулятор, пример либо в старом учебнике, либо по ссылке http://vscode.ru/prog-lessons/kalkulyator-windows-forms-c-shapr.html
13	Работа с файлами	См.ниже после таблиц
14	Работа со строками в Си-шарп. Класс String	См.ниже после таблиц
15	Турнир. Этап 2.	
16	Обработка исключений try catch	Герберт Шилдт C# Полное руководство 2011 стр.404 – 418 задание см.ниже после таблиц

2 семестр

№ занятия	Тема	Описание
17	Понятие объектно-ориентированного программирования (ООП). Классы и объекты	Герберт Шилдт C# Полное руководство 2011 стр.147 – 155
18	Статичные и простые методы. Разница между ними. Перегрузка методов.	задание см.ниже после таблиц
19	Конструкторы в Си-шарп. Оператор this	
20	Свойства. Аксессоры get и set.	
21	Наследование 1	
22	Наследование 2	
23	Турнир. Этап 3.	
24	Полиморфизм 1	
25	Полиморфизм 2 Виртуальные методы	
26	Абстрактные классы	
27	Интерфейсы 1	
28	Интерфейсы 2	
29	Инкапсуляция 1	
30	Инкапсуляция 2	
31	Турнир. Этап 4. Финал.	
32	Перегрузка операторов	

Тема 11. Класс List

Класс List служит для работы со списками, о чем и говорит его название. Это такой «навороченный» массив. Главное отличие от простого массива в том, что он динамический – вы можете вставлять и удалять элементы в любое время, в то время как в простом массиве размер указывается при создании и сделать его больше или меньше нельзя.

```
static void Main(string[] args)
{
  List<string> teams = new List<string>(); // создание списка
  teams.Add("Barcelona"); // добавление элемента
  teams.Add("Chelsea");
  teams.Add("Arsenal");
  List<string> teams2 = new List<string>() {"Dynamo", "CSKA" }; // инициализация
}
```

Добавление элементов

Для добавления элементов в список в нем реализовано несколько методов:

Метод	**Описание**
Add([элемент])	добавляет элемент в конец списка
AddRange([список элементов])	добавляет в конец списка элементы указанного списка
Insert([индекс],[элемент])	вставляет элемент на позицию соответствующую индексу, все элементы «правее» будут сдвинуты на одну позицию
InsertRange([индекс], [список элементов])	то же самое, только вставляется множество элементов

Удаление элементов

Метод	**Описание**
Remove([элемент])	удаляет первое вхождение указанного элемента из списка
RemoveRange([индекс], [количество])	удаляет указанное количество элементов, начиная с указанной позиции
RemoveAt([индекс])	удаляет элемент, который находится на указанной позиции
Clear()	удаляет все элементы списка

Свойство Count соответствует свойству обычного массива – Length – количество элементов.

```
static void Main(string[] args)
{
  List<string> teams = new List<string>() { "Inter", "Milan", "Bayern", "Juventus"};
  teams.Insert(2,"Barcelona"); // вставляем в список элемент "Barcelona" на позицию 2
  teams.Remove("Milan"); // удаляем первое вхождение элемента "Milan" из списка
  List<string> newTeams = new List<string>() { "Liverpool", "Roma", "Borussia", "Valencia" };
  teams.AddRange(newTeams); // добавляем в конец списка элементы списка newTeams
}
```

Стоит помнить, что простые массивы работают быстрее, чем списки List. Если в вашей программе не особо важна производительность и вы не работаете с большими количествами данных, то удобнее использовать список, в противном случае нужно использовать простые массивы.

Задание №1.

Написать программу, использующую разные методы, написанные в таблице.

Задание №2.

Петя успевает по математике лучше всех в классе, поэтому учитель задал ему сложное домашнее задание, в котором нужно в заданном наборе целых чисел найти сумму всех положительных элементов, затем найти где в заданной последовательности находятся максимальный и минимальный элемент и вычислить произведение чисел, расположенных между ними. Так же известно, что минимальный и максимальный элемент встречаются в заданном множестве чисел только один раз. Поскольку задач такого рода учитель дал Пете около ста, то Петя как сильный программист смог написать программу, которая по заданному набору чисел самостоятельно находит решение. А Вам слабо?

На входе пользователь вводит единственное число N – количество элементов списка. Далее список заполняется N случайными элементами в диапазоне от -100 до 100.

На выходе нужно вывести два числа, разделенных пробелом: сумму положительных элементов и произведение чисел, расположенных между минимальным и максимальным элементами.

Пример:

На входе	На выходе
5 -7 5 -1 3 9	17 -15

Тема 13. Работа с файлами

Файл – это набор данных, который хранится на внешнем запоминающем устройстве (например на жестком диске). Файл имеет имя и расширение. Расширение позволяет идентифицировать, какие данные и в каком формате хранятся в файле.

Под работой с файлами подразумевается:

- создание файлов;
- удаление файлов;
- чтение данных;
- запись данных;
- изменение параметров файла (имя, расширение…);
- другое.

В Си-шарп есть пространство имен **System.IO**, в котором реализованы все необходимые нам классы для работы с файлами. Чтобы подключить это пространство имен, необходимо в самом начале программы добавить строку using System.IO.

```
using System.IO;
```

Как создать файл?

Для создания пустого файла, в классе **File** есть метод **Create()**. Он принимает один аргумент – путь. Ниже приведен пример создания пустого текстового файла new_file.txt на диске D:

```
static void Main(string[] args)
{
  File.Create("D:\\new_file.txt");
}
```

Если файл с таким именем уже существует, он будет переписан на новый пустой файл.

Метод **WriteAllText()** создает новый файл (если такого нет), либо открывает существующий и записывает текст, заменяя всё, что было в файле:

```
static void Main(string[] args)
{
  File. WriteAllText("D:\\new_file.txt", "текст");
}
```

Метод **AppendAllText()** работает, как и метод WriteAllText() за исключением того, что новый текст дописывается в конец файла, а не переписывает всё что было в файле:

```
static void Main(string[] args)
{
  File.AppendAllText("D:\\new_file.txt", "текст метода AppendAllText ()"); //допишет текст в конец файла
}
```

Как удалить файл?

Метод **Delete()** удаляет файл по указаному пути:

```
static void Main(string[] args)
{
  File.Delete("d:\\test.txt"); //удаление файла
}
```

Как создать папку?

С помощью статического метода **CreateDirectory()** класса **Directory**:

```
static void Main(string[] args)
{
 Directory.CreateDirectory("d:\\new_folder");
}
```

Как удалить папку?

Для удаления папок используется метод **Delete()**:

```
static void Main(string[] args)
{
  Directory.Delete("d:\\new_folder"); //удаление пустой папки
}
```

Если папка не пустая, необходимо указать параметр рекурсивного удаления - true:

```
static void Main(string[] args)
{
  Directory.Delete("d:\\new_folder", true); //удаление папки, и всего, что внутри
}
```

Получение пути к системным папкам

```
string path = Environment.GetFolderPath(Environment.SpecialFolder.System);
```

Задание №1.

Попробуй создать фаил, затем его удалить, создать и удалить папку.

Задание №2.

Используя циклы создай в папке «Документы» 20 файлов с именем file1.txt, file2.txt … file20.txt

Задание №3.

Реализуй создание файлов по нажатии на кнопку в Winforms

Тема 14. Работа со строками в Си-шарп. Класс String

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.

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.IsNullOrWhiteSpace(s1); // True
  String.IsNullOrWhiteSpace(s2); // True
  String.IsNullOrWhiteSpace(s3); // True
  String.IsNullOrWhiteSpace(s4); // False
}
```

Как проверить, является ли одна строка "больше" другой?
Для сравнивания 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

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".

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();
}
```

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();
}
```

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();
}
```

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.

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();
}
```

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'…
}
```

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"
}
```

Задание № 1.

Есть некий текст. Необходимо заменить в этом тексте все слова "Nikolay" на "Oleg". Записать полученный текст в фаил names.txt на рабочий стол.

Задание №2.

Дана строка, которая содержит имена пользователей, разделенные запятой – "Login1,LOgin2,login3,loGin4". Необходимо разбить эту строку на массив строк (чтобы отдельно были логины), и перевести их все в нижний регистр.

Задание №3.

На WinForms реализовать задание №1, какое слово и каким словом заменить вводит пользователь. Текст для изменения помещается в textbox.

Тема 16

Задание 1.

Есть массив целых чисел размером 10. С клавиатуры вводится два числа - порядковые номера элементов массива, которые необходимо суммировать. Например, если ввели 3 и 5 - суммируются 3-й и 5-й элементы. Нужно предусмотреть случаи, когда были введены не числа, и когда одно из чисел, или оба больше размера массива.

Тема 18.

Задание 1.

Создайте класс Телевизор. В нем есть поле текущий канал. Предусмотрите в нем возможность переключения каналов: следующий канал, предыдущий канал, переход к каналу по номеру. Учтите, что канал не может иметь отрицательный номер.

Задание 2.

В зависимости от выбора пользователя вычислить площадь круга, прямоугольника или треугольника (использовать перегрузку методов)

Задание 3.

Имеется список имен. Создайте метод, который будет выводить на экран эти имена через запятую. Перегрузите этот метод так, чтобы можно было изменять разделитель – вместо запятых между именами любой символ, переданный параметром.