# **Среда Visual** [Studio.NET](http://Studio.NET)

Среда разработки Visual [Studio.NET](http://Studio.NET) предоставляет мощные и удобные средства написания, корректировки, компиляции, отладки и запуска приложений, использующих .NET-совместимые языки. Корпорация Microsoft включила в платформу средства разработки для четырех языков: С#, [VB.NET](http://VB.NET), С++ и J#.

Платформа .NET является открытой средой. Это значит, что компиляторы для нее могут поставляться и сторонними разработчиками. К настоящему времени разработаны десятки компиляторов для .NET, например, Ada, COBOL, Delphi, Eiffel, Fortran, Lisp, Obe-ron, Perl и Python.

Приложение в процессе разработки называется проектом. Про­ект объединяет все необходимое для создания приложения: файлы, папки, ссылки и прочие ресурсы. Среда Visual [Studio.NET](http://Studio.NET) позволя­ет создавать проекты различных типов, например:

* Windows-приложение использует элементы интерфейса Win­dows, включая формы, кнопки, флажки и пр.;
* консольное приложение выполняет вывод «на консоль», то есть в окно командного процессора;
* библиотека классов объединяет классы, которые предназначе­ны для использования в других приложениях;
* веб-сервис - компонент, методы которого могут вызываться через Интернет.

Консольные приложения наилучшим образом подходят для изучения языка, так как в них не используется множество стан­дартных объектов, необходимых для создания графического интер­фейса.

Знакомство со средой Microsoft Visual Studio 2010 (язык С#). Microsoft Visual Studio 2010 доступна в следующих вариантах:

Express - бесплатная среда разработки, включающая только базовый набор возможностей и библиотек;

Professional - поставка, ориентированная на профессиональ­ное создание программного обеспечения, и командную разработку, при которой созданием программы одновременно зани­маются несколько человек;

Premium - издание, включающее дополнительные инструмен­ты для работы и исходным кодом программ и создания баз дан­ных;

Ultimate - наиболее полное издание Visual Studio, включаю­щие все доступные инструменты для написания, тестирования, отладки и анализа программ, а также дополнительные инстру­менты для работы с базами данных и проектирования архитек­туры ПО.

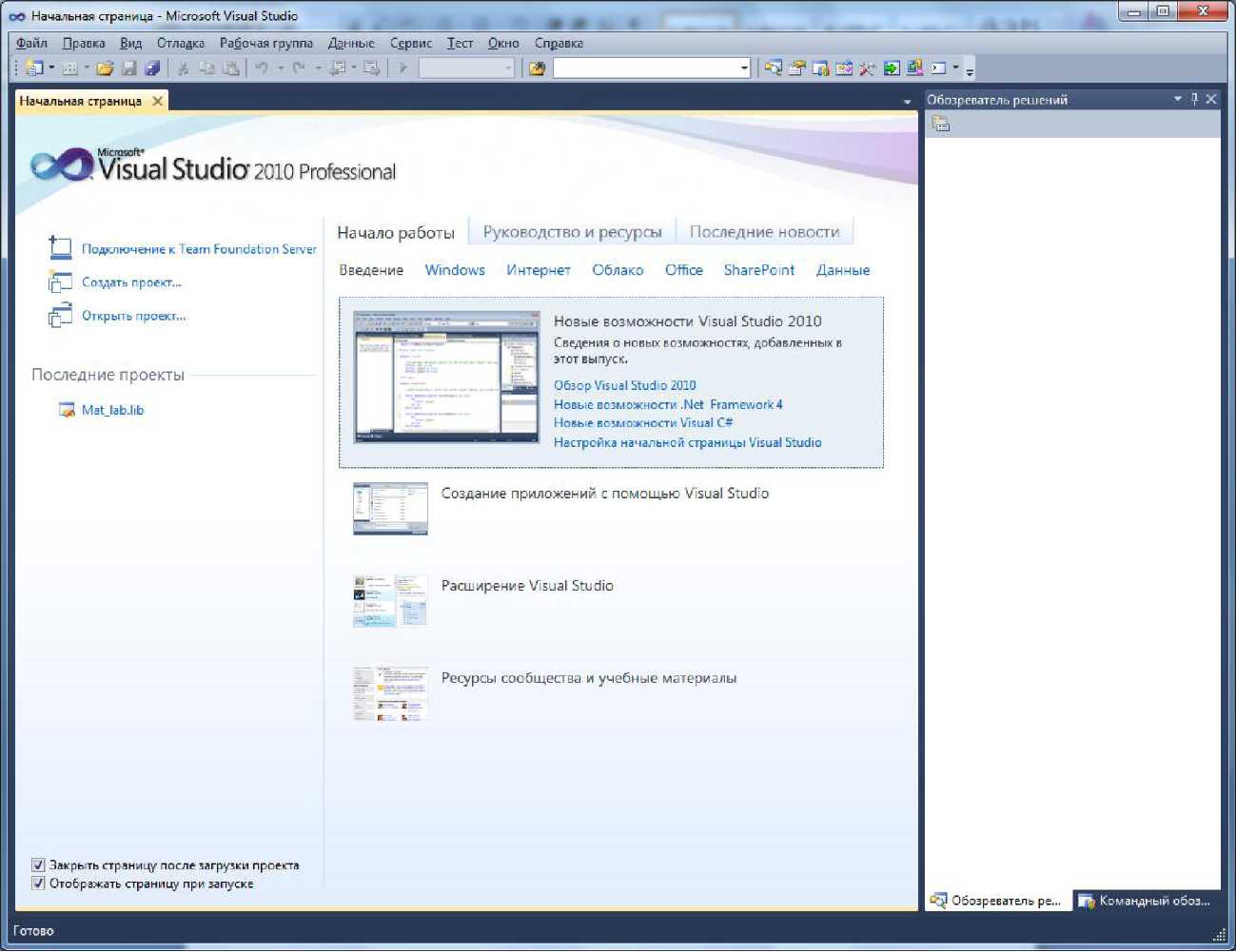


Рис. 0.2. Создание проекта

Отличительной особенностью среды Microsoft Visual Studio 2010 является то, что она поддерживает работу с несколькими язы­ками программирования и программными платформами. Поэтому, перед тем, как начать создание программы на языке С#, необходи­мо выполнить несколько подготовительных шагов по созданию проекта и выбора и настройки компилятора языка С# для трансля­ции исходного кода.

После щелчка на кнопке ОК среда создаст решение и проект с указанным именем. Примерный вид экрана приведен на рис. 2.

В верхней части экрана располагается главное меню (с разде­лами Файл, Правка, Вид и т. д.) и панели инструментов. Панелей инструментов в среде великое множество, и если включить их все, они займут половину экрана. При создании проекта необходимо указать язык C# и тип проекта

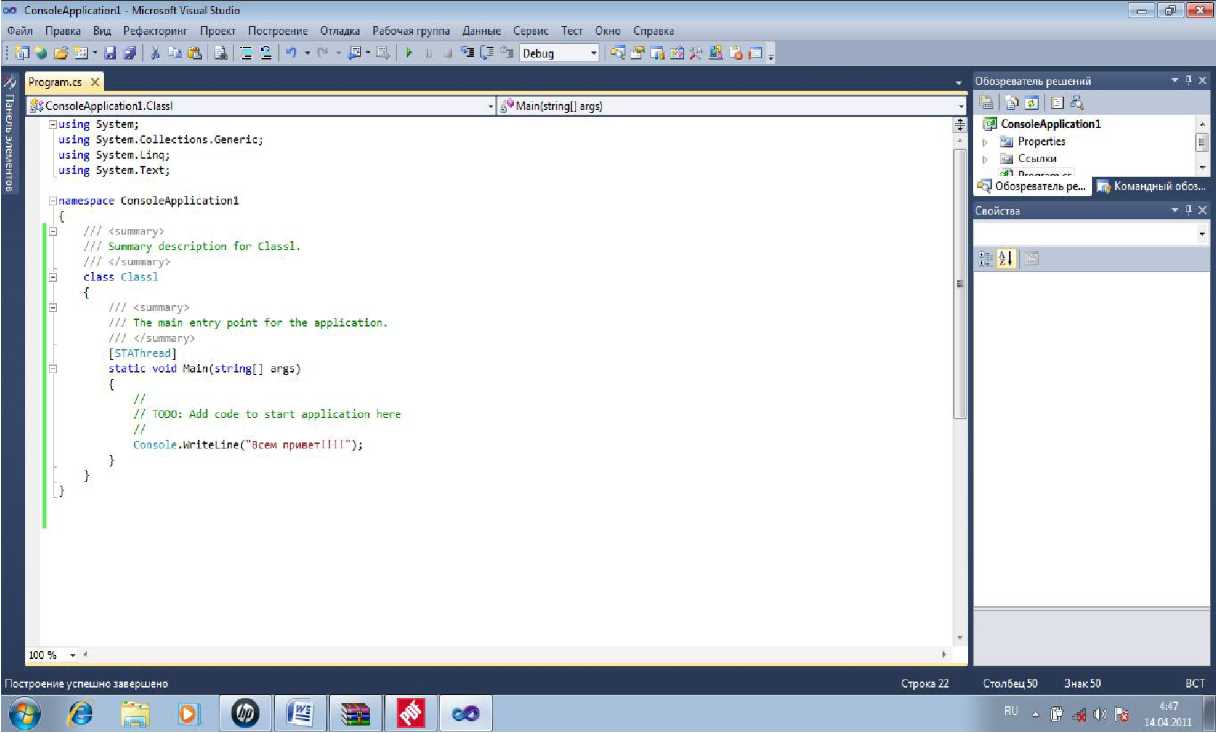


Рис. 0.3. Примерный вид экрана после создания проекта консольного приложения

В верхней правой части экрана располагается окно управления проектом. Обозреватель решений (если оно не отображается, сле­дует воспользоваться командой Вид • Обозреватель решений гла­вного меню). В окне перечислены все ресурсы, входящие в проект: ссылки на библиотеку, и информация о сборке и файл с исходным текстом класса ([Classl.cs](http://Classl.cs)). В этом же окне можно увидеть и дру­гую информацию,если перейти на вкладку Командный обозрева­тель, ярлычок которой находится в верхней части окна. На этой вкладке представлен список всех классов, входящих в приложение, их элементов и предков. С помощью проводника Windows можно увидеть какие файлы создала среда для поддержки проекта. На за­данном диске появилась папка с указанным именем, содержащая несколько других файлов и вложенных папок. Среди них - файл проекта (с расширением csproj), файл решения (с расширением sln) и файл с кодом класса ([Classl.cs](http://Classl.cs)).

Основное пространство экрана занимает окно редактора, в ко­тором располагается текст программы, созданный средой автома­тически. Текст представляет собой каркас, в который программист добавляет код по мере необходимости. Ключевые (зарезервиро­ванные) слова отображаются синим цветом, комментарии (2 раз­личных типов) - серым и темно-зеленым, остальной текст - чер­ным.

Слева от текста находятся символы структуры: щелкнув на любом квадратике с минусом, можно скрыть соответствующий блок кода. При этом минус превращается в плюс, щелкнув на ко­тором, можно опять вывести блок на экран. Это средство хорошо визуально структурирует код и позволяет сфокусировать внимание на нужных фрагментах.

# ЛАБОРАТОРНАЯ РАБОТА 1

ВСТРОЕННЫЕ ТИПЫ ДАННЫХ В С#. МАССИВЫ. СТРОКИ. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

## Цель работы

* изучить классификацию типов данных и отличительные особенности синтаксических конструкций языка C# от C++;
* изучить базовые типы: Array, String, StringBuilder, а также средства стандартного ввода/вывода и возможности форматирования вы­вода;
* получить понятие о регулярных выражениях и их применении для поиска, замены и разбиения текста на синтаксические лексемы.

## ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

1. Проработать примеры программ 1-8, данные в теоретических сведениях. Создать на их основе программы. Получить результаты работы программ и уметь их объяснить. Внести их в отчет по работе с комментариями.
2. Выполнить задания на двумерный массив по указанию препода­вателя.
3. Точки на плоскости заданы координатами х и у, которые хра­нятся в двумерном массиве. Найти пару самых удаленных друг от друга точек.
4. Найти суммы элементов двухмерного массива целых чисел, расположенных на линиях, параллельных главной диагонали, и ниже ее.
5. Найти номер столбца двухмерного массива целых чисел, для которого среднеарифметическое значение его элементов максимально.
6. В двухмерном массиве вещественных чисел поменять местами строки и столбцы с одинаковыми номерами.
7. В двухмерном массиве целых чисел поменять местами столбцы, симметричные относительно середины массива (вертикальной линии).
8. В двухмерном массиве целых чисел поменять местами строки, симметричные относительно середины массива (горизонтальной линии).
9. Поменять местами значения элементов двухмерного массива вещественных чисел, симметричных относительно побочной диагонали.
10. Найти максимальный элемент среди минимальных элементов строк двухмерного массива целых чисел. Определить номер строки и столбца для такого элемента.
11. Найти максимальный среди минимальных элементов столбцов двухмерного массива целых чисел. Определить номер строки и столбца для такого элемента.
12. Удалить столбец двухмерного массива вещественных чисел, в котором находится максимальный элемент этого массива.
13. Найти все неповторяющиеся элементы двухмерного массива целых чисел.
14. Заполнить двухмерный массив целыми числами от 1 до 100 по спирали.
15. Определить:

а) сумму элементов главной диагонали массива;

б) сумму элементов побочной диагонали массива;

в) среднее арифметическое элементов главной диагонали массива;

г) среднее арифметическое элементов побочной диагонали массива.

3. Выполнить задания на строки по указанию преподавателя. Ис­пользовать в задачах два класса строк: String и StringBuilder.

3.1. Текстовые сообщения часто печатаются строчными буквами, но многие сотовые телефоны имеют встроенные средства преобразования строчной буквы в прописную после символа пунктуации, как точка или знак вопроса. Составить программу, которая будет вводить сообщение в переменную String (на одной строке), а затем обрабатывать его с получением новой строки с прописными буквами в соответствующих местах.

1. Составить программу, которая будет вводить строку в переменную String. Подсчитать, сколько различных символов встречается в ней. Вывести их на экран.
2. Составить программу, которая будет вводить строку в переменную String. Найти в ней те слова, которые начинаются и оканчиваются одной и той же буквой.
3. Составить программу, которая будет вводить строку в переменную String. Определить, сколько раз в строке встречается заданное слово.
4. Строка, содержащая произвольный русский текст, состоит не более чем из 200 символов. Написать, какие буквы и сколько раз встречаются в этом тексте. Ответ должен приводиться в грамматиче­ски правильной форме: например: а - 25 раз, к - 3 раза и т. д.
5. Двумерный массив n x m содержит некоторые буквы русского алфавита, расположенные в произвольном порядке. Написать программу, проверяющую, можно ли из этих букв составить данное слово S. Каждая буква массива используется не более одного раза.
6. Составить программу, которая будет вводить строку в переменную String. Удалить из нее все лишние пробелы, оставив между словами не более одного. Результат поместить в новую строку.
7. Составить программу, которая будет вводить строку в переменную String. Напечатать в алфавитном порядке все слова из данной строки, имеющие заданную длину n.
8. Составить программу, которая будет вводить строку в переменную String. Найти слово, встречающееся в каждом предложении, или сообщить, что такого слова нет.
9. Дана строка, содержащая текст на русском языке. В предложениях некоторые из слов записаны подряд несколько раз (предложение заканчивается точкой или знаком восклицания). Получить в новой строке отредактированный текст, в котором удалены подряд идущие вхождения слов в предложениях.
10. Даны две строки А и B. Составьте программу, проверяющую, можно ли из букв, входящих в А, составить В (буквы можно использовать не более одного раза и можно переставлять). Например, А: ИНТЕ­ГРАЛ; В: АГЕНТ - составить можно; В: ГРАФ - нельзя.
11. Дана строка, содержащая текст на русском языке. Заменить все вхождения заданного слова на другое.

3.13. С клавиатуры вводится предложение, слова в котором разде­лены символом Напечатать все предложения, которые получаются путем перестановки слов исходного текста.

4. Выполнить задание на применение регулярных выражений.

1. Задан текст. Определить, входит ли в него заданное слово и сколько раз.
2. Задан текст. Определить, является ли он кодом HTML : содер­жит теги <html>, <form>, <h1>.
3. Задан текст. Определить, является ли он текстом на англий­ском языке.
4. Задан текст. После каждой буквы «о» вставить сочетание «Ок».
5. Задан текст. Определить, является ли он текстом на русском языке.
6. Задан текст. Определить, содержит ли он цифры.
7. Задан текст. Определить, сколько предложений начинается со слова "Информатика".
8. Задан текст. Выбрать из него все семизначные номера теле­фонов.
9. Задан текст. Определить, содержит ли он цифры.
10. Задан текст. Выбрать из него все e-mail адреса.
11. Задан текст, содержащий буквы и цифры. Найти произведе­ние всех чисел в тексте.
12. Задано предложение. Распечатать все слова в столбик.
13. Задан текст. Определить количество согласных букв в нем и распечатать их.

## КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Для создания нового проекта C# заходим в меню File и далее выбираем New, Project. В появившемся окне New Project слева выбираем Visual C#, а справа тип приложения - Console Application. Дайте имя проекту - ConsoleHello, укажите, где будет храниться проект.

Главное окно редактора кода, в котором отображается програм­мный код, хранимый в файле [ConcoleHello.cs](http://ConcoleHello.cs), изображен на рис. 1.1. Ниже главного окна расположены окна вывода (Error List, Output), в которых выводится вся служебная информация.

В правой части окна находится Solution Explorer, где показывается список файлов, содержащийся в "решении", которое, может состоять из нескольких "проектов". Вкладки вверху главного окна позволяют легко перемещаться от одного открытого файла к другому.

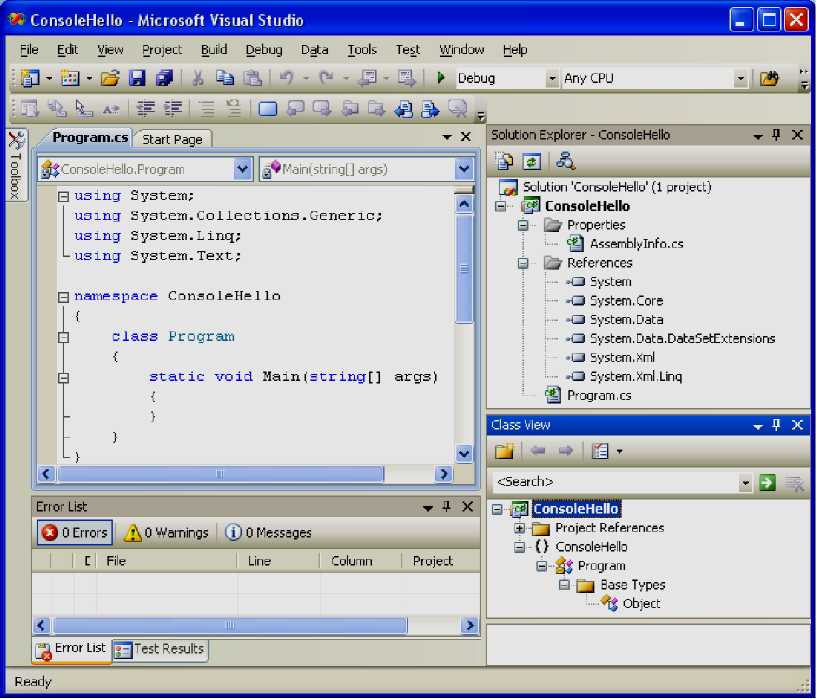


Рис. 1.1. Консольный проект в Visual Studio NET

Добавим в проект код:

namespace ConsoleHello // создаваемое пространство имен  
{

class Program // имя класса по умолчанию

{

static void Main(string[] args) // вывод строки на экран

{

Console.WriteLine("Введите Ваше имя");

string name;

name = Console.ReadLine(); // ввод строки с клавиатуры

if (name == "") Console.WriteLine("Здравствуй, мир!");

else Console.WriteLine("Здравствуй, " + name + "!");

}

}

}

## ТИПЫ ДАННЫХ C#

С# является типизированным языком. Необходимо всегда объявлять тип каждого объекта, который создаете.

C# подразделяет типы на два вида: встроенные типы, которые определены в языке, и определяемые пользователем типы, которые создает программист. Также все типы разделяются на две основные разновидности: размерные (структурные) типы (value-based) и ссылочные типы (reference-based). К структурным типам относятся все числовые типы данных (int, float и др.), а также перечисления и структуры. К ссылочным типам относятся массивы, строки и классы.

### **Встроенные типы**

Для каждого встроенного типа существует соответствующий тип в CRL (Common Language Runtime). Это означает, что каждый тип имеет два названия - полный (из CLR) и

сокращенный, используемый в C# (см. табл. 1.1).

Таблица 1. Имена встроенных типов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Логический тип | | | | | |
| Имя типа | Системный тип | Значения | | | Размер |
| bool | System.Boolean | true, false | | | 8 бит |
| Арифметические целочисленные типы | | | | | |
| Имя типа | Системный тип | Диапазон | | Размер | |
| sbyte | System.SByte | -128 ... - 128 | | Знаковое, 8 бит | |
| byte | System.Byte | 0 - 255 | | Беззнаковое, 8 бит | |
| short | System.Short | -32768 ... - 32767 | | Знаковое, 16 бит | |
| ushort | System.UShort | 0 - 65535 | | Беззнаковое, 16 бит | |
| int | System.Int32 | ~(-2 • 109 ... -2 • 109) | | Знаковое, 32 бит | |
| uint | System.UInt32 | ~(0 ... - 4 • 109) | | Беззнаковое, 32 бит | |
| long | System.Int64 | ~(-9 • 1018 ... 9 • 1018) | | Знаковое, 64 бит | |
| ulong | System.UInt64 | ~(0 ... 18 • 1018) | | Беззнаковое, 64 бит | |
| Арифметический тип с плавающей точкой | | | | | |
| Имя типа | Системный тип | Диапазон | | Точность | |
| float | System.Single | -1.5 • 10-45 ... +3.4 • 1038 | | 7 цифр | |
| double | System.Double | -5.0 • 10-324 ... +1.7- •10308 | | 15-16 цифр | |
| Арифметический тип с фиксированной точкой | | | | | |
| Имя типа | Системный тип | Диапазон | | Точность | |
| decimal | System.Decimal | -1.0 • 10-28 ... +7.9 • 1028 | | 28-29 значащих цифр | |
| Символьные типы | | | | | |
| Имя типа | Системный тип | Диапазон | | Точность | |
| char | System.Char | U+0000 - U+ffff | | Unicode символ | |
| string | System.String | Строка из символов Unicode | | | |
| Объектный тип | | | | | |
| Имя типа | Системный тип | | Примечание | | |
| Object | System.Object | | Прародитель всех типов | | |

В языке C# сглажено различие между типом и классом. Все типы одновременно являются классами, связанными отношением наследования. Родительским, базовым, классом является класс Object. Все остальные типы являются его потомками, наследуя методы этого класса. У класса Object есть четыре наследуемых метода:

bool Equals(object obj) - проверяет эквивалентность текущего объекта и объекта, переданного в качестве аргумента;

System.Type GetType() - возвращает системный тип текущего объекта;

string ToString() - возвращает строку, связанную с объектом. Для арифметических типов возвращается значение, преобразованное в строку;

int GetHashCode() - служит как хэш-функция в соответствующих алгоритмах поиска по ключу при хранении данных в хэш-таблицах.

Естественно, что все встроенные типы нужным образом переопре­деляют методы родителя и добавляют собственные методы и свойства.

//определение целой переменной встроенного типа int x=11, v = new Int32(); v = 007;

//определение строковой переменной string s1 = "Agent";

#### Преобразования типов.

Преобразования в строковый тип всегда определены, поскольку все типы наследуют метод ToString(). Метод можно вызывать явно или он вызывается неявно, когда по контексту требуется преобразование к строковому типу:

public void ToStringTest()

{

string s -'Владимир Петров ", s1 =" Возраст: ";

int ux = 27;

s = s + s1 + ux.ToString();

Console.WriteLine (s);

s1 =" Зарплата: ";

float dy = (float)2700.50;

s = s + s1 + dy;

Console.WriteLine (s);

}

#### Преобразования строк в число.

Класс Convert пространства имен System обеспечивает необходимые преобразования между раз­личными типами. Класс содержит 15 статических методов вида

To<Type> (ToBoolean(), ...ToUInt64()). Все методы многократно пе­регружены.

public void FromStringTest()

{

s - 'Введите возраст ";

Console.WriteLine(s);

s1 = Console.ReadLine();

ux = Convert.ToUInt32(s1);

Console.WriteLine("Возраст: "+ ux);

Console.WriteLine("Введите зарплату ");

dy = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Зарплата: "+ dy);

}

Данные, читаемые с консоли методом ReadLine или Read, всегда представляют собой строку, которую затем необходимо преобразовать в нужный тип.

### **Ссылочные типы**

#### Массивы в C#

Массивом называют упорядоченную совокупность элементов од­ного типа. Число индексов характеризует размерность массива. При объявлении массива границы задаются выражениями. Если все грани­цы заданы константами, то такие массивы называются статическими. Если же выражения, задающие границы, зависят от переменных, то такие массивы называются динамическими, память им отводится в процессе выполнения программы.

В языке C# имеются одномерные массивы, массивы массивов и многомерные ступенчатые массивы.

##### ***Определение одномерных массивов:***

int[] k; //k - одномерный массив

k=new int [3]; //Определяем массив из трех целых

k[0]=-5; k[1]=4; k[2]=55; //Задаем элементы массива

Элементы массива можно задавать сразу при объявлении: int[] k = {-5, 4, 55};

##### ***Создание динамического массива:***

Console.WriteLine("Введите число элементов массива ");

int size = Int32.Parse(Console.ReadLine());

int[] A1 = new int[size]; //создание динамического массива

##### ***Определение многомерных массивов:***

int[,] k = new int [2,3];

Обратите внимение, что пара квадратных скобок только одна. Аналогично можно задавать многомерные массивы. Вот пример трехмерного массива:

int[,,] k = new int [10,10,10];

Многомерные массивы можно сразу инициализировать:

int[,] k = {{2,-2},{3,-22},{0,4}};

##### ***Определение ступенчатых массивов:***

int[][] k = new int [2][]; //Объявляем второй ступенчатый массив  
k[0]=new int[3]; //Определяем нулевой элемент  
k[1]=new int[4]; //Определяем первый элемент  
k[1][3]=22; //записываем 22 в последний элемент

Обратите внимание, что у ступенчатых массивов задается несколь­ко пар квадратных скобок (столько, сколько размерностей у массива), табл. 1.2.

Массив имеет два уровня. Можно считать, что у него три элемента, каждый из которых является массивом. Для каждого внутреннего мас­сива необходимо вызвать конструктор new.

##### ***Базовый класс System.Array***

Все классы-массивы являются потомками класса Array из библиотеки FCL. Класс имеет большое число методов и свойств (табл. 1.3, 1,4). Благодаря такому родителю над массивами определены самые разнообразные операции - копирование, поиск, обращение, сортиров­ка, получение различных характеристик. Массивы можно рассматривать как коллекции и устраивать циклы foreach для перебора всех элементов.

Таблица 2 Создание ступенчатых массивов

|  |  |  |
| --- | --- | --- |
| Объявление и инициализация значениями | Объявление и инициализация нулевыми значениями | Объявление и инициализация нулевыми значениями |
| int[][] jagger = new int[3][] {  new int[] {5,7,9,11}, new int[] {2,8},  new int[] {6,12,4} | int[][] jagger1 = new  int[3][] {  new int[4], new int[2], new int[3]  }; | int[][] jagger2 =  {  new int[4], new int[2], new int[3] |

Таблица 3 Статические методы класса Array

|  |  |
| --- | --- |
| Метод | Описание |
| BinarySearch() | Двоичный поиск |
| Clear() | Выполняет начальную инициализацию элементов в зависимости от типа: 0 - для арифметического типа, false - для логического типа, null - для ссы­лок, "" - для строк |
| CopyTo() | Копирование части или всего массива в другой массив. Описание и примеры даны в тексте |
| GetLength() | Используется для определения количества эле­ментов в указанном измерении массива |
| IndexOf() | Индекс первого вхождения образца в массив. Описание и примеры даны в тексте |
| LastIndexOf() | Индекс последнего вхождения образца в массив. Описание и примеры даны в тексте |
| Reverse() | Обращение одномерного массива |
| Sort() | Сортировка одномерного массива встроенных ти­пов данных |
| GetVahie()  SetValue() | Возвращает или устанавливает значение указанно­го индекса для массива |

##### *Программа* **1.** Применение методов класса Array

public static int Main(string[] args)

{

string[]firstNames={“Саша”, "Маша", "Олег", "Света", "Игорь"};

Console.WriteLine("Here is the array:");

for(int i=0; i< firstNames.Length;

Console.WriteLine(firstNames[i]+"\t");

Console.WriteLine("\n");

Array.Reverse(firstNames);

for(int i=0; i< firstNames.Length;Console.WriteLine(firstNames[i]+"\t");

Console.WriteLine("\n");

Console.WriteLine("Cleared out all but one...");

Array.Clear(firstNames,1,4);

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

Console.WriteLine(firstNames[i]+"\t\n");

return 0;

}

В процедуре PrintAr формальный аргумент класса Array, следова­тельно, можно передавать массив любого класса в качестве фактиче­ского аргумента

##### **Программа 2.** Примение методов класса Array

public static void PrintAr(string name, Array A)

{

Console.WriteLine(name);

switch (A.Rank)

{

case 1:for(int i = 0; i<A.GetLength(0);i++)

Console.Write("\t" + name + "[{0}]={1}", i, A.GetValue(i)); Console.WriteLine();

break;

case 2:for(int i = 0; i<A.GetLength(0);i++)

{

for(int j = 0; j<A.GetLength(l);j++)

Console.Write("\t" + name + "[{0},{1}]={2}",A.GetValue(i,j));

Console.WriteLine();

}

break;

default: break;

}

}

#### Строки в С#

##### ***Класс Char.***

Использует двухбайтную кодировку символов Unicode. Константу можно задавать:

- символом, заключенным в одинарные кавычки;

- escape-последовательностью, задающей код символа;

- Unicode-последовательностью.

char ch1='A', ch2 ='\x5A', ch3='\u0058';

char ch = new Char(); int code;

string s; ch = ch1; //преобразование символьного типа в тип int

code = ch;

ch1=(char) (code +1) ;

//преобразование символьного типа в строку s = ch1.ToString()+ch2.ToString()+ch3.ToString();

Таблица 5 Статические методы и свойства класса

|  |  |
| --- | --- |
| Метод | Описание |
| GetNumericValue | Возвращает численное значение символа, если он является цифрой, и (-1) в противном случае |
| IsDigit | Возвращает true, если символ является десятичной цифрой |
| IsLetter | Возвращает true, если символ является буквой |
| Метод | Описание |
| IsLetterOrDigit | Возвращает true, если символ является буквой или цифрой |
| IsLower | Возвращает true, если символ задан в нижнем регист- |
| IsNumber | Возвращает true, если символ является числом (десятичной или шестнадцатеричной цифрой) |
| IsUpper | Возвращает true, если символ задан в верхнем регист- |
| ToLower | Приводит символ к нижнему регистру |
| ToUpper | Приводит символ к верхнему регистру |

Класс Char имеет большое число методов (см. табл.1.5).

Класс Char[] - массив символов. Можно использовать для пред­ставления строк постоянной длины. Массив char[] - это обычный мас­сив. Он не задает строку, заканчивающуюся нулем. В C# не определе­ны взаимные преобразования между классами String и Char[]. Для этого можно применить метод ToCharArray() класса String или посим­вольно передать содержимое переменной string в массив символов:

###### **Программа 3.** Массивы символов Char[]

string CharArrayToString(char[] ar)

{

string result-'";

for(int i = 0; i< ar.Length;

result += ar[i]; return(result);

}

void PrintCharAr(string name,char[] ar)

{

Console.WriteLine(name);

for(int i=0; i < ar.Length; Console.Write(ar[i]); Console.WriteLine();

}

public void TestCharArAndString()

{

string hello = "Здравствуй, Мир!";

char[] strM1 = hello.ToCharArray();

PrintCharAr("strM1",strM 1);

char[] World = new char[3];

Array.Copy(strM1,12,World,0,3); //копирование подстроки

PrintCharAr("World",World);

Console.WriteLine(CharArrayToString(World));

}

Класс Char[] является наследником классов Object и класса Array и обладает всеми методами родительских классов.

##### ***Класс String.***

Яляется основным типом при работе со строками. За­дает строки переменной длины. Над объектами этого класса определен широкий набор операций, соответствующий современному представ­лению о том, как должен быть устроен строковый тип. Объекты класса String объявляются как все прочие объекты простых типов - с явной или отложенной инициализацией, с явным или неявным вызовом кон­структора класса. Чаще всего при объявлении конструктор явно не вы­зывается, а инициализация задается строковой константой. Но у класса Sring достаточно много конструкторов. Они позволяют сконструиро­вать строку:

* из символа, повторенного заданное число раз;
* массива символов char[]; части массива символов.

string world = "Мир";

string sssss = new string('s',5);

char[] yes = "Yes".ToCharArray();

string stryes = new string(yes);

string strye = new string(yes,0,2);

Console.WriteLine("world = {0}; sssss={1}; stryes={2};"+ " strye= {3}", world, sssss, stryes, strye);

Над строками определены следующие операции:

* присваивание (=);
* две операции проверки эквивалентности (= =) и (!=);
* конкатенация или сцепление строк (+); взятие индекса ([]).

В результате присваивания создается ссылка на константную строку, хранимую в "куче". Операции, проверяющие эквивалент­ность, сравнивают значения строк, а не ссылки. Бинарная опера­ция "+" сцепляет две строки, приписывая вторую строку к хвосту первой. Взятие индекса при работе со строками отражает тот факт, что строку можно рассматривать как массив и получать каждый ее символ. Символ строки имеет тип char, доступный только для чтения, но не для записи.

Таблица 6 Статические методы и свойства класса

|  |  |
| --- | --- |
| Метод | Описание |
| Empty | Возвращается пустая строка. Свойство со статусом read only |
| Compare | Сравнение двух строк. Реализации метода позволяют сравнивать строки или подстроки, учитывать или не учи­тывать регистр, особенности национального формати­рования дат, чисел и т.д. |
| CompareOrdinal | Сравнение двух строк. Реализации метода позволяют сравнивать строки и подстроки. Сравниваются коды символов |
| Concat | Конкатенация строк. Допускает сцепление произволь­ного числа строк |
| Метод | Описание |
| Copy | Создается копия строки |
| Format | Выполняет форматирование в соответствии с заданны­ми спецификациями формата |
| Join | Конкатенация массива строк в единую строку. При конкатенации между элементами массива вставляются разделители |

Класс String относится к неизменяемым классам (immutable). Ни один из его методов не меняет значения существующих объектов. Ме­тоды создают новые значения и возвращают в качестве результата но­вые строки. Методы класса String описаны в табл. 1.6 и 1.7.

###### **Строковые константы.**

В C# существуют два вида строковых констант:

• обычные константы, которые представляют строку символов, за­ключенную в кавычки;

• @-константы, заданные обычной константой c предшествую­щим знаком @.

В обычных константах некоторые символы интерпретируются осо­бым образом. Например, управляющие символы, начинающиеся сим­волом "\". В @-константах все символы трактуются в полном соответ­ствии с их изображением. Пример задания констант: s1 = "c:\\c#book\\ch5\\chapter5.doc"; s2 = @"c:\c#book\ch5\chapter5.doc".

###### **Класс StringBuilder - построитель строк.**

Компенсирует недос­таток класса String. Класс принадлежит к изменяемым классам и нахо­дится в пространстве имен System.Text. Объекты класса объявляются с явным вызовом конструктора класса. Конструктор без параметров соз­дает пустую строку.

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;

Таблица 7. Динамические методы и свойства класса String

|  |  |
| --- | --- |
| Метод | Описание |
| Insert | Вставляет подстроку в заданную позицию |
| Remove | Удаляет подстроку в заданной позиции |
| Replace | Заменяет подстроку в заданной позиции на новую подстроку |
| Substring | Выделяет подстроку в заданной позиции |
| IndexOf,IndexOfAny, LastIndexOf,  LastIndexOfAny | Определяются индексы первого и последнего вхождения заданной подстроки или любого сим­вола из заданного набора |
| StartsWith, EndsWith | Возвращается true или false в зависимости от того, начинается или заканчивается строка заданной подстрокой |
| ToCharArray | Преобразование строки в массив символов |

public StringBuilder (string str, int start, int len, int cap). Параметры str, start, len задают строку инициализации, cap - емкость объекта. Над строками класса определены операции:

- присваивание (=);

- две операции проверки эквивалентности (= =) и (!=);

- взятие индекса ([]).

- Операция конкатенации (+) не определена, ее роль играет метод Append.

Со строкой этого класса можно работать как с массивом, допуска­ется не только чтение отдельного символа, но и его изменение.

###### **Программа 4.** Строки класса StringBuilder

public void TestStringBuilder()

{

StringBuilder s1 =new StringBuilder("ABC"),

s2 =new StringBuilder("CDE"),

s3 = new StringBuilder();

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);

StringBuilder s = new StringBuilder("Zenon");

s[0]='L'; Console.WriteLine(s);

}

###### Основные методы класса:

public StringBuilder Append (<объект>). К строке, вызвавшей ме­тод, присоединяется строка, полученная в качестве параметра. Метод перегружен и может принимать на входе объекты всех простых ти­пов. В качестве результата возвращается ссылка на объект, вызвав­ший метод.

public StringBuilder Insert (int location,<объект>). Метод вставляет строку в позицию, указанную параметром location.

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 strl 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. Полученные в результате форматиро­вания строки присоединяются в конец исходной строки.

За исключением метода Remove, все рассмотренные методы явля­ются перегруженными.

###### **Пространство имен RegularExpression**

Регулярные выражения - это один из способов поиска подстрок (соответствий) в строках. Осуществляется с помощью просмотра стро­ки в поисках некоторого шаблона (табл. 1.8). Очень эффективны биб­лиотеки, интерпретирующие регулярные выражения, обычно пишутся на низкоуровневых высокопроизводительных языках (С, C++, Assembler). С помощью регулярных выражений выполняются три дей­ствия:

проверка наличия соответствующей шаблону подстроки;

поиск и выдача пользователю соответствующих шаблону под­строк;

замена соответствующих шаблону подстрок.

Синтаксис регулярных выражений. Регулярное выражение на C# задается строковой константой. Обычно используется @-константа. В С# работа с регулярными выражениями выглядит сле­дующим образом:

Regex re = new Regex(«образец», «опции»);

MatchCollection me = re.Matches("строка для поиска");

где re - это объект типа Regex. В конструкторе ему передается образец поиска и опции.

Таблица 8. Символы описания

|  |  |
| --- | --- |
| Символ | Интерпретация |
|  | Категория: подмножества (классы) символов |
| • | Соответствует любому символу, за исключением символа конца строки |
| [aeiou] | Соответствует любому символу из множества, заданного в квадратных скобках |
| [^aeiou] | Отрицание. Соответствует любому символу, за исключе­нием символов, заданных в квадратных скобках |
| [0-9a-fA-F] | Задание диапазона символов, упорядоченных по коду. Так, 0-9 задает любую цифру |
| Символ | Интерпретация |
| \w | Множество символов, используемых при задании иден­тификаторов - большие и малые символы латиницы, цифры и знак подчеркивания |
| \s | Соответствует символам белого пробела |
| \d | Соответствует любому символу из множества цифр |
|  | Категория: Операции (модификаторы) |
| \* | Итерация. Задает ноль или более соответствий; например, \w\* или (abc)\*. Аналогично {0,} |
| + | Положительная итерация. Задает одно или более соответ­ствий; например, \w+ или (abc)+. Аналогично {1,} |
| ? | Задает ноль или одно соответствие; например, \w? Или (abc)? Аналогично {0,1} |
| {n} | Задает в точности n соответствий; например, \w{2} |
|  | Задает, по меньшей мере n соответствий; например, (abc){2,} |
|  | Категория: Группирование |
| (?<Name>) | При обнаружении соответствия выражению, заданному в круглых скобках, создается именованная группа, кото­рой дается имя Name |
| () | Круглые скобки разбивают регулярное выражение на группы. Для каждого подвыражения, заключенного в круглые скобки, создается группа, автоматически полу­чающая номер |

Класс Regex. Это основной класс, объекты которого определяют регулярные выражения. В конструктор класса передается в качестве

параметра строка, задающая регулярное выражение. Основные методы класса Regex:

метод Match запускает поиск первого соответствия. Параметром передается строка поиска. Метод возвращает объект класса Match, описывающий результат поиска.

###### Программа 5. Поиск первого соответствия шаблону

string FindMatch(string str, string strpat)

{

Regex pat = new Regex(strpat);

Match match =pat.Match(str);

string found = "";

if (match.Success)

{

found =match. Value;

Console.WriteLine("Строка ={0}\Юбразец={1}^ Найдено={2}", str,strpat,found);

}

return(found);

}

public void TestSinglePat()

{

string str,

strpat,

found;

Console.WriteLine("Поиск по образцу"); //образец задает подстроку, начинающуюся с

//символа a, далее идут буквы или цифры.

str ="start";

strpat =@"a\w+";

found = FindMatch(str,strpat); //art str ="fab77cd efg";

found = FindMatch(str,strpat); //ab77cd //образец задает подстроку, начинающуюся

//с символa а, заканчивающуюся f с возможными символами b и d в середине

strpat = "a(b|d)\*f';

str = "fabadddbdf';

found = FindMatch(str,strpat); //adddbdf

}

- метод Matches позволяет разыскать все непересекающиеся вхо­ждения подстрок, удовлетворяющие образцу. В качестве результата возвращается объект MatchCollection, представляющий коллекцию объектов Match.

###### **Программа 6.** Поиск всех соответствий шаблону

void FindMatches(string str, string strpat)

{

Regex pat = new Regex(strpat);

MatchCollection match =pat.Matches(str);

Console.WriteLine(" Строка ={0}\Юбразец={1}^ Найдено={2}", str,strpat,match.Count);

}

Console.WriteLine("око и рококо"); strpat-'око"; str = "рококо";

FindMatches(str, strpat); //найдено одно соответствие

- метод NextMatch запускает новый поиск.

- метод Split является обобщением метода Split класса String. Он позволяет, используя образец, разделить искомую строку на элементы.

static void Main()

{

string si = "Один, Два, Три, Строка для разбора";

Regex theRegex = new Regex(" |, |,");

int id = 1;

foreach (string substring in theRegex.Split(si));

Console.WriteLine("{0}: {1}", id++, substring);

}

метод Replace - позволяет делать замену найденного образца. Метод перегружен. При вызове метода передаются две строки: первая задает строку, в которой необходимо произвести замену, а вторая - на что нужно заменить найденную подстроку.

Regex r = new Regex(@"(a+)");

string s="bacghghaaab";

s=r.Replace(s,"\_$1\_"); // $1 - соответствует группе (а+)

Console.WriteLine("{0}",s);

Третий параметр указывает, сколько замен нужно произвести:

Regex r = new Regex(@"(dotsite)"); string s="dotsitedotsitedotsiterulez"; s=r.Replace(s,"f",1);

Console.WriteLine("{0}",s);

Четвертый параметр указывает, с какого вхождения производить замены:

Regex r = new Regex(@"(dotsite)"); string s="dotsitedotsitedotsiterulez"; s=r.Replace(s,"f',2,1); Console.WriteLine("{0}",s);

## Классы Match и MatchCollection.

Коллекция MatchCollection, по­зволяет получить доступ к каждому ее элементу - объекту Match. Для этого можно использовать цикл foreach.

При работе с объектами класса Match наибольший интерес пред­ставляют свойства класса. Рассмотрим основные свойства:

- свойства Index, Length и Value наследованы от прародителя Capture. Они описывают найденную подстроку - индекс начала под­строки в искомой строке, длину подстроки и ее значение;

- свойство Groups класса Match возвращает коллекцию групп -объект GroupCollection, который позволяет работать с группами, соз­данными в процессе поиска соответствия;

- свойство Captures, наследованное от объекта Group, возвращает коллекцию CaptureCollection.

### **Программа 7.** Поиск всех образцов, соответствующих регулярно­му выражению

public static void Main( )

{

string si = "Это строка для поиска"; // найти любой пробельный символ следующий за

//непробельным

Regex theReg = new Regex(@"(\S+)\s"); // получить коллекцию результата поиска

MatchCollection theMatches = theReg.Matches (si); // перебор всей коллекции

foreach (Match theMatch in theMatches)

{

Console.WriteLine( "theMatch.Length: {0}",

theMatch.Length);

if (theMatch.Length != 0)

Console.WriteLine("theMatch: {0}", theMatch.ToString( ));

}

}

## Классы Group и GroupCollection.

Коллекция GroupCollection воз­вращается при вызове свойства Group объекта Match. Имея эту кол­лекцию, можно добраться до каждого объекта Group.

Свойства создаваемых групп:

при обнаружении одной подстроки, удовлетворяющей условию поиска, создается не одна группа, а коллекция групп;

• группа с индексом 0 содержит информацию о найденном соот­ветствии;

число групп в коллекции зависит от числа круглых скобок в за­писи регулярного выражения. Каждая пара круглых скобок создает дополнительную группу;

группы могут быть индексированы, но могут быть и именован­ными, в круглых скобках разрешается указывать имя группы.

Создание именованных групп крайне полезно при разборе строк, содержащих разнородную информацию. Например:

### **Программа 8.** Создание именованных групп

public static void Main( )

{

string stringl = "04:03:27 127.0.0.0 [GotDotNet.ru](http://GotDotNet.ru)";

Regex theReg = new Regex( @"(?<время>0#:)+)^" + @"(?<ip>(\d|\ .)+)\s" + @"(?<url>\S+)");

// группа time - одна и более цифр или двоеточий, за которыми сле­дует пробельный символ

// группа ip адрес - одна и более цифр или точек, за которыми сле­дует пробельный символ

// группа url - один и более непробельных символов MatchCollection theMatches = theReg.Matches (stringl); foreach (Match theMatch in theMatches)

{

if (theMatch.Length != 0)

{ // выводим найденную подстроку

Console.WriteLine("\ntheMatch: {0}", theMatch.ToString ()); // выводим группу "time"

Console.WriteLine ("время: {0}", theMatch.Groups["время"]); // выводим группу "ip"

Console.WriteLine("ip: {0}", theMatch.Groups["ip"]); // выводим группу "url"

Console.WriteLine("url: {0}", theMatch.Groups["url"]);

}

}

}

# **ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ**

1. На какие группы и разновидности разделяются все типы данных в C#? Примеры.
2. Основное отличие структурных типов от ссылочных?
3. Перечислить все целочисленные арифметические типы данных в C# и их названия в CLR. Указать их диапазон и размер занимаемой памяти.
4. Перечислить все вещественные арифметические типы данных в C# и их названия в CLR. Указать их диапазон и размер занимаемой памяти.
5. Перечислить все символьные типы данных в C# и их названия в CLR. Указать их диапазон и размер занимаемой памяти.
6. Оператор цикла foreach и его применение в программах.
7. Определение одномерного массива в С#. Инициализация одно­мерного массива.
8. Определение многомерного массива в С#. Инициализация мно­гомерного массива.
9. Определение ступенчатых массивов и их инициализация.
10. Базовый класс Array, его методы и свойства.
11. Тип char и принимаемые значения переменными типа char. Ме­тоды и свойства класса Char.
12. Тип char[] и его отличительные особенности от С/С++.
13. Тип string и способы его конструирования.
14. Операции над строками типа string.
15. Способы задания строковых констант.
16. Методы и свойства класса String.
17. Класс StringBuilder. Конструкторы.
18. Операции над строками StringBuilder.
19. Основные методы класса StringBuilder.
20. Что такое регулярные выражения? Для чего применяются регу­лярные выражения?
21. Задание регулярного выражения. Поиск подстрок с помощью ре­гулярного выражения.
22. Класс Regex и его методы.
23. Поиск первого вхождения образца с текст.
24. Поиск всех вхождений образца в текст.
25. Замена образца в тексте.
26. Разбор текста на лесемы.
27. Свойства класса Match.
28. Классы Match, MatchCollection и их свойства.
29. Классы Group, GroupCollection и их свойства