Анатомия класса.
Схожесть с др. языками

- C# похож на язык Java в том, что он требует, чтобы вся программная логика была заключена в определения типов
- В отличие от C (и C++) глобальные функции и глобальные переменные в чистом видев C# использовать нельзя.

Определение простейшего класса в C#

```
using System:
class HelloClass
{
public static int Main(string[] args)
    {
    Console.WriteLine ("Hello, World");
    return 0;
    }
}
```

Определение простейшего класса в C#

ключевое слово public в определении метода означает, что этот метод будет доступен извне, а ключевое слово static говорит о том, что этот метод выступает как класс, а не как отдельный объект и будет доступен ,когда еще не создано ни одного экземпляра объекта данного класса.

Объявление **Main()**

```
public static void Main (string[] args);
    public static void MaIn()
  {
  public static int MaIn()

  }
```

Обработка параметров строки

```
using System;
class HelloClass
   {
public static int Main (string[] args)
      {
    for(int x=0; x < args. Length; x++)
       {
```

Обработка параметров строки

```
    Console. WnteLine(“"Arg: {0}", args[x]):
    }
Console. WriteLine("Hello. World!");
return 0;
  }
}
```

Создание объектов: конструкторы

«Класс» и «объект» - два разных понятия.

В отличие от класса объектом называется конкретный экземпляр определенного класса, с помощью которого обычно и производятся определенные действия.

Для создания объекта используется слово «new»

Создание объектов: конструкторы

```
Using System;
class HelloClass
 {
   public static int Main(string[] args)
    {
    HelloClass cl = new HelloClassO;
    HelloClass c2;
    c2 = new HelloClassO;
    return 0;
   }
 }
```

конструкторы

```
using System;
class HelloClass
  {
   public HelloClass()
       {
    Console.WriteLine(“Default ctor called!")
       }

   public HelloClass(int x, int y)
       {
```

конструкторы

```
    Console.WriteLine("Custom ctor called!");
    intX = x;
    intY = y;
}

public int intX, intY;
public static int Main(string[] args)
```

конструкторы

```
{
HelloClass c1 = new HelloClass();

Console.WriteLine("c1.intX = {0)\nc1.intY = {1}\n",
    c1.IntX, c1.intY);

HelloClass c2 = new HelloClass(100, 200);
Console.WnteLine("c2.intX = {0}\nc2.intY = {I}\n", c2.intX.
    c2.intY);
return 0;
```

Инициализация членов

можно инициализировать переменные прямо в момент их объявления:

```
class Text
{
private int MyInt = 90;
private string MyString = "My initial value";
private HotRod viper = new HotRod {200,
   "Chucky“, Color.Red);
}
```

Ввод и вывод

В большинстве созданных нами приложений использовался класс System. Console —

один из многих классов, определенных внутри пространства имен System.

Главные методы класса Console — это методы ReadLine() и WriteLlne() (оба этих метода определены как статические).

Средства форматирования строк в C#

```
using System;
class BasicIO()
{
public static void Main(string[] args)
{
int theInt = 90:
```

Средства форматирования строк в C#

```
float theFloat = 9.99;

BasicIO my10 - new BasicIO();
Console.WnteLine(
“Int is: {0}\nFloat is: {1}\nYou are: {2}”,
 theInt, theFloat, myIO.ToStringO):
}
```

Средства форматирования строк в C#

В каждом подстановочном выражении при желании можно использовать параметры форматирования, представленные в табл.

Средства форматирования строк в C#

- C или c
  - Используется для вывода значений в денежном (currency) формате. По умолчанию перед выводимым значением подставляется символ доллара ($), хотя можно отменить подстановку этого символа при помощи объекта NumberFormatInfo
- D или d
  - Используется для вывода десятичных значений. После этого символа можно указать количество выводимых символов после запятой
- E или e
  - Для вывода значений в экспоненциальном формате

Средства форматирования строк в C#

- F или f
  - Вывод значений с фиксированной точностью
- G или g
  - Общий (general) формат. Применяется для вывода значений с фиксированной точностью или в экспоненциальном формате
- N или n
  - N или n Стандартное числовое форматирование с использованием разделителей (запятых) между разрядами
- X или x
  - Вывод значений в шесгнадцатеричном формате. Если вы использовали прописную X, то буквенные символы в шестнадцатеричных символах также будут прописными

Структурные типы.

Разрядность всех встроенных типов фиксирована и постоянна.

К структурными типам относятся все числовые типы данных (int, float и пр.), а также перечисления и структуры. Память для структурных типов выделяется из стека. При присвоении одного структурного типа другому присваивается его побитовая копия.

Ссылочные типы

Ссылочные типы (классы и интерфейсы) ведут себя совершенно по-другому.

Память для них выделяется не в стеке, а в области управляемой кучи. При копировании ссылочного типа создается еще одна ссылка, которая указывает на ту же область оперативной памяти.

Сравнение типов

Вопрос	Структурные	Ссылочные
Размещение	**В области стека**	**В области управляемой кучи**
Представление переменной	**В виде локальной копии типа**	**В виде указателя на область опер памяти**
Может выступать как базовый?	**Нет. Структурные типы всегда закрыты и дополнение их другими свойствами нет**	**Да, если этот ссылочный тип не определен внутренне как закрытый**

Сравнение типов

Передача параметров	Как значений (то есть передаются только локальные копии значений переменных)	Как ссылок
Переопределение Object. Finalize()	Нет. Структурные типы никогда не размещаются в куче, и поэтому к ним не применяется функция завершения	Да, но не напрямую
Существует ли контруктор	Да, все должны принимать параметры	Конечно!

System Object

Все типы данных от него.

Главные методы объекта System.Object:

- EqualsQ

для сравнения объектов ссылочных типов, но не структурных.

2. GetHashCode{)

Возвращает целочисленное значение, идентифицирующее конкретный экземпляр объекта данного типа

3. GetTypeC)

Метод возвращает объект Type(), полностью описывающий тот объект, из которого метод был вызван.

4.ToStringQ

Возвращает символьное представление объекта в формате <имя_пространства_имен>.<имя_класса>

5. Finalize()

Освободить все ресурсы, занятые объектом данного класса, перед удалением этого объекта.

6. MemberwiseCloneQ

создания еще одной ссылки на область, занимаемую объектом данного типа в оперативной памяти. Этот метод не может быть замещен.

Упаковка и распаковка

Предположим, что у нас есть переменная простого структурного типа данных — short:

```
// Создаем переменную типа short и присваиваем ей значение
short s = 25;
```

Процесс упаковки:

```
// Упаковываем переменно s:
object objShort = s:
```

Упаковка — это процесс явного преобразования структурного типа в ссылочный.

Обратная распаковка объекта

```
short anotherShort = (short)objShort;
```

Распаковка —это преобразование ссылки на объект в оперативной памяти обратно в структурный тип.