ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ , C# (C sharp)

Объе́ктно-ориенти́рованное программи́рование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

C# является полноценным объектно-ориентированным языком. Это значит, что программу на C# можно представить в виде взаимосвязанных взаимодействующих между собой объектов.

Описанием объекта является **класс**, а **объект** представляет экземпляр этого класса.
Можно еще провести следующую аналогию. У нас у всех есть некоторое представление о человеке, у которого есть имя, возраст, какие-то другие характеристики.

То есть некоторый шаблон - этот шаблон можно назвать классом. Конкретное воплощение этого шаблона может отличаться, например, одни люди имеют одно имя, другие - другое имя. И реально существующий человек (фактически экземпляр данного класса) будет представлять объект этого класса.

Типы значений:

- Целочисленные типы (byte, sbyte, short, ushort, int, uint, long, ulong)
- Типы с плавающей запятой (float, double)
- Тип decimal
- Тип bool
- Тип char
- Перечисления enum
- Структуры (struct)

Ссылочные типы:

- Тип object
- Тип string
- Классы (class)
- Интерфейсы (interface)
- Делегаты (delegate)

Все члены класса - поля, методы, свойства - все они имеют **модификаторы доступа**. Модификаторы доступа позволяют задать допустимую область видимости для членов класса. То есть модификаторы доступа определяют контекст, в котором можно употреблять данную переменную или метод. В предыдущих темах мы уже с ним сталкивались, когда объявляли поля класса публичными (то есть с модификатором public).

В C# применяются следующие модификаторы доступа:

public: публичный, общедоступный класс или член класса. Такой член класса доступен из любого места в коде, а также из других программ и сборок.

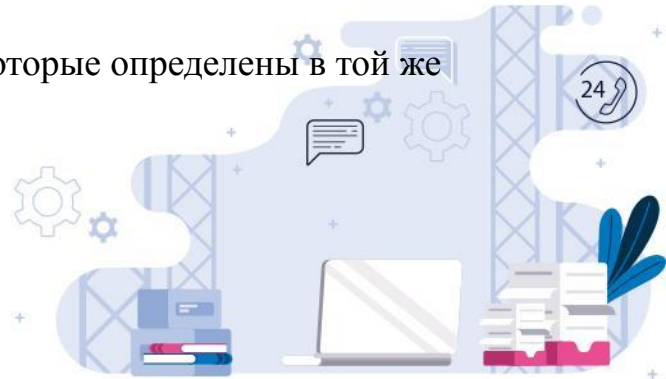
private: закрытый класс или член класса. Представляет полную противоположность модификатору public. Такой закрытый класс или член класса доступен только из кода в том же классе или контексте.

protected: такой член класса доступен из любого места в текущем классе или в производных классах. При этом производные классы могут располагаться в других сборках.

internal: класс и члены класса с подобным модификатором доступны из любого места кода в той же сборке, однако он недоступен для других программ и сборок (как в случае с модификатором public).

protected internal: совмещает функционал двух модификаторов. Классы и члены класса с таким модификатором доступны из текущей сборки и из производных классов.

private protected: такой член класса доступен из любого места в текущем классе или в производных классах, которые определены в той же сборке.

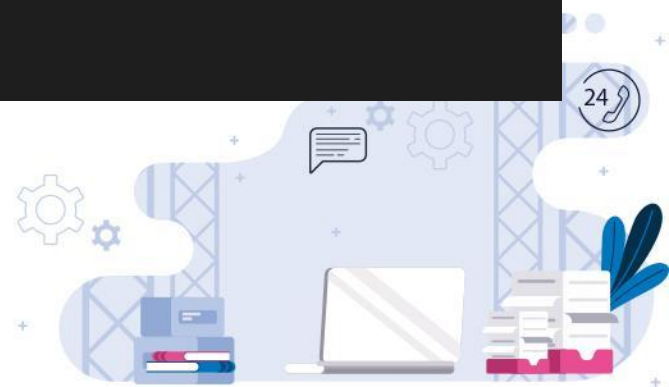


Кроме обычных методов в языке C# предусмотрены специальные методы доступа, которые называют **свойства**. Они обеспечивают простой доступ к полям классов и структур, узнать их значение или выполнить их установку.

Стандартное описание свойства имеет следующий синтаксис:

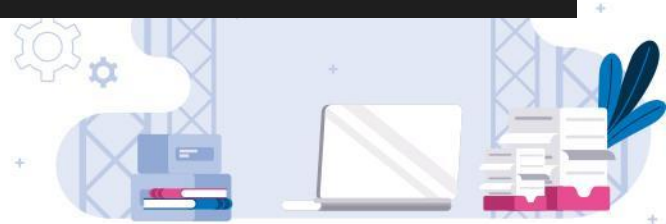
```
1  [модификатор_доступа] возвращаемый_тип произвольное_название
2  {
3      // код свойства
4  }
```

```
ссылка: 1
private void button3_Click(object sender, EventArgs e)
{
    //Конвертировать можно не только string в int но также и в double
    //float и тд.
    double a = Convert.ToDouble(textBox3.Text);
    double b = Convert.ToDouble(textBox4.Text);
    double c =Math.Pow(a,b);
    //для того чтобы возводить в степень или же нахожить корень
    //то нужно использовать тип переменных с плавающей точкой
    label2.Text = c.ToString();
}
```

```
ссылка: 1
private void button4_Click(object sender, EventArgs e)
    //при нажатии на кнопку 4
{
    string a = textBox1.Text;
    //Переменная а равна тому что мы ввели в textbox1
    label1.Text = a;
    //Выводим переменную а на label1
}
```

```
ссылка: 1
private void button2_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox1.Text);
    //Конвертируем переменную с типа string в int и записываем в a
    int b = Convert.ToInt32(textBox2.Text);
    //то же самое делаем с переменной b
    int c = a - b;//Математические вычисления
    label1.Text = c.ToString();
    //получившуюся переменную конвертируем с типа int в string
    //чтобы мы могли ее вывести в label
    //Label,textbox,listbox и тд используют тип string
}
```

1. Пользователь вводит два числа. Программа должна определить и вывести наибольшее из введенных чисел.
2. Квадратное уравнение имеет вид ax^2+bx+c=0. Пользователь вводит 3 числа (a, b, c) и программа должна вывести корни уравнения.
3. Вычислить и вывести факториал введенного пользователем числа.