# **Первое знакомство с C#**

**Числовые типы данных**

Переменная – именованная область памяти используется, чтобы хранить в ней объекты или ссылки на объекты

У каждой переменной есть тип.

При объявлении переменной сначала нужно написать её тип, затем название

int a = 45; (переменная a с типом int, в которой хранится число 45)

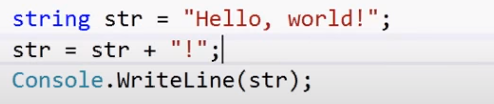
Типы переменных:

1. int – используется для целых чисел (1, 100, 34)
2. double – используется если нужно число с “плавающей точкой” (75.5, 45.6, 0.23)
3. long – для длинных целых чисел (30000000000)
4. float – тоже самое что и double с большей точностью для чисел с “плавающей точкой “ (56.34f)

чтобы каждый раз самому не объявлять тип переменной можно использовать var, тогда его определит компьютер

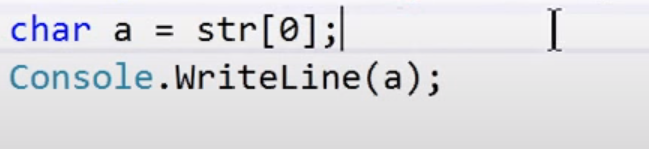
Строки

string - тип данных, значениями которого является произвольная последовательность (строка) символов алфавита.



Вывод: Hello, world!!

char – тип данных для отдельных символов (строки) определяется одинарными кавычками



Вывод: H



null – отсутствие всякой строки (даже пустой)

Ввод строки с консоли:



Чтобы конвертировать строку в число нужно использовать метод Parse:



Конверсия числа в строку:



Операции деления:

/ - целочисленное деление

3/2 = 1

3.0/2 = 1.54

% - Выводится остаток от деления

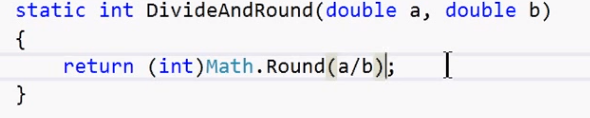
10%3 = 1

10%2 =0

Чтобы использовать сложные математические операции нужно использовать класс Math

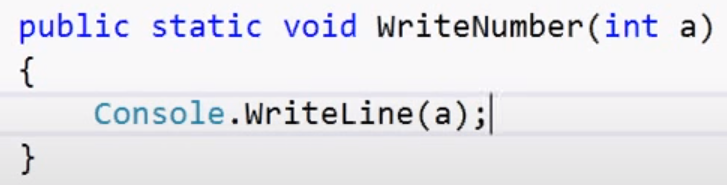
a++ - увеличивает значение на еденицу

Методы

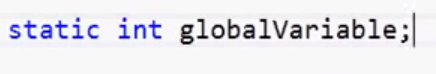


1. int – показывает что метод вернёт значение типа int
2. DivideAndRound – название метода
3. (double a, double b) – входные данные с которыми будет работать метод
4. return - то что вернёт метод

При такой записи метод не будет ничего возвращать

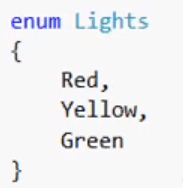


Глобальные переменные – переменные с которыми могут работать все методы



# **Ошибки**

enum - cущность, позволяющая решить заданную проблему



константа — способ адресации данных, изменение которых рассматриваемой программой не предполагается или запрещается.



# **Ветвления**

Сравнения

Для сравнений используются знаки:

1. == (определяет равны ли значения двух элементов),
2. != (определяет неравность двух элементов)
3. > , >=
4. <, <=

bool – тип переменной которая хранит либо True, либо False

Логические операции для переменных типа bool:

* && - Конъюнкция (И)
* || - Дизъюнкция (ИЛИ)
* ! – отрицание (НЕ)

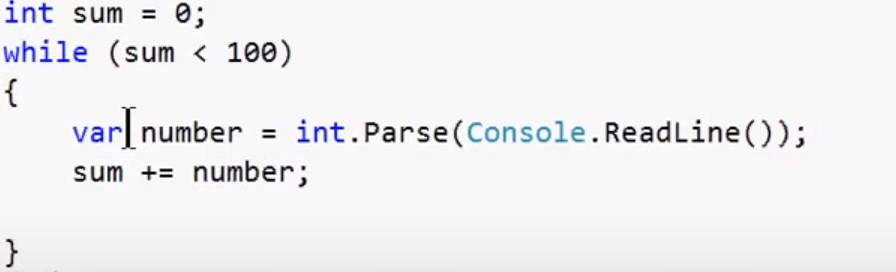
Логические выражения:

* if (Условие) Если условие правдиво произойдёт действие
* else Иначе другое действие
* else if – иначе если ещё одно действие

# **Циклы**

**while** (Условие)

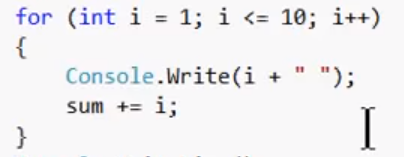
Пока выполняется условие, выполняются действия



Чтобы выйти из цикла досрочно можно написать break

Чтобы перейти к следующей итерации нужно написать continue

**for** (Инициализация переменной; условие работы цикла; действие после каждой итерации цикла)



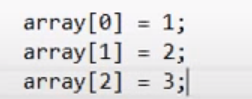
# **Массивы**

Массив — это структура данных, которая хранит упорядоченный набор однотипных элементов

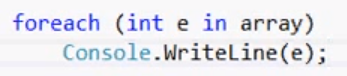
Объявление и инициализация массива с элементами типа int длинной 3



Действия с его элементами можно совершать с помощью индексаций



Для того чтобы пройтись по всем элементам массива используется цикл **foreach**



Массив может быть двумерный, трёхмерный и тд.

Пример двумерного массива:



# **Коллекции, строки, файлы**

**list**

Список (бесконечный массив) - это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза

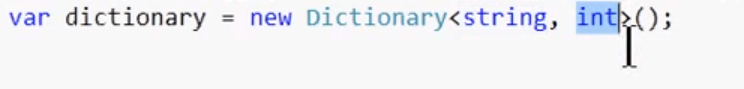
В списке может хранится сколько угодно элементов

Инициализация листа:



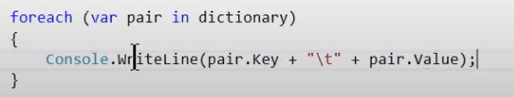
**Словари**

Инициализация словаря:

****

Dictionary<Ключ, Значение>

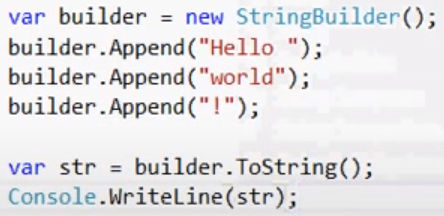
foreach’ем также можно проходится по словарю:



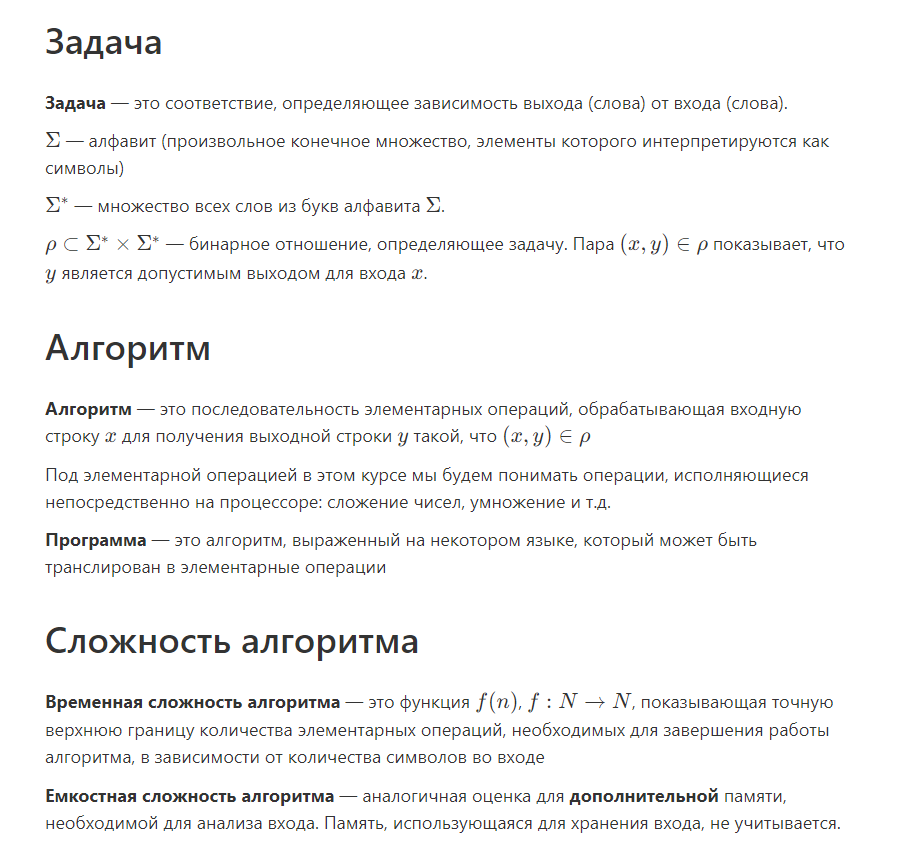
**СтрингБилдер**

В обычных строках мы не можем изменять элементы строки, добавлять или удалять подстроки. Для этого существует стрингбилдер

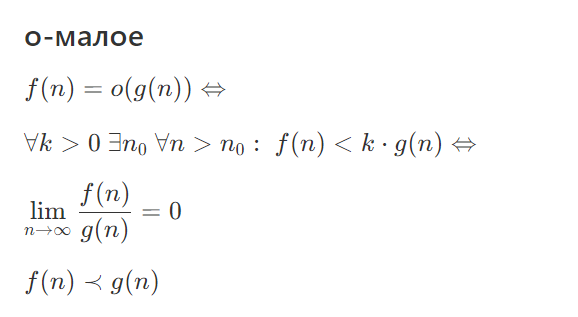
Пример работы со стрингбилдером

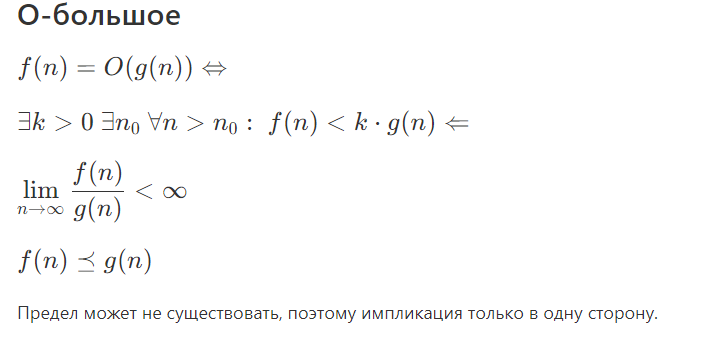


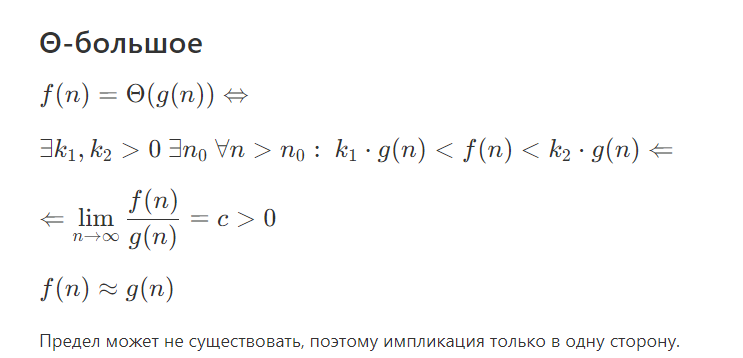
# **Сложность алгоритмов**



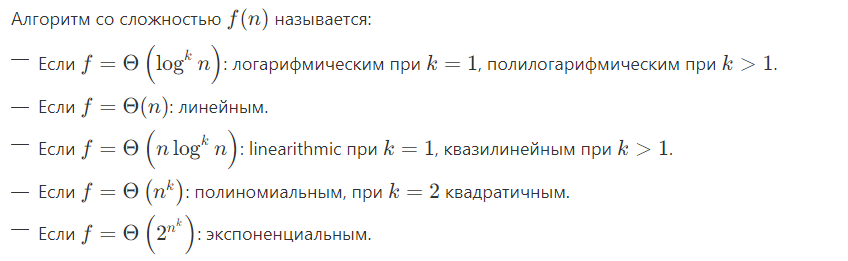
Оценки сложности алгоритма:







Классы сложности:



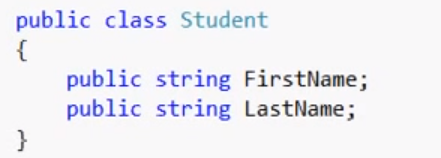
# **Основы ООП**

**Классы**

С помощью классов можно выделять сущности внутри которых могут лежать методы,

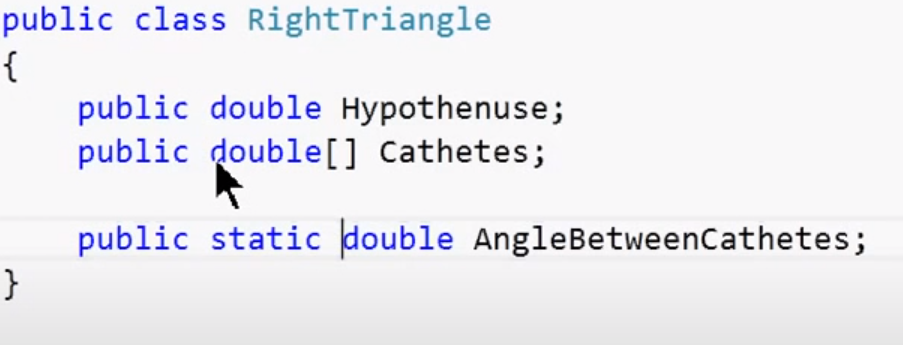
переменные и тд.

Пример:



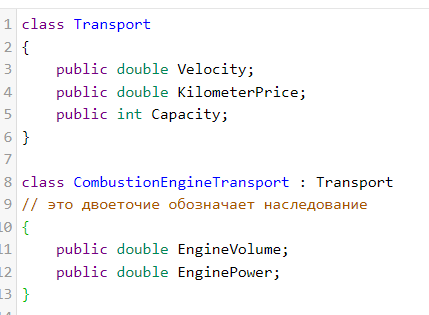
**Статическое и Динамическое**

Данные сущностей есть статические данные (данные одинаковые для каждой сущности) и динамические данные (индивидуальные данные для каждой сущности)

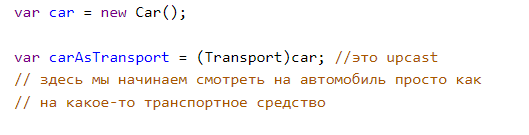


# **Наследование**

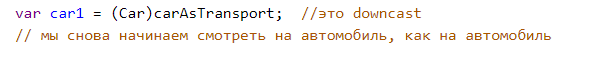
Для ситуаций когда класс должен содержать в себе элементы предыдущего класса и плюсом добавлять что-то новое существует наследование



Upcast – приведение типа от конкретного к общему

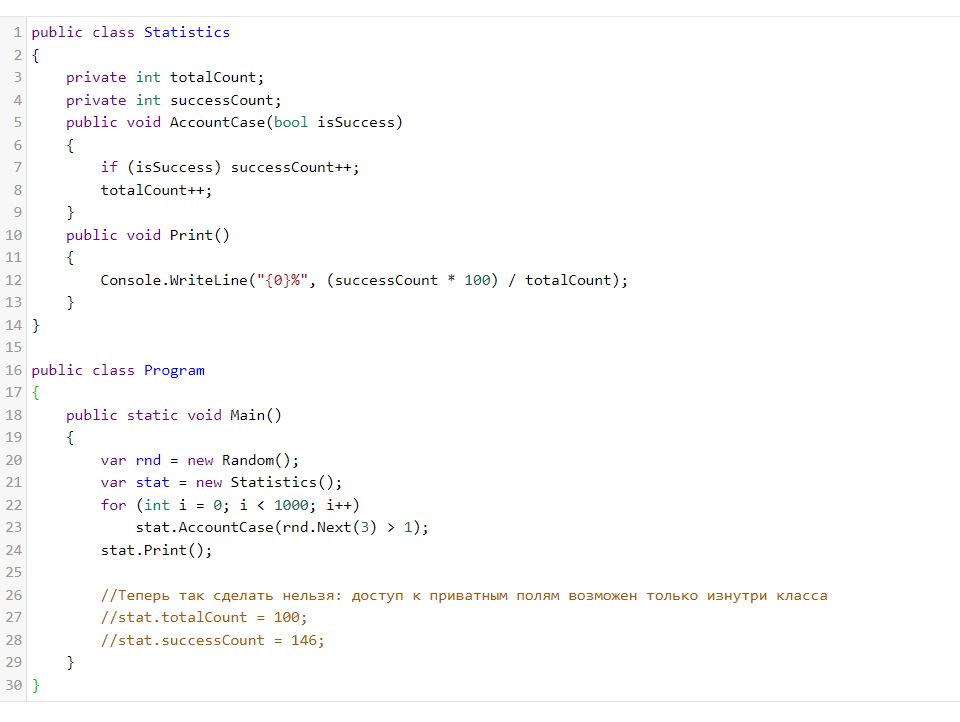


Downcast - приведение типа от более общего к более конкретному.

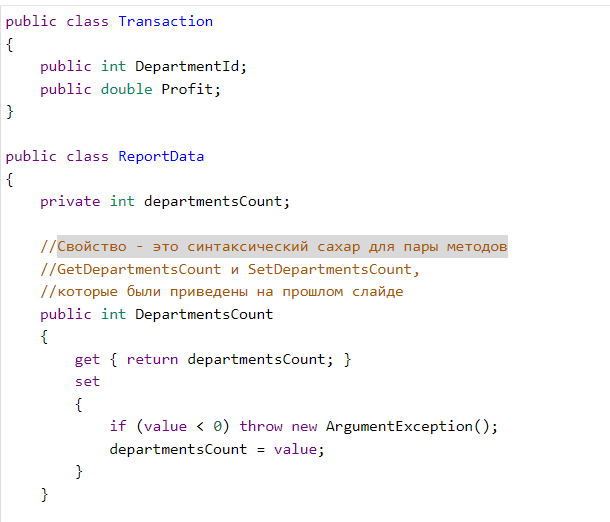


# **Целостность данных**

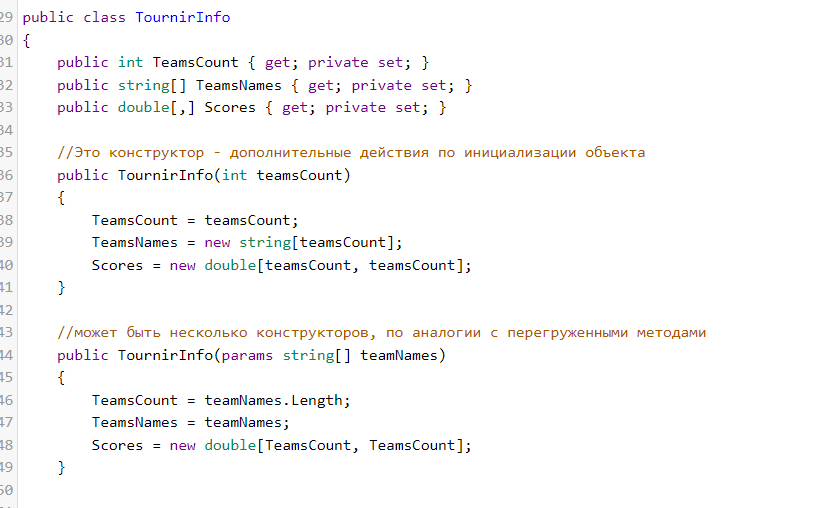
**Private** — наиболее строгий модификатор доступа. Он ограничивает видимость данных и методов пределами одного класса.

  
**Свойства**

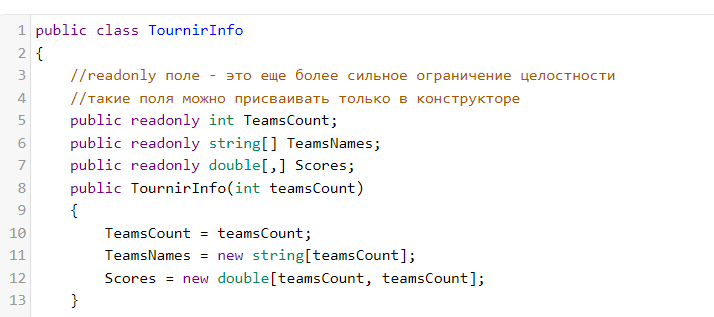
Свойство - это синтаксический сахар для пары методов set чтобы устанавливать значения и get, чтобы получать



Конструктор - это специальный метод, который используется для инициализации объекта. Каждый класс имеет конструктор, если мы явно не объявляем конструктор для любого класса C#, компилятор строит конструктор по умолчанию для этого класса.

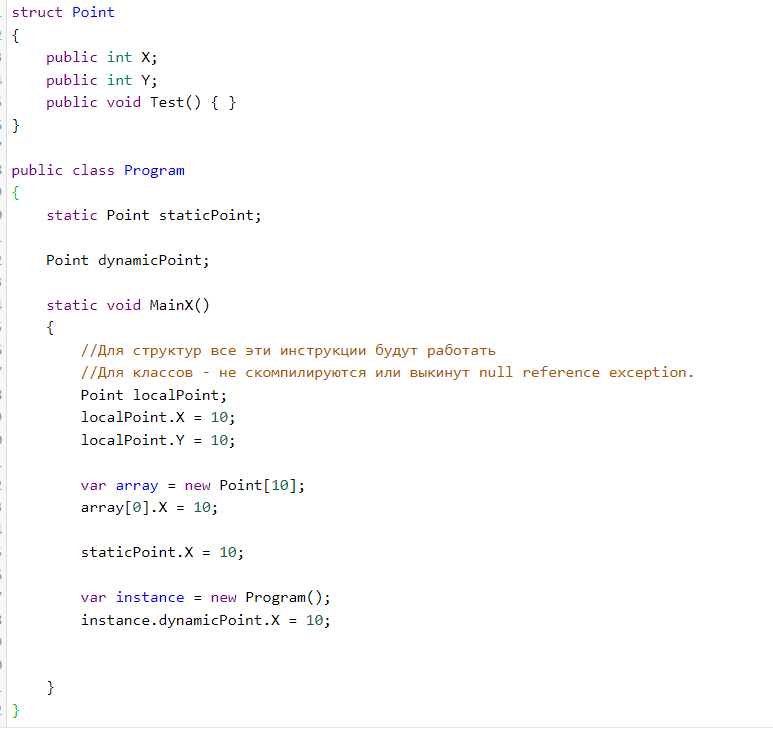


Поле readonly – не даёт изменять объект после его инициализации

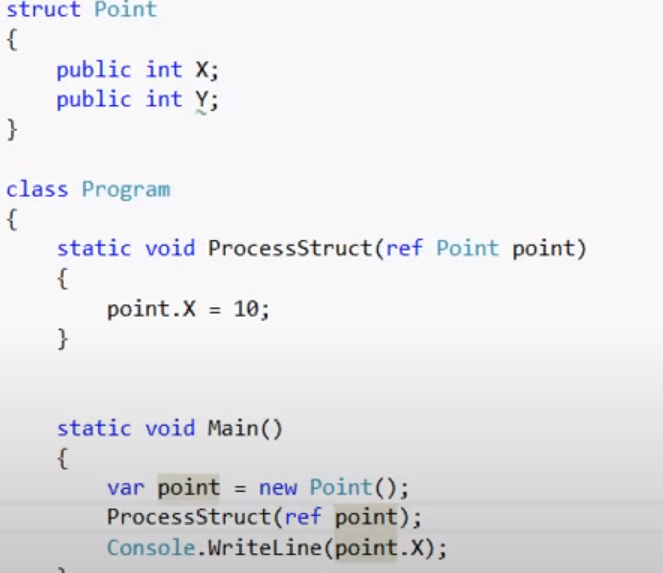


# **Структуры**

**Структура** — это пользовательский тип данных, который используется наряду с классами и может содержать какие-либо данные и методы. В отличии от класса у структуры собственный не ссылочный тип



**ref** передает аргументы по ссылке. Это означает, что любые изменения, внесенные в этот аргумент в методе, будут отражены в этой переменной, когда управление вернется к вызывающему методу.

****

Благодаря ref выведется 10