/\*

Комментарий

многострочный

\*/

// комментарий по строке

// комментарий по строке Ctrl+E,C

/// <summary>

/// Название подписание

/// </summary>

/// <param name="args">параметры запуска</param> - зачем параметры запуска  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

WriteLine("Hello World!"); показать

ReadKey(); получить

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

using System;  
using static System.Console;  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Переменная (имя) – идентификатор, «контейнер» хранения данных.

var a = 28;  
var b = 12,2;  
var c = “Текст”;

Числовые типы:  
  
Целые числа со знаком.  
sbyte int8 = 2; // от -128 до +127  
short int16 = 1; // от -32768 до +32767;  
int int32 = 0; // от - 2 147 483 648 до 2 147 483 647  
long int64 = -1; // от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807  
  
Целые числа без знака.  
byte uInt = 0; // от 0 до 255  
ushort uInt16 = 1; // от 0 до 65535  
uint uInt32 = 2; // от 0 до 4 294 967 296  
ulong uInt64 = 3; // от 0 до 18 446 744 073 709 551 616  
  
int \_numb = 1\_999\_222; для удобсва  
  
Числа с плавающей точкой.  
float floatD = 3.14F; // точность 7 цифр 1.1234567  
double doubleD = 3.14D; // точность 15 цифр 1.\*\*\*\*\*\*\*\*\*\*\*\* (d в конце необязательный)  
decimal decimalD = 3.14M; // точность 28 цифр 1.\*\*\*\*\*\*\*\*\*\*\*  
  
Console.WriteLine(floatVariable);  
Console.WriteLine(doubleVariable);  
Console.WriteLine(decimalVariable);  
Символьный - ‘a’ юникод символ  
char VarChar1 = ‘a’; Console.WriteLine(VarChar1);  
  
Логический – true, false

Bool flag = true;  
flag = false;  
  
Строковые типы – “Слова“  
String text = “текст”;

Контактенация (“Возвраст: “ + aslld)

String name = "Александр";

String nameFirst = "Лазарев";

Console.WriteLine("Меня зовут {0} можно писать внутри кавычек {1}",

name,

nameFirst);

String name = "Александр";

String nameFirst = "Лазарев";

String pattern = "Меня зовут {0} можно писать внутри кавычек {1}";

Console.WriteLine(pattern,

name,

nameFirst);

String fname = "Марина";

String lname = "Лазарева";

Console.WriteLine (pattern,

fname,

lname);

pattern пишем один раз, а используем в разных местах с разными значениями. Escape последовательности.

\n новая строка

\r возврат каретки

\t горизонтальная табуляция

\’ одиночная кавычка

\” двойная кавычка

\\ обратная косая черта

\? Литерал вопросительного знака

Интерполяция строк. Вывод строк значений переменных. $ перед кавычками.  
Можно подставить переменные {} напрямую, не заморачи С контактенацией.

int a = 3;

int b = 4;

int c = 5;

Console.WriteLine($"{a} + {b} = {c}");

String name = "Александр", lastName = "Лазарев";

String name1 = "Марина", lastName1 = "Лазарева";

Console.WriteLine($"{name} {lastName} \n{name1} {lastName1}");

Console.WriteLine($"{name1} {lastName1}");   
  
 Console.WriteLine($"{name,10} {lastName,14}");

Console.WriteLine($"{name1,10} {lastName1,14}");  
  
Числа количество отведенных символов под это значение.

Числовые типы, количество чисел после запятой и разделение пробелами

double d = 1233.1231231;

String dFormated = d.ToString("## ##.## ##");

Console.WriteLine(dFormated);

//Вывод по заданому формату ToString

Console.WriteLine("{0:00 00.00 00 0}", d);

//Вывод по строке и формату через :

Console.WriteLine(d.ToString("## ##.## ## #"));  
 //Вывод по заданому формату ToString

Дата (в зависимости от настроек языкового форматы могут различаться)  
Вывод даты в разных орматах  
  
 var date = new DateTime(2020, 02, 11, 01, 22, 58);

Console.WriteLine(date);

Console.WriteLine($"{date:HH:mm}");

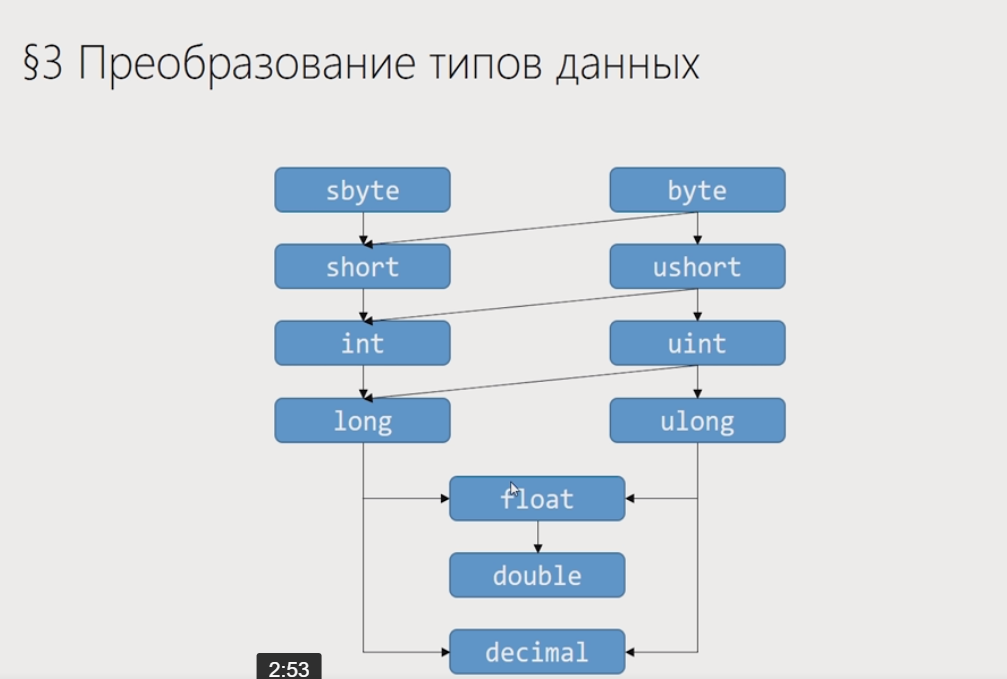
Console.WriteLine($"{date:yyyy-MM-dd}");

Console.WriteLine($"{date:dd.MM.yyy}");

------------------------------------------------------------------------------------

Числовые типы:  
  
Целые числа со знаком.  
sbyte int8 = 2; // от -128 до +127  
short int16 = 1; // от -32768 до +32767;  
int int32 = 0; // от - 2 147 483 648 до 2 147 483 647  
long int64 = -1; // от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807  
  
Целые числа без знака.  
byte uInt = 0; // от 0 до 255  
ushort uInt16 = 1; // от 0 до 65535  
uint uInt32 = 2; // от 0 до 4 294 967 296  
ulong uInt64 = 3; // от 0 до 18 446 744 073 709 551 616  
  
int \_numb = 1\_999\_222; для удобства  
  
Числа с плавающей точкой.  
float floatD = 3.14F; // точность 7 цифр 1.1234567  
double doubleD = 3.14D; // точность 15 цифр 1.\*\*\*\*\*\*\*\*\*\*\*\* (d в конце необязательный)  
decimal decimalD = 3.14M; // точность 28 цифр 1.\*\*\*\*\*\*\*\*\*\*\*

Что во что можно преобразовать.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Неявное преобразование, так как число max число байт, меньше max числа int

Меньшие типы чисел, в большие.  
**Неявные преобразования**. Специальный синтаксис не требуется, так как преобразование всегда завершается успешно и данные не будут потеряны. Примеры включают преобразования из меньших в большие целочисленные типы и преобразования из производных классов в базовые классы.

byte b = 100;

int a = b;

Console.WriteLine(a);

double d = a;

Console.WriteLine(d);

\_\_\_\_\_\_

byte byteValue = 1;

short shortValue = byteValue;

int intValue = shortValue;

long longValue = intValue;

double doubleValue = longValue;

char c = 'k';

int d = c;

с таблицы кодировок.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Явное преобразование. (приведение типов) Ответственность лежит на нас, данные могут быть некорректны и потеряны.

int i = 330;

byte j = i; - ошибка

int i = 330;

byte j = (byte)i; = 74

\_\_\_\_\_\_\_

double d = 1.77;

int h = (int)d;

Console.WriteLine(h);

int f = (byte)d; явное присвоение типа double к байт, int к byte.

Console.WriteLine(f);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Преобразования с использованием вспомогательных классов. Чтобы выполнить преобразование между несовместимыми типами.**   
Метод Parse встроенный внутри числовых типов int.Parse(String); double.Parse(String);

string inputString = "2029";

int intValue = int.Parse(inputString);

double doubleValue = double.Parse(inputString);

ulong ulongValue = ulong.Parse(inputString);

//byte byteValue = byte.Parse(inputString);

Console.WriteLine($"{intValue} {doubleValue} {ulongValue} byte тут не будет работать)))");

\_\_\_\_\_\_\_\_\_\_\_

Метод Convert встроенный внутри любой тип в любые типы.  
int – ToInt32 – синоним .NET посмотреть можно через F1, выделив тип данных.

string inputString = "2029";

int intValue = Convert.ToInt32(inputString);

long longValue = Convert.ToInt32(intValue); - приведение стринг-инт-лонг

ulong ulongValue = Convert.ToUInt64(inputString);

double doubleValue = Convert.ToDouble(longValue);

Console.WriteLine($"{intValue} {longValue} {ulongValue}");  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Посмотреть названия чисел .Net можно через F1 справка.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Арифметические операции   
+ - \* /**

int valueA = 2\_233\_222; int valueB = 778;

int valueC = valueA + valueB;

Console.WriteLine(valueC);

long valueL = 2332; byte valueR = 22;

var valueS = valueL + valueR; - VAR универсальное, сам подберет.

Console.WriteLine(valueS);

Деление. Целое число. Остаток остается. 59/8 = 7+3

\_\_\_\_\_\_\_\_\_\_\_\_

Остаток от деления %.

a = b \* q + r

Console.WriteLine("Деление в целых числах. Остаток");

Console.WriteLine($"5 / 2 = {5 / 2}"); //2

Console.WriteLine($"5 % 2 = {5 % 2}"); //1

Console.WriteLine($"59 % 8 = {59 % 8}"); //3

Console.WriteLine($"21 % 4 = {21 % 4}"); //1

\_\_\_\_\_\_\_\_\_\_\_\_

Сокращение арифметика

int a = 12;

int b = 13;

int c = a + b;

Console.WriteLine(a); //12

a = a + b;

a += b;

a -= b;

a /= b;

a %= b;

Console.WriteLine(a); //25

\_\_\_\_\_\_\_\_\_\_\_\_

++ --

int a = 10;

a += 20; //30

a += 1; //31

a++; //постфиксный инкремент +1

++a; //префиксный инкремент +1

\_\_\_\_\_\_\_\_\_\_\_\_

// Приоритет операций/

// Унарные операции – 10 и +10 одно и то же.

// Унарный минус –

int a = 10;

a = -a; // a = -1 \* a, -10

// Префиксный инкремент ++a

int a = 10;

Console.WriteLine(a++); //10

Console.WriteLine(a); //11

Console.WriteLine(++a); //12

Console.WriteLine(a); //12

// %, /, \*

// +, -

// Постфиксный инкремент a++

// () – скобки имеют приоритет.

------------------------------------------------------------------------------------------------------------------------------------------

Логические операции. bool true (1), false (0)   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Инверсия / Отрицание / Логическое «НЕ» / !

bool variable1 = true;

bool variable2 = false;

Console.WriteLine($"variable1 = {variable1} !variable1 = {!variable1}");

Console.WriteLine($"variable1 = {variable2} !variable1 = {!variable2}");  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Конъюнкция / Логическое умножение / Логическое «И» / &&  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Дизъюнкция / Логическое сложение / Логическое «ИЛИ» / ||  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Разделительная дизъюнкция / Строгая дизъюнкция / Исключающее «ИЛИ» / ^  
Когда одна из нескольких true, тогда true, если одинаковы – false.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 char c = 'f';

bool flag1 = 4 > 5; Console.WriteLine(flag1); //false

bool flag2 = 1 < 3; Console.WriteLine(flag2); //true

bool flag3 = c == 'f'; Console.WriteLine(flag3); //true

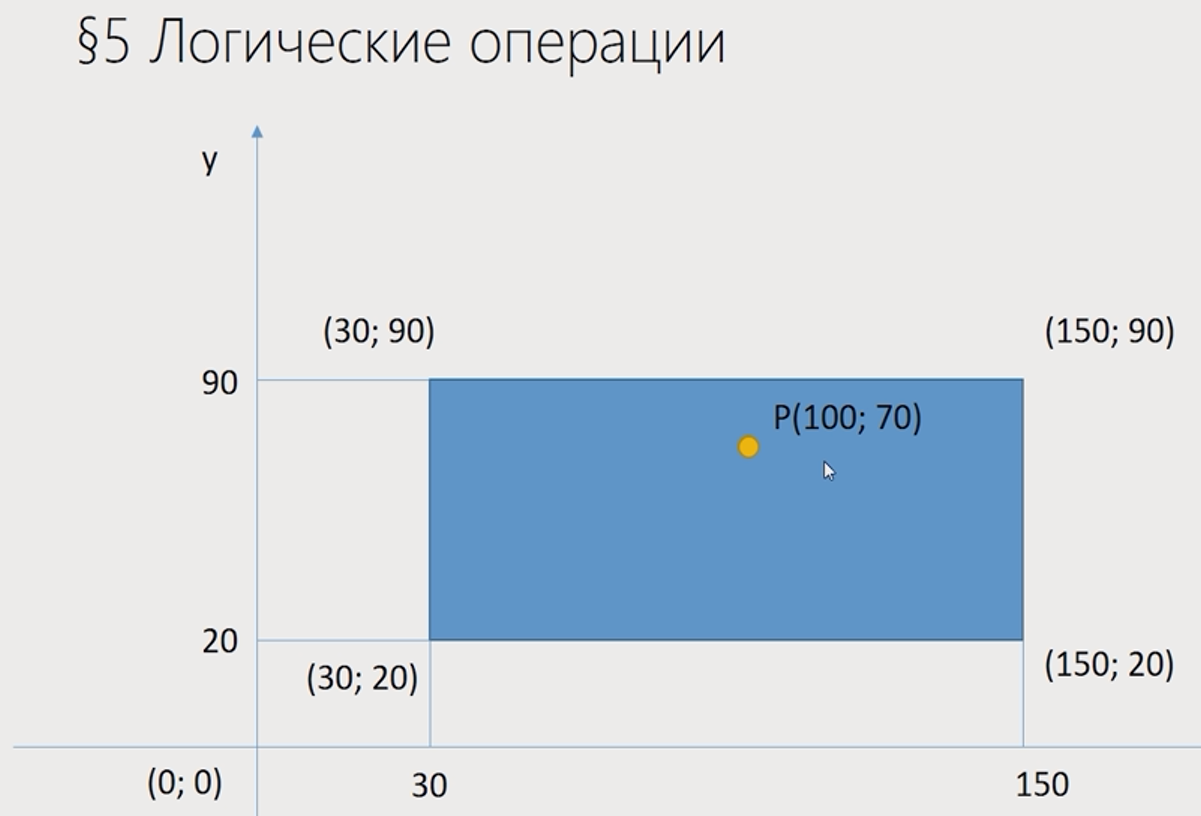
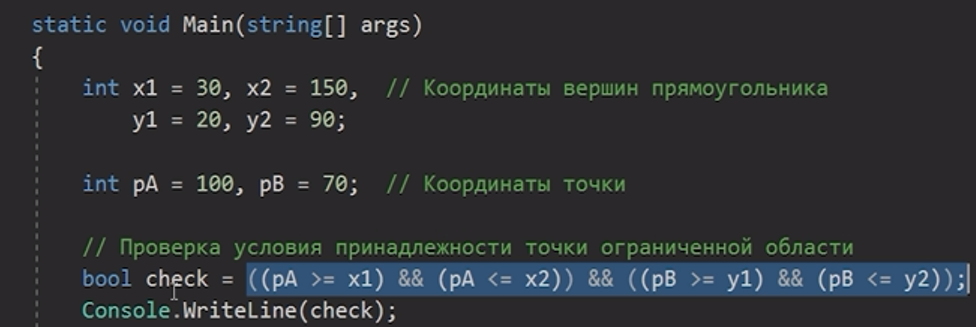
bool flag4 = 28 == 28; Console.WriteLine(flag4); //true

bool flag5 = 28 >= 20; Console.WriteLine(flag5); //true

bool flag6 = 9 <= 9; Console.WriteLine(flag6); //true

bool flag7 = 28 != 90; Console.WriteLine(flag7); //true  
 string s = "C#";

bool flag8 = 28 != 90 && s == "C#"; Console.WriteLine(flag8); //true  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_