Основы С#

Самообучение

# 

# Info

.NET - бесплатная кроссплатформенная среда разработки открытый код.

документация:

[Документация по .NET | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/?view=net-9.0)

[Руководство по C# — управляемый язык .NET | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/csharp/)

# VS Code

При создании .NET проекта VS Соde не может выполнить в консоли dotnet run, тк нужно ещё зайти в папку проекта.

PS D:\stuff\1 code\C#\test 6 console> dotnet run

Не удалось найти выполняемый проект. Убедитесь в том, что проект существует в D:\stuff\1 code\C#\test 6 console, или передайте путь к проекту с помощью --project.

PS D:\stuff\1 code\C#\test 6 console> cd ConsoleApp1

PS D:\stuff\1 code\C#\test 6 console\ConsoleApp1> dotnet run

Hello world!

# First screen

using System;

namespace ConsolApp1

{

class Program

{

static void **Main()**

{

Console.WriteLine("Hello world!");

}

}

}

# Запуск программы

В C# исполнение программы происходит из ключевой функции Main()

using System;

namespace ConsolApp1

{

class Program

{

static void **Main()**

{

Console.WriteLine("Hello world!");

}

}

}

# Библиотеки и using

Библиотеки — это сборки (assemblies), которые содержат классы, интерфейсы и методы, упрощающие разработку. Для доступа к библиотекам в C# используется директива using.

Пример подключения стандартной библиотеки:

using System; // Подключение библиотеки для работы с базовыми функциями

using System.Collections.Generic; // Работа с коллекциями, такими как List, Dictionary и др.

using System.Linq; // Методы LINQ для работы с данными

using System.IO; // Работа с файлами и потоками

Если библиотека не подключена, вы можете столкнуться с ошибками компиляции, например, при использовании классов из неё.

Пример использования подключённых библиотек:

using System;

class Program

{

static void Main()

{

Console.WriteLine("Hello, World!"); // Console — это часть библиотеки System

}

}

## System

using System;

док

[System Пространство имен | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/api/system?view=net-9.0)

# Пространства имён (namespace)

Пространство имён позволяет группировать названия классов, переменных, структур по разным ПРОСТРАНСТВАМ, чтобы избежать конфликтов имен.

Так пространство имён объявляется:

namespace firstNamespace {

….

}

Объявление пространства имён:

namespace MyNamespace

{

class MyClass

{

public void SayHello()

{

Console.WriteLine("Hello from MyNamespace!");

}

}

}

## Ключевые особенности пространств имён

1. **По умолчанию:** Все файлы C# автоматически находятся в пространстве имён проекта.
2. **Рекомендация:** Название пространства имён должно совпадать с названием проекта или логически отражать его структуру.

**Конфликты имён:** Если два класса из разных пространств имеют одинаковое имя, можно указать их полные имена:  
 System.Console.WriteLine("Hello!");

MyNamespace.Console.WriteLine("Custom Console!");

## Использование пространств имён

### Прямое обращение через полное имя:

Если не использовать директиву using, доступ к классу осуществляется через указание полного имени пространства имён:  
  
 MyNamespace.MyClass myClass = new MyNamespace.MyClass();

myClass.SayHello();

### 

### Подключение пространства имён через using:

using MyNamespace;

class Program

{

static void Main()

{

MyClass myClass = new MyClass();

myClass.SayHello();

}

}

### Вложенные пространства имён:

Пространства имён можно вкладывать одно в другое для организации больших проектов:  
  
 namespace OuterNamespace

{

namespace InnerNamespace

{

class InnerClass

{

public void DisplayMessage()

{

Console.WriteLine("Hello from InnerNamespace!");

}

}

}

}

Использование:  
  
 using OuterNamespace.InnerNamespace;

class Program

{

static void Main()

{

InnerClass innerClass = new InnerClass();

innerClass.DisplayMessage();

}

}

### Сокращение имени через using alias

Если два пространства имён имеют одинаковые классы, можно использовать псевдонимы:

using MyAlias = System.Collections.Generic.Dictionary<string, int>;

class Program

{

static void Main()

{

MyAlias myDict = new MyAlias();

myDict.Add("key", 42);

Console.WriteLine(myDict["key"]);

}

}

# Console

[Console Класс (System) | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/api/system.console?view=net-9.0)

1. Console.ReadLine() - всегда получает STRING
2. float в терминале

Чтобы число отпарсилось как float или double, то в терминале нужно вписать число с ЗАПЯТОЙ.

НО в самом коде через точку.

1. вывод разных типов данных



# Типы данных

## Переменные

Объявление

type name;

name = value;

### Типы переменных

using System;

byte 8

short 16

int - целые 32

uint - неотрицательные

long

float

double

string

char

## Работа с разными типами данных

### Конвертирование

System.Convert

[Convert Класс (System) | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/api/system.convert?view=net-9.0)

У каждого типа данных есть свой парсер

float num;

num = float.Parse(Console.ReadLine());

# 

# Классы

## Наследование

класс наследует от класса Window

public partial class MainWindow : Window