

# **Вводная лекция по дисциплине «Инструментальные средства разработки ПО»**

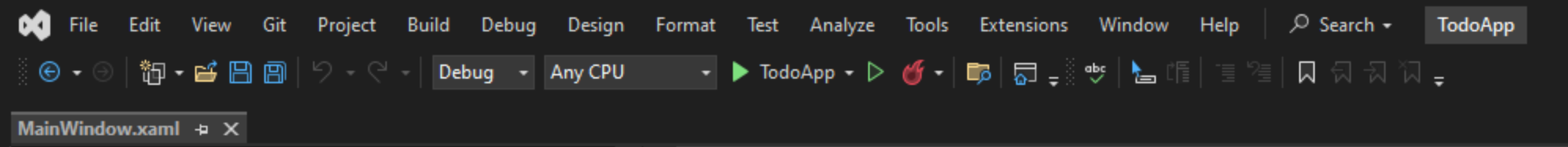
**Автор: Леонтьев Д.А.**

**[denis.leontev92@yandex.ru](mailto:denis.leontev92@yandex.ru)**

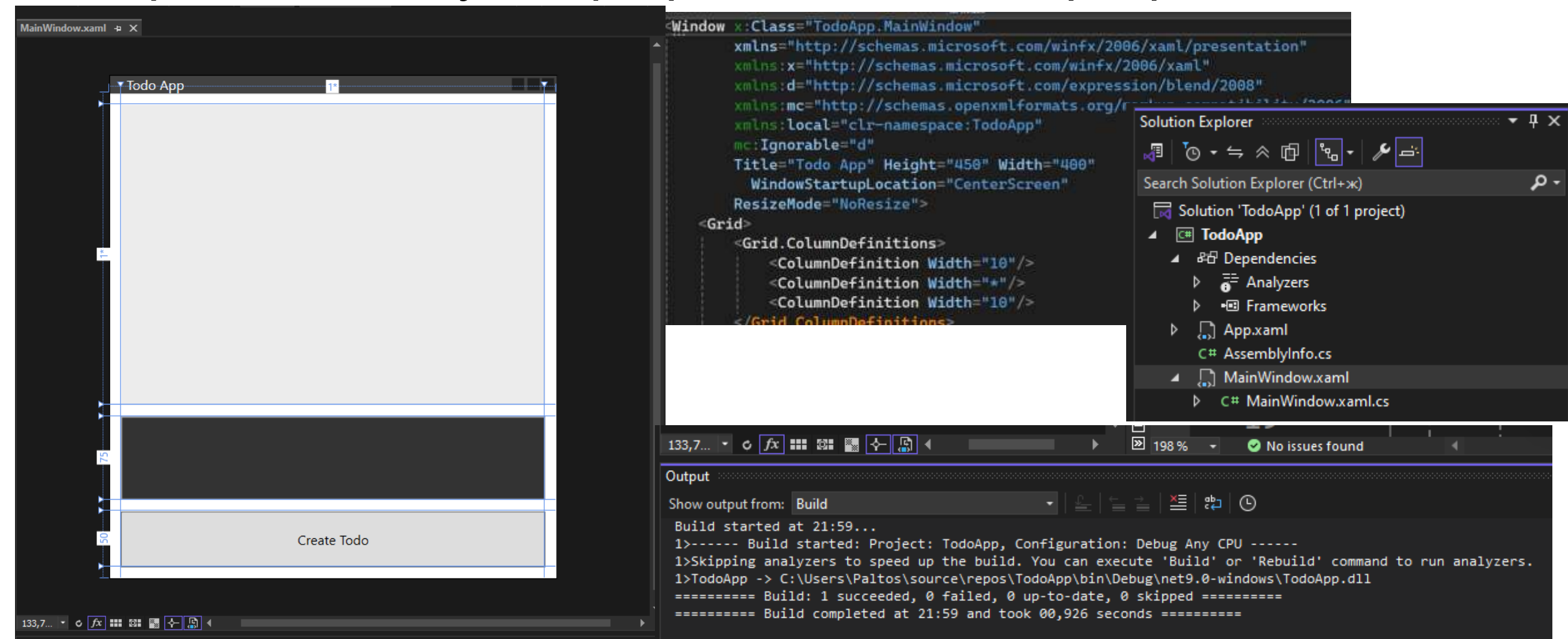
# Введение в дисциплину

# Чем отличается от курса по Python

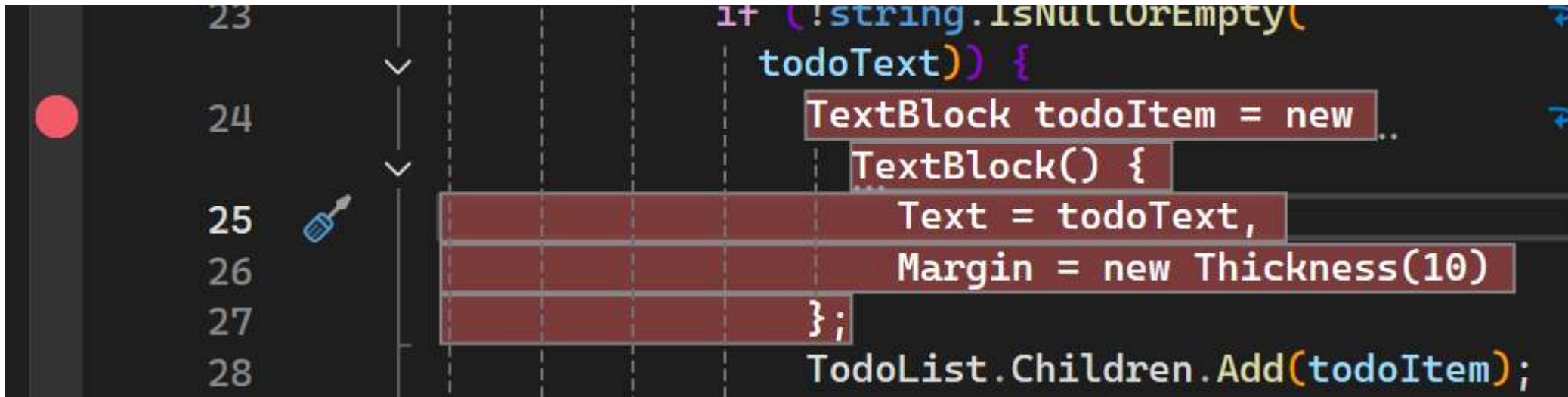
- Python: акцент на **основах программирования и синтаксисе.**
- Этот курс: акцент на **инструментах разработки и организации проекта.**
- Python = быстрый старт, простота.
- C#/.NET = мощные IDE, строгая структура, профессиональная среда.



- Изучаем не только язык, но и среду и инструменты, которые используют профессиональные разработчики.



- Цель: научиться писать, отлаживать, управлять и сопровождать код с помощью современных средств.

A screenshot of a code editor with a dark theme. The code is in C# and shows a conditional block where a new TextBlock is created and added to a list. The code is as follows:

```
23 if (!string.IsNullOrEmpty(  
24     todoText)) {  
25         TextBlock todoItem = new  
26             TextBlock() {  
27                 Text = todoText,  
28                 Margin = new Thickness(10)  
29             };  
30         TodoList.Children.Add(todoItem);  
31     }
```

The code is highlighted with a red selection box. The line numbers 23 through 28 are visible on the left side of the editor. A red circle is visible on the left margin next to line 24. A blue icon of a syringe is visible on the left margin next to line 25.

# • Рефакторинг

## ▣ Баг 2: Можно открыть паузу, пока открыт магазин

**Причина:** В `PauseManager` кнопка паузы не учитывает `ShopManager.IsShopOpen`.

✂ **Решение:** Добавить проверку в метод `PauseGame()`.

Открываем скрипт `PauseManager`. И вносим правки:

```
asset usage 1 usage
public void PauseGame()
{
    if (ShopManager.IsShopOpen) return; // Проверяем,
    // открыт ли магазин перед паузой

    _isPaused = true;
    IsPaused = true;
    pauseMenu.SetActive(true);
    Time.timeScale = 0f;
}
```

➤ Вынесем код из метода `Start()` в новый `InitializeSnake()`:

```
private void Start()
{
    InitializeSnake();
}

1 reference
private void InitializeSnake()
{
    _segments = new List<Transform>();
    _segments.Add(transform);
}
```

Разделим в методе `FixedUpdate` код на 3 метода:

```
private void FixedUpdate()
{
    MoveSegments();
    MovePlayer();
    CheckCollision();
}
```

## 30. Рефакторинг кода.

Наш код неплох, но мы можем немного улучшить его:

- ☒ Оптимизировать работу с музыкой (избежать повторного `StopMusic()`)
- ☒ Сделать названия методов более точными
- ☒ Объединить логику `Pause` и `StopMusic()`
- ☒ Объединить `PipeReset()` с `ClearPipes()` для удобства

Paltosik92 / WPFZooManager

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

Settings

WPFZooManager

Private

Watch

master

1 Branch

0 Tags

Go to file

Add file

<> Code

Paltosik92

remove

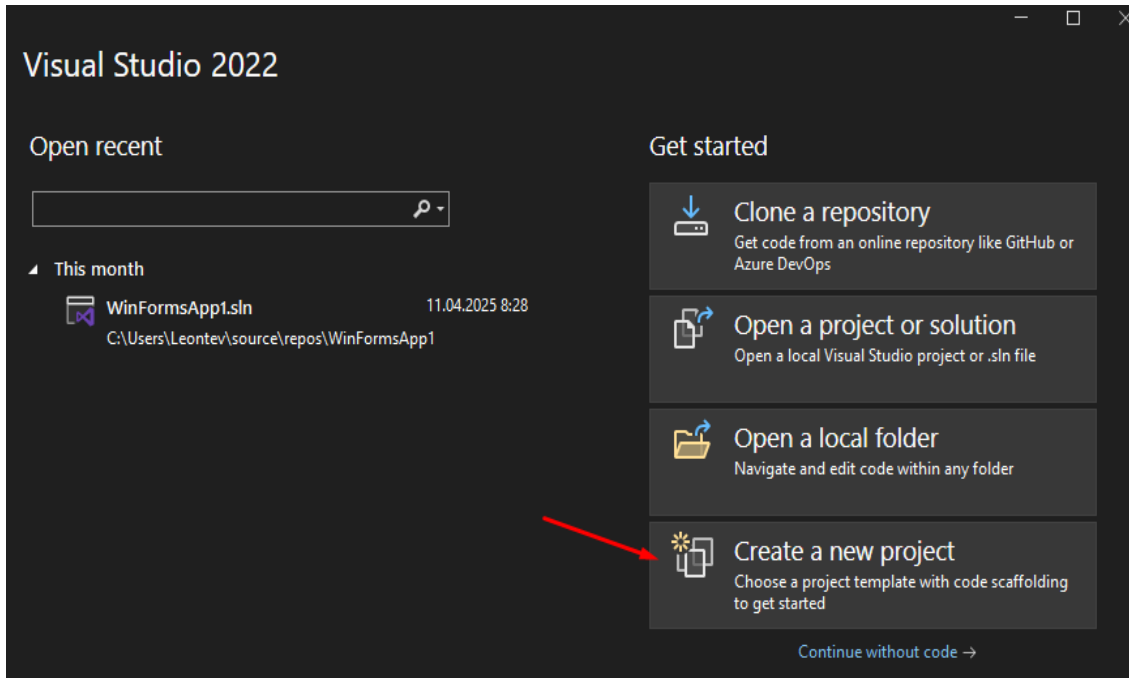
78a5d16 · 2 days ago

9 Commits

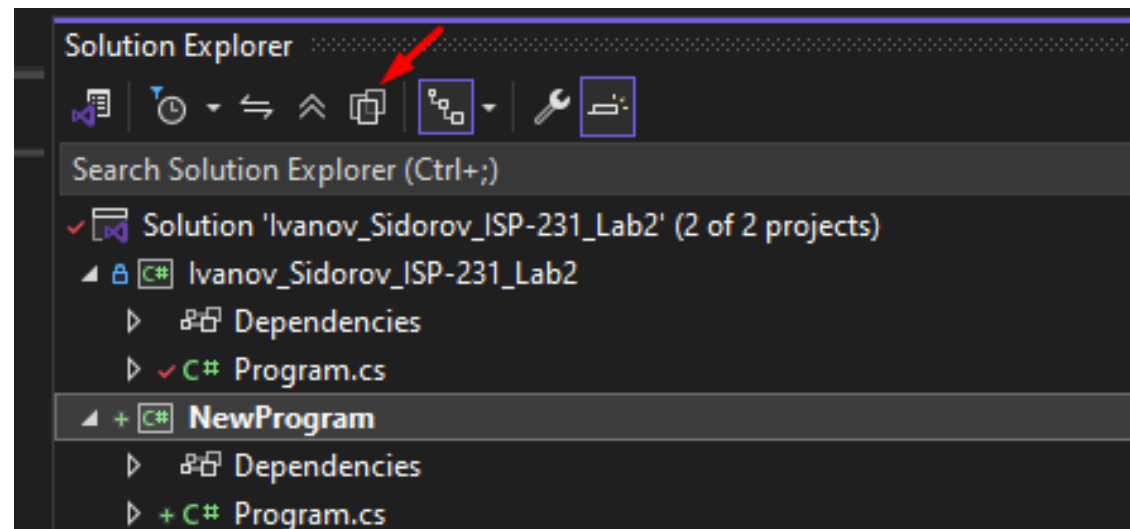
Properties	add db, xaml, list code	3 days ago
.gitattributes	Add .gitattributes and .gitignore.	3 days ago
.gitignore	Add .gitattributes and .gitignore.	3 days ago
AnimalZooDBDataSet.Designer.cs	add db, xaml, list code	3 days ago
AnimalZooDBDataSet.xsc	add db, xaml, list code	3 days ago
AnimalZooDBDataSet.xsd	add db, xaml, list code	3 days ago
AnimalZooDBDataSet.xss	add db, xaml, list code	3 days ago
App.config	add db, xaml, list code	3 days ago
App.xaml	Add project files.	3 days ago
App.xaml.cs	Add project files.	3 days ago
MainWindow.xaml	add two button	2 days ago
MainWindow.xaml.cs	remove	2 days ago
WPFZooManager.csproj	add db, xaml, list code	3 days ago
WPFZooManager.sln	Add project files.	3 days ago

- Практика > теория: больше работы в IDE и с Git.

# Что будем изучать. 1 Семестр



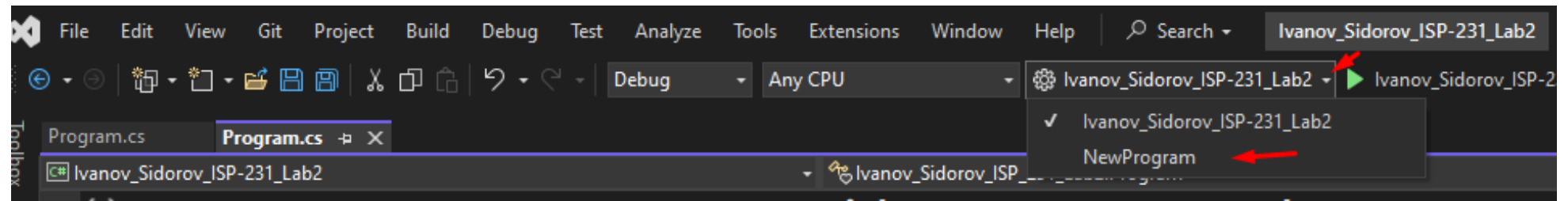
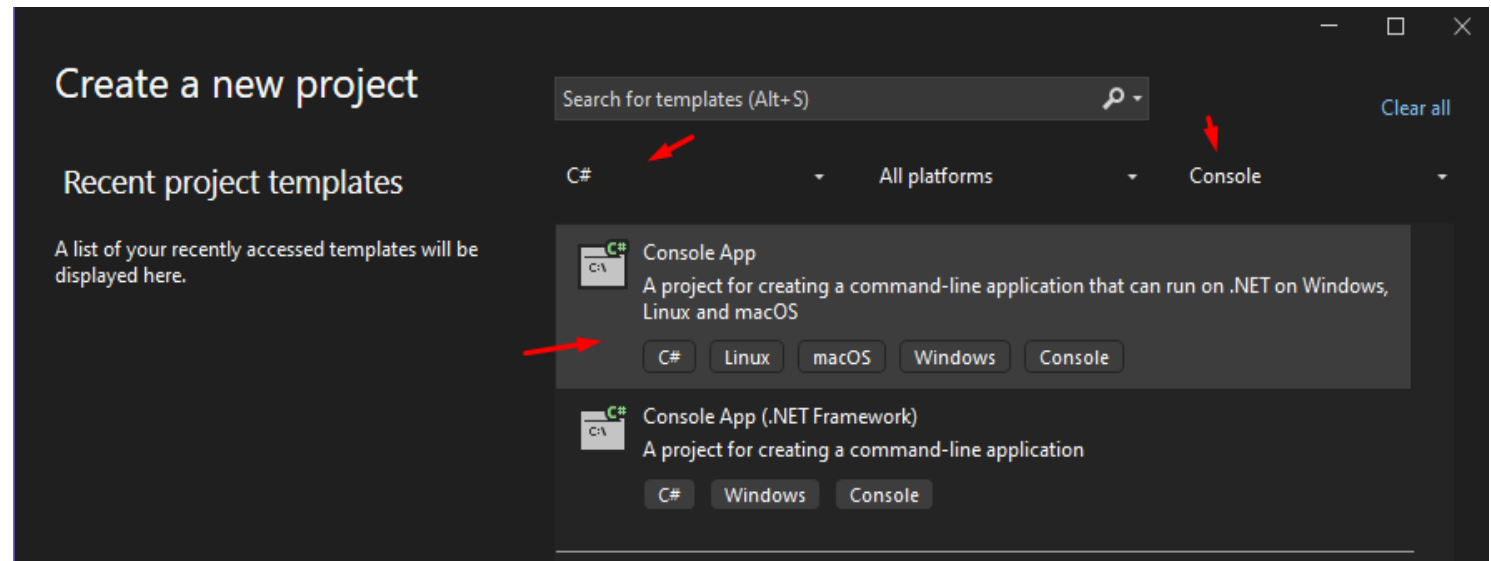
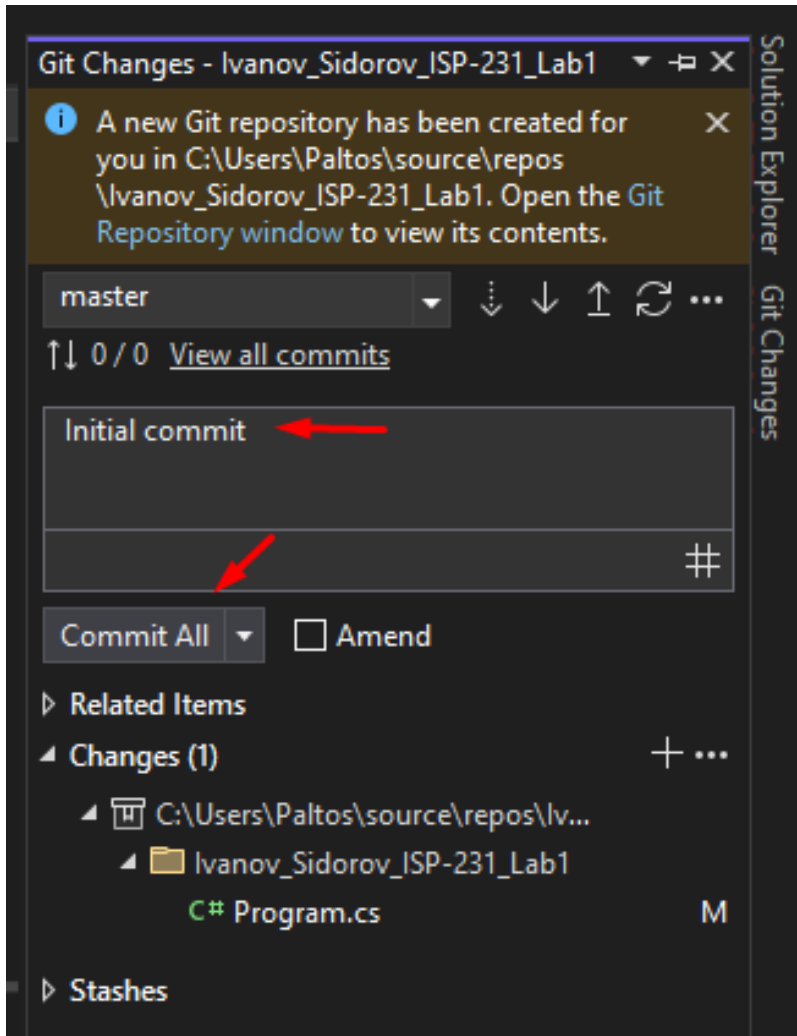
- Освоить работу в Visual Studio 2022
- Понять структуру консольного проекта
- Изучить базовый синтаксис языка C#



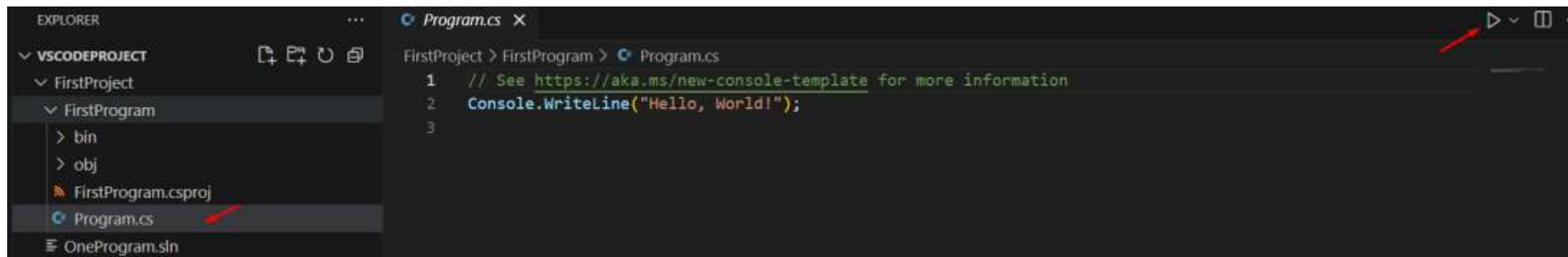
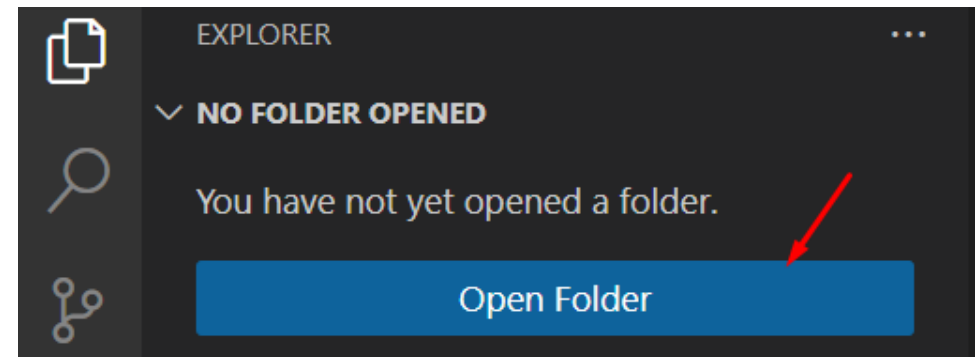
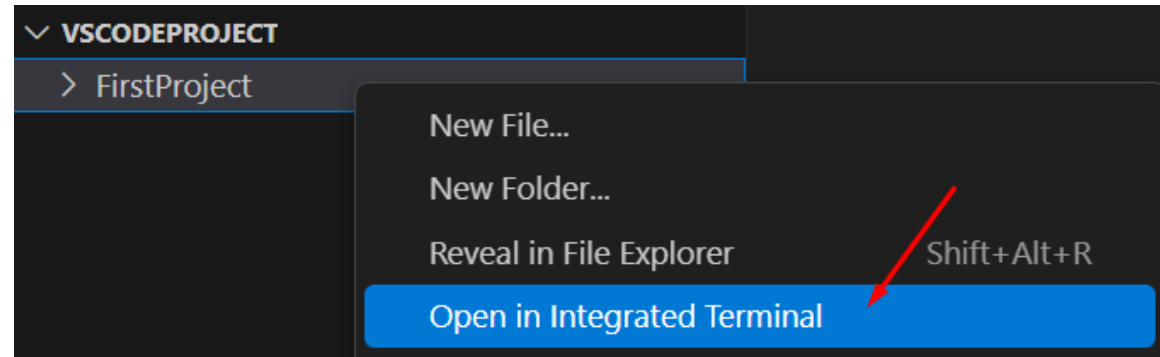
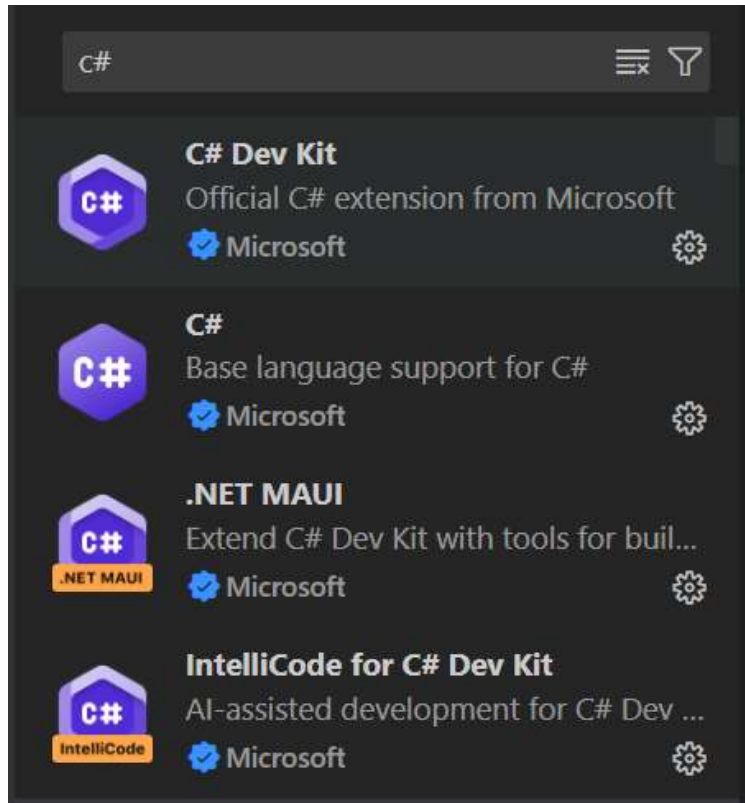
```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <RootNamespace>Ivanov_Sidorov_ISP_231_Lab1</RootNamespace>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>
</Project>
```



- Освоить создание и коммит проекта в GitHub через Visual Studio



- Научиться работать в среде Visual Studio Code;



- Установить и использовать последнюю версию .NET SDK;

Хотите узнать больше о .NET 10.0.0-preview.4? .NET 10.0.0-preview.4 теперь доступен. [Ознакомьтесь с объявлением о выпуске](#)

## Скачать .NET 10.0

Это не то, что вы искали? Посетите страницу [скачиваний](#) для получения дополнительных возможностей.

### 10.0.0-preview.4 Последний релиз

[Повторное издание](#) Последняя дата выпуска 13 мая 2025 г.

#### Создание приложений — SDK

##### SDK 10.0.100-preview.4

ОС	Установщики	Двоичные файлы
Linux		<a href="#">Arm32</a>   <a href="#">Arm32 Alpine</a>   <a href="#">Arm64</a>   <a href="#">Arm64 Alpine</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a>
macOS	<a href="#">Arm64</a>   <a href="#">x64</a>	<a href="#">Arm64</a>   <a href="#">x64</a>
Windows	<a href="#">x64</a>   <a href="#">x86</a>   <a href="#">Arm64</a>   <a href="#">инструкции.winget</a>	<a href="#">x64</a>   <a href="#">x86</a>   <a href="#">Arm64</a>
Все	<a href="#">dotnet-install scripts</a>	

Полная версия  
10.0.100-preview.4.25258.110

Поддержка Visual Studio  
Visual Studio 2022 (v17.14)

Включен в  
Visual Studio 17.14

Включенные среды выполнения  
Среда выполнения .NET 10.0.0-preview.4.25258.110  
Среда выполнения ядра ASP.NET 10.0.0-preview.4.25258.110  
Среда выполнения рабочего стола .NET 10.0.0-preview.4.25258.110

#### Запуск приложений — время выполнения

##### Среда выполнения ядра ASP.NET 10.0.0-preview.4

Среда выполнения ASP.NET Core позволяет запускать существующие веб-приложения и серверные приложения. В Windows мы рекомендуем установить пакет хостинга, который включает среду выполнения .NET и поддержку IIS.

Полная версия  
10.0.0-preview.4.25258.110

ОС	Установщики	Двоичные файлы
Linux		<a href="#">Arm32</a>   <a href="#">Arm32 Alpine</a>   <a href="#">Arm64</a>   <a href="#">Arm64 Alpine</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a>
macOS		<a href="#">Arm64</a>   <a href="#">x64</a>
Windows	<a href="#">x64</a>   <a href="#">x86</a>   <a href="#">Arm64</a>   <a href="#">Hosting Bundle</a>   <a href="#">инструкции.winget</a>	<a href="#">x64</a>   <a href="#">x86</a>   <a href="#">Arm64</a>

##### Среда выполнения рабочего стола .NET 10.0.0-preview.4

Среда выполнения .NET Desktop позволит запускать существующие классические приложения Windows. В этот выпуск входит среда выполнения .NET; вам не нужно устанавливать ее отдельно.

Полная версия  
10.0.0-preview.4.25258.110

```
C:\Users\Leontev>dotnet --list-sdks
8.0.408 [C:\Program Files\dotnet\sdk]
9.0.203 [C:\Program Files\dotnet\sdk]
10.0.100-preview.4.25258.110 [C:\Program Files\dotnet\sdk]
```

- Освоить базовую работу с проектами C# через терминал;
- Попрактиковаться в создании и запуске нескольких проектов в VS Code;

```
PS C:\Users\Leontev\Documents\VSCodeProject\FirstProject> dotnet new sln --name OneProgram
Шаблон "Файл решения" успешно создан.
```

```
PS C:\Users\Leontev\Documents\VSCodeProject\FirstProject> dotnet new console --output FirstProgram
Шаблон "Консольное приложение" успешно создан.
```

Идет обработка действий после создания...

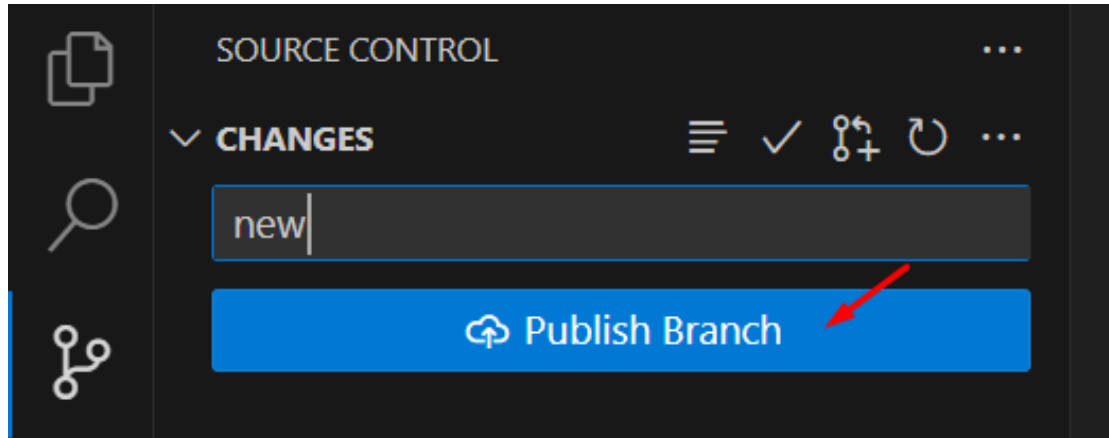
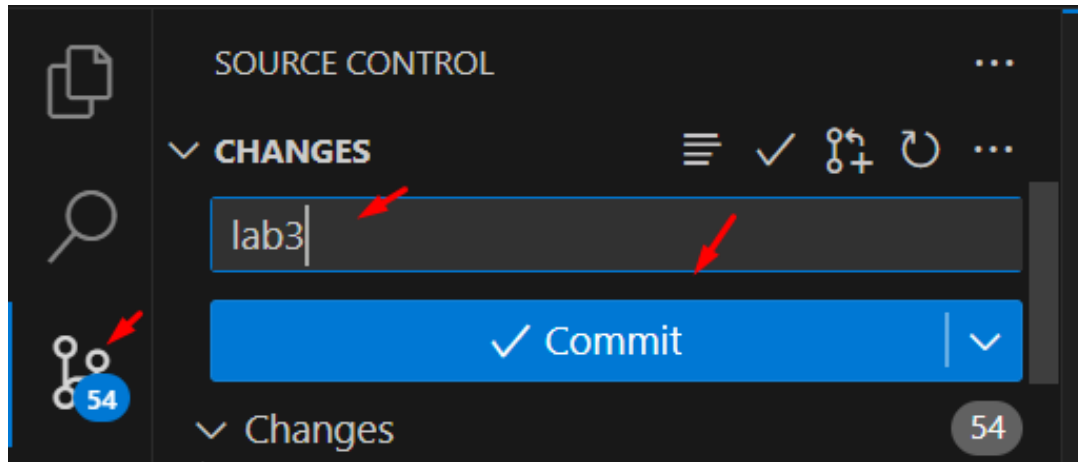
```
Restoring C:\Users\Leontev\Documents\VSCodeProject\FirstProject\FirstProgram\FirstProgram.csproj:
Restore succeeded.
```

```
PS C:\Users\Leontev\Documents\VSCodeProject\FirstProject> dotnet sln add FirstProgram
Project `FirstProgram\FirstProgram.csproj` added to the solution.
```

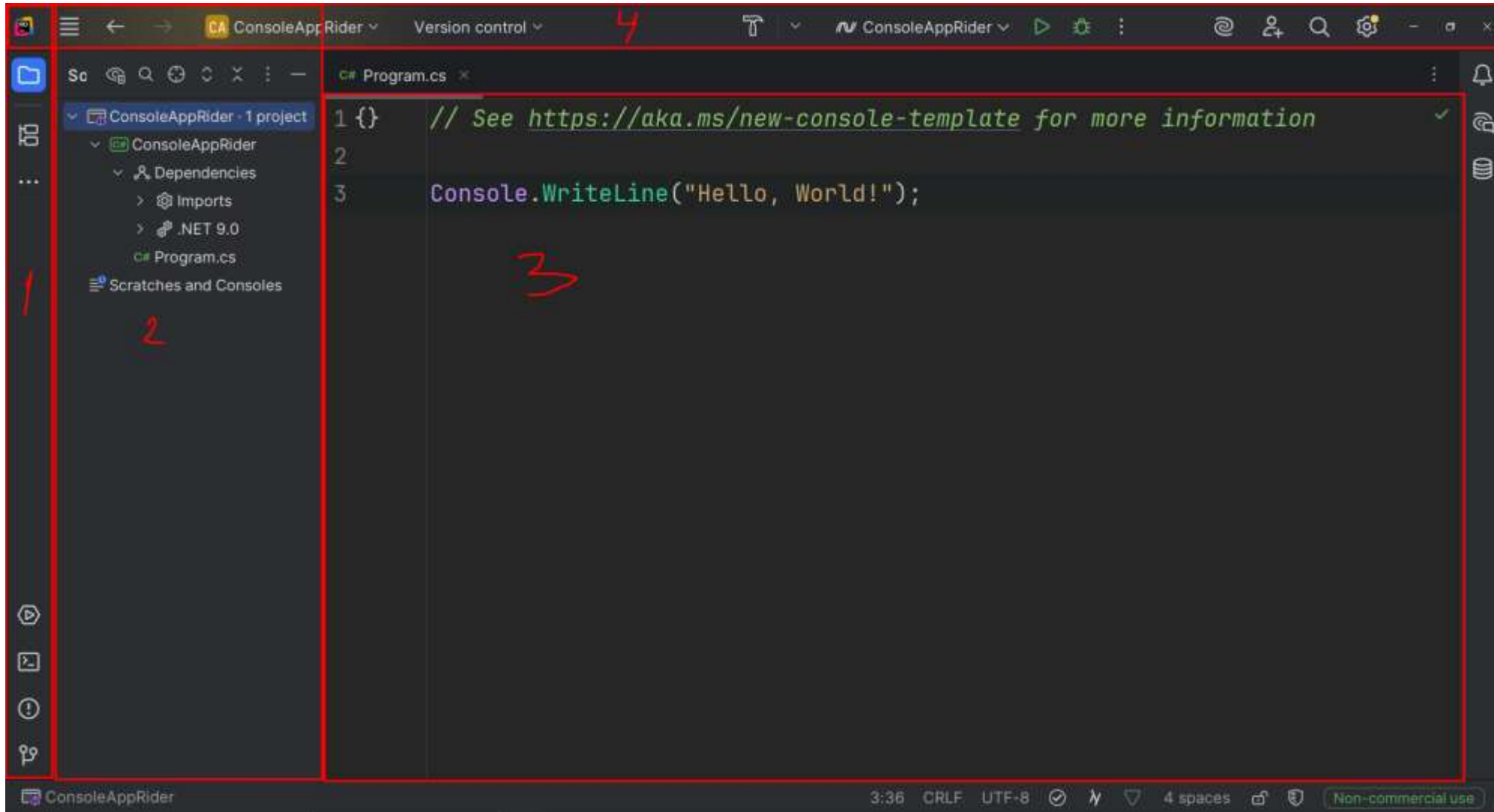
```
PS C:\Users\Leontev\Documents\VSCodeProject\FirstProject\FirstProgram> dotnet run
Восстановление завершено (0,8 с)
You are using a preview version of .NET. See: https://aka.ms/dotnet-support-policy
FirstProgram успешно выполнено (0,4 с) → bin\Debug\net10.0\FirstProgram.dll
```

```
Сборка успешно выполнено через 2,2 с
Hello, World!
```

- Повторить основы Git в Visual Studio Code;



# Познакомиться с возможностями среды разработки **JetBrains Rider**





# Изучить основные команды системы контроля версий Git, включая создание веток, добавление и фиксацию изменений

```
Leontev@DESKTOP-6MFOUBL MINGW64 ~  
$ git --version  
git version 2.45.1.windows.1
```

```
Leontev@DESKTOP-6MFOUBL MINGW64 ~  
$ git config --global user.name "Denis"
```

```
Leontev@DESKTOP-6MFOUBL MINGW64 ~  
$ git config --global user.email denis.leontev92@yandex.ru
```

```
Leontev@DESKTOP-6MFOUBL MINGW64 /c  
$ mkdir projectGit
```

```
Leontev@DESKTOP-6MFOUBL MINGW64 /c/projectGit  
$ git init  
Initialized empty Git repository in C:/projectGit/.git/
```

```
Leontev@DESKTOP-6MFOUBL MINGW64 /c/projectGit (master)  
$ git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   index.html  
    new file:   style/style.css
```

```
Leontev@DESKTOP-6MFOUBL MINGW64 /c/projectGit (master)  
$ git rm --cached index.html  
rm 'index.html'  
  
Leontev@DESKTOP-6MFOUBL MINGW64 /c/projectGit (master)  
$ git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   style/style.css  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    index.html
```

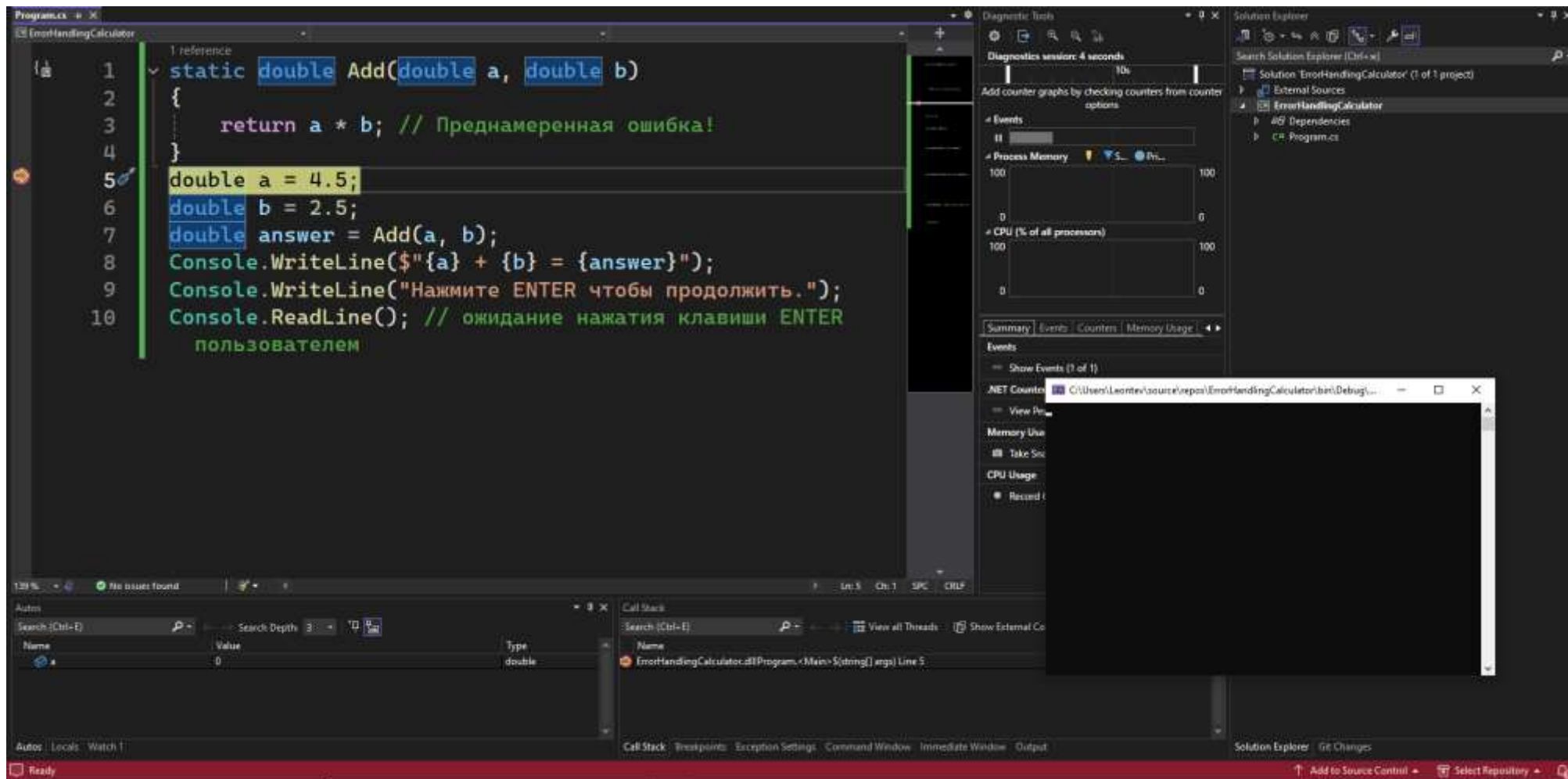
# Основы обработки ошибок в C# с использованием конструкции try-catch.

```
try
{
    int x = 5;
    int y = x / 0;
    Console.WriteLine(y);
}
catch
{
    Console.WriteLine("Возникло исключение!");
}
finally
{
    Console.WriteLine("Блок finally");
}
Console.WriteLine("Конец программы");
```

```
catch (FormatException ex)
{
    // Обрабатываем ошибку неверного формата ввода
    Console.WriteLine($"Ошибка: {ex.Message}");
}
```



# Базовые навыки отладки программ с помощью средств IDE (Visual Studio, Visual Studio Code, JetBrains Rider).



```
1 static double Add(double a, double b)
2 {
3     return a * b; // Преднамеренная ошибка!
4 }
5 double a = 4.5;
6 double b = 2.5;
7 double answer = Add(a, b);
```

```
return a * b;
```

a 4.5

139 % No issues found

Watch 1

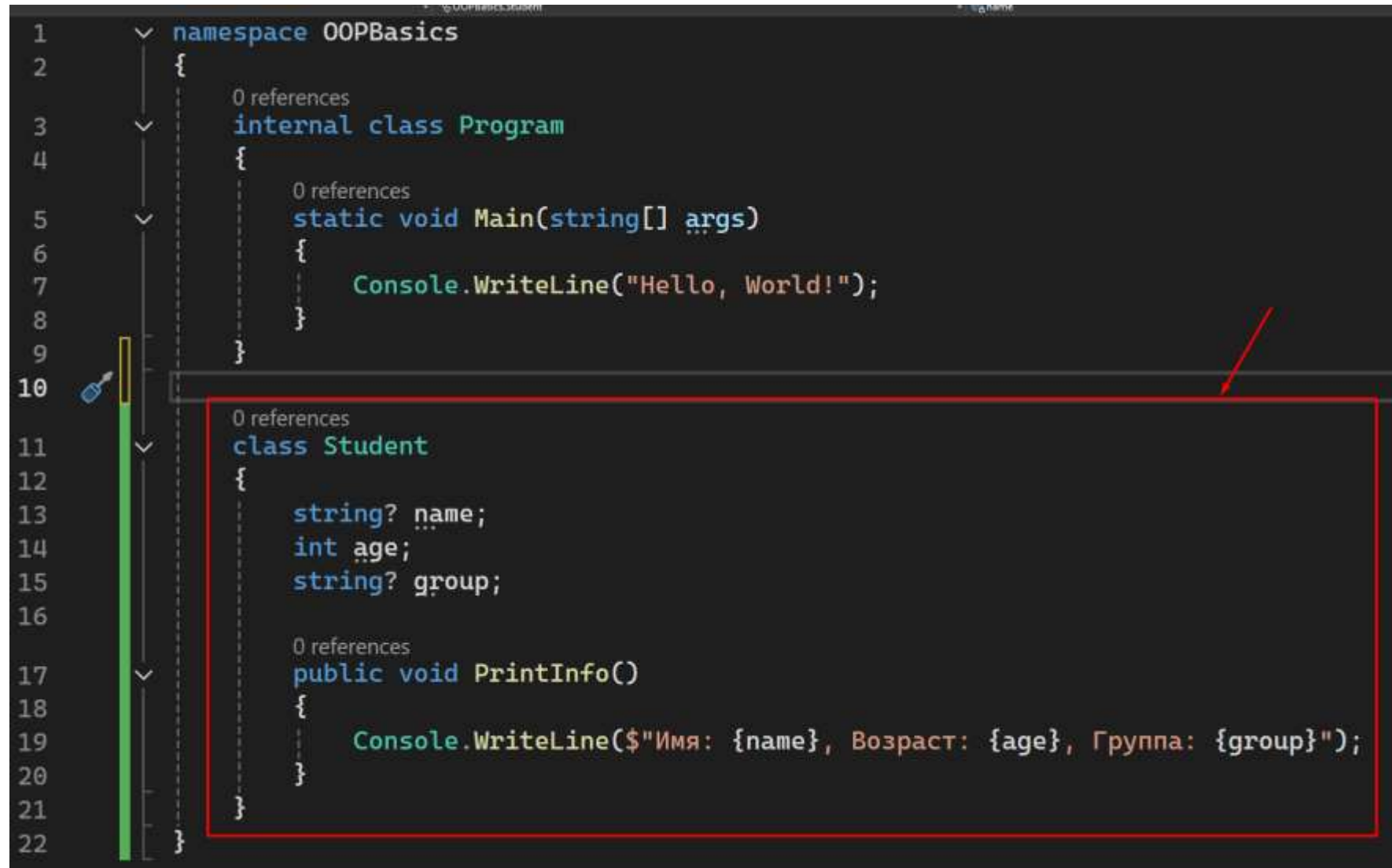
Search (Ctrl+E) Search Depth: 3

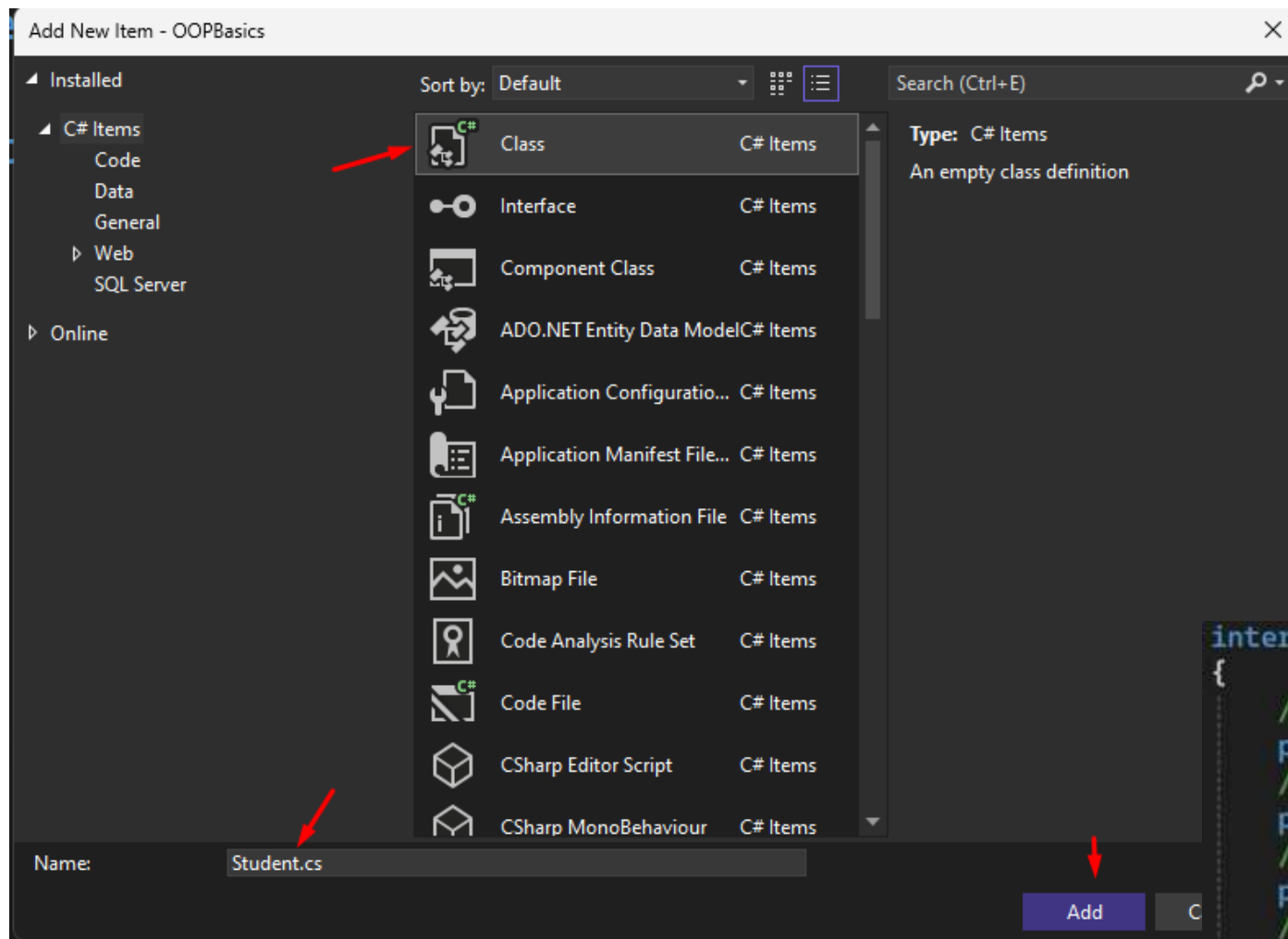
Name	Value	Type
✖ return a * b;	error CS1525: Invalid expression term 'return'	
Add item to watch		

Autos Locals Watch 1

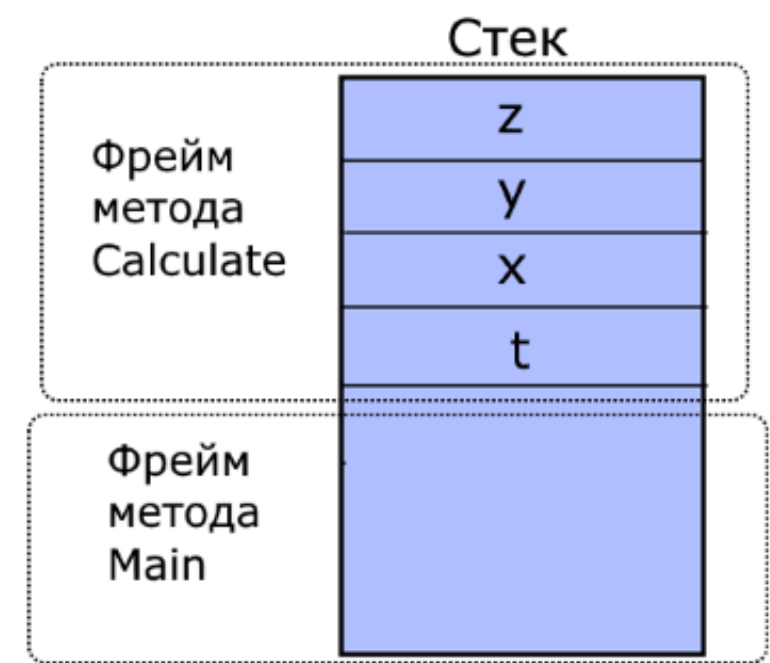
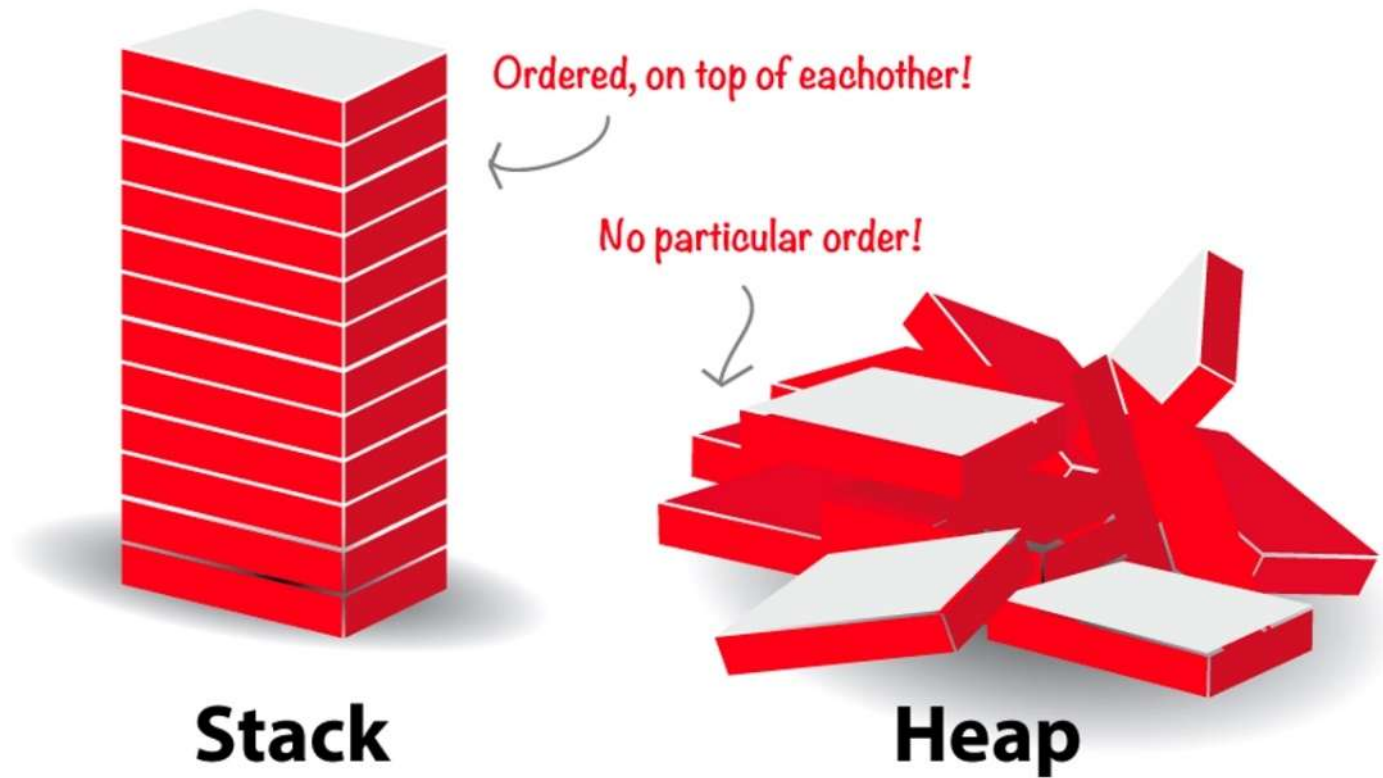
# ООП

```
1 namespace OOPBasics
2 {
3     0 references
4     internal class Program
5     {
6         0 references
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Hello, World!");
10        }
11    }
12
13    0 references
14    class Student
15    {
16        string? name;
17        int age;
18        string? group;
19
20        0 references
21        public void PrintInfo()
22        {
23            Console.WriteLine($"Имя: {name}, Возраст: {age}, Группа: {group}");
24        }
25    }
26 }
```



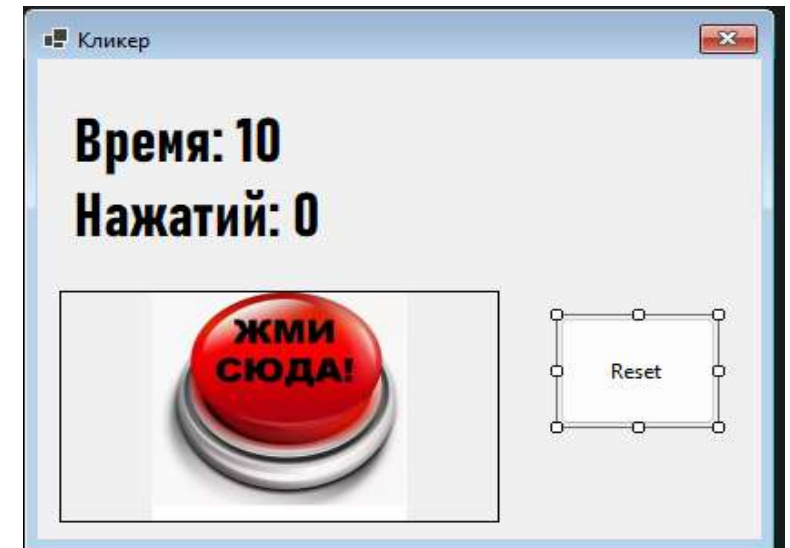
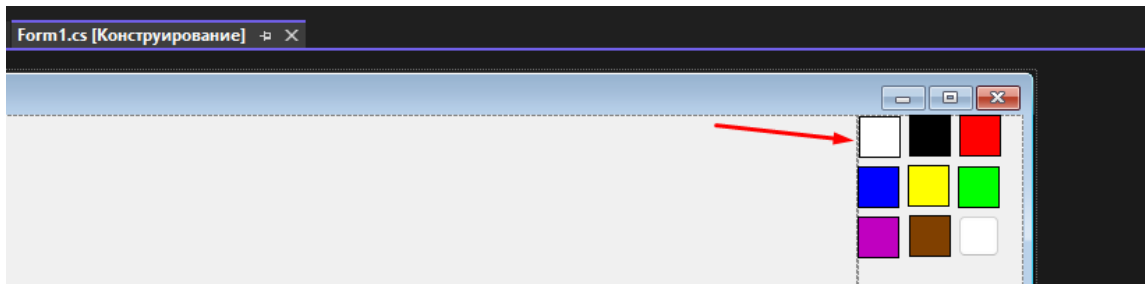
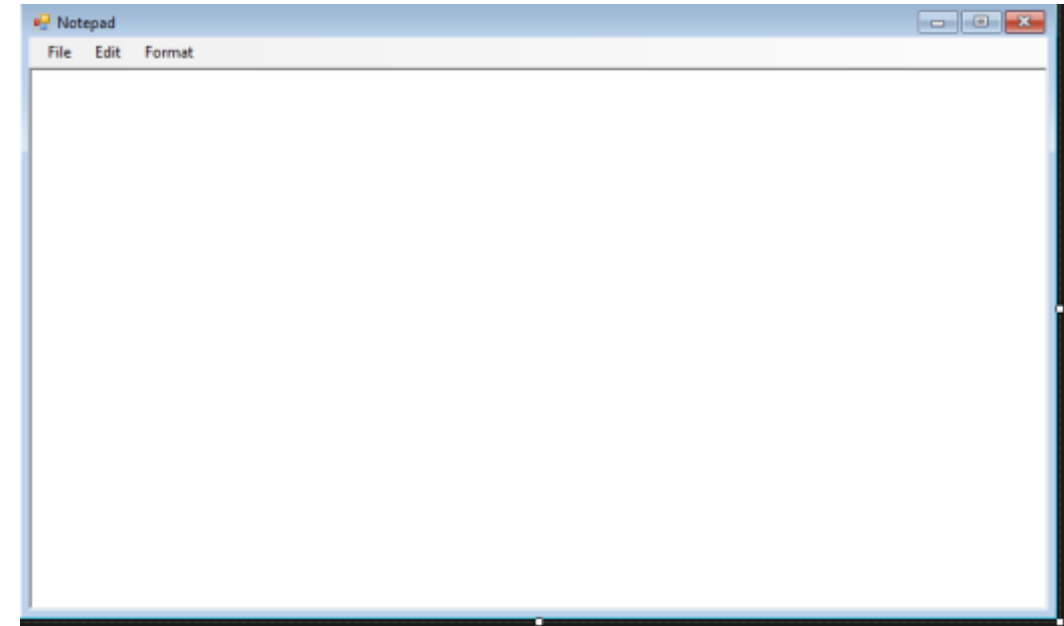
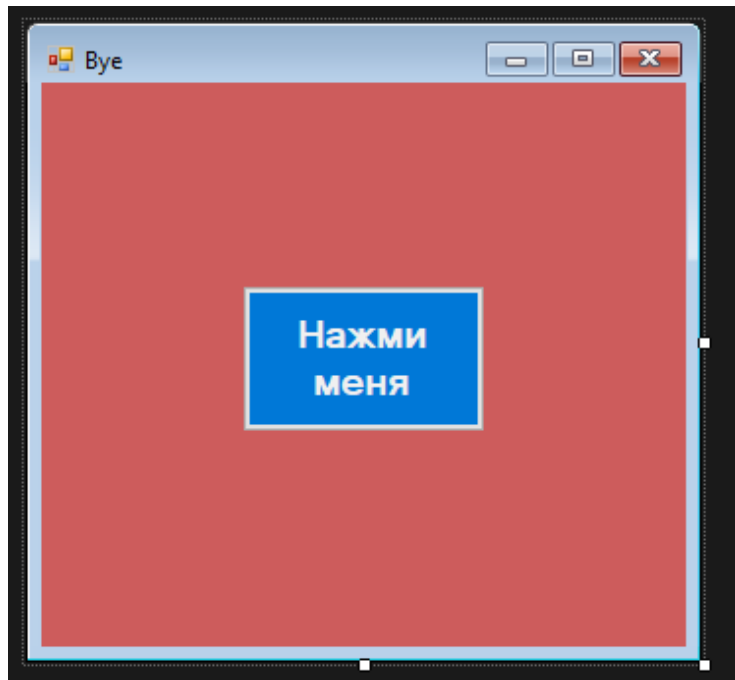


```
internal class Car
{
    // Публичное поле (можно менять снаружи)
    public string? model;
    // Приватное поле (только для внутреннего использования)
    private int _currentSpeed;
    // Protected поле (доступно в наследниках)
    protected string? engineType;
    // Публичный метод (можно вызывать снаружи)
    public void Accelerate(int speed)
    {
        if (speed > 0)
        {
            _currentSpeed += speed;
            Console.WriteLine($"Разгон до {_currentSpeed} км/ч");
        }
    }
}
```

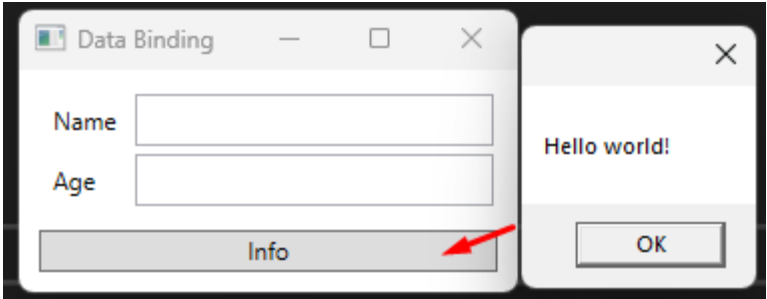
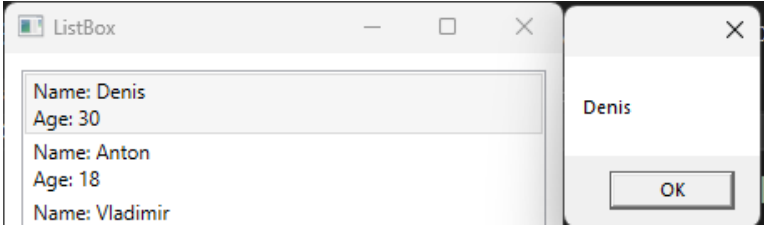
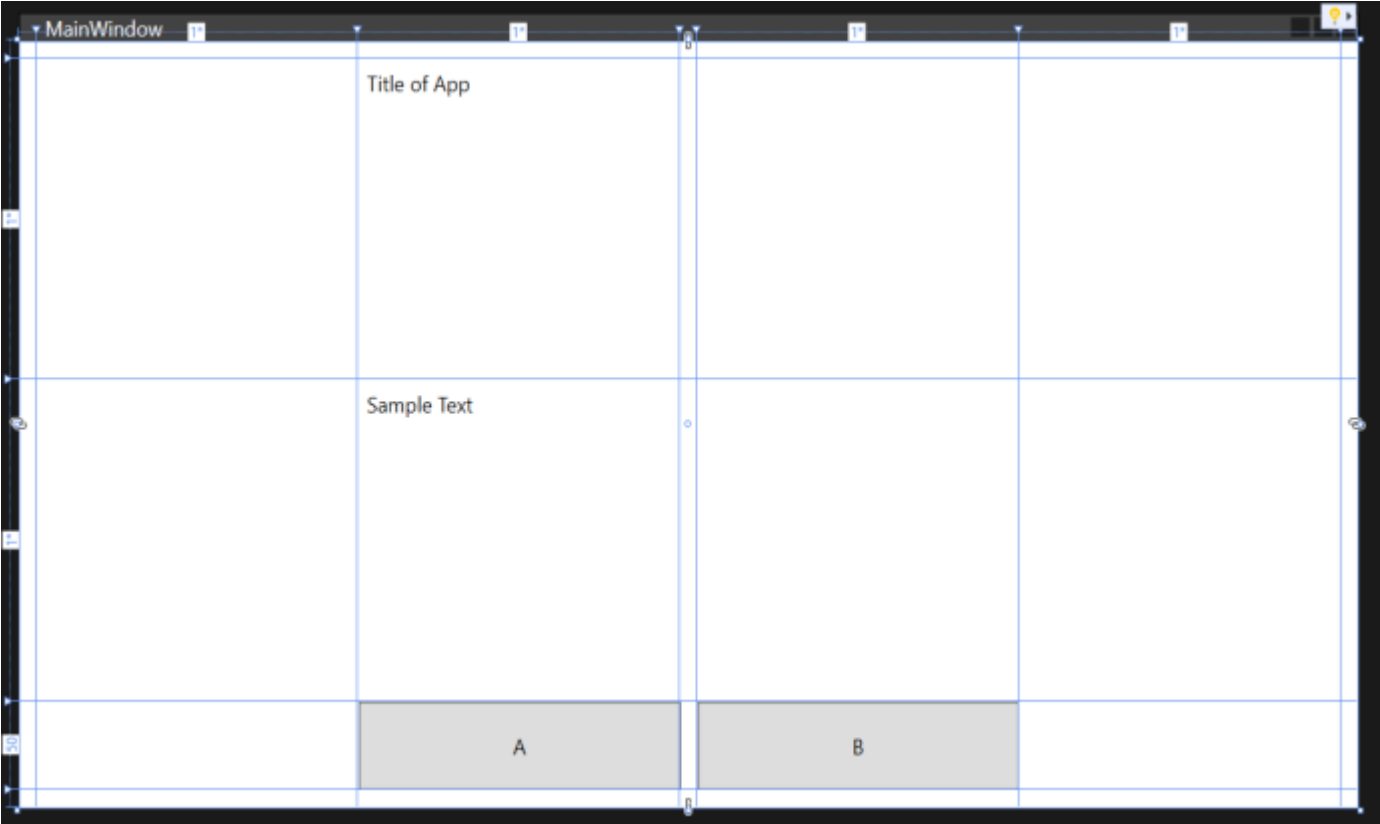


# 11-20 занятие. Графические приложения

## Windows Forms



# WPF

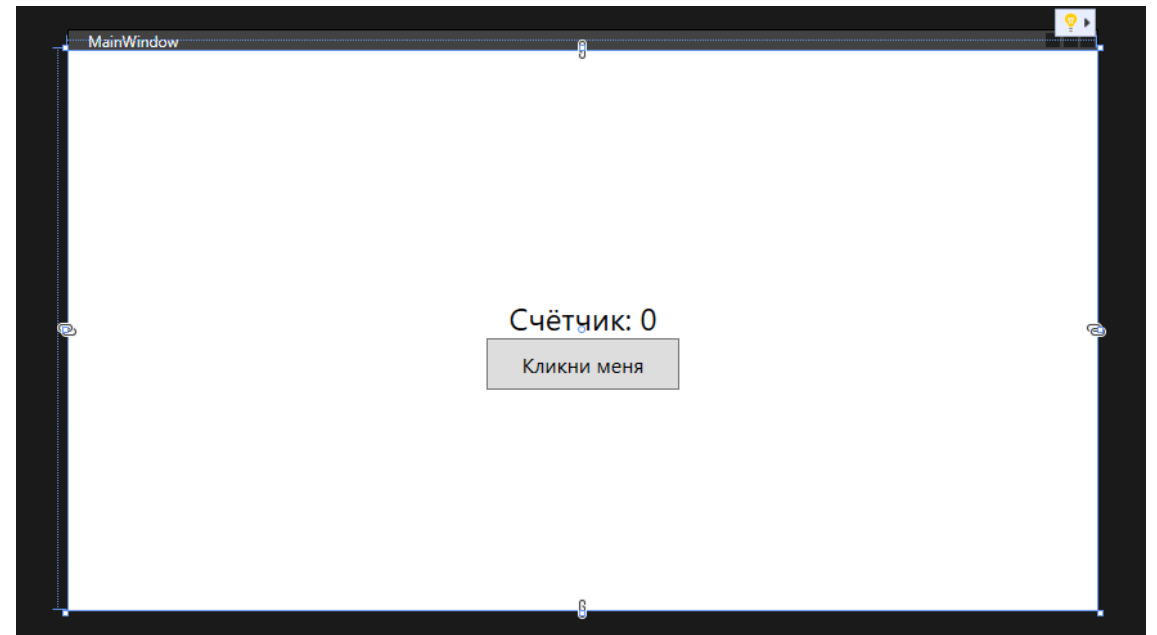


# WPF

Login

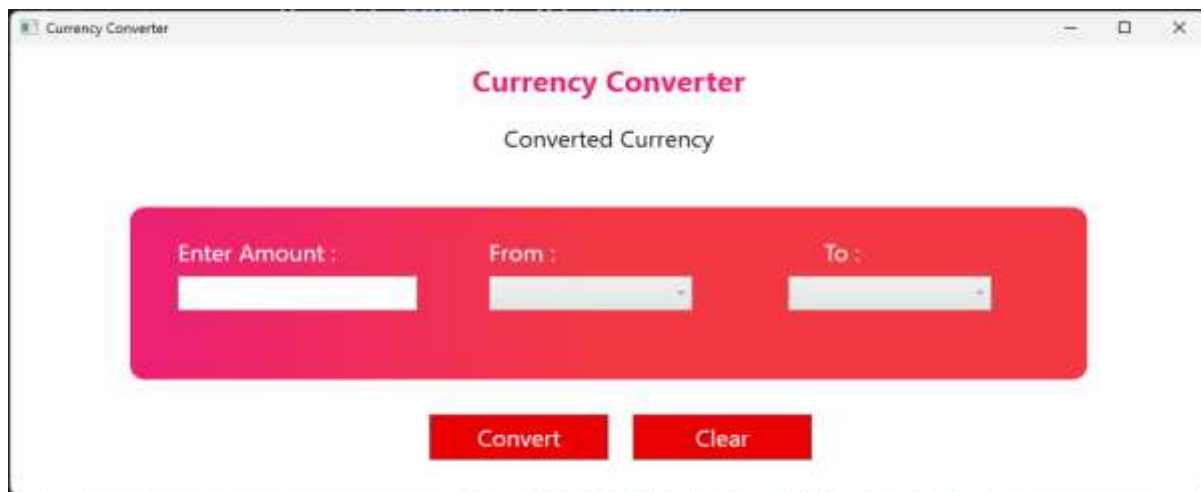
Enter Password

Login





# WPF



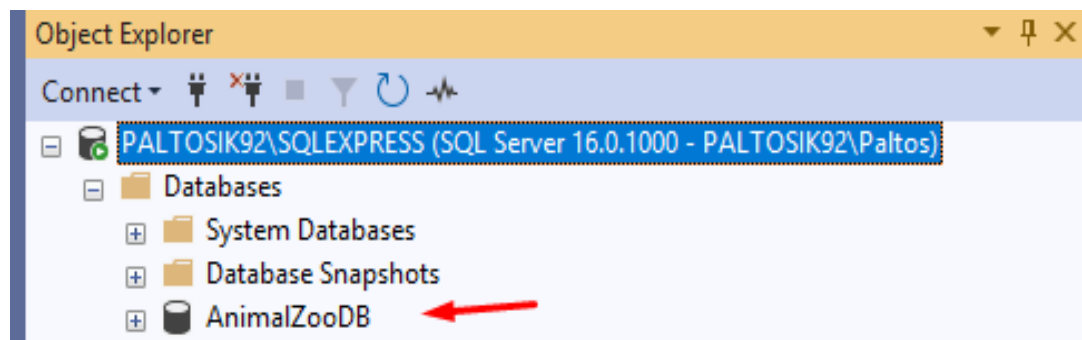
Игра с подбором пар животных



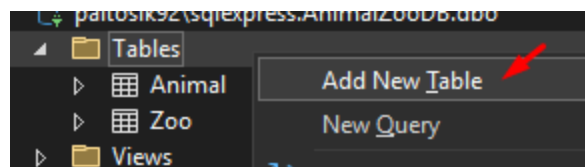
На экране выводятся  
восемь пар животных,  
которые случайно  
распределяются в окне.  
Игрок должен щёлкнуть на  
двух одинаковых  
животных, чтобы пара  
исчезала из окна.

Таймер показывает сколько времени  
потребовалось для завершения игры.

# WPF



```
CREATE TABLE [dbo].[Animal]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Name] NVARCHAR(50) NOT NULL
)
```



Zoo List

Волгоград

Москва

Delete Zoo

Add Zoo

Update Zoo

Associated Animals

Monkey

Crocodile

Owl

Owl

Crocodile

Remove Animal

Add Animal Zoo

Update Animal

Add Animal to Zoo

Shark

Monkey

Wolf

Crocodile

Owl

Тигр

Панда


Delete Animal

dbo.Animal [Data]		
dbo.Animal [Design]		
Max Rows: 1000		
	Id	Name
	1	Shark
	2	Monkey
	3	Wolf
	4	Crocodile
	5	Owl
	6	Parrot
	NULL	NULL

# Avalonia

### Temperature Converter


Celsius	<input type="text" value="0"/>
Fahrenheit	<input type="text" value="0"/>
<input type="button" value="Calculate"/>	



## Avalonia for VSCode Preview

Avalonia Team | 58,163 | ★★★★★ (17) | ♥ Sponsor

Avalonia tools for Visual Studio Code

☒ Auto Update 

### AvaloniaClicker2

7

### BasicMvvmSample

Simple ViewModel Reactive UI

Enter your Name:

## NotePage

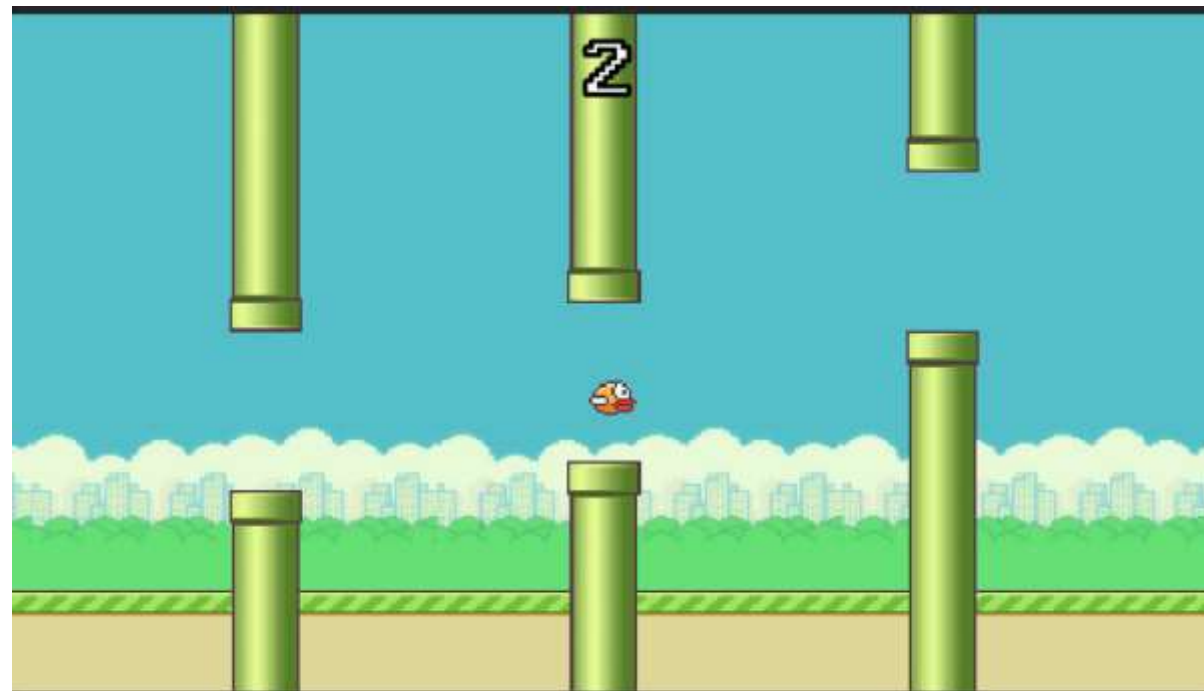
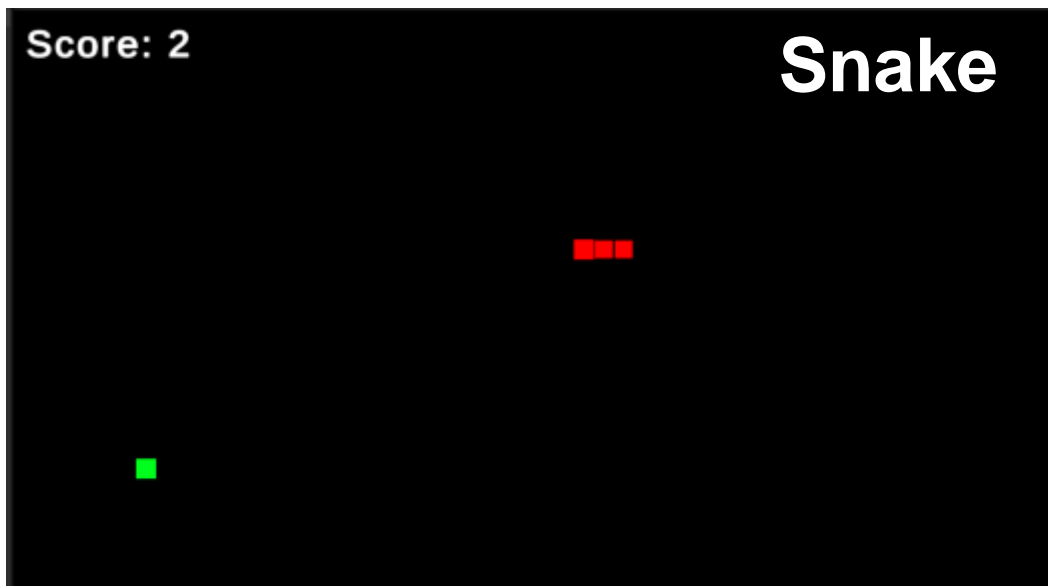
 Notes  About

Получить C#.

Сохранить

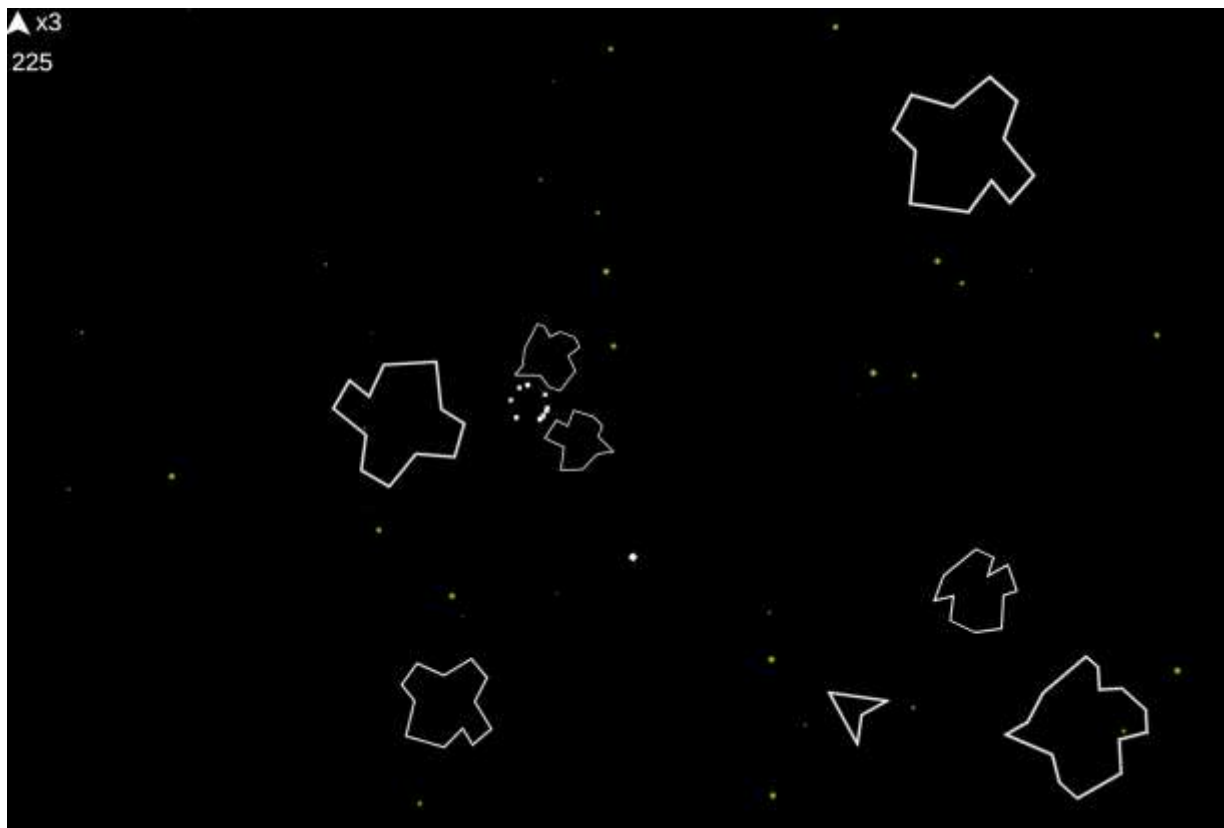
Удалить

# Что будем изучать. 2 Семестр

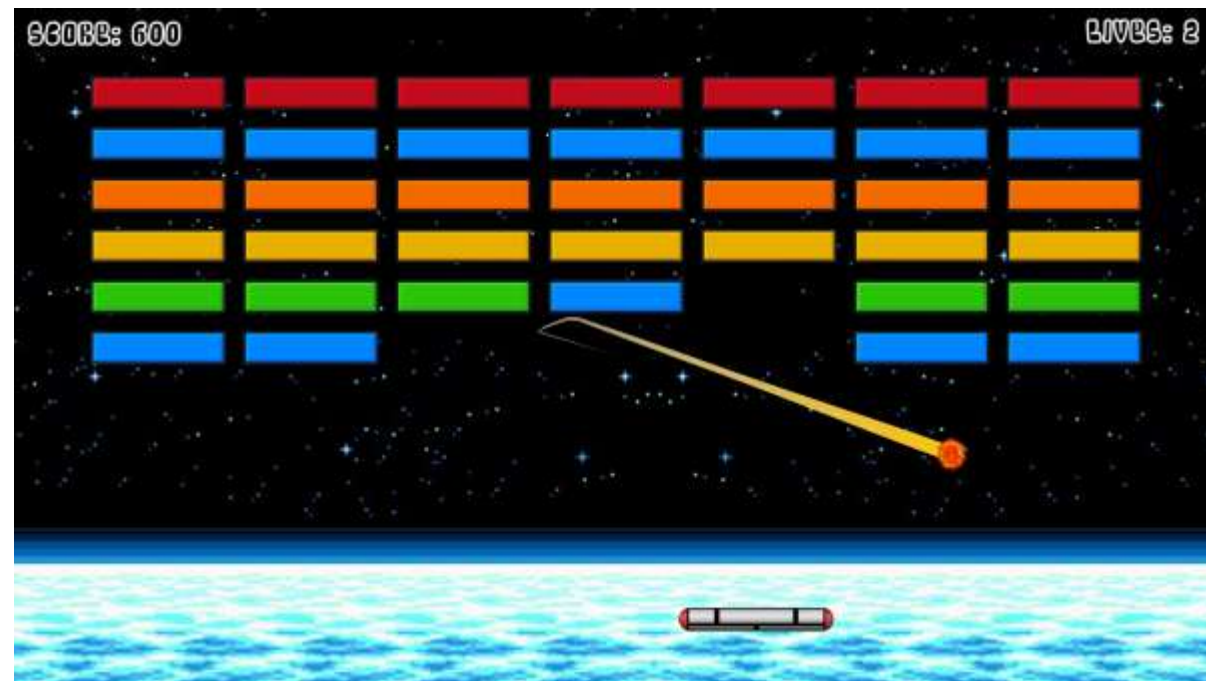


Flappy Bird

# Что будем изучать. 2 Семестр



**Asteroid**

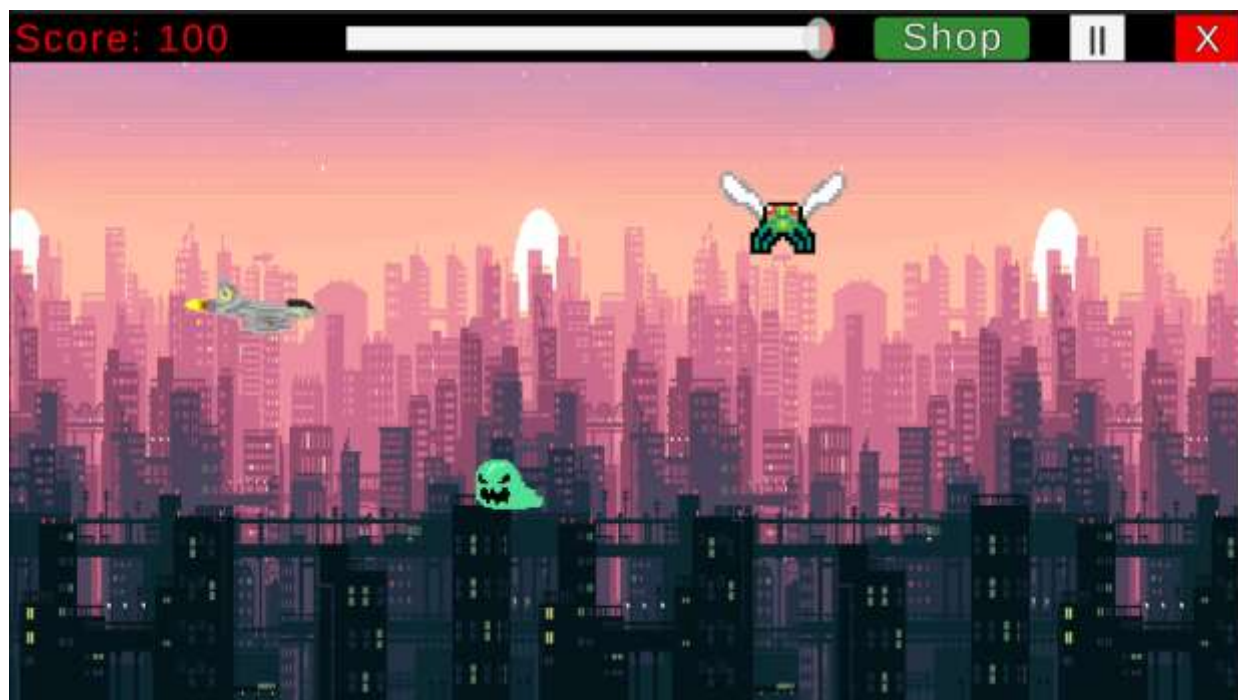


**Brick Breaker**





## Полноценная мини игра



# Простой скрипт для случайного появления объекта еды на игровой области

```
1 using UnityEngine;
2
3 public class Food : MonoBehaviour
4 {
5     public BoxCollider2D areaFood;
6     private void Start()
7     {
8         RandomPosition();
9     }
10
11     private void OnTriggerEnter2D(Collider2D other)
12     {
13         if (other.CompareTag("Player"))
14         {
15             RandomPosition();
16         }
17     }
18
19     private void RandomPosition()
20     {
21         var bounds = areaFood.bounds;
22         var x :float = Random.Range(bounds.min.x, bounds.max.x);
23         var y :float = Random.Range(bounds.min.y, bounds.max.y);
24         transform.position = new Vector2(Mathf.Round(x),Mathf.Round(y));
25     }
26 }
```



```

1 using System.Collections;
2 using TMPro;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 * 1 asset usage * Paltosik92 *
7 public class BossSpawner : MonoBehaviour
8 {
9     [Header("Time Settings")]
10    [SerializeField] private float timeBeforeBoss = 90f; * Unchanged
11    [SerializeField] private float bossSpawnDelay = 5f; * Unchanged
12
13    [Header("Objects")]
14    [SerializeField] private GameObject bossPrefab; * Boss
15    [SerializeField] private Slider bossHealthBar; * BossHP (Slider)
16    [SerializeField] private TextMeshProUGUI warningText; * Warning (TextMeshProUGUI)
17    [SerializeField] private GameObject enemySpawner; * EnemySpawner
18
19    * Event Function * Paltosik92
20    private void Start()
21    {
22        StartCoroutine(Routine.BossSequence());
23    }
24
25    * Frequently called * 1 usage * new *
26    private void ShowWarning()
27    {
28        warningText.gameObject.SetActive(true);
29        AudioManager.Instance?.PlayWarningSound();
30    }
31
32    private IEnumerator BossSequence()
33    {
34        yield return new WaitForSeconds(timeBeforeBoss);
35        ShowWarning();
36        yield return StartCoroutine(Routine.FlashWarningText(flashes: 10, interval: 0.5f));
37        HideWarning();
38        StopEnemySpawner();
39        yield return new WaitForSeconds(bossSpawnDelay);
40        SpawnBoss();
41    }

```

```

42    * Frequently called * 1 usage * Paltosik92
43    private IEnumerator FlashWarningText(int flashes, float interval)
44    {
45        for (int i = 0; i < flashes; i++)
46        {
47            warningText.enabled = !warningText.enabled;
48            yield return new WaitForSeconds(interval);
49        }
50    }
51
52    * Frequently called * 1 usage * Paltosik92
53    private void HideWarning()
54    {
55        warningText.enabled = true;
56        warningText.gameObject.SetActive(false);
57    }
58
59    private void StopEnemySpawner()
60    {
61        if (enemySpawner != null)
62        {
63            enemySpawner.GetComponent<EnemySpawner>()?.StopSpawning();
64            enemySpawner.SetActive(false);
65        }
66        AudioManager.Instance?.PlayBossMusic();
67    }
68
69    * Frequently called * 1 usage * Paltosik92
70    private void SpawnBoss()
71    {
72        var boss :GameObject = Instantiate(bossPrefab, new Vector3(6.5f, 0, 0), Quaternion.identity);
73
74        if (bossHealthBar != null && boss.TryGetComponent(out BossHealth bossHealth))
75        {
76            bossHealth.AssignHealthBar(bossHealthBar);
77            bossHealthBar.gameObject.SetActive(true);
78        }
79    }

```

Код для спавна босса в Unity

# Баллы

- Посещение занятий;
- Работа на парах;
- Выполнение домашних работ;
- Защита мини-проектов.

ИСРПО (1 семестр)			
Кол-во занятий	Работа на парах (баллы)	ДЗ (баллы)	Инд.проекты (баллы)
1	2	1	15
2	2	1	
3	2	1	
4	2	1	
5	2	1	
6	2	1	
7	2	1	
8	2	1	
9	2	1	
10	2	1	
11	2	1	25
12	2	1	
13	2	1	
14	2	1	
15	2	1	
16	2	1	
17	2	1	
18	2	1	
19	2	1	
20	2	1	
Итог:	40	20	40
Общий итог:			100

Необходимо создать учётную запись на **github**.  
Все домашние работы высылать **ссылкой** на  
репозиторий (предварительно не забудьте  
сделать его **публичным**).

*Лучше заранее создать google учётную запись*

# Сравнение C# с языком Python

# Сравнение программы “Hello world” на двух языках

```
print("Hello world!")
```

На Python

```
using System;

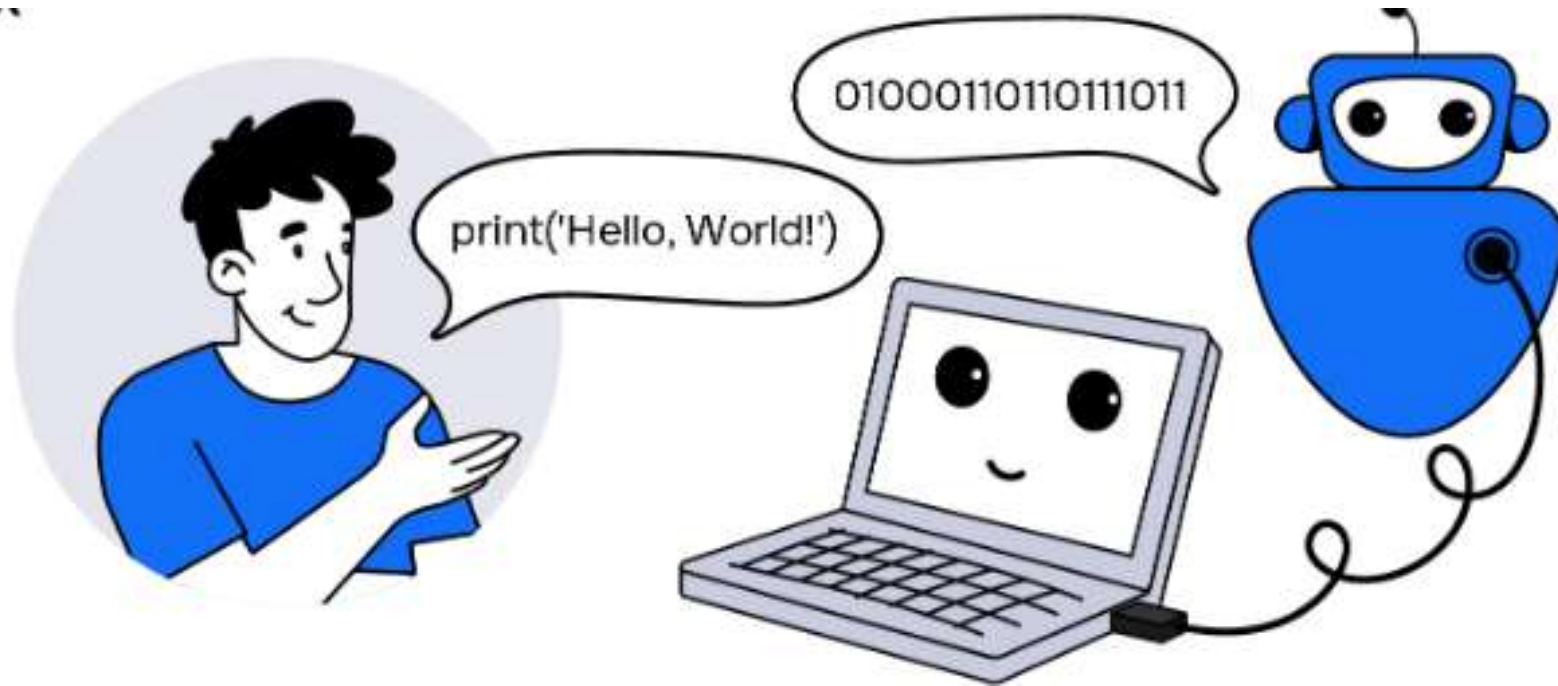
namespace HelloWorld
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello world!");
        }
    }
}
```

На C#

<b>Характеристика</b>	<b>C#</b>	<b>Python</b>
<b>Скорость выполнения</b>	Выше (JIT-компиляция)	Ниже (интерпретация)
<b>Поддержка ООП</b>	Очень сильная, строгая модель	Гибкая, но менее строгая модель
<b>Использование</b>	Корпоративные приложения, игры (Unity), сервисы под Windows	Скрипты, анализ данных, веб-разработка, машинное обучение
<b>Запуск программы</b>	Компилируется перед выполнением	Выполняется сразу (интерпретация)

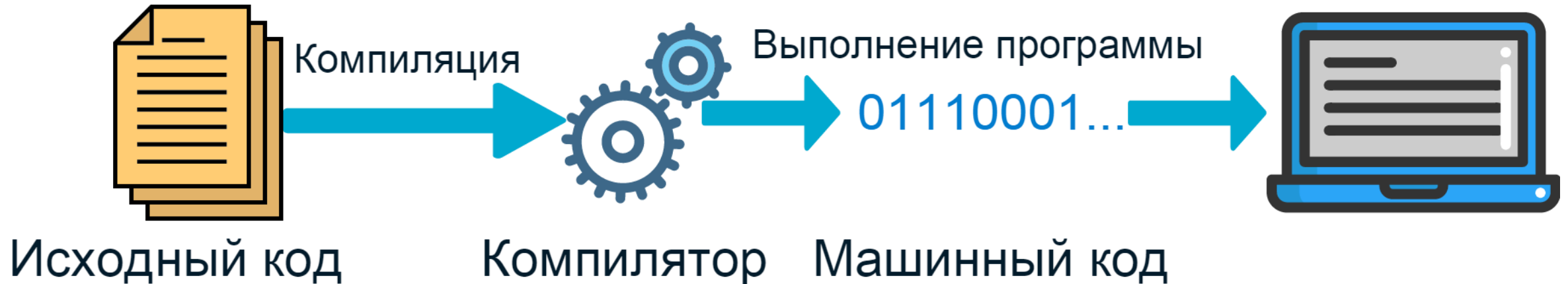
# Компилируемые и интерпретируемые языки

**Компилятор** переводит весь код в машинный (состоящий из нулей и единиц) сразу же при запуске программы.



# Компилируемые и интерпретируемые языки

Программист пишет программу на языке программирования → Запускает компилятор → Компилятор переводит всю программу в машинный код и кладет его в исполняемый файл (.exe в Windows)

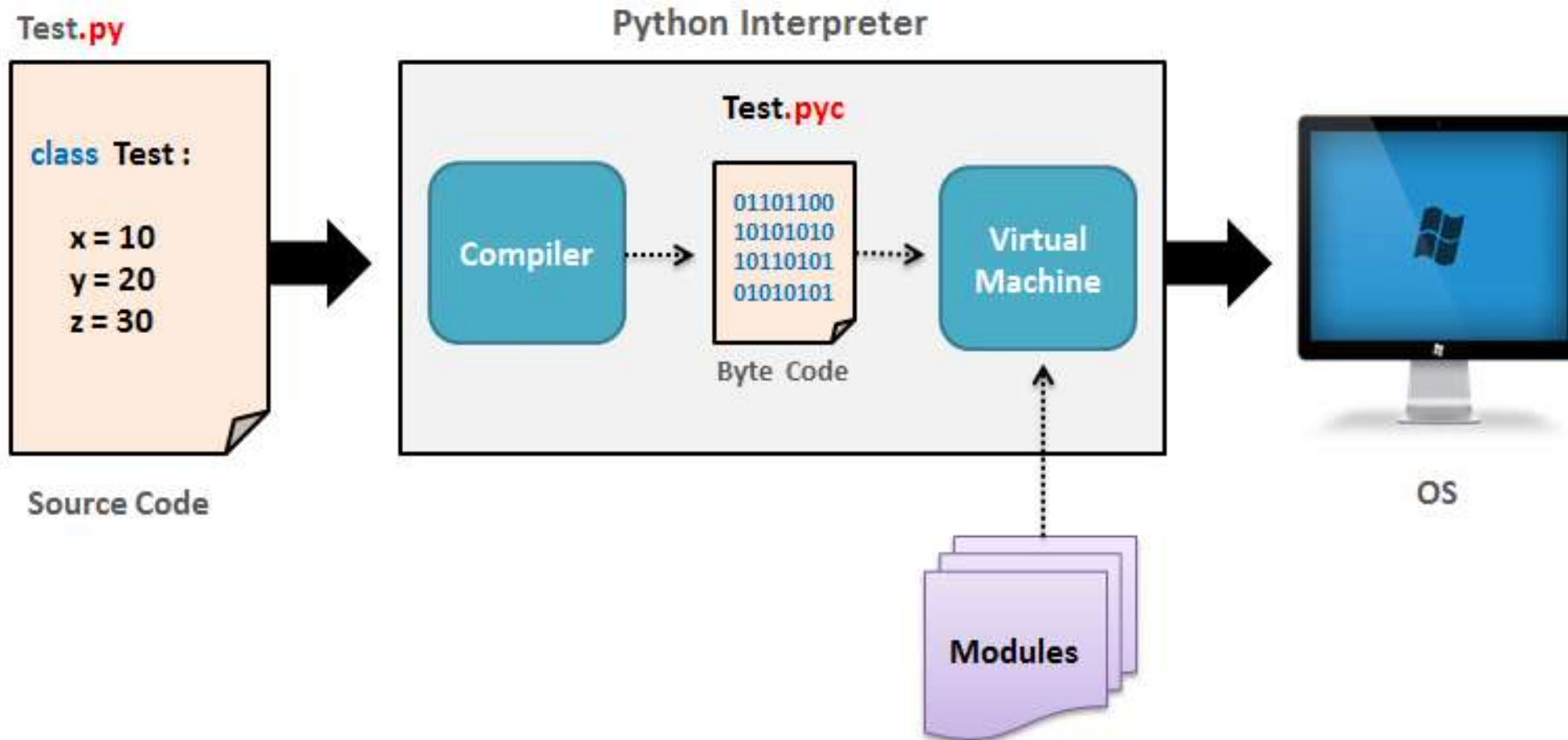


**C, C++, Java, C#, Swift, Go.**



# Компилируемые и интерпретируемые языки

**Интерпретатор** переводит код в машинный построочно каждый раз, когда запускается программа.

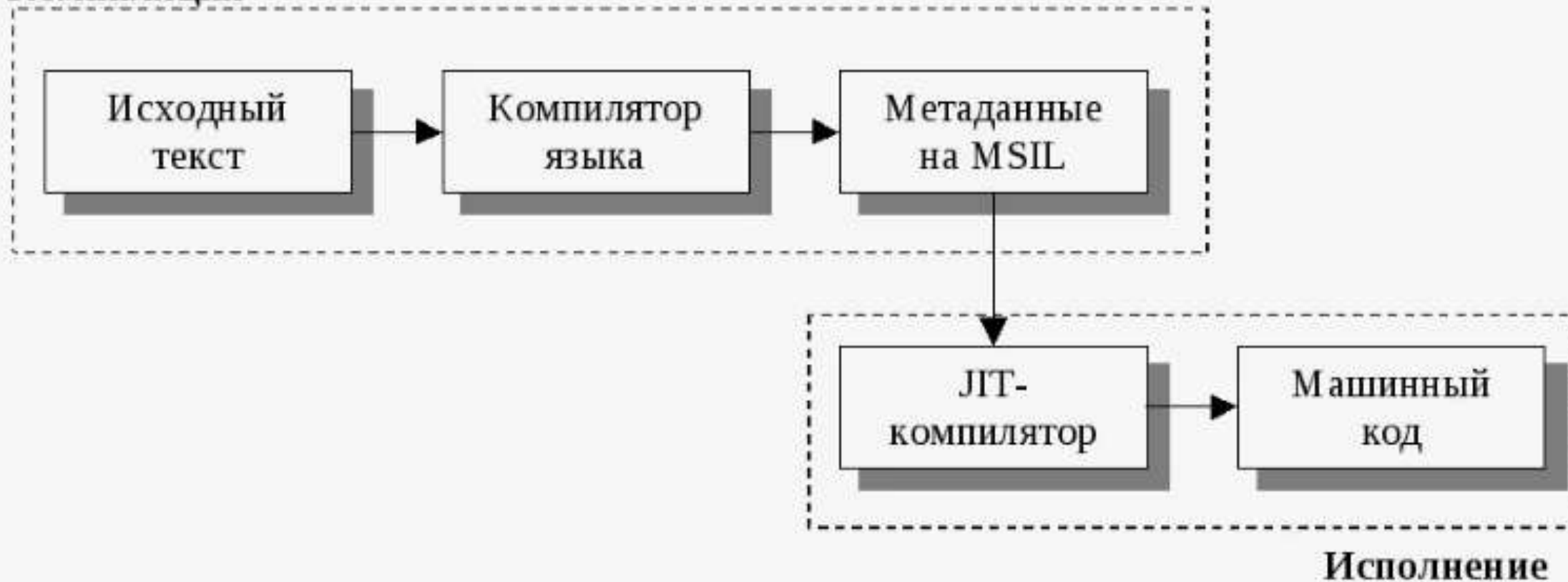


# Компилируемые и интерпретируемые языки

**Python, PHP, JavaScript.**

Программы, написанные на интерпретируемых языках, чаще всего запускаются медленнее компилируемых программ — как раз из-за строчного перевода кода. Однако процесс разработки на интерпретируемом языке быстрее, чем на компилируемом, так как программисту не нужно снова и снова компилировать программу в машинный язык.

## Компиляция



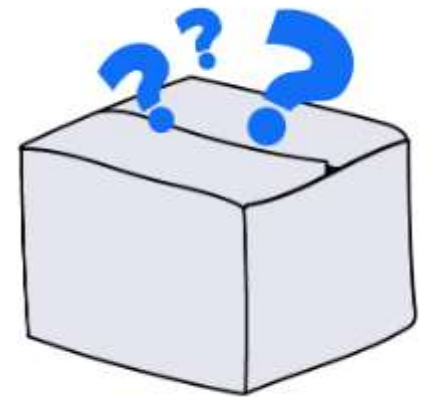
# C#

# Языки со статической типизацией и динамической типизацией

**Типизация** — это набор правил, по которым язык программирования классифицирует информацию. Если у языка нет типизации, программист может присваивать переменной любой тип данных (строку, число), а потом отнести к этой же переменной другой тип данных. Это позволяет быстрее писать код, но в таком коде проще запутаться.



Статическая типизация



Динамическая типизация

**Статическая** типизация определяет типы данных в программе до ее запуска (во время компиляции) (Java, C++, Swift).

В **динамически-типизированных** языках тип переменной определяется во время запуска программы. Объявлять тип переменной в явном виде не нужно: интерпретатор определяет его в процессе работы программы (Python, JavaScript, Ruby).

```
Python Shell
>>> x = 4
>>> x = 10
>>> x = 3.4
>>> print(x)
3.4
>>>
```

**Python**

```
C#
int x = 4;
x = 10;
x = 3.4;
```

**C#**

Но бывает и исключения (динамический тип данных в C#):

```
static void Main(string[] args)
{
    dynamic x = 4;
    Console.WriteLine(x);
    Console.WriteLine(x.GetType());
    x = "hello";
    Console.WriteLine(x);
    Console.WriteLine(x.GetType());
    x = 3.4;
    Console.WriteLine(x);
    Console.WriteLine(x.GetType());
}
```

Консоль отладки Microsoft V

```
4
System.Int32
hello
System.String
3,4
System.Double
C:\Users\Palto
```

# Цикл for (вывод чисел от 1 до 10)

```
for i in range(1, 11):  
    print(i)
```

На Python

```
for (int i = 1; i <= 10; i++)  
{  
    Console.WriteLine(i);  
}
```

На C#

# Условия (проверка числа)

```
num = 12
if num < 10:
    print("Меньше 10")
elif 10 <= num <= 15:
    print("От 10 до 15")
else:
    print("Больше 15")
```

На Python

```
int num = 12;
if (num < 10)
{
    Console.WriteLine("Меньше 10");
}
else if (num >= 10 && num <= 15)
{
    Console.WriteLine("От 10 до 15");
}
else
{
    Console.WriteLine("Больше 15");
}
```

На C#



# Функция/Метод для суммы двух чисел

```
def sum_numbers(a, b):  
    return a + b  
  
result = sum_numbers(5, 3)  
print(result) # 8
```

На Python

```
static int SumNumbers(int a, int b)  
{  
    return a + b;  
}  
  
int result = SumNumbers(5, 3);  
Console.WriteLine(result); // 8
```

На C#

# Простой класс Студент

```
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Привет, меня зовут {self.name}!")

# Использование
student = Student("Алексей", 20)
student.greet() # Привет, меня зовут Алексей!
```

На Python

На C#

```
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        // Использование
        Student student = new("Алексей", 20);
        student.Greet(); // Привет, меня зовут Алексей!
    }
}

3 references
public class Student
{
    2 references
    public string Name { get; set; }
    1 reference
    public int Age { get; set; }

    1 reference
    public Student(string name, int age)
    {
        Name = name;
        Age = age;
    }

    1 reference
    public void Greet()
    {
        Console.WriteLine($"Привет, меня зовут {Name}!");
    }
}
```

# Преимущества и недостатки C#

## Плюсы:

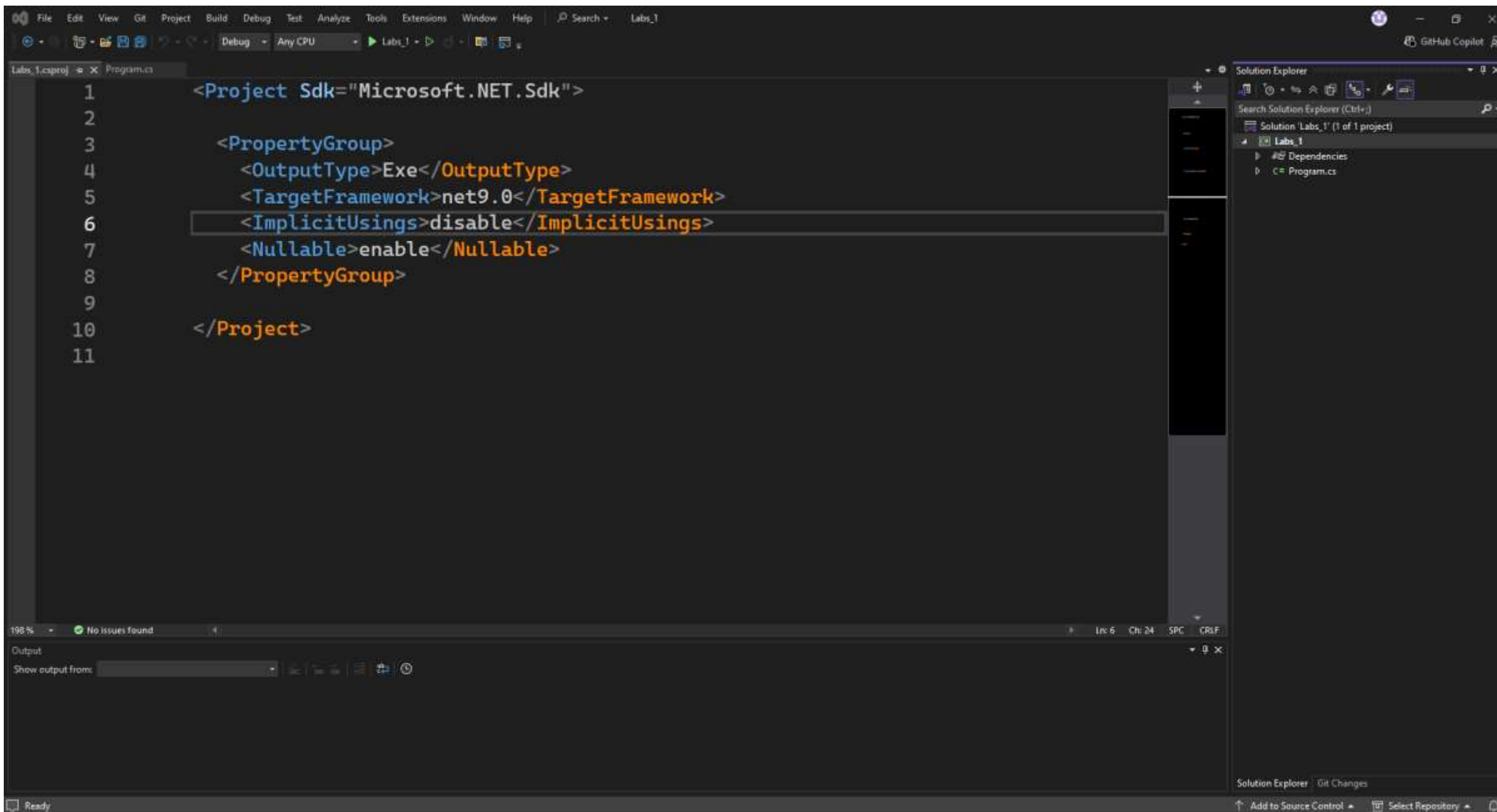
- Отличная документация и поддержка.
- Высокий уровень инструментов (Visual Studio).
- Подходит как для новичков, так и для крупных проектов.

## Минусы:

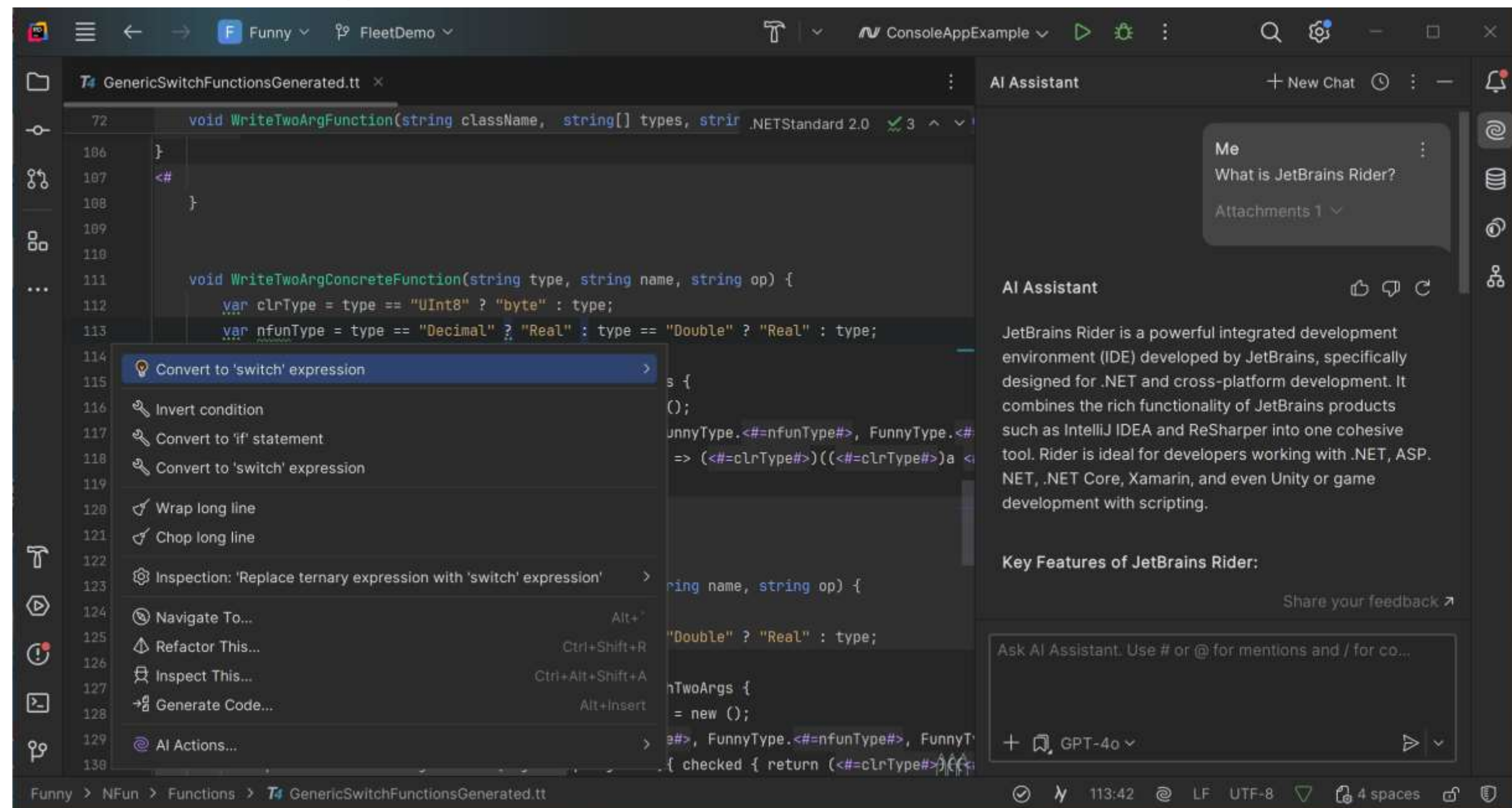
- Немного «шумный» синтаксис по сравнению с Python.
- Входной порог выше.
- Не везде подходит.

**Инструменты, которые  
будем использовать**

# • IDE: Visual Studio 2022 (основная)

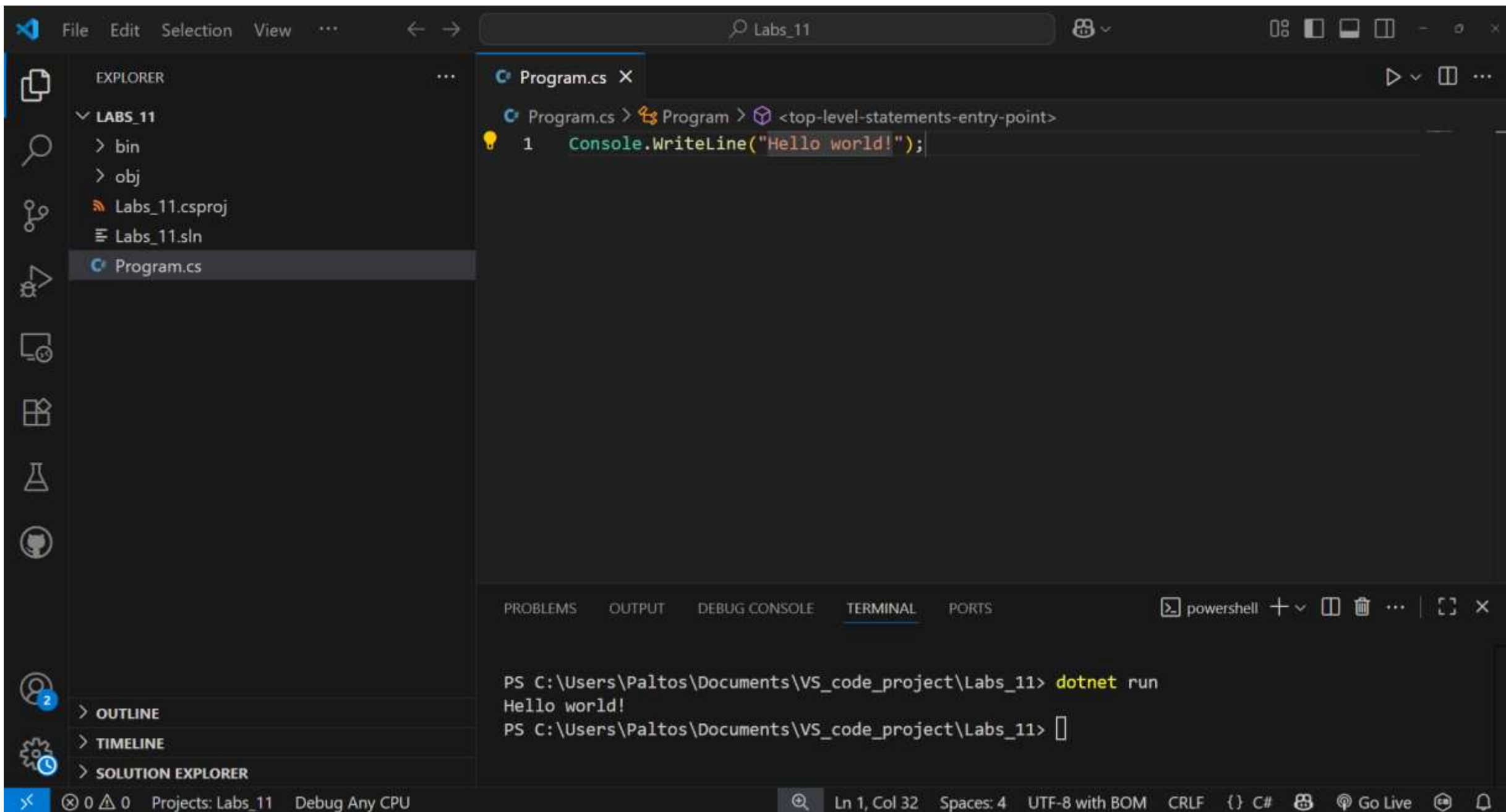


# • IDE: Rider (JetBrains)





# • VS Code (легковесная).





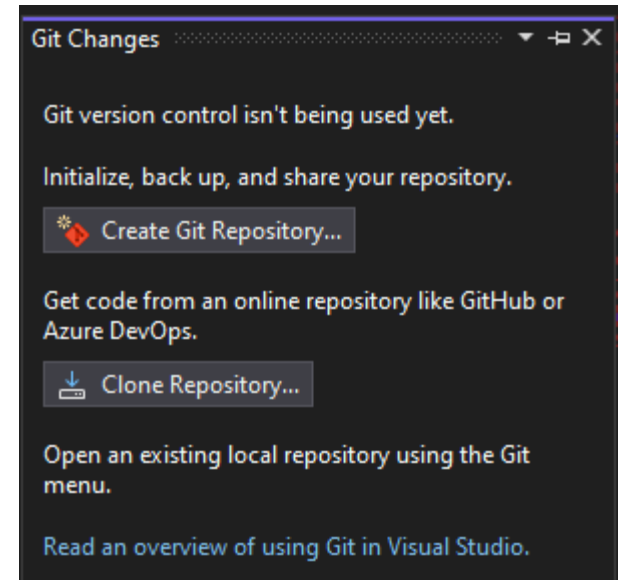
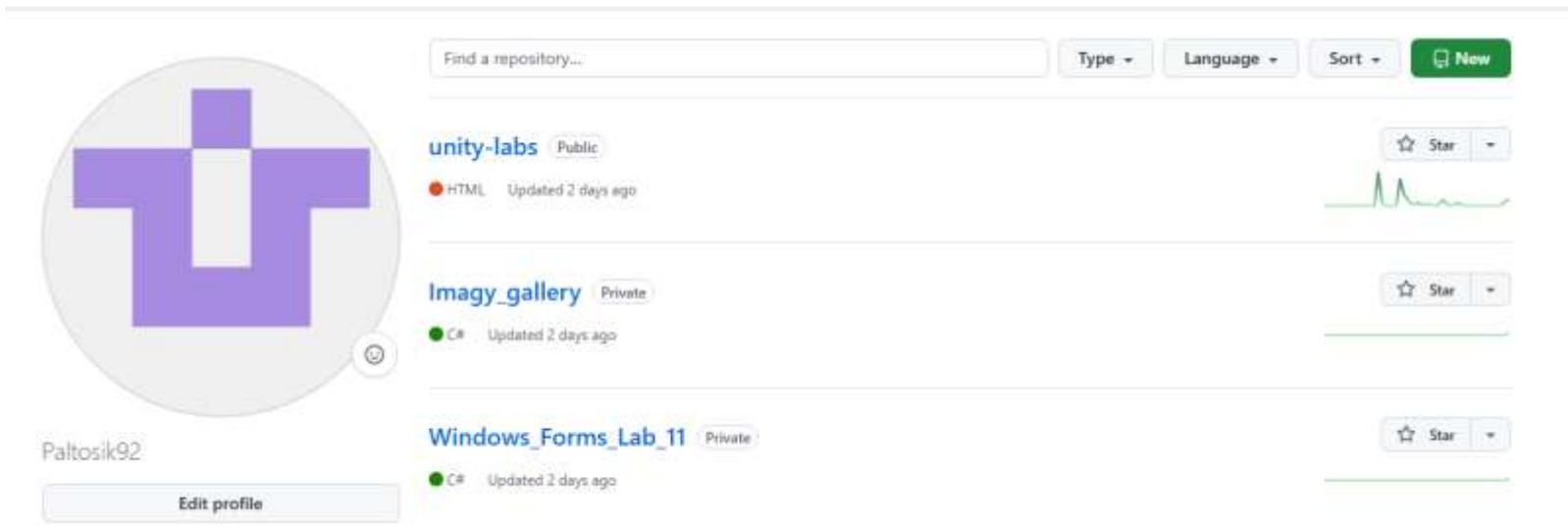
# Инструменты, которые будем использовать

- Git + GitHub: зачем нужен контроль версий, как вести свой репозиторий.

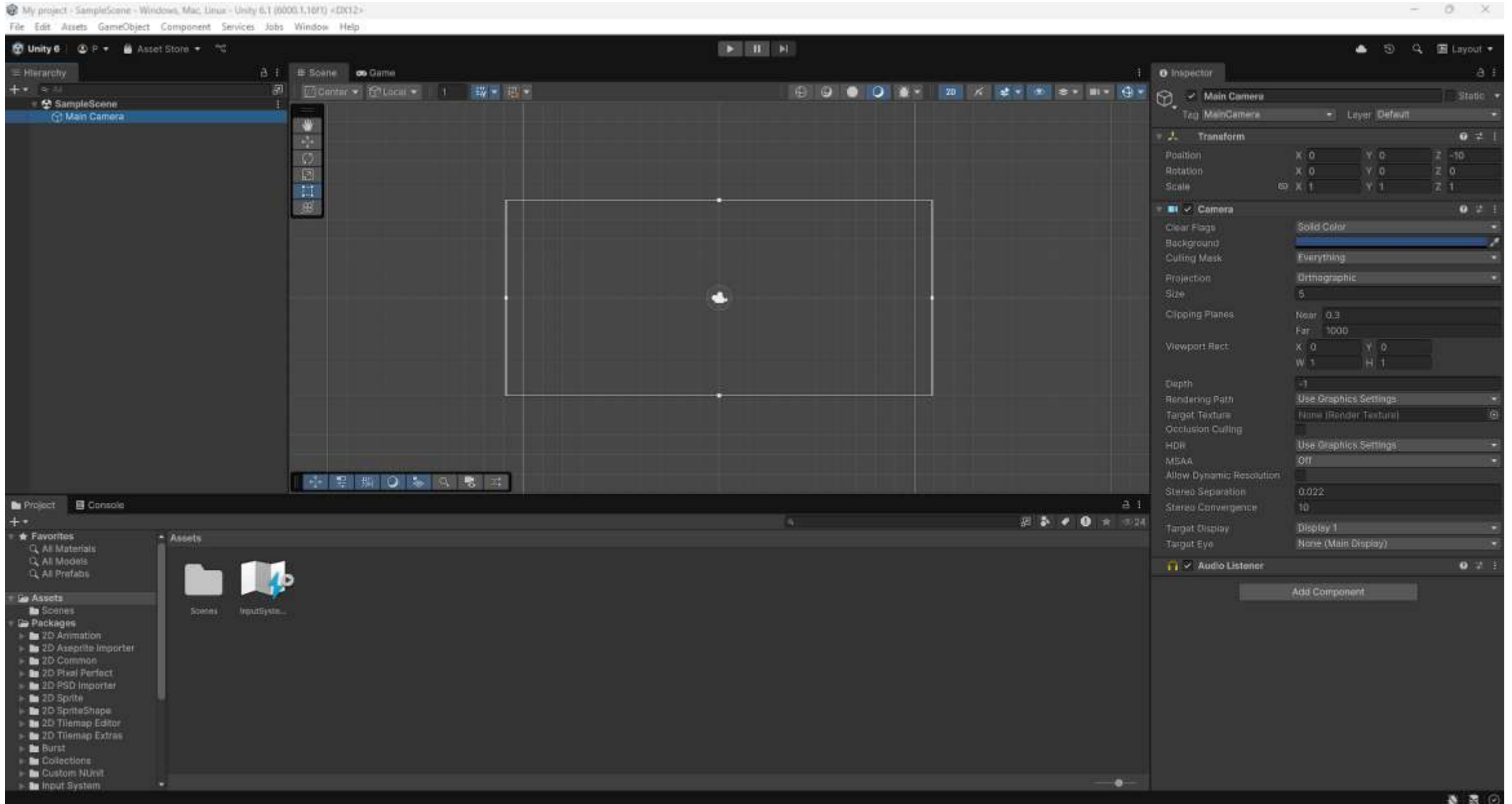
```
MINGW64:/c/Users/Paltos

Paltos@DESKTOP-L950PDM MINGW64 ~
$ git --version
git version 2.50.1.windows.1

Paltos@DESKTOP-L950PDM MINGW64 ~
$
```



- Unity Editor: только во втором семестре.



# Инструменты для работы:

## VS 2022 vs VS Code vs Rider

Инструмент	Особенности
<b>Visual Studio 2022</b>	Полноценная IDE, мощный отладчик, дизайнер форм, интеграция с .NET
<b>VS Code</b>	Лёгкий редактор, подходит для новичков, требует настройки, есть на всех платформах
<b>JetBrains Rider</b>	Удобен, особенно для опытных, классный AI, есть на всех платформах

# **Рекомендуемая литература и ресурсы**












# Топ-1

**METANIT.COM**

Сайт о программировании

**Полное руководство по языку программирования C# 13 и платформе .NET 9**

<https://metanit.com/sharp/tutorial/>

- ▶  Глава 1. Введение в C#
- ▶  Глава 2. Основы программирования на C#
- ▶  Глава 3. Классы, структуры и пространства имен
- ▶  Глава 4. Объектно-ориентированное программирование
- ▶  Глава 5. Обработка исключений
- ▶  Глава 6. Делегаты, события и лямбды
- ▶  Глава 7. Интерфейсы
- ▶  Глава 8. Дополнительные возможности ООП в C#
- ▶  Глава 9. Pattern matching
- ▶  Глава 10. Коллекции
- ▶  Глава 11. Работа со строками

# Документация Microsoft

Документация по C#

> Начало работы

▼ Основы

▼ Структура программы

Обзор

Метод Main

Инструкции верхнего уровня

> Система типов

> Объектно-ориентированное программирование

> Функциональные методики

> Исключения и ошибки

> Стил код

> Учебные пособия

> Новые возможности C#

> Учебные пособия

> Языково-интегрированный запрос (LINQ)

> Асинхронное программирование

> Основные понятия C#

> Статьи с практическими инструкциями по C#

> Дополнительные разделы

> Пакет SDK для платформы компилятора .NET (API-интерфейсы Roslyn)

> Руководство по программированию на C#

> Другая документация по C#

## Операторы верхнего уровня — программы без Main методов

01.07.2025

Не нужно явно включать `Main` метод в проект консольного приложения. Вместо этого можно использовать *операторы верхнего уровня* для минимизации объема создаваемого кода.

Инструкции верхнего уровня позволяют создавать исполняемый код непосредственно в корне файла, устраняя необходимость упаковки кода в класс или метод. Это означает, что вы можете создавать программы без церемонии `Program` класса и `Main` метода. В этом случае компилятор создает `Program` класс с методом точки входа для приложения. Имя созданного метода не `Main` является именем, это сведения о реализации, на которые код не может ссылаться напрямую.

Ниже приведен файл `Program.cs`, который является полной программой C#:

C#

Копировать

```
Console.WriteLine("Hello World!");
```

Операторы верхнего уровня позволяют создавать простой программный код для небольших служебных программ, таких как Функции Azure и GitHub Actions. Они также помогают начинающим программистам C# начать обучение и приступить к написанию кода.

В следующих разделах описываются правила, которые определяют использование операторов верхнего уровня.

<https://learn.microsoft.com/ru-ru/dotnet/csharp/>

# Платформа для изучения технологий Microsoft

Microsoft Learn.



## Write your first C# code

34 min • Module • 7 Units

[Feedback](#)

Beginner Developer .NET

Get started by writing code examples to learn the basics of the C# syntax.

### Learning objectives

900 XP

Learn / Training / Browse / Write your first C# code /

Unit 2 of 7

Ask Learn

100 XP

## Exercise - Write your first code

6 minutes

In this first hands-on exercise, you'll use C# to print a hallowed programmer's phrase to the standard output of a console.

### Write your first line of code

There's a long-standing tradition among software developers to print the phrase "Hello World!" to the console output window. As you'll experience, you can learn a lot about programming and the C# programming language from this simple exercise.

### Enter code into the .NET Editor

The .NET Editor and output console provide a great in-browser experience that's perfect for this tutorial approach. The .NET Editor is located on the right-hand side of this web page. The output console is below it.

1. Enter this code exactly as it appears into the .NET Editor on the right:

C#

Copy

```
Console.WriteLine("Hello World!");
```

.NET Editor

Press CTRL + N, TAB to exit the editor

Clear

Run

ⓘ

1

✓

✓

✓

✓

✓

✓

✓

[Review the solution](#)

3 min

[Module assessment](#)

10 min

[Summary](#)

1 min

<https://learn.microsoft.com/>



## PRO C#. Основы программирования

Курс посвящен базовым понятиям программирования: типы данных, операторы, переменные, условия, циклы, массивы и функции. Он является вводным и подойдет слушателям с небольшим опытом или вообще без опыта программирования. Обучение будет проходить на языке программирования C#.



Начальный  
уровень



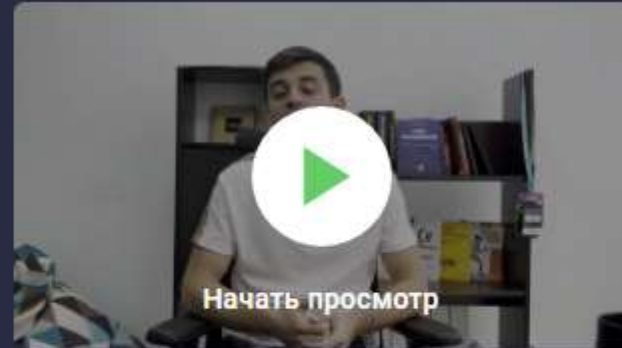
3-6 часов в  
неделю



Часть  
5 программ →



Сертификат  
Stepik



★★★★★ 4.9  
98 751 учащийся

[990 отзывов](#)

<https://stepik.org/course/5482/promo>

**КНИГИ**

EXPERT INSIGHT

# C# 13 and .NET 9

## Modern Cross-Platform Development Fundamentals

Start building websites and services with ASP.NET Core 9,  
Blazor, and EF Core 9



Ninth Edition



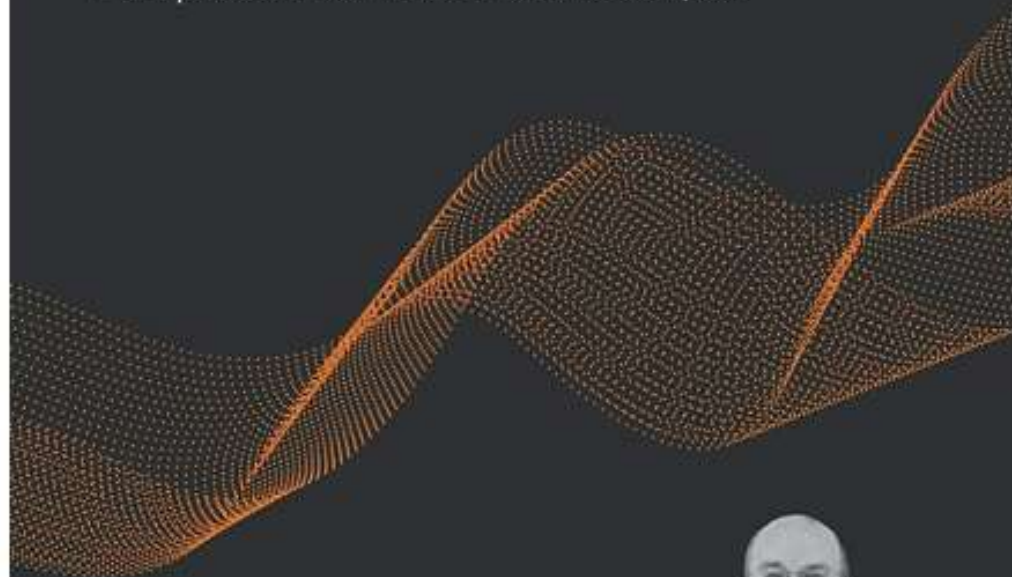
Mark J. Price

packt

EXPERT INSIGHT

# C# 9 и .NET 5

Разработка и оптимизация



Пятое издание



Марк Прайс

Packt

Всё обо всём



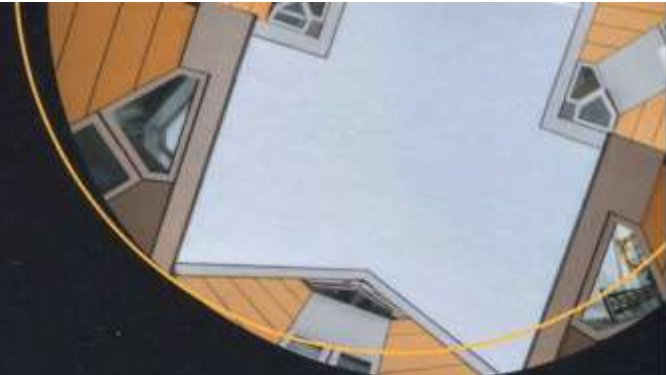
# Pro C# 10 with .NET 6

Foundational Principles and Practices  
in Programming

—  
*Eleventh Edition*

—  
Andrew Troelsen  
Phil Japikse

Apress®



# Язык программирования C# 9 и платформа .NET 5

ОСНОВНЫЕ ПРИНЦИПЫ И ПРАКТИКИ ПРОГРАММИРОВАНИЯ

10-Е ИЗДАНИЕ

Эндрю Троелсен, Филипп Джепикс

 ДИАЛЕКТИКА  
[www.dialektika.com](http://www.dialektika.com)

  
[www.apress.com](http://www.apress.com)

## Всё обо всём #2

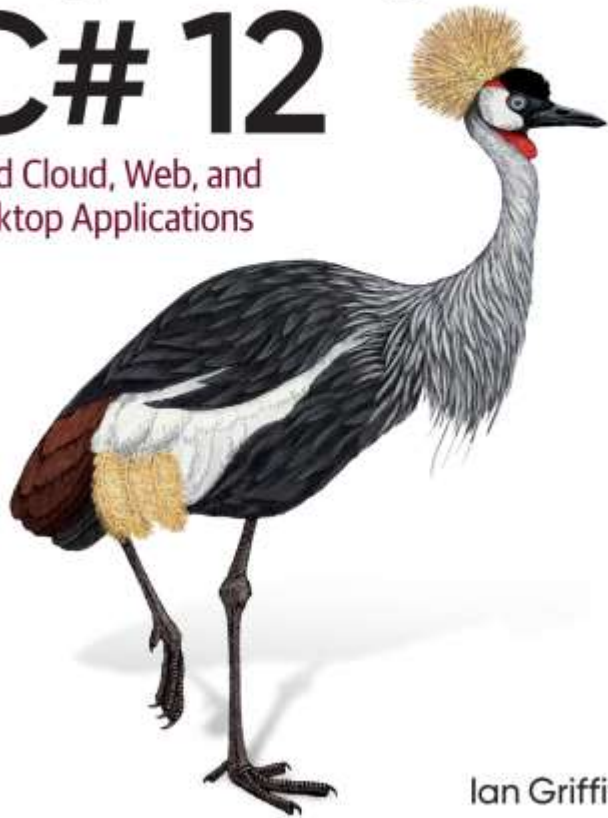


# Для любителей пожёстче. Ныряем в глубины C#.

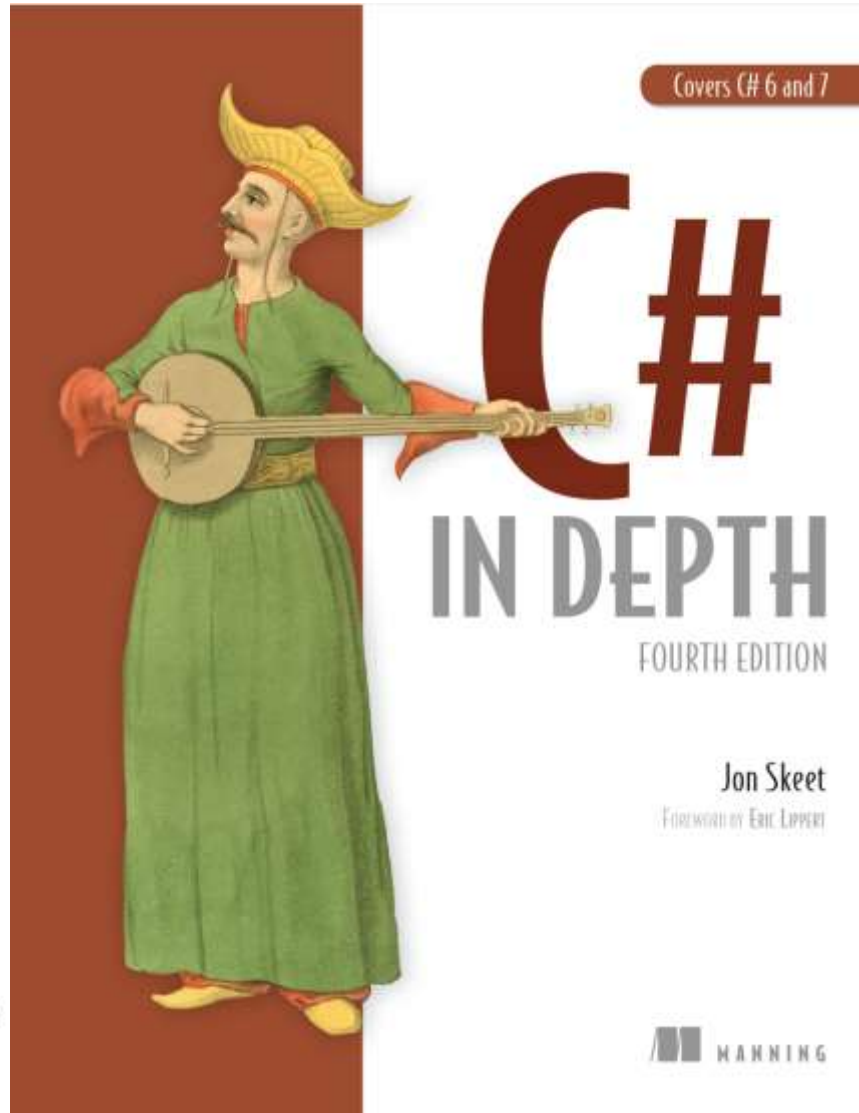
O'REILLY®

## Programming C# 12

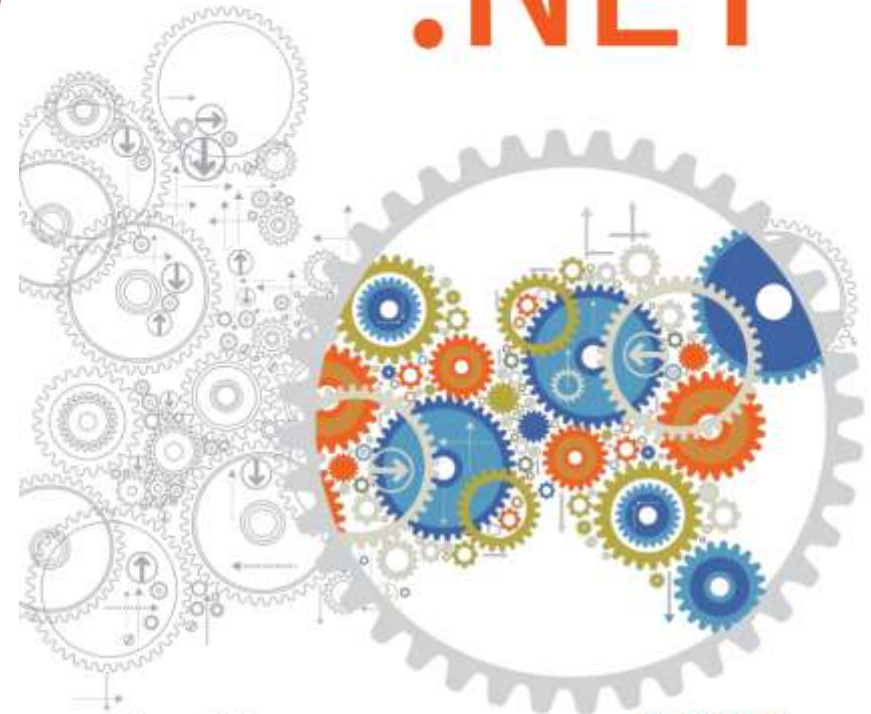
Build Cloud, Web, and  
Desktop Applications



Ian Griffiths



## ПАТТЕРНЫ проектирования на платформе .NET



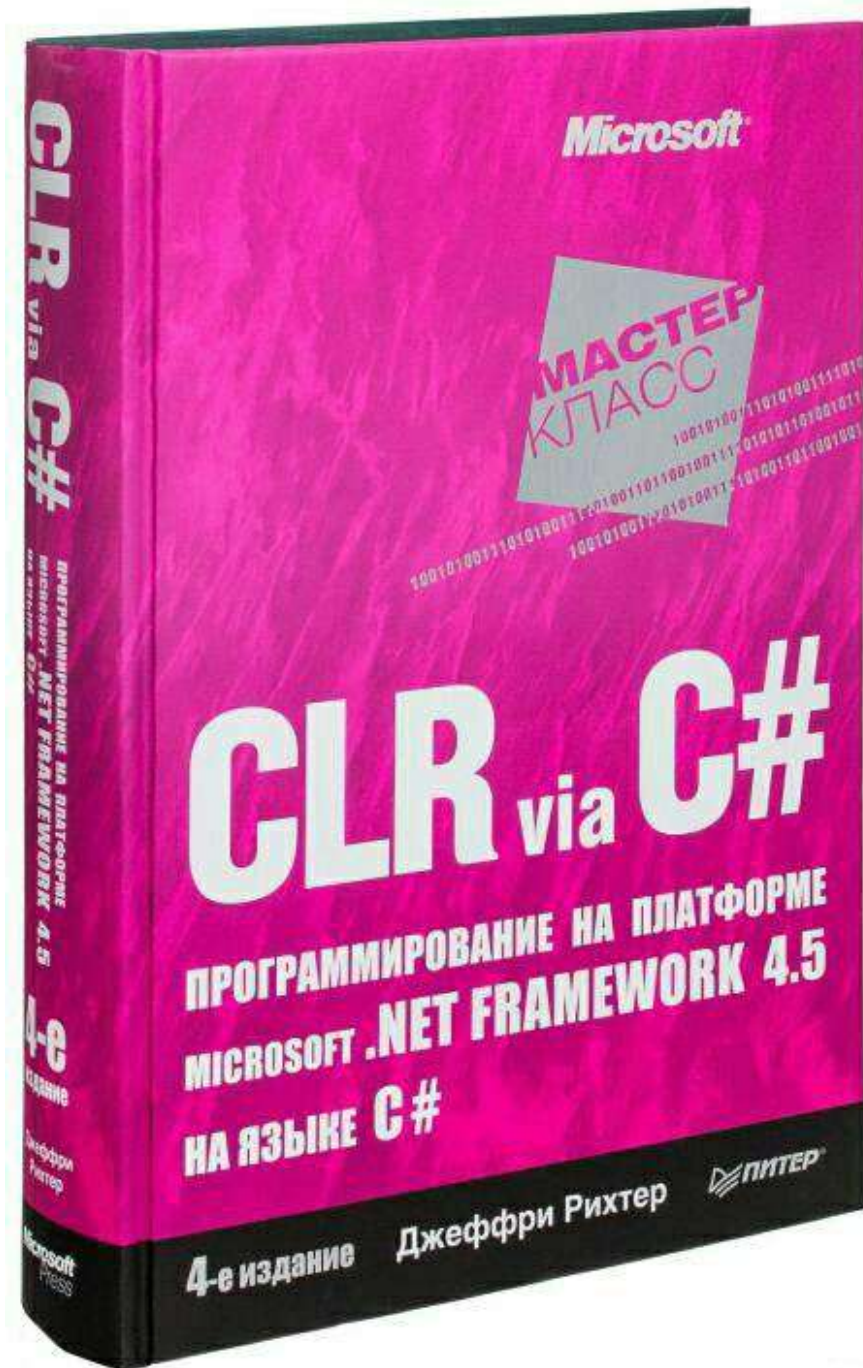
Сергей Тепляков

ПИТЕР®

# Бестселлер

«Азбука» для программистов на C#. В ней довольно мало внимания уделено конкретному языку программирования — упор сделан на платформу .Net и основные принципы ее работы.

Также на страницах этой книги вы найдете ответы на большую часть вопросов, которые любят задавать на собеседованиях по C# и .Net.



Если хотите уже сейчас  
погрузиться в **Unity**



# Изучаем C# через разработку игр на Unity

Пятое издание



Харрисон Ферроне



Packt

EXPERT INSIGHT

# Learning C# by Developing Games with Unity

Get to grips with coding in C# and build simple 3D games in Unity 2022  
from the ground up

Seventh Edition



<packt>

Harrison Ferrone

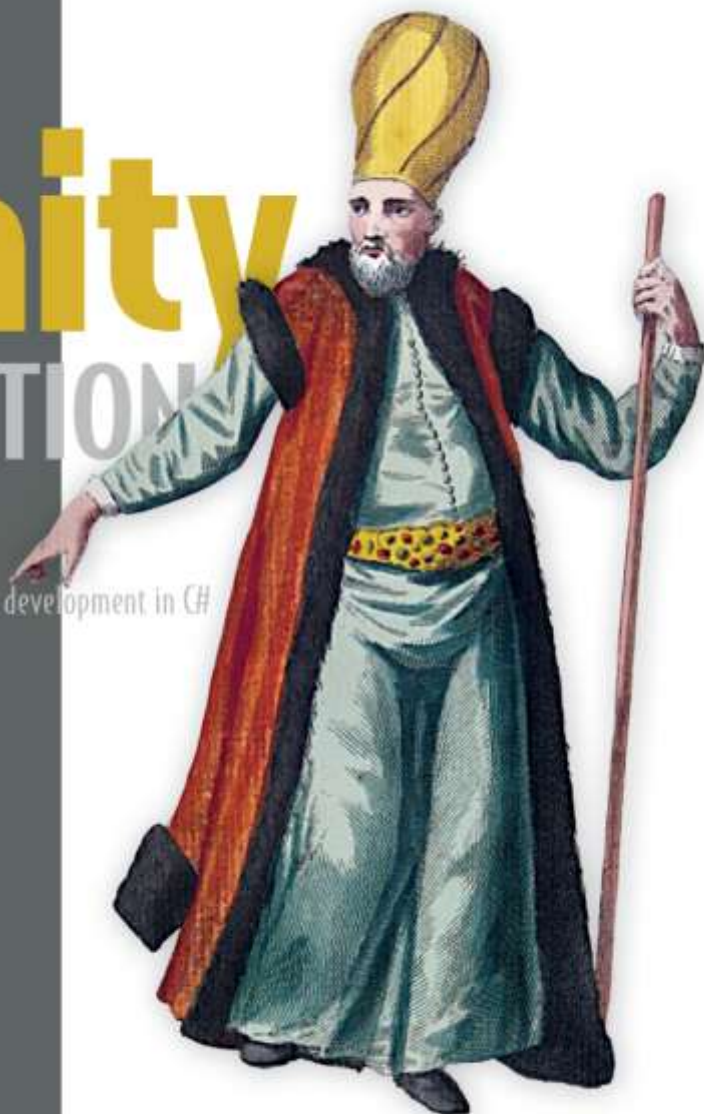
THIRD EDITION

# Unity IN ACTION

Multiplatform game development in C#

Joseph Hocking  
Foreword by Jesse Schell

 MANNING



# Unity В ДЕЙСТВИИ

Джозеф Хокинг

2-е международное  
издание

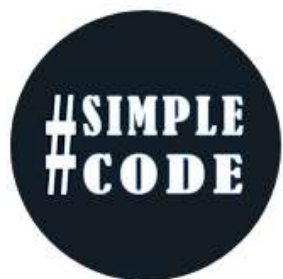
 MANNING



 ПИТЕР®



# Для любителей видео



## #SimpleCode

@SimpleCodeIT · 375K subscribers · 373 videos

Подробные уроки программирования на самых разных языках для новичков и н

[udemy.com/course/simplecode-csharp/?referralCode=53C0314076F77DCC2...](https://www.udemy.com/course/simplecode-csharp/?referralCode=53C0314076F77DCC2...) and 3

Subscribed

Home Videos Shorts Live Playlists Posts

### For You



События (Events) в C# | Проблемы Инкапсуляции и Делегаты | C# ОТ НОВИЧКА К...

4.5K views · 3 weeks ago



Искусственный интеллект ЗАМЕНИТ программистов ИЛИ НЕТ?

12K views · 2 months ago

<https://ulearn.me/>

Ulearn.me

Ошибки

- 01 Ошибки на этапе компиляции
- 02 Ошибки компиляции 0/0
- 03 Ошибки
- 04 Ошибки на этапе выполнения
- 05 Минимум функций 0/0
- 06 Отладка
- 07 Горячие клавиши отладки 0/1
- 08 Эксперименты
- 09 Стилистические ошибки
- 10 Рефакторинг
- 11 Дизайн кода 0/0
- 12 Константы и инициализация
- 13 Выделение методов
- 14 Рефакторинг и улучшение кода 0/4
- 15 Сделай то, не знаю что 0/0

Отладка

Basic Programming 3.9K subscribers

Subscribe

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Program
    {
        static double GetDiscriminant(double a, double b, double c)
        {
            return b * b - 4 * a * c;
        }

        static double GetFirstRoot(double a, double b, double c)
        {
            var discriminant = GetDiscriminant(a, b, c);
            var squareRoot = Math.Sqrt(discriminant);
            return (-b - squareRoot) / (2 * a);
        }

        static void Main()
        {
            GetFirstRoot(1, 1, 1);
        }
    }
}
```

Меню видео

00:05 — Об отладке программы в целом.

01:53 — Программа для решения квадратных уравнений.

05:00 — Иллюстрация поиска причины ошибки с помощью отладчика.

07:20 — Специальные значения double, NaN, zInfinity.

Ошибка в содержании? Предложить исправление!



@CodeMonkeyUnity · 565K subscribers · 1K videos

Hello and Welcome! ...more

unitycodemonkey.com and 4 more links



[Home](#)

## Videos

## Shorts

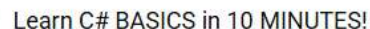
Live

## Playlists

## Posts



C#



Code Monkey  861K views • 4 years ago

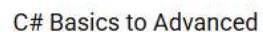

👉 Let's learn the Basics of C# in a quick Crash Course! 🌐 [FREE C# Beginner Complete Course!](#)

<https://www.youtube.com/watch?v=pReR6Z9rK-o> 📺 C# Basics to Advanced Playlist <https://www.youtube.com/watch?v=pReR6Z9rK-o>



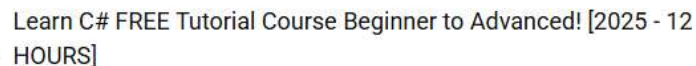
Code Monkey  27K views • 2 months ago

🌐 Watch my FREE 12 HOUR C# Course! [https://www.youtube.com/watch?v=qZpMX8Re\\_2Q](https://www.youtube.com/watch?v=qZpMX8Re_2Q) ✅ Get the Premium Version with Interactive Exercises! <https://cmonkey.co/csharpcourse> 🌐 FREE Game Dev...

Code Monkey · Playlist 

Learn C# BASICS in 10 MINUTES! • 10:55

What are Events? (C# Basics) • 15:05

[VIEW FULL PLAYLIST](#)

Code Monkey ✓ 153K views • 5 months ago

Get the Premium Course! <https://cmonkey.co/csharpcompletecourse> Learn by doing the Interactive Exercises and everything else in the companion project! Watch my FREE Complete...

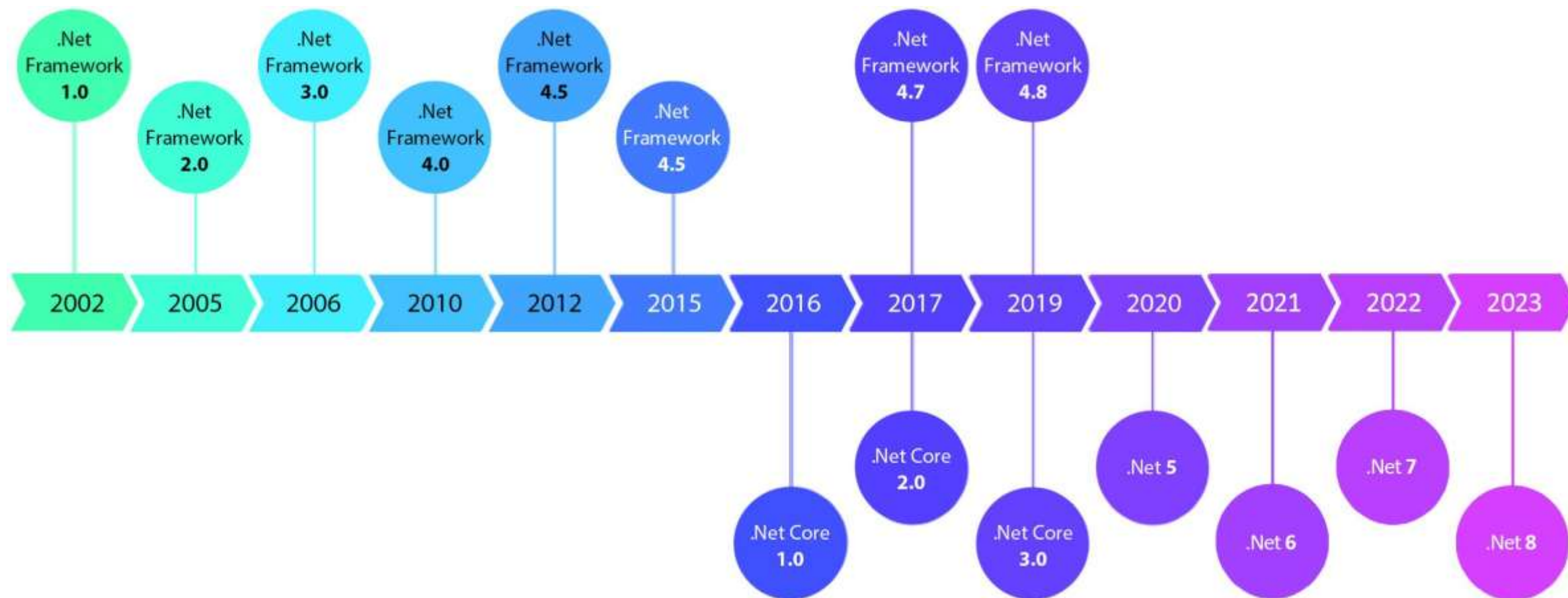
# Code Monkey

# **Введение в платформу .NET и язык программирования C#**

На сегодняшний момент язык программирования **C#** один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

**C#** уже не молодой язык и как и вся платформа **.NET** уже прошел большой путь. Первая версия языка вышла вместе с релизом **Microsoft Visual Studio .NET** в феврале **2002 года**. Текущей версией языка является версия **C# 13**, которая вышла 12 ноября **2024 года** вместе с релизом **.NET 9**.

# Краткая история





**C#** является языком с **Си-подобным** синтаксисом и близок в этом отношении к **C++** и **Java**.

**C#** является **объектно-ориентированным** и в этом плане много перенял у **Java** и **C++**. Например, **C#** поддерживает **полиморфизм**, **наследование**, **перегрузку операторов**, **статическую типизацию**. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

# Роль платформы .NET

Когда говорят **C#**, нередко имеют в виду технологии платформы **.NET** (**Windows Forms, WPF, ASP.NET, .NET MAUI**). И, наоборот, когда говорят **.NET**, нередко имеют в виду **C#**. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык **C#** был создан специально для работы с фреймворком **.NET**, однако само понятие **.NET** несколько шире.

## Разработка с помощью .NET



### Браузер

Создавайте веб-приложения и службы для macOS, Windows, Linux и Docker.



### Мобильные устройства и ПК

Используйте единую кодовую базу для создания собственных приложений Windows, macOS, iOS и Android.



### Облако

Создание масштабируемых и устойчивых облачных приложений, работающих со всеми основными поставщиками облачных служб.

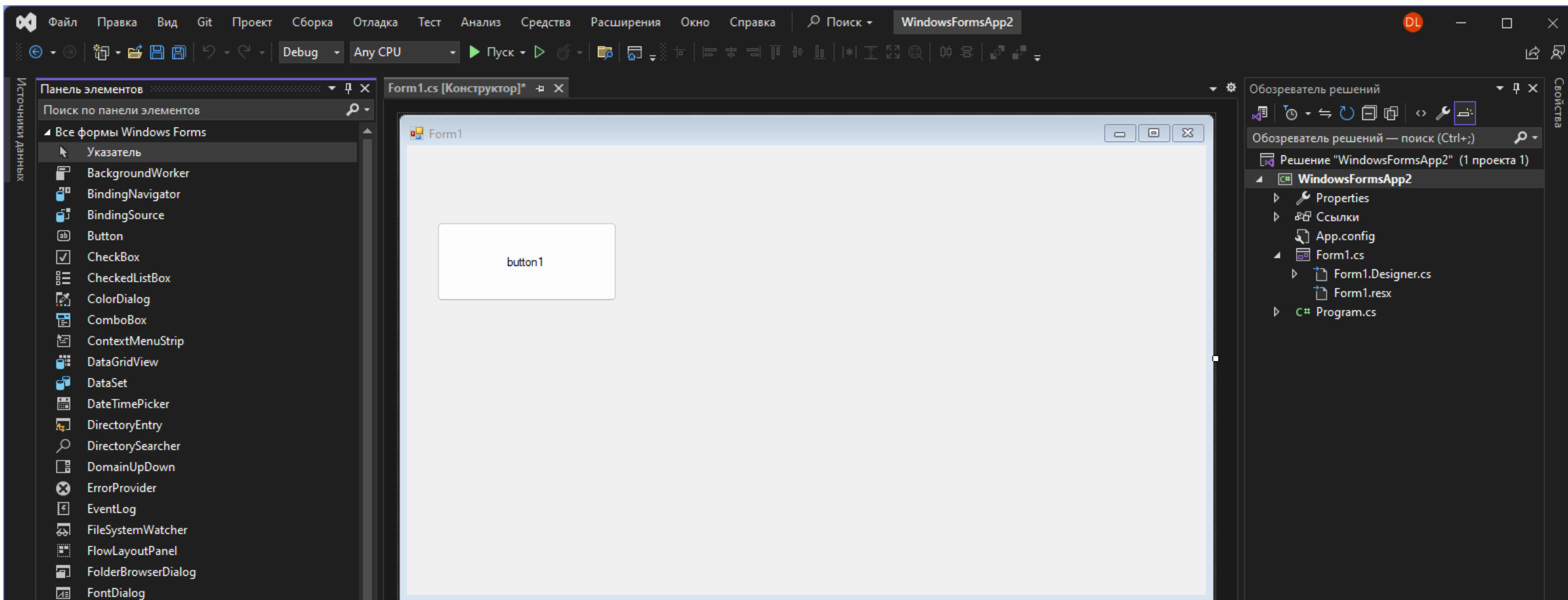


### Искусственный интеллект и ML

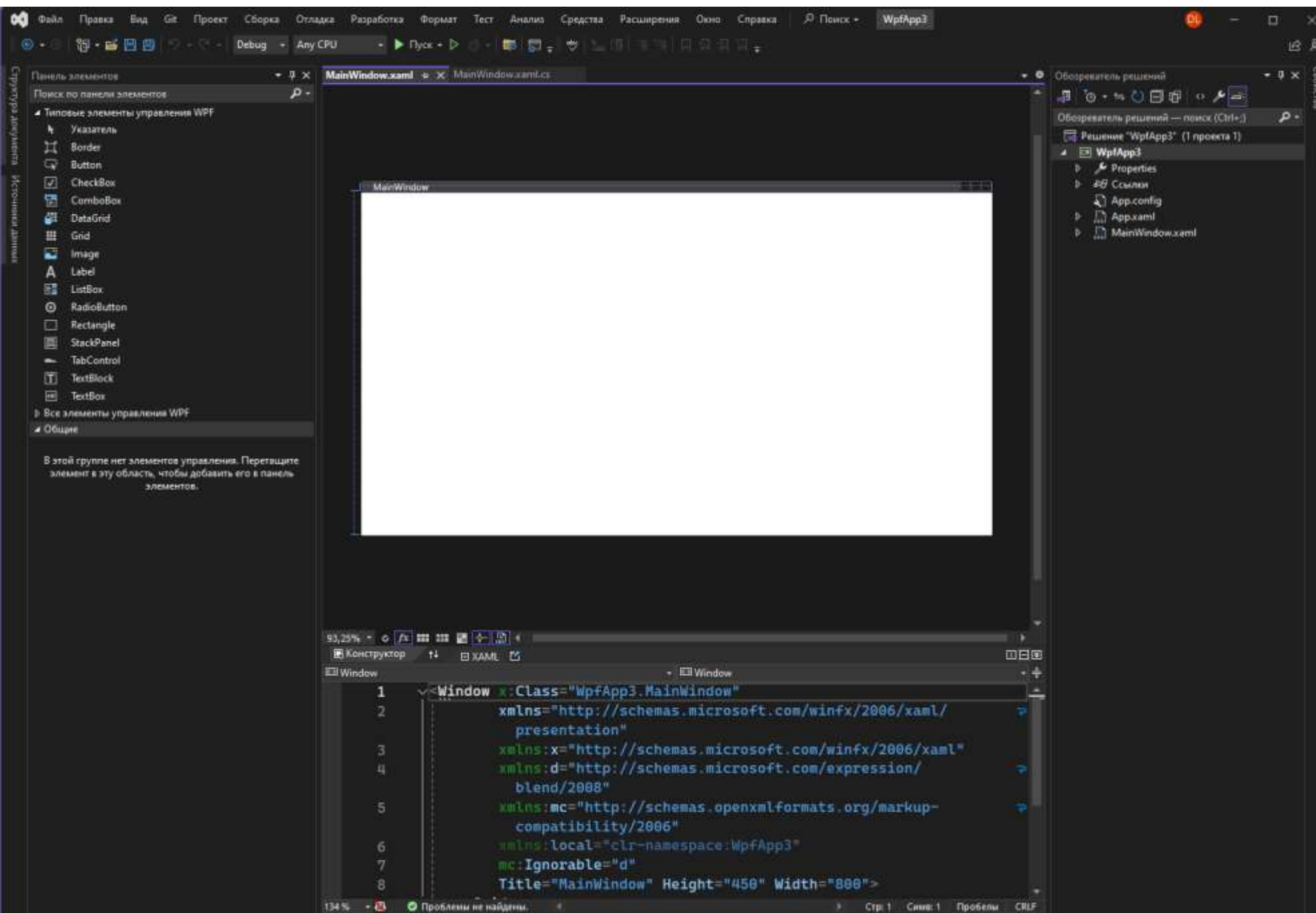
Создавайте интеллектуальные приложения с помощью C#, OpenAI и Azure.

Микросервисы → Разработка игр → Интернет вещей → Mobile → Desktop → Веб-интерфейсы → Серверные API-интерфейсы →

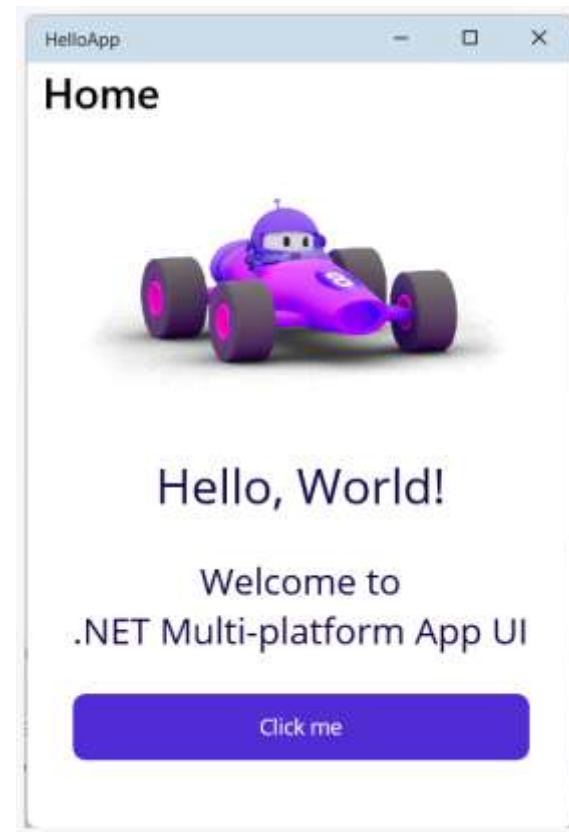
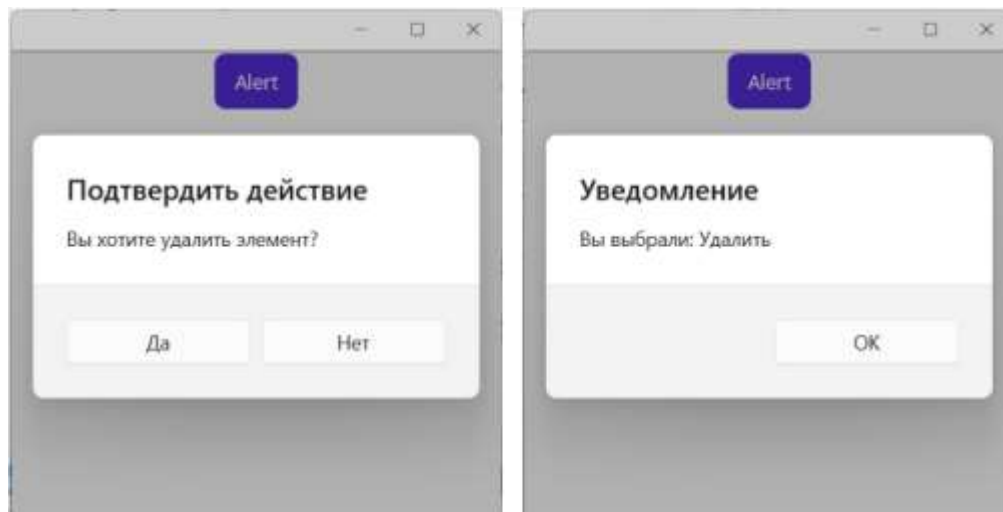
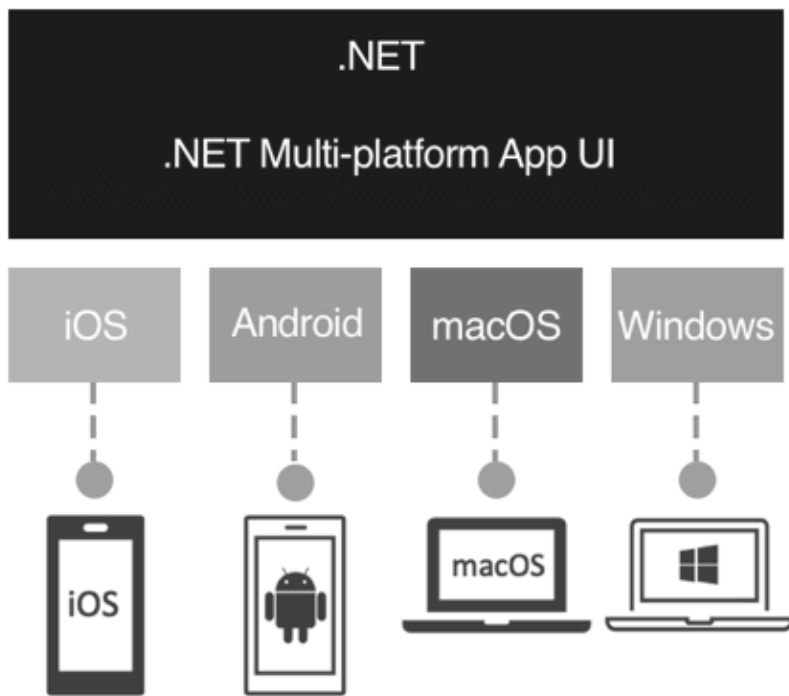
Данные →



# Windows Forms

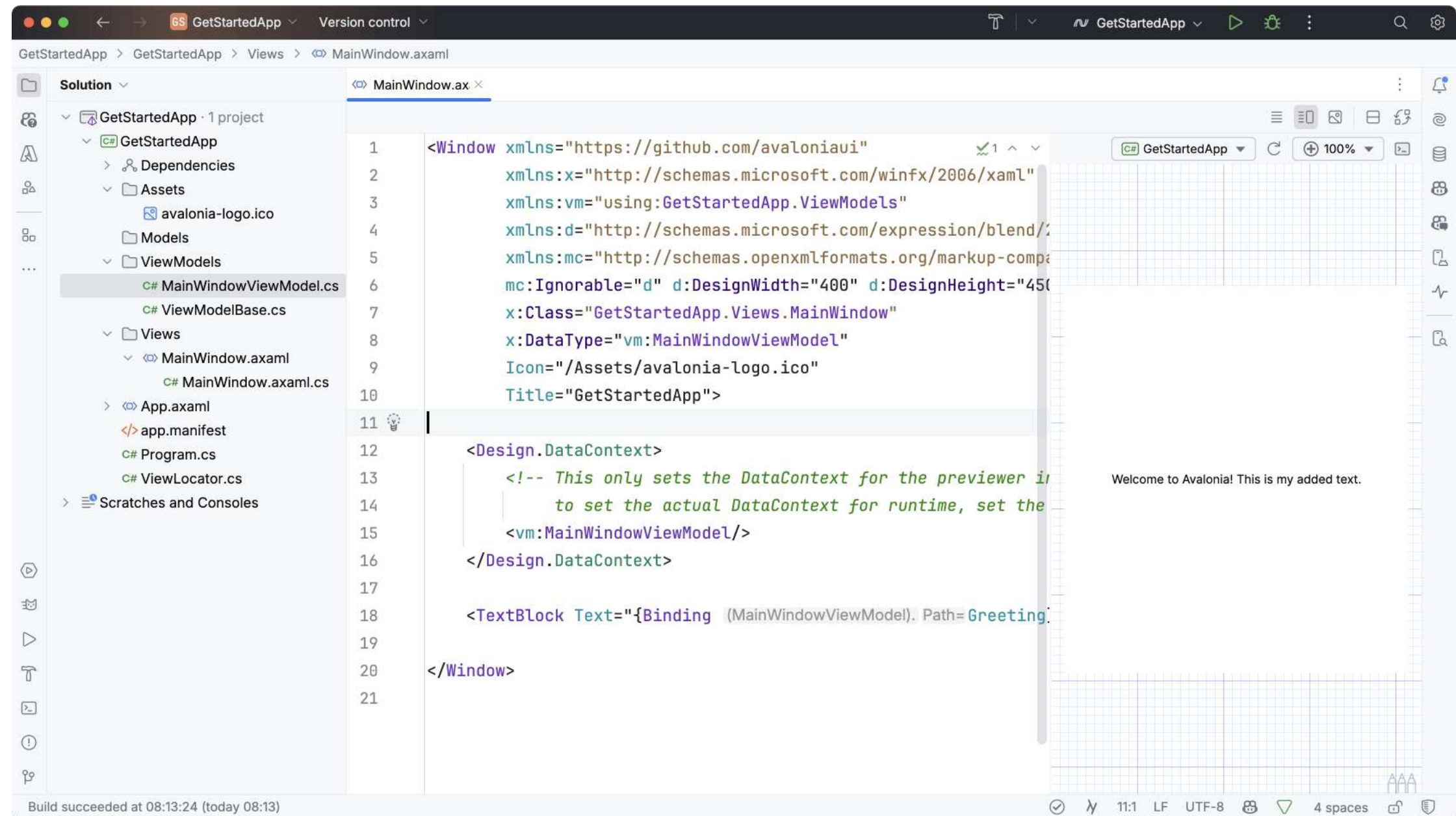


# Windows Presentation Foundation (WPF)



**.NET Multi-platform App UI** или сокращенно **MAUI** представляет кроссплатформенный фреймворк от компании **Microsoft** для создания нативных мобильных и десктопных приложений с использованием языка программирования **C#** и языка разметки **XAML**.

# Avalonia





# Asp.NET

Search...

Websites & Domains

Mail

Applications

Files

Databases

Statistics

Users

Account

Websites & Domains

ASP.NET Configuration for example.com

Restore Default Configuration

Restore the default ASP.NET settings.

Framework version

Version4.5.1

Connection string manager

Connection strings

LocalSqlServerdata source=.\SQLEXPRESS;Integrated Security=SSFRemove

Name / Connection Parameters

Add connection string

Custom error settings

Custom error mode

RemoteOnly

Custom errors

Status Code / Redirect URLRemove

Add custom error

Compilation and debugging

Default webpage language

vb

Switch on debugging

Globalization settings

Request encoding

utf-8

# Blazor

The image displays a Blazor application running in a web browser on the left and its source code in Visual Studio on the right.

**Blazor Application (Left):**

- Blazortron** header.
- Navigation menu: Home, Counter (selected), Fetch data.
- Counter** page content:
  - Current count: 0
  - Click me button

**Visual Studio (Right):**

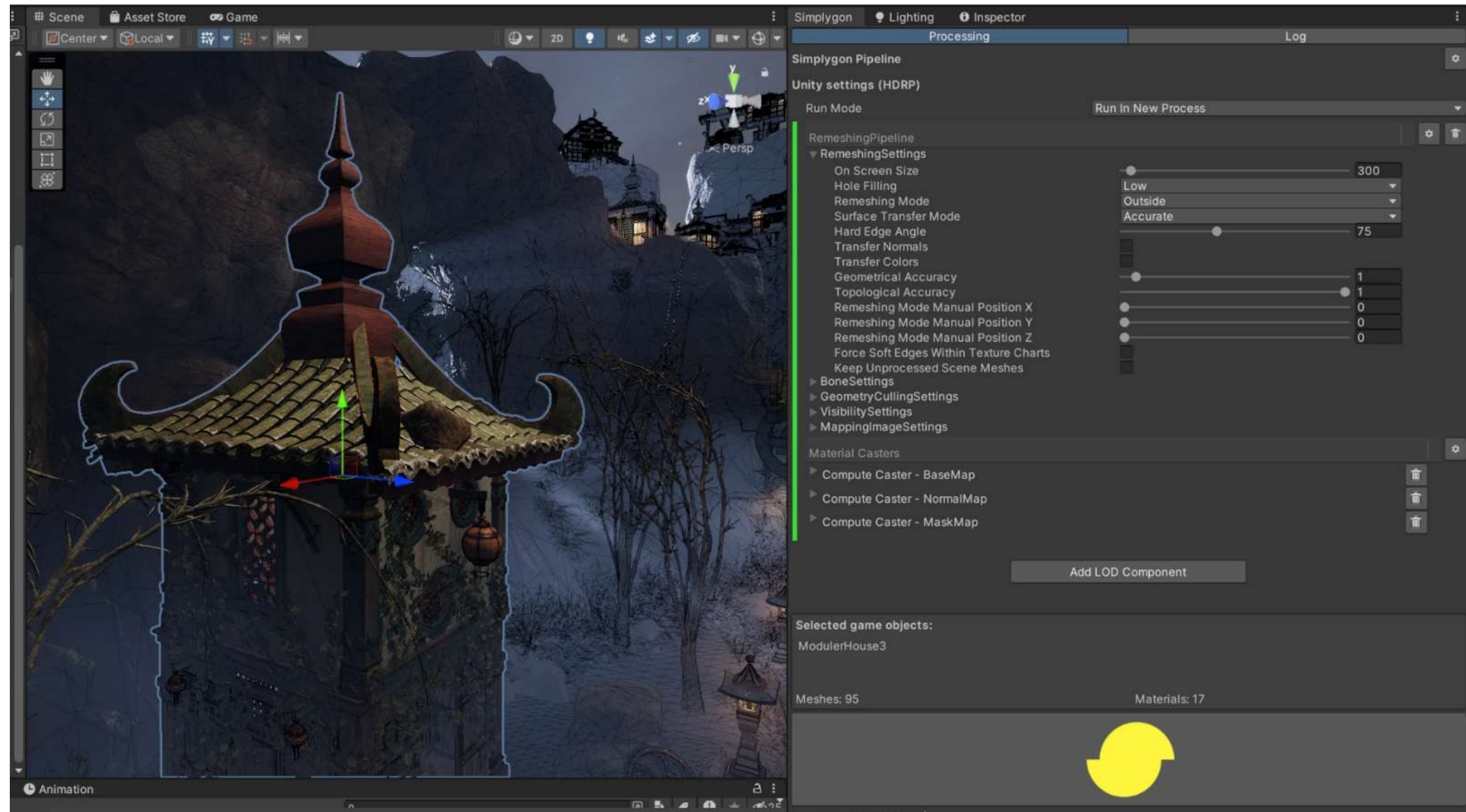
- Debugger window shows **Counter.IncrementCount() in** [External code: 43 frames].
- Source code for **Counter.razor** is visible, showing the `currentCount++` increment operation.

```
@page "/counter"

<h1>Counter</h1>
<p>Current count: @currentCount</p>
<button class="btn btn-primary" @onclick=
@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        currentCount++;
    }
}
```

# Unity



## Можно выделить следующие ее основные черты:

- **Поддержка нескольких языков.** Основой платформы является общезыковая среда исполнения **Common Language Runtime (CLR)**, благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#.

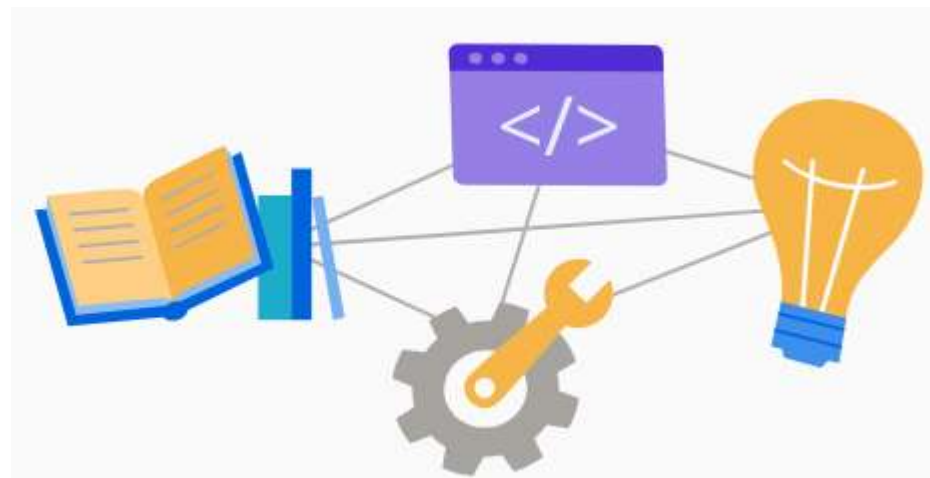


➤ **Кроссплатформенность.** .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент - .NET 8 поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

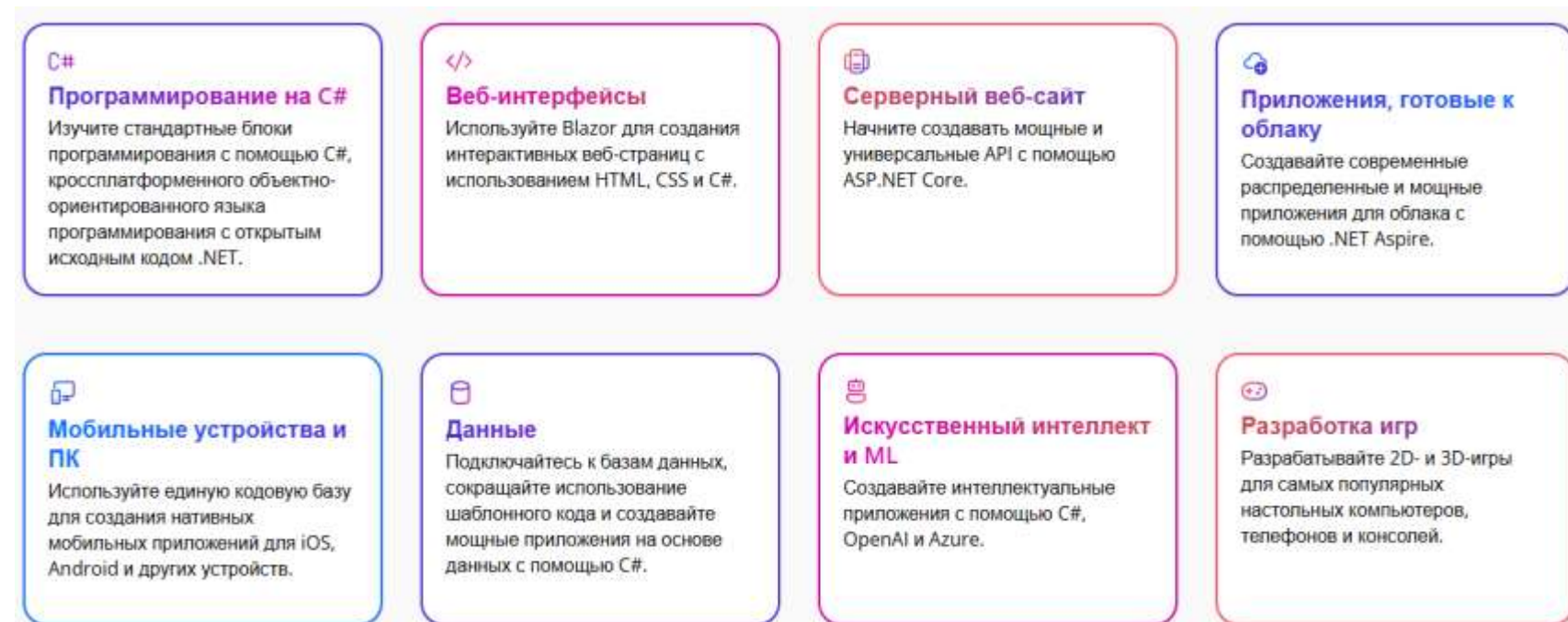




➤ **Мощная библиотека классов.** .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.



➤ **Разнообразие технологий.** Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных - ADO.NET и Entity Framework Core. Для построения графических приложений WPF, WinUI, Windows Forms. Для разработки кроссплатформенных мобильных и десктопных приложений - Xamarin/MAUI. Для создания веб-сайтов и веб-приложений - ASP.NET, Blazor

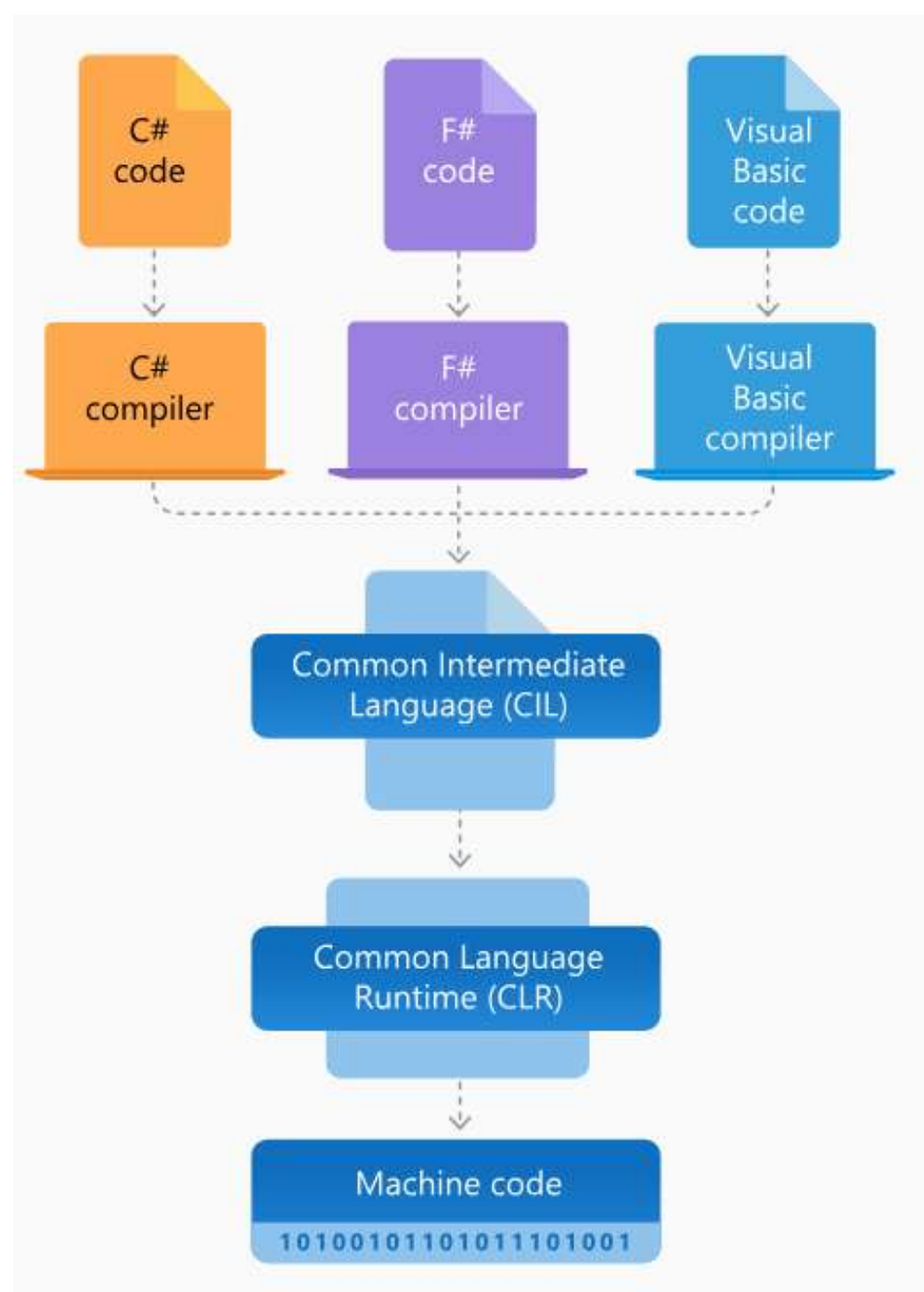




➤ **Производительность.** Согласно ряду тестов веб-приложения на .NET в ряде категорий сильно опережают веб-приложения, построенные с помощью других технологий. Приложения на .NET в принципе отличаются высокой производительностью.

# .NET Framework и .NET 9

Стоит отметить, что .NET долгое время развивался преимущественно как платформа для Windows под названием **.NET Framework**. В 2019 вышла последняя версия этой платформы - **.NET Framework 4.8**. Она больше не развивается

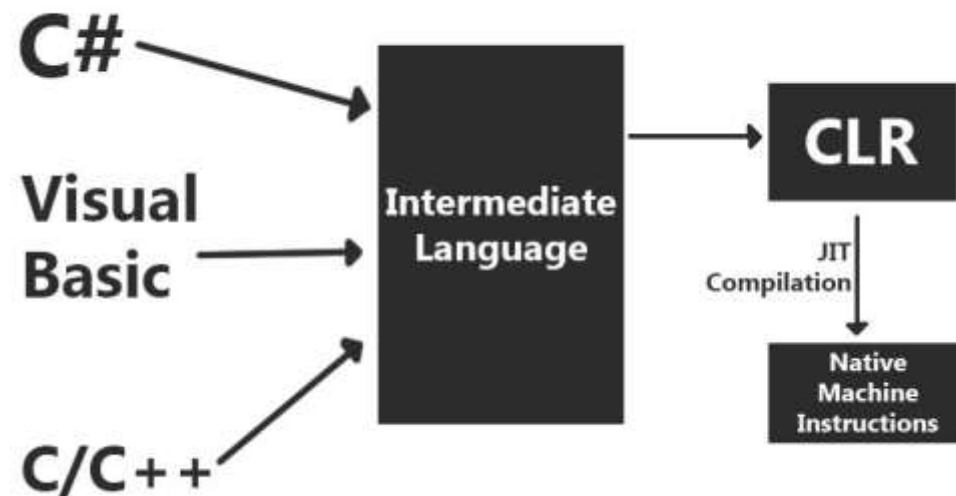


С 2014 Microsoft стал развивать альтернативную платформу - **.NET Core**, которая уже предназначалась для разных платформ и должна была вобрать в себя все возможности устаревшего **.NET Framework** и добавить новую функциональность. Затем Microsoft последовательно выпустил ряд версий этой платформы: **.NET Core 1**, **.NET Core 2**, **.NET Core 3**, **.NET 5**. И текущей версией является платформа **.NET 9**. Поэтому следует различать **.NET Framework**, который предназначен преимущественно для Windows, и кроссплатформенный **.NET 9**.

# Управляемый и неуправляемый код

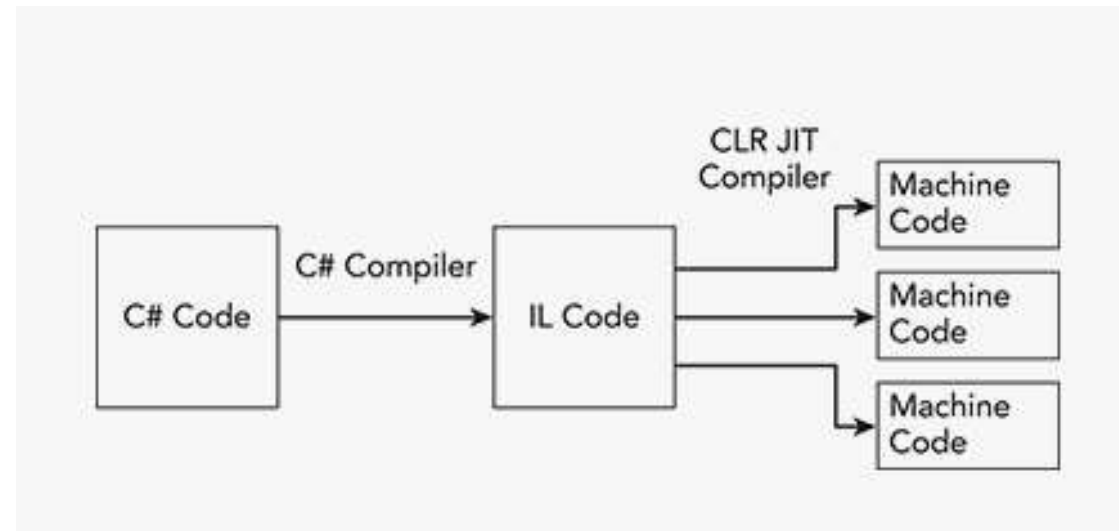
Нередко приложение, созданное на **C#**, называют управляемым кодом (**managed code**). Это значит, что данное приложение создано на основе платформы **.NET** и поэтому управляется общеязыковой средой **CLR**, которая загружает приложение и при необходимости очищает память. Но есть также приложения, например, созданные на языке **C++**, которые компилируются не в общий язык **CIL**, как **C#**, **VB.NET** или **F#**, а в обычный машинный код. В этом случае **.NET** не управляет приложением.

В то же время платформа **.NET** предоставляет возможности для взаимодействия с неуправляемым кодом.



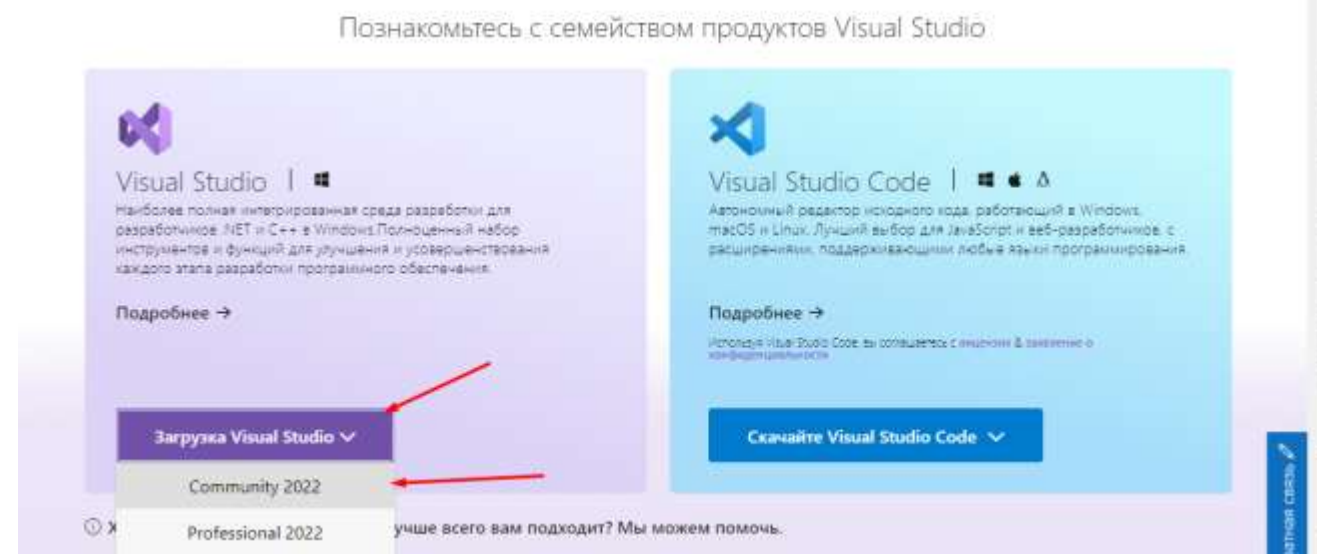
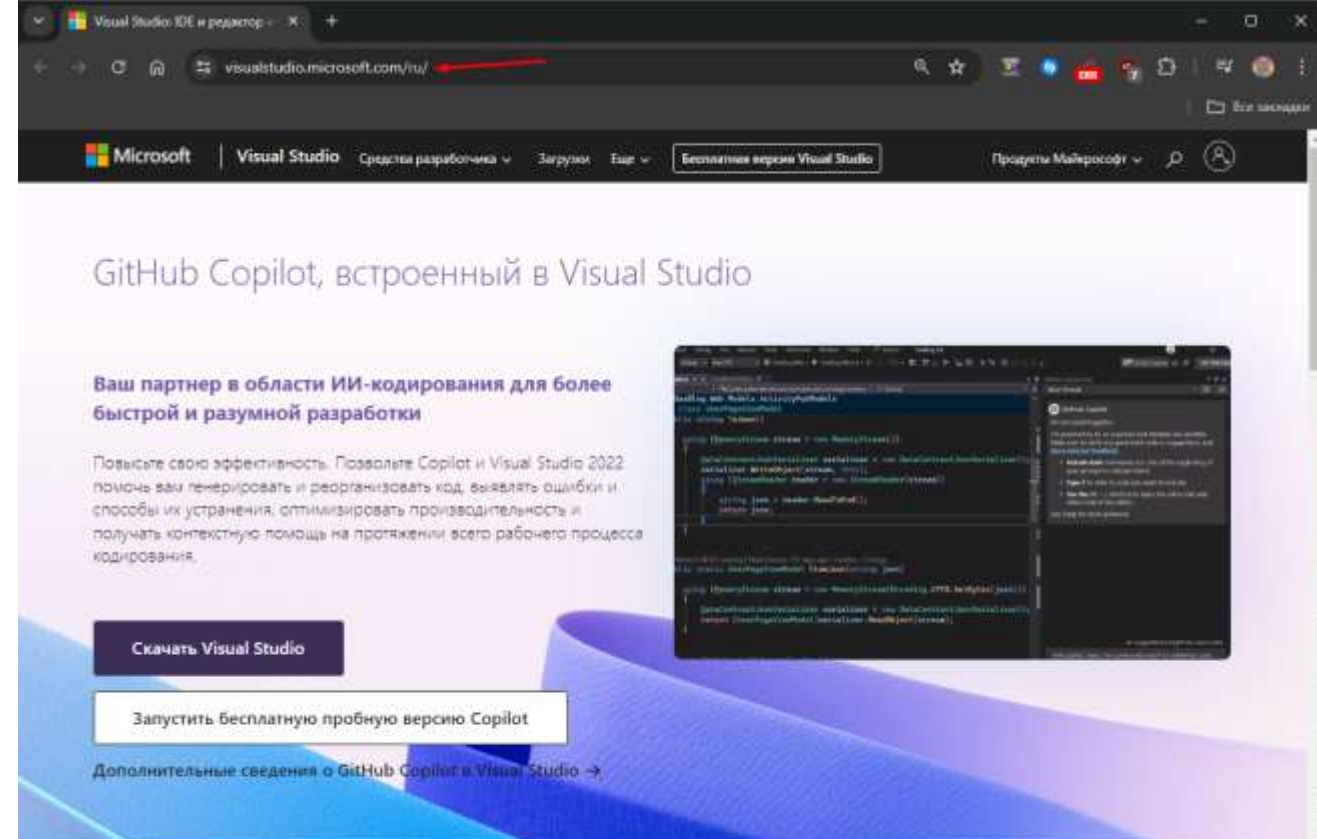
# JIT-компиляция

Код на **C#** компилируется в приложения или сборки с расширениями **.exe** или **.dll** на языке **CIL**. Далее при запуске на выполнение подобного приложения происходит **JIT-компиляция (Just-In-Time)** в машинный код, который затем выполняется. При этом, поскольку наше приложение может быть большим и содержать кучу инструкций, в текущий момент времени будет компилироваться лишь та часть приложения, к которой непосредственно идет обращение. Если мы обратимся к другой части кода, то она будет скомпилирована из **CIL** в машинный код. При том уже скомпилированная часть приложения сохраняется до завершения работы программы. В итоге это повышает производительность.







# **Среда разработки Visual Studio: установка и базовая конфигурация**

Бесплатная и полнофункциональная среда Visual Studio Community 2022 загружается с официального сайта:  
<https://visualstudio.microsoft.com/ru/>










## Веб-разработка и облако (4)

-  **ASP.NET и разработка веб-приложений**  
Создание веб-приложений с использованием ASP.NET Core, ASP.NET, HTML/JavaScript и контейнеров, включ...
-  **Разработка на Python**  
Редактирование, отладка, интерактивная разработка и система управления версиями для Python.
-  **Разработка для Azure**  
Пакеты SDK Azure, средства и проекты для разработки облачных приложений и создания ресурсов с исполь...
-  **Разработка Node.js**  
Создавайте масштабируемые сетевые приложения с помощью Node.js, асинхронной среды выполнения Ja...

## Классические и мобильные приложения (5)

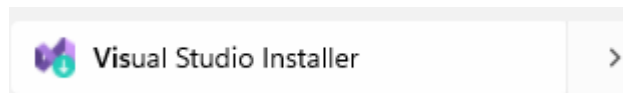
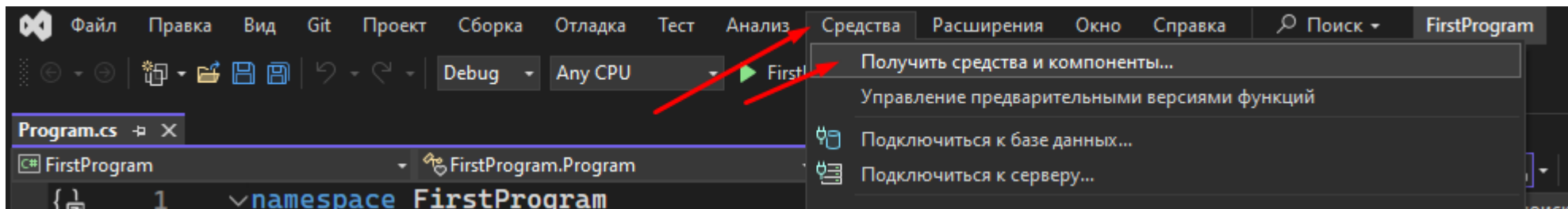
-  **Разработка с помощью .NET Multi-Platform App UI**  
Создавайте приложения Android, iOS, Windows и Mac из общей базы кода на языке C# с помощью .NET MA...
-  **Разработка классических приложений .NET**  
Создание приложений WPF, Windows Forms и консольных приложений с использованием C#, Visual...
-  **Разработка классических приложений на C++**  
Создавайте современные приложения C++ для Windows с использованием любых удобных инструме...
-  **Разработка приложений WinUI**  
Создавайте приложения для платформы Windows, используя WinUI с C# или, при необходимости, C++.
-  **Разработка мобильных приложений на языке C++**  
Создавайте кросс-платформенные приложения для iOS, Android или Windows на языке C++.

## Игры (2)

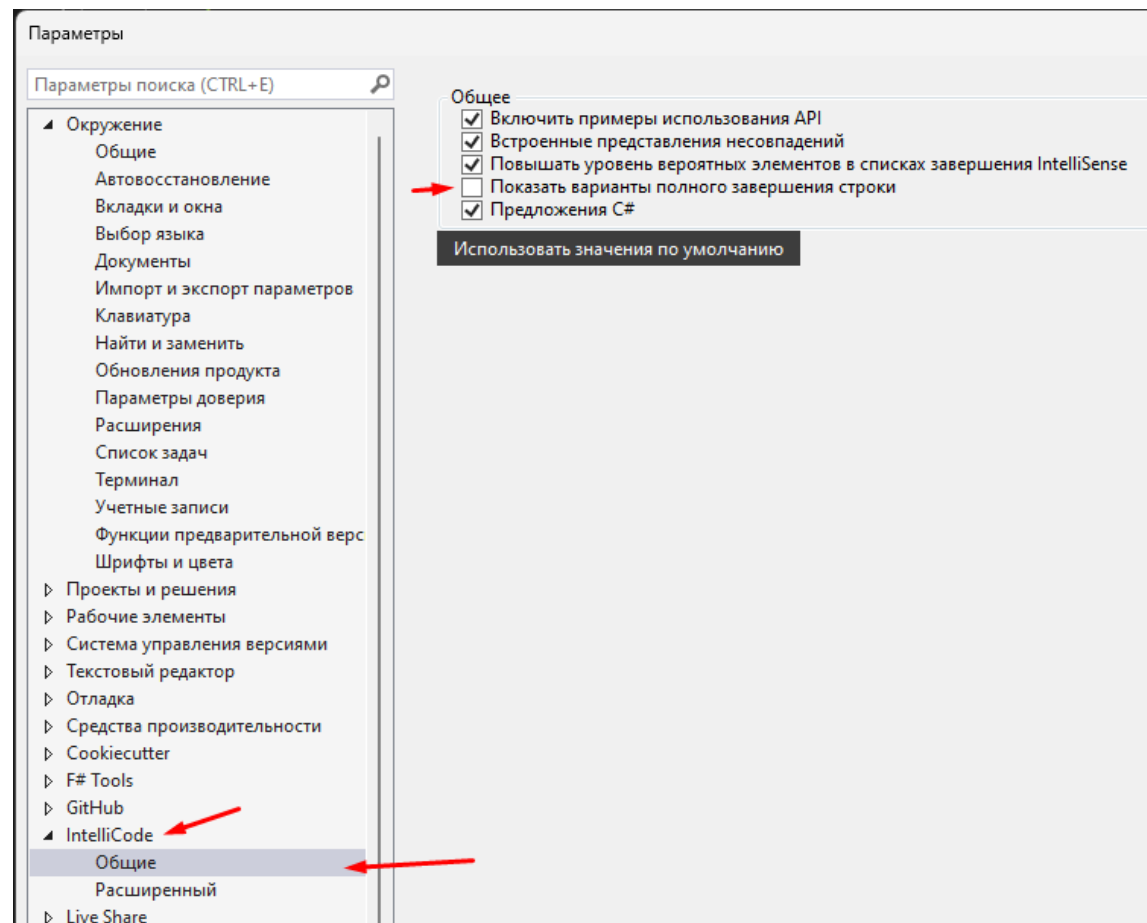
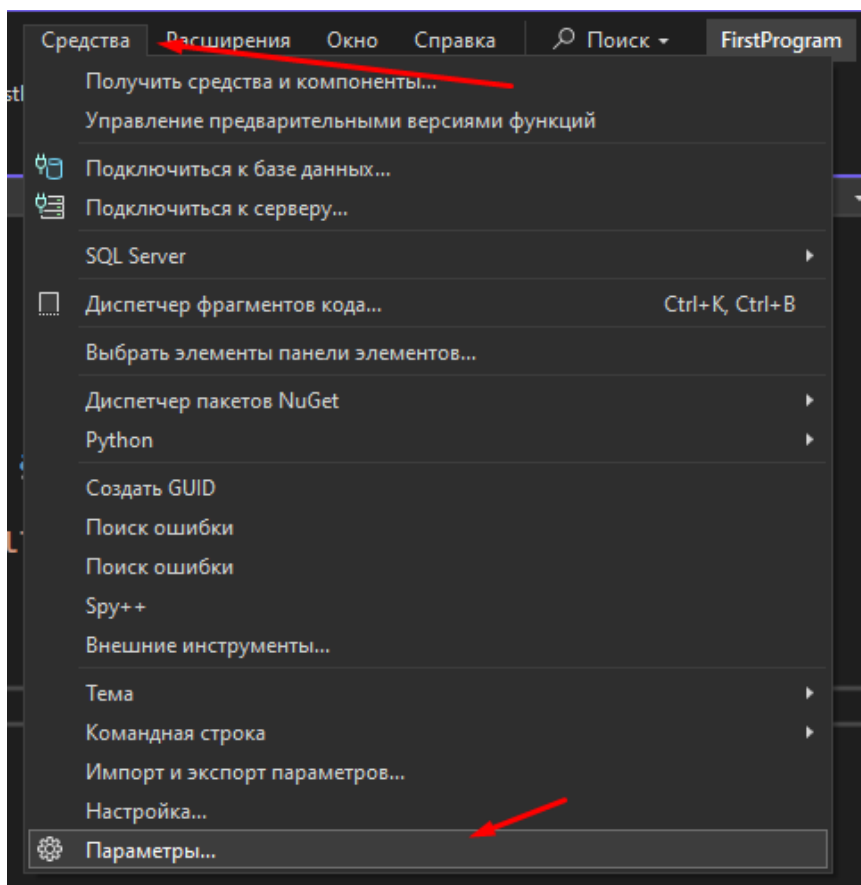
-  **Разработка игр с помощью Unity**  
В Unity, мощной кросс-платформенной среде разработки, можно создавать игры в формате 2D и 3D.
-  **Разработка игр на C++**  
Все возможности C++ для создания высококлассных игр на платформе DirectX, Unreal или Cocos2d.

После загрузки запустим программу установщика. В открывшемся окне нам будет предложено выбрать те компоненты, которые мы хотим установить вместе Visual Studio. Стоит отметить, что Visual Studio - очень функциональная среда разработки и позволяет разрабатывать приложения с помощью множества языков и платформ. В нашем случае нас будет интересовать прежде всего **C#** и **.NET**.

Если вы закроете окно, его можно будет открыть снова  
или через программу – **Средства – получить средства и компоненты...**  
или через **Visual Studio Installer**



Для лучшего изучения, я рекомендую перейти в **Средства – Параметры** и в меню **IntelliCode-Общие** снять галочку с пункта – **Показывать варианты полного завершения строки**




# Консольная программа на C# в Visual Studio

# Visual Studio 2022

## Открыть последние

### Сегодня

 ConsoleApp22.sln 27.07.2024 12:47  
C:\Users\Paltos\C#\ConsoleApp22


### Вчера


 My project.sln 26.07.2024 23:38  
C:\Users\Paltos\My project

 ConsoleApp21.sln 26.07.2024 23:35  
C:\Users\Paltos\C#\ConsoleApp21

 Hello.sln 26.07.2024 14:17  
C:\Users\Paltos\C#\Hello

### На этой неделе

 Pong.sln 25.07.2024 13:12  
C:\Users\Paltos\Documents\Unity\Pong

 Snake\_3.sln 24.07.2024 15:45  
C:\Users\Paltos\Documents\Unity\Snake\_3

 ConsoleApp20.sln 24.07.2024 9:06  
C:\Users\Paltos\C#\ConsoleApp20

## Начало работы



### Клонирование репозитория

Получить код из интернет-репозитория, например, GitHub или Azure DevOps



### Открыть проект или решение

Открыть локальный проект Visual Studio или SLN-файл



### Открыть локальную папку

Перейти и изменить код в любой папке

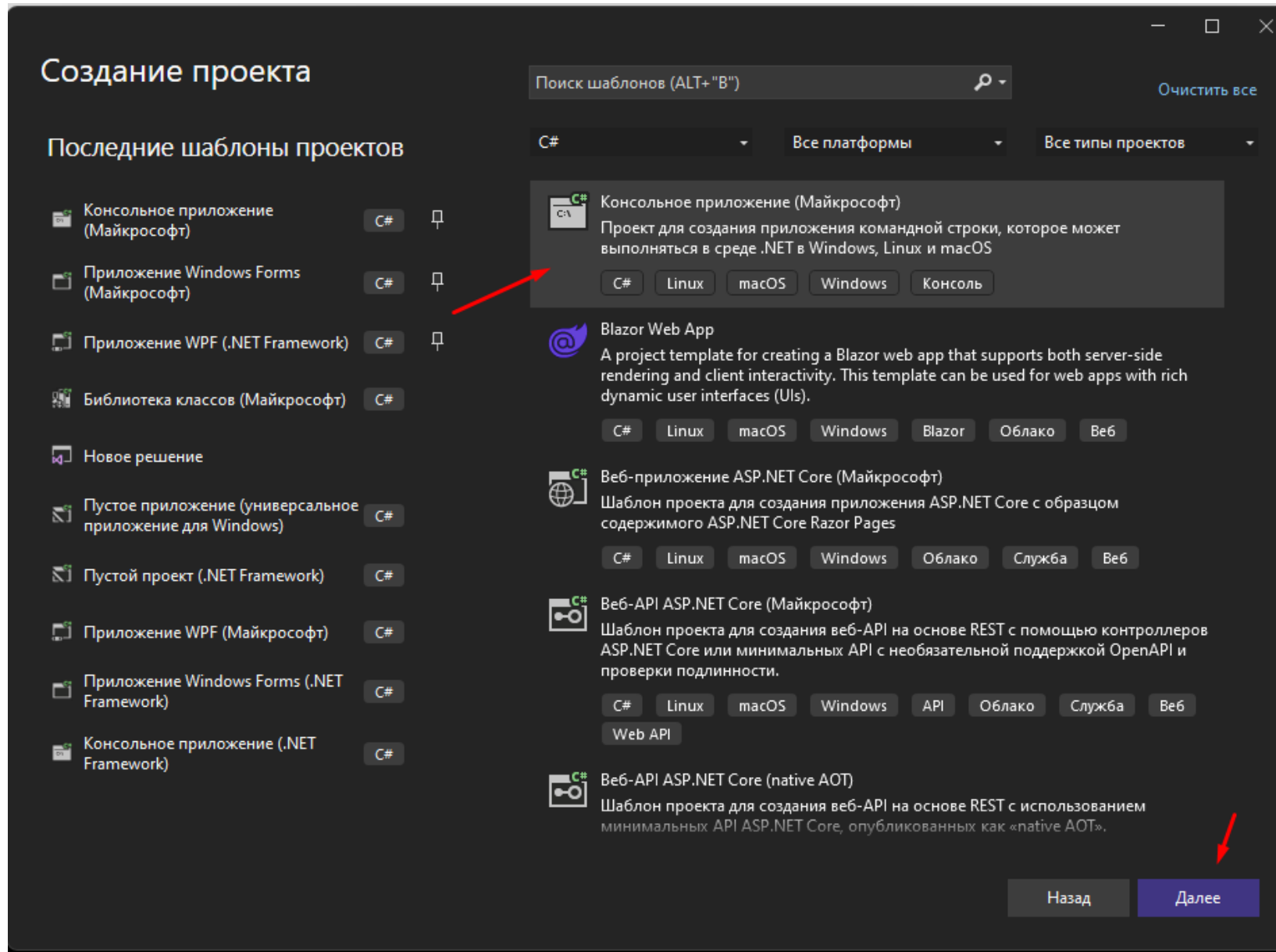


### Создание проекта

Выберите шаблон проекта с формированием шаблонов кода, чтобы начать работу

[Продолжить без кода →](#)

Открываем  
**Visual Studio 2022.**  
На стартовом экране  
выбираем  
**Создание проекта**



На следующем окне  
в качестве типа  
проекта выберем  
**Консольное  
приложение  
(Майкрософт)**



## Настроить новый проект

Консольное приложение (Майкрософт)

C#

Linux

macOS

Windows

Консоль

Имя проекта

FirstProgram

Расположение

C:\Users\Paltos\C#\

Имя решения ⓘ

FirstProgram

☐ Поместить решение и проект в одном каталоге

Проект будет создан в "C:\Users\Paltos\C#\FirstProgram\FirstProgram\"

Назад

Далее

Далее на следующем этапе нам будет предложено указать **имя проекта** и **каталог**, где будет располагаться проект.

## Дополнительные сведения

Консольное приложение (Майкрософт)

C#

Linux

macOS

Windows

Консоль

Платформа ⓘ

.NET 8.0 (долгосрочная поддержка)

☒ Не использовать операторы верхнего уровня ⓘ

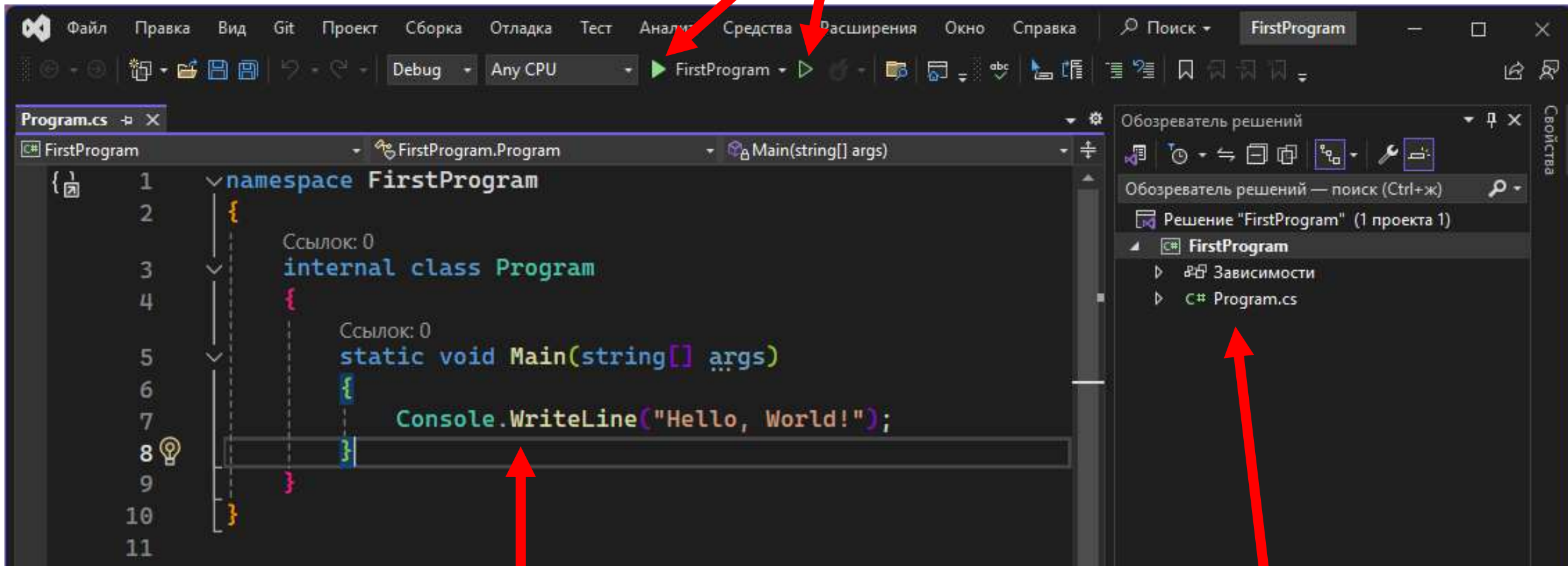
☐ Включить публикацию native AOT ⓘ

Назад

Создать

На следующем окне **Visual Studio** предложит нам выбрать версию **.NET**, которая будет использоваться для проекта. Выберем последнюю на данный момент версию - **.NET 8.0**. Также поставьте галочку **Не использовать операторы верхнего уровня**.

Запуск проекта

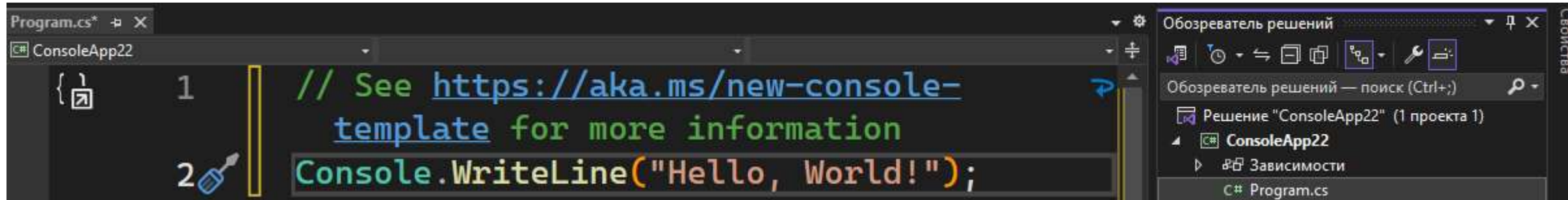


Исходный код на языке C#

Структура проекта

# Структура программы

Весь код программы на языке **C#** помещается в файлы с расширением **.cs**. По умолчанию в проекте, который создается в **Visual Studio** (а также при использовании **.NET CLI**) уже есть один файл с кодом **C#** - файл **Program.cs** со следующим содержимым:

A screenshot of the Visual Studio IDE. The main editor window shows a file named 'Program.cs' with the following code:

```
1 // See https://aka.ms/new-console-  
2 template for more information  
   Console.WriteLine("Hello, World!");
```

The code is color-coded: comments are green, the URL is blue, and the code itself is black. The right sidebar shows the 'Обозреватель решений' (Solution Explorer) with the project 'ConsoleApp22' and the file 'C# Program.cs' selected.

Именно код файла **Program.cs** выполняется по умолчанию, если мы запустим проект на выполнение. Но при необходимости мы также можем добавлять другие файлы с кодом **C#**

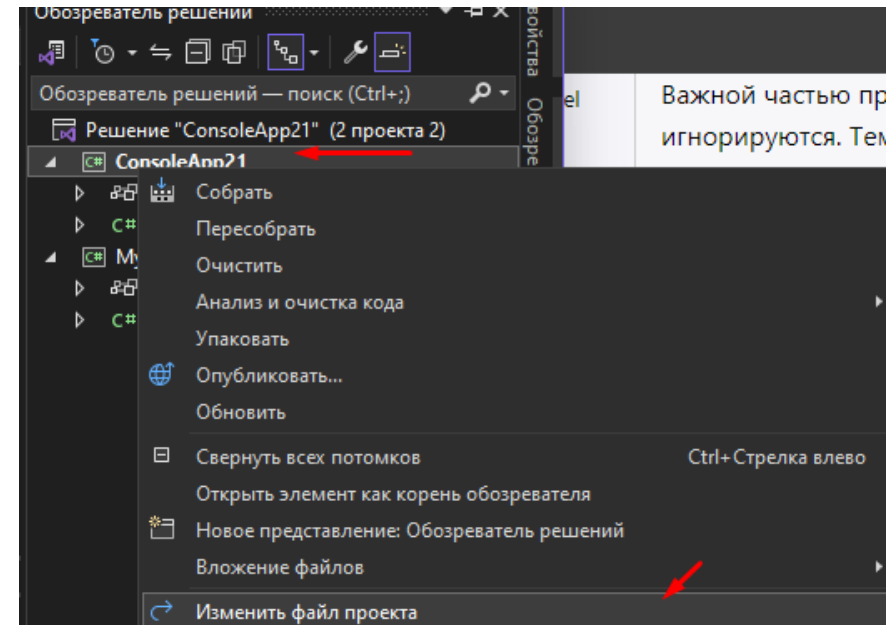
# Инструкции

Базовым строительным блоком программы являются инструкции (**statement**). Инструкция представляет некоторое действие, например, арифметическую операцию, вызов метода, объявление переменной и присвоение ей значения. В конце каждой инструкции в **C#** ставится точка с запятой (;). Данный знак указывает компилятору на конец инструкции.

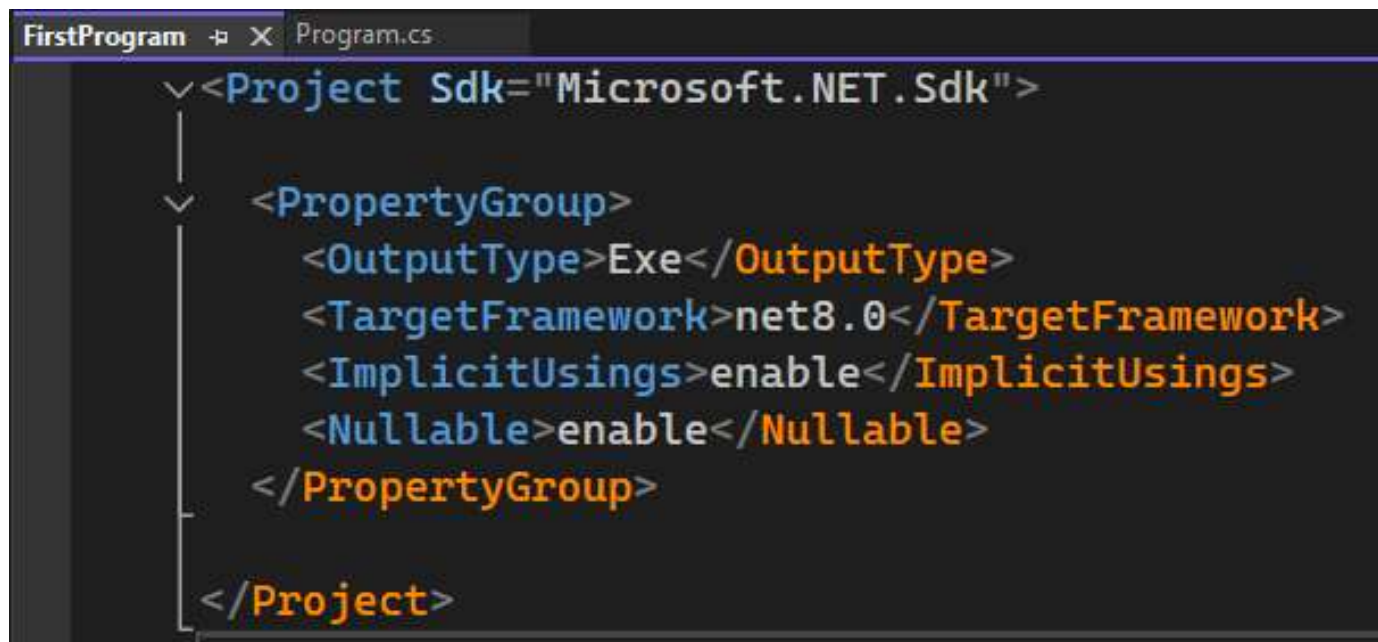


# Файл проекта

В каждом проекте **C#** есть файл, который отвечает за общую конфигурацию проекта. По умолчанию этот файл называется **Название\_проекта.csproj**. Итак, откроем данный файл. Для этого либо двойным кликом левой кнопкой мыши нажмем на название проекта, либо нажмем на название проекта правой кнопкой мыши и в появившемся меню выберем пункт **Изменить файл проекта**



После этого Visual Studio откроет нам файл проекта, который будет выглядеть следующим образом:



```
FirstProgram x Program.cs
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>
</Project>
```

Этот файл в виде кода **xml** определяет конфигурацию проекта и он может содержать множество элементов.