

В.В. Подбельский, О.В. Максименкова

# Программирование на C#

## Семинар 1

**Знакомство с IDE**

**Консольный Ввод-Вывод**

# Где Искать Материалы Курса?

Материалы семинаров находятся в SmartLMS

Материалы дублируются на Google-диск

[https://drive.google.com/drive/folders/1fFRagLpDDj\\_jGj82riAXpUEpe3cqTv0A?usp=sharing](https://drive.google.com/drive/folders/1fFRagLpDDj_jGj82riAXpUEpe3cqTv0A?usp=sharing)

# Работа в Компьютерном Классе

- Рекомендуется создать папку на диске **D:\** и назвать её своей фамилией (используйте только латинские символы, иначе возможны проблемы при работе внутри IDE);
- В папке следует сохранять результаты работы на практическом занятии – при желании Вы можете выгрузить материал на своё хранилище данных.

**Важно:** после выключения компьютера все папки с рабочего стола и диска C:\ автоматически удаляются – после этого восстановление не является возможным!

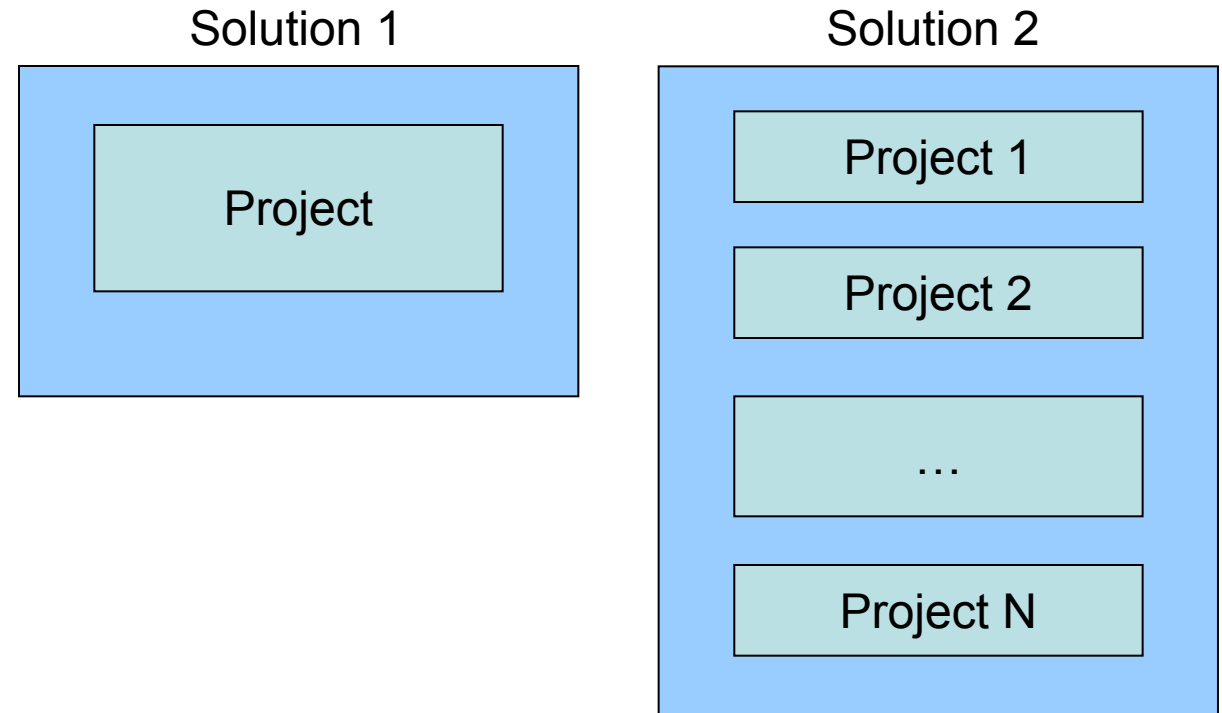


# Проекты и Решения

- Для организации C# программ вводятся две основные логические единицы – решения и проекты;
- Решение является контейнером для набора проектов, каждый из которых может определять запускаемое приложение, библиотеку классов, набор тестов и т. д.
- Пример:

Решение [*Solution*]

- Проект1 [*Project1*]
- Проект2 [*Project2*]
- ...
- ПроектN [*ProjectN*]

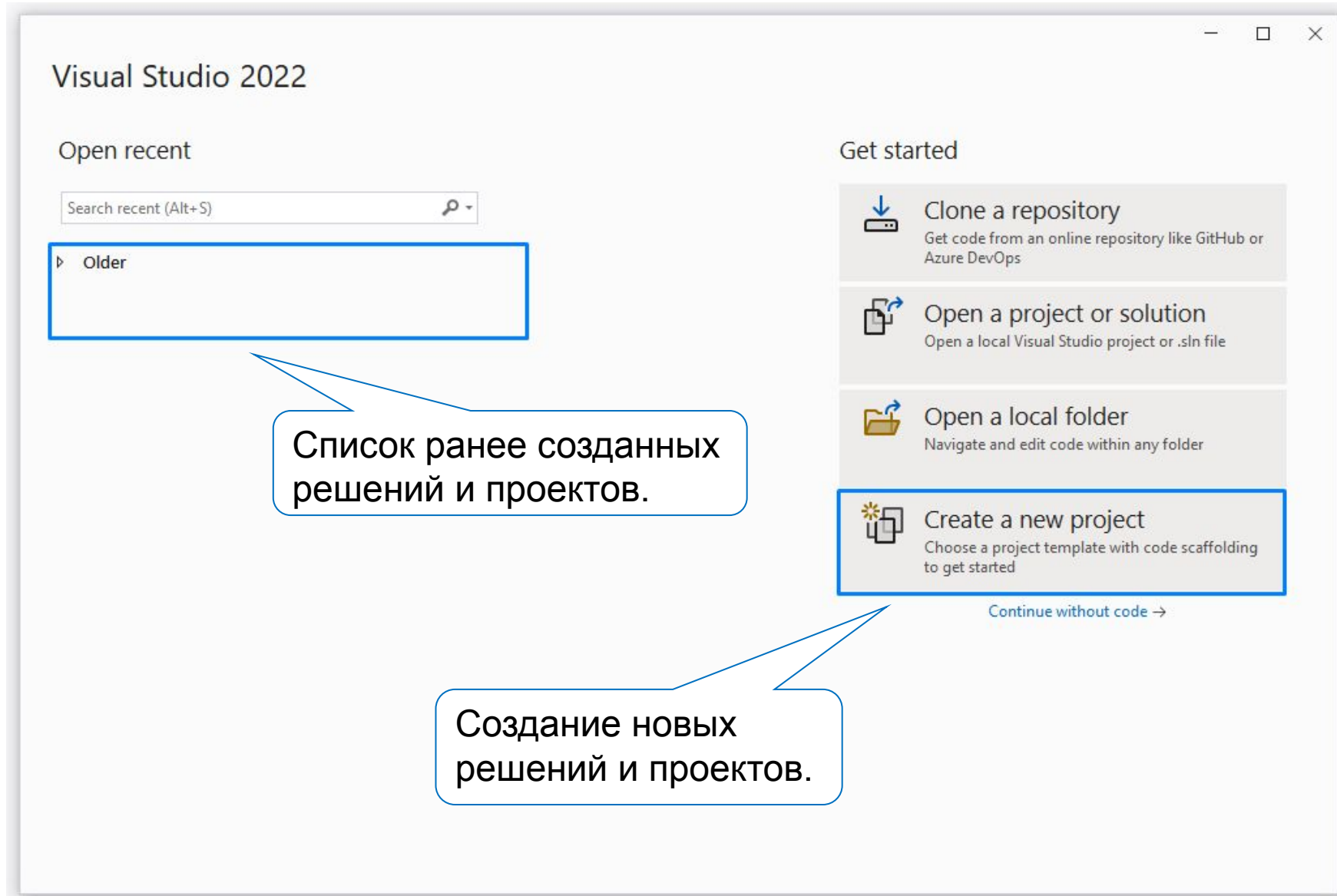


# Последовательность Действий

- Запустить Microsoft Visual Studio 2022 (далее – VS 2022);
- Ознакомиться с инструкцией по созданию первого решения можно по ссылке: <https://visualstudio.microsoft.com/vs/getting-started/>



# Стартовый Экран VS 2022



# Выбор Шаблона Решения

Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Search for templates (Alt+S)

All languages All platforms All project types

**Azure Functions** New  
A template to create an Azure Function project.  
C# Azure Cloud

**Console App**  
A project for creating a command-line application that can run on .NET on Windows, Linux and macOS  
C# Linux macOS Windows Console

**ASP.NET Core Web App**  
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.  
C# Linux macOS Windows Cloud Service Web

**Blazor Server App**  
A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).  
C# Linux macOS Windows Blazor Cloud Web

**ASP.NET Core Web API**  
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Back Next

**Важно:** необходимо выбрать именно тип **“Console App”**. Если создать **“Class Library”**, то проект просто не запустится.

# Название Проекта и Решения

The image shows a screenshot of the 'Configure your new project' dialog box in Visual Studio. The dialog is titled 'Configure your new project' and has a window title bar with standard Windows window controls. Below the title, there are several tabs: 'Console App', 'C#', 'Linux', 'macOS', 'Windows', and 'Console'. The 'Console App' tab is selected. The dialog contains the following fields and options:

- Project name:** A text box containing 'Sagalov\_Daniil\_First\_Console\_App'. A callout bubble points to this field with the text 'Название проекта.'
- Location:** A dropdown menu showing 'D:\Coding\_Projects\Visual\_Studio' and a browse button (three dots). A callout bubble points to the browse button with the text 'расположение решения на диске.'
- Solution name:** A text box containing 'Sagalov\_Daniil\_First\_Console\_App'. A callout bubble points to this field with the text 'Название решения.'
- Place solution and project in the same directory:** An unchecked checkbox.
- Navigation:** 'Back' and 'Next' buttons at the bottom right.



# Выбор Версии и Создание Решения

Additional information

Console App C# Linux macOS Windows Console

Framework ⓘ

.NET 6.0 (Long-term support)

Do not use top-level statements ⓘ

Back Create

Нужно ли использовать упрощённый шаблон основной программы\*

Версия .NET, неявно определяет версию C#.

\*Различия между стандартным и упрощённым шаблонами будут рассматриваться далее

# Первая Программа в Редакторе VS 2022

The screenshot displays the Visual Studio 2022 interface. The main editor window shows the file `Program.cs` with the following code:

```
1 // See https://aka.ms/new-console-template for more information
2 Console.WriteLine("Hello, World!");
3
```

Two callout boxes provide additional information:

- Left Callout:** Стандартный шаблон программы, печатающей строку "Hello, World!" на консоль. **Помните:** данный шаблон содержит много «компиляторной магии», которую мы будем разбирать.
- Right Callout:** Обзорщик решений – в нём можно смотреть структуру проектов и решений, работать с необходимыми файлами.

The Solution Explorer on the right shows the project structure for 'Sagalov\_Daniil\_First\_Console\_App', including a 'Dependencies' folder and the 'Program.cs' file. The bottom status bar shows '128%' zoom, 'No issues found', and 'Ln: 3 Ch: 1 SPC CRLF'.

# «Классическая» Первая Программа в VS 2022

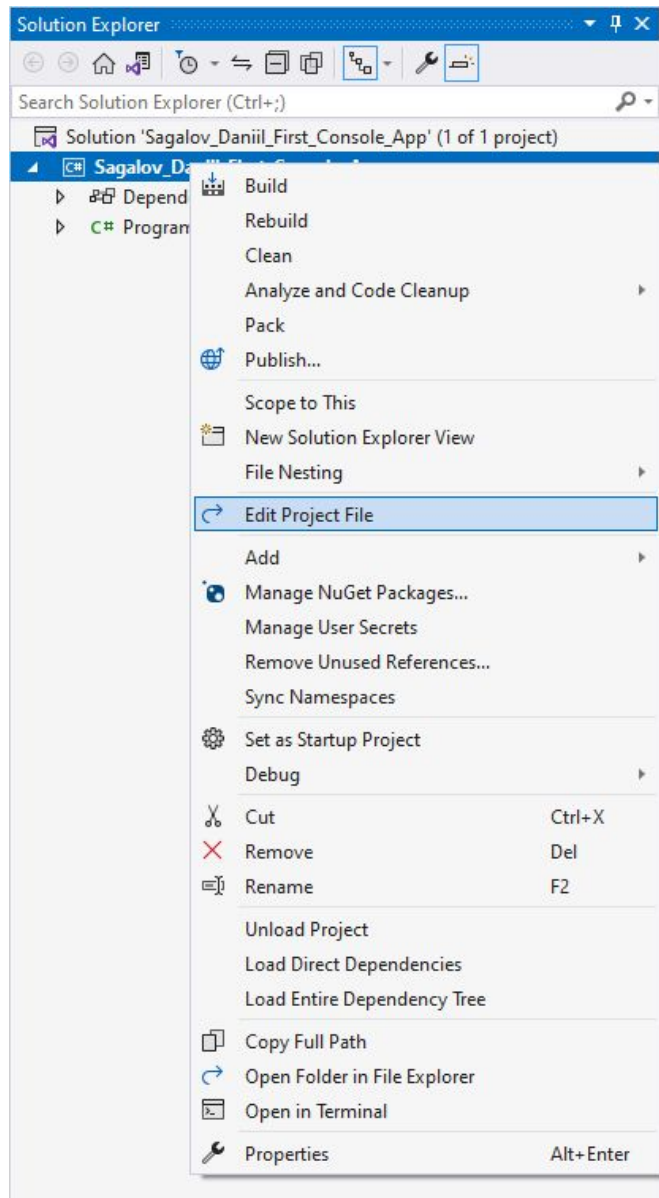
Список неявно импортированных пространств имён – позволяет использовать обращение к Console сокращённо (без явного написания System.Console каждый раз).

```
Program.cs [X]
[C#] Sagalov_Daniil_First_C... Sagalov_Daniil_First_Console_App.Program Main(string[] args)
1 namespace Sagalov_Daniil_First_Console_App
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    }
}
```

**class Program** – объявление типа. Помните: в C# все методы пишутся внутри классов. В прошлом примере данный класс дописан неявно компилятором.

**Метод Main** – точка входа в программу. В прошлом примере был неявно дописан компилятором за нас.

# Файл с Настройками Проекта в VS 2022



В данном файле можно увидеть тип проекта (например, исполняемый exe-файл), отключить неявные using.

```
Sagalov_Danii...ole_App.csproj  Program.cs
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>
</Project>
```

# Запуск Проекта в VS 2022

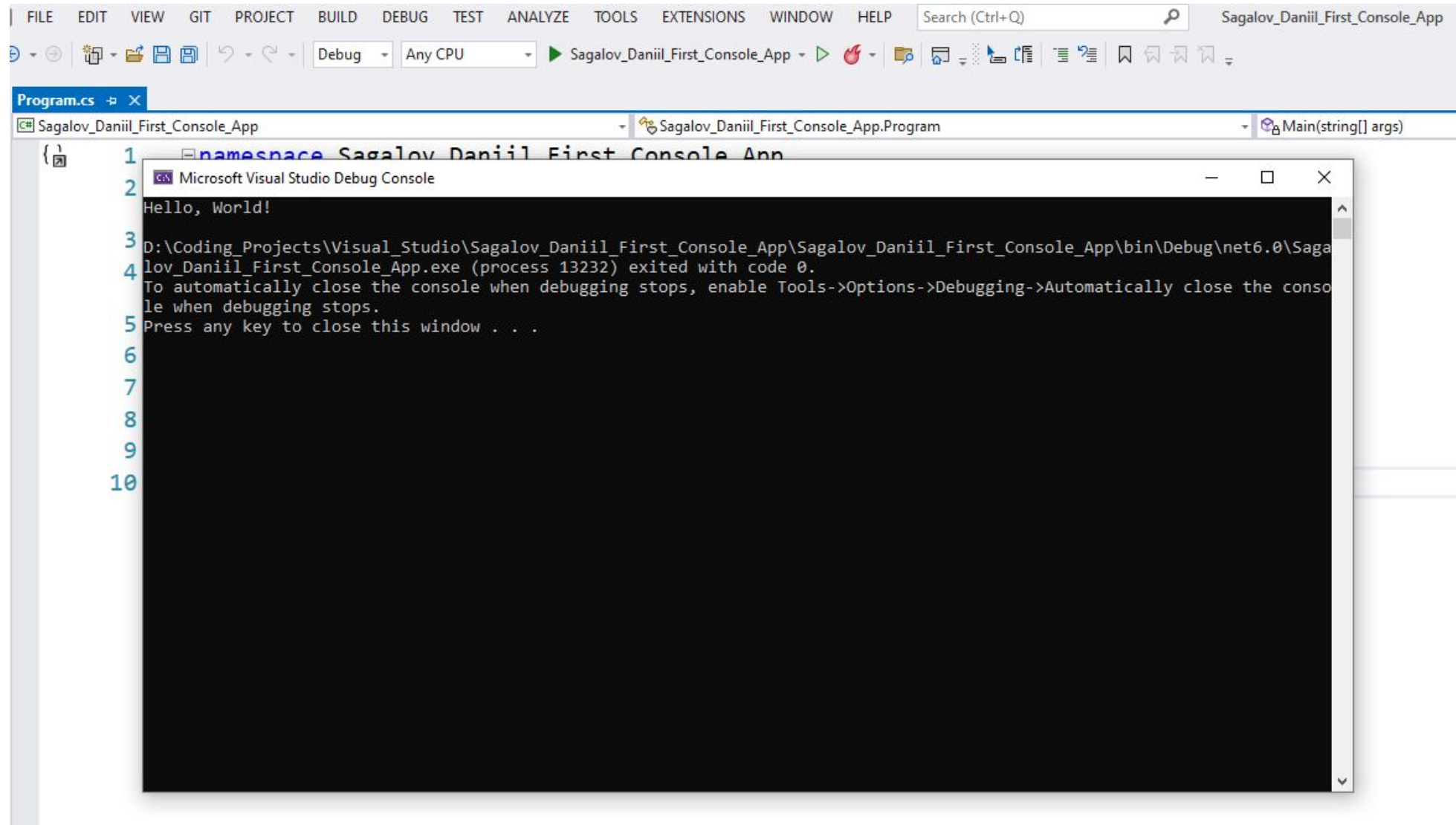
- Комбинация клавиш Ctrl + F5 (Windows);
- Вкладка с именем активного проекта → ЛКМ;
- Частично прозрачная стрелка (запуск без отладки).



```
1 namespace Sagalov_Daniil_First_Console_App
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    }
```

# Результат Запуска Программы в VS 2022

VS 2022 запускает программу в отдельном консольном окне.



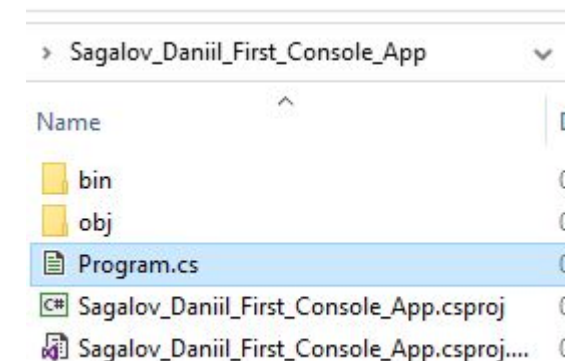
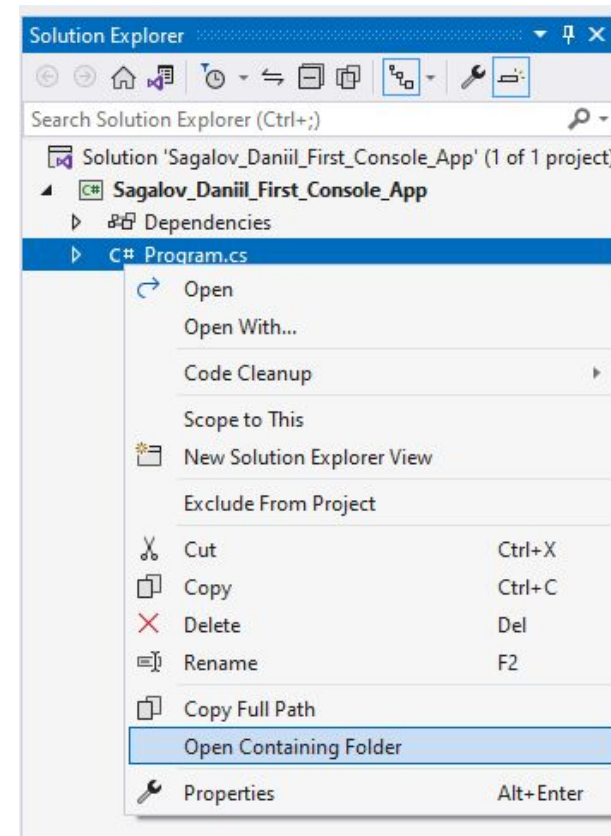
The screenshot displays the Visual Studio 2022 interface during a debug session. The main window shows the source code for `Program.cs` in the `namespace Sagalov Daniil First Console App`. A `Microsoft Visual Studio Debug Console` window is overlaid on top, showing the following output:

```
1 namespace Sagalov Daniil First Console App
2
3 Hello, World!
4 D:\Coding_Projects\Visual_Studio\Sagalov_Daniil_First_Console_App\Sagalov_Daniil_First_Console_App\bin\Debug\net6.0\Saga
5 lov_Daniil_First_Console_App.exe (process 13232) exited with code 0.
6 To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
7 le when debugging stops.
8 Press any key to close this window . . .
9
10
```

# Запуск Программы вне VS 2022

Простой способ запустить программу вне VS 2022 – ПКМ по файлу с исходным кодом → с помощью **“Open containing folder”** перейти в папку, содержащую файл с исходным кодом, затем спуститься в **bin** → **Профиль сборки (по умолчанию Debug)** → **Версия .NET** → **ЛКМ по .exe файлу**.

**Важно:** программа мгновенно завершится, т. к. не ожидает никаких действий со стороны пользователя!



# Модификация Программы в Vs 2022

```
namespace Sagalov_Daniil_First_Console_App
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
            Console.WriteLine("Press any key to exit...");
            Console.ReadKey();
        }
    }
}
```

Теперь программа ожидает нажатия любой клавиши, поэтому не закроется автоматически.

Попробуйте запустить данную программу вне VS 2022 – теперь она не закроется мгновенно.

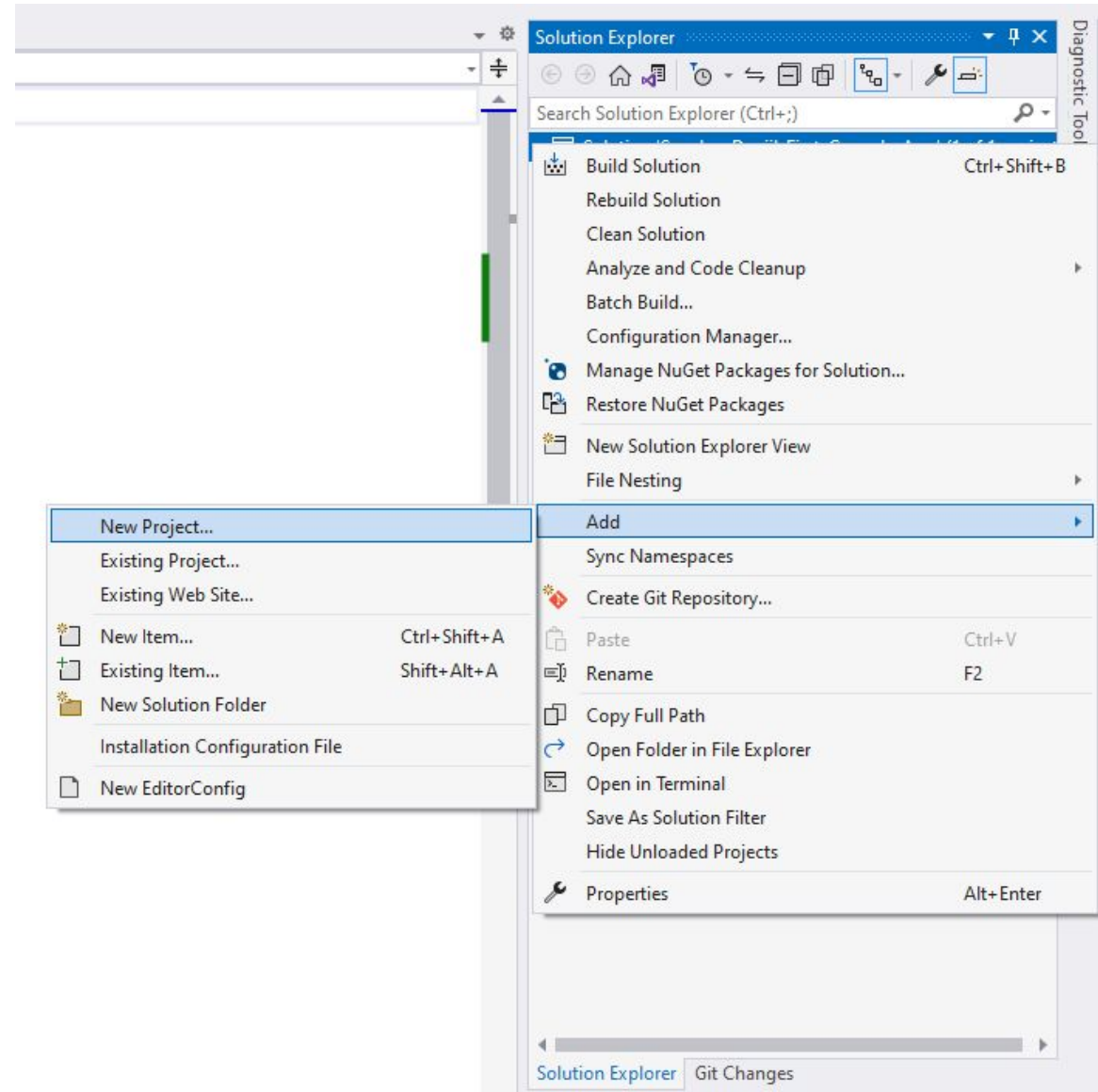


# Создание Второго Проекта в VS 2022

Удобно иметь несколько запускаемых приложений в одном решении.

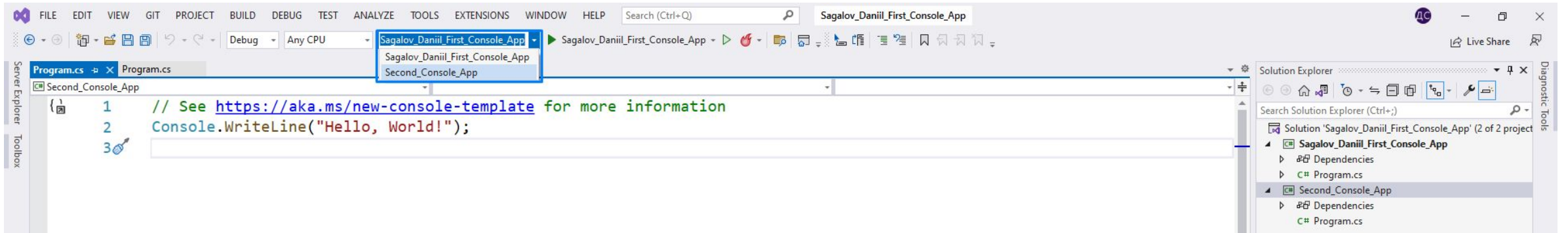
Для этого:

**ПКМ по названию решения (не проекта!)  
→ Add → New Project.**



# Выбор Запускаемого Проекта в VS 2022

VS 2022 позволяет выбрать запускаемый проект с помощью вкладки выбора запускаемого проекта, расположенной левее кнопки запуска программы:



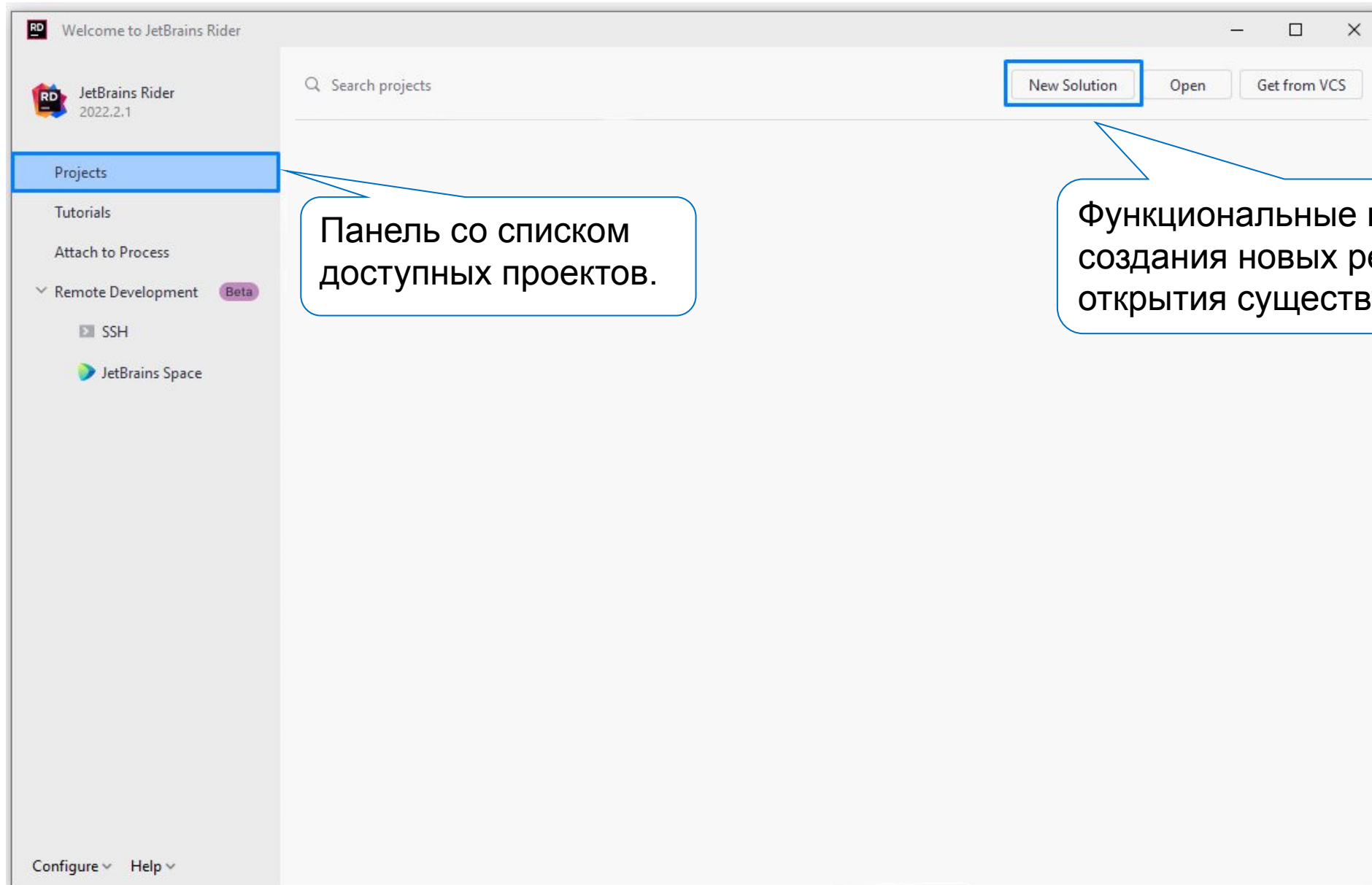
**На заметку:** Visual Studio выделяет текущий выбранный для запуска проект жирным шрифтом в обозревателе решений.

# Последовательность Действий

- Запустить JetBrains Rider;
- Ознакомиться с инструкцией по созданию первого решения можно по ссылке:  
[https://www.jetbrains.com/help/rider/Creating\\_and\\_Opening\\_Projects\\_and\\_Solutions.html](https://www.jetbrains.com/help/rider/Creating_and_Opening_Projects_and_Solutions.html)



# Стартовый Экран Rider



# Создание Решения в Rider

RD New Solution

.NET / .NET Core > Console Application

Template author: Microsoft

Solution name: Sagalov\_Daniil\_First\_Console\_App

Project name: Sagalov\_Daniil\_First\_Console\_App

Solution directory: D:\Coding\_Projects\Rider\_Sharp

Put solution and project in the same directory

Create Git repository

Language: C#

Framework: net6.0

Docker Support: Disabled

Resulting project structure Project template info

```
\Rider_Sharp\Sagalov_Daniil_First_Console_App\Sagalov_Daniil_First_Console_App.sln
\Rider_Sharp\Sagalov_Daniil_First_Console_App\Sagalov_Daniil_First_Console_App\<project files>
```

Callouts:

- Название решения.
- Название проекта.
- расположение решения на диске.
- Версия платформы.

**Важно:** необходимо выбрать именно тип “**Console Application**”. Если создать “Class Library”, то проект просто не запустится.

Create Cancel

# Первая Программа в Редакторе Rider

Обозреватель решений – в нём можно смотреть структуру проектов и решений, работать с необходимыми файлами.

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

Стандартный шаблон программы, печатающей строку "Hello, World!" на консоль.

**Помните:** данный шаблон содержит много «компиляторной магии», которую мы будем разбирать.

Sagalov\_Daniil\_First\_Console\_App

Version Control | Search everywhere | Sagalov\_Daniil\_First\_Console\_App | Run | Debug

Solution Explorer: Sagalov\_Daniil\_First\_Console\_App · 1 project

- Sagalov\_Daniil\_First\_Console\_App
  - Dependencies
  - C# Program.cs
  - Scratches and Consoles

Bottom status bar: Sagalov\_Daniil\_First\_Console\_App | <top-level-entry-point> | 3:36 CRLF UTF-8 4 spaces

# «Классическая» Первая Программа в Rider

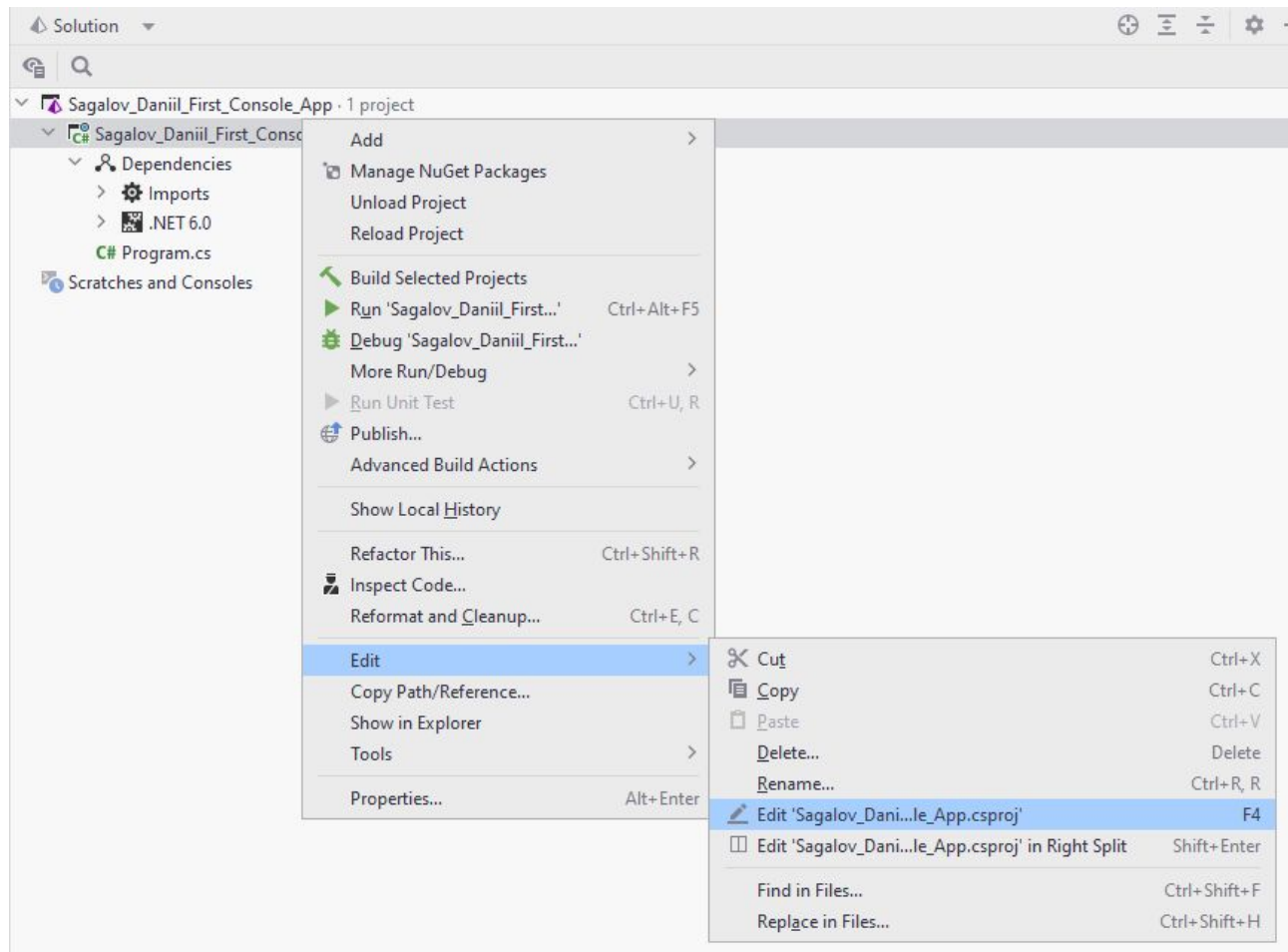
```
C# Program.cs x
1  using System;
2
3  namespace Sagalov_Daniil_First_Console_App
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Hello, World!");
10         }
11     }
12 }
```

**using System;** – позволяет использовать обращение к Console сокращённо (без явного написания System.Console каждый раз).

**class Program** – объявление типа. Помните: в C# все методы пишутся внутри классов. В прошлом примере данный класс дописан неявно компилятором.

**Метод Main** – точка входа в программу. В прошлом примере был неявно дописан компилятором за нас.

# Файл с Настройками Проекта в Rider



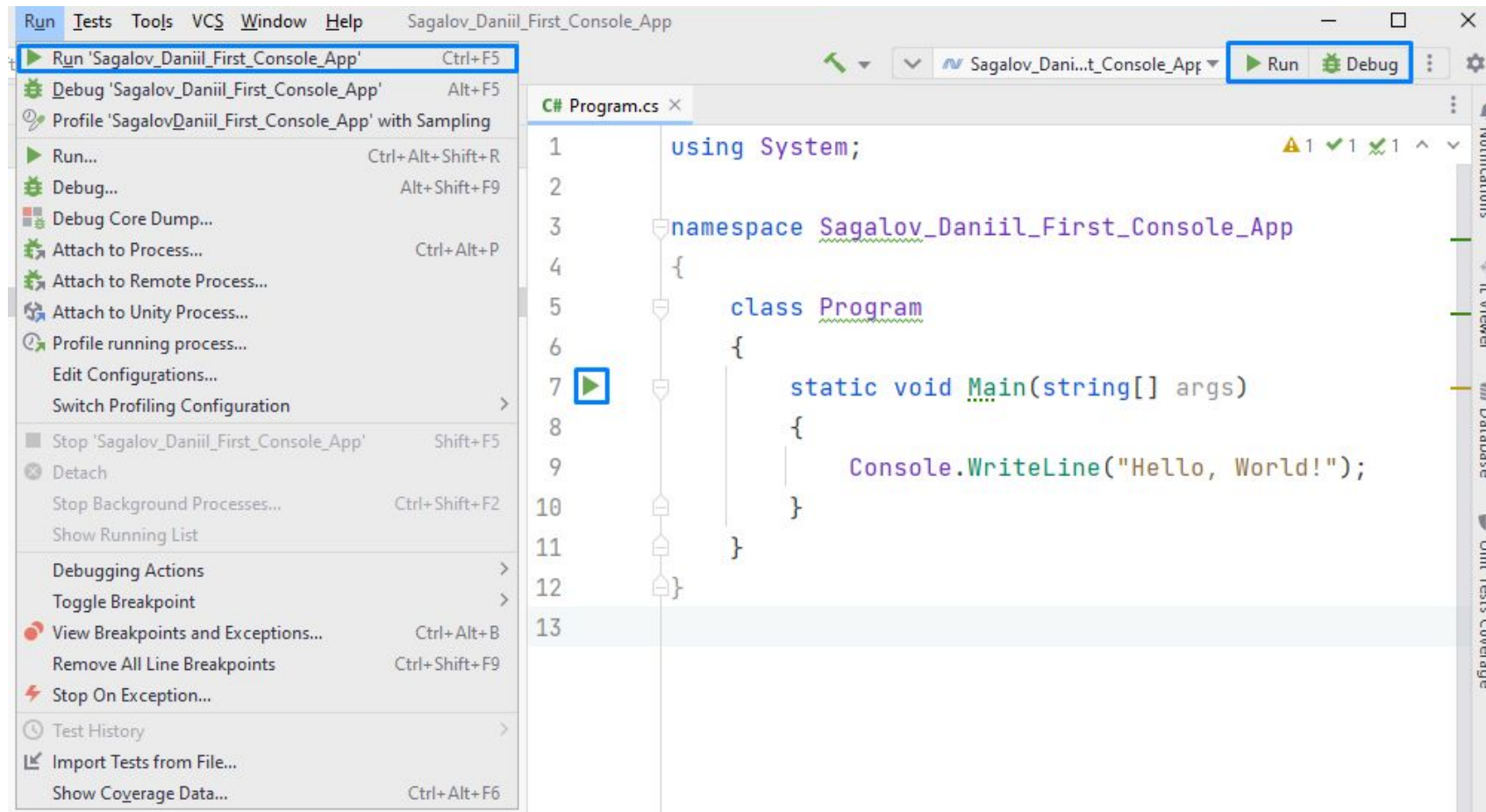
В данном файле можно увидеть тип проекта (например, исполняемый exe-файл), отключить неявные using.

```
C# Sagalov_Daniil_First_Console_App.csproj x
1  <Project Sdk="Microsoft.NET.Sdk">
2
3      <PropertyGroup>
4          <OutputType>Exe</OutputType>
5          <TargetFramework>net6.0</TargetFramework>
6          <ImplicitUsings>enable</ImplicitUsings>
7          <Nullable>enable</Nullable>
8      </PropertyGroup>
9
10 </Project>
11 |
```



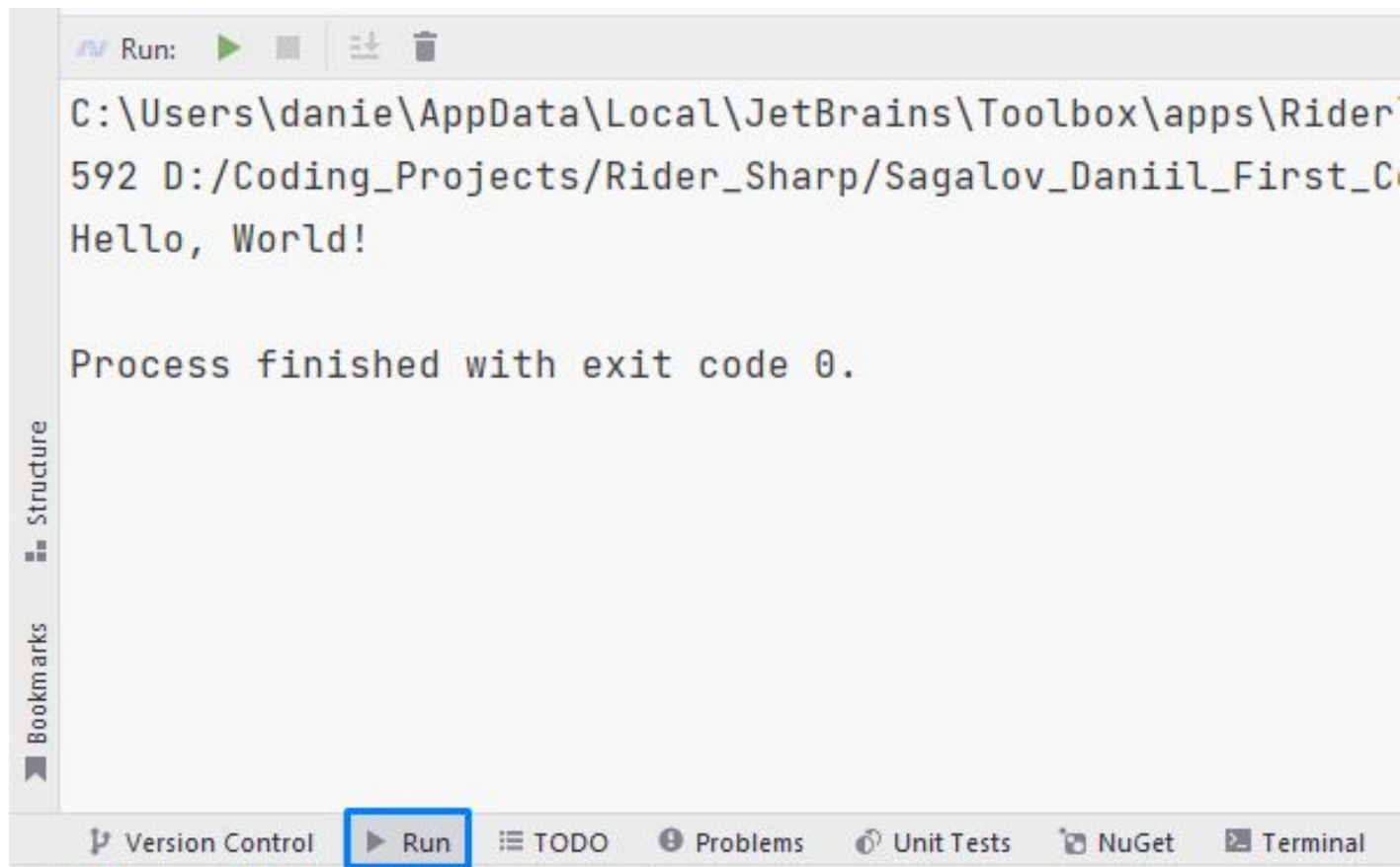
# Запуск Проекта в Rider (Множеством Способов)

- Комбинация клавиш Ctrl + F5 (Windows);
- Вкладка Run → Выбрать подходящий вариант;
- Кнопки Run или Debug в правой верхней части экрана;
- Значок запуска рядом с основным методом запускаемого приложения.



# Результат Запуска Программы в Rider

Rider визуализирует результаты работы программы в нижней части экрана во вкладке Run.



The screenshot shows the 'Run' window in the Rider IDE. The window title is 'Run:' and it contains the following text:

```
C:\Users\danie\AppData\Local\JetBrains\Toolbox\apps\Rider
592 D:/Coding_Projects/Rider_Sharp/Sagalov_Daniil_First_C
Hello, World!

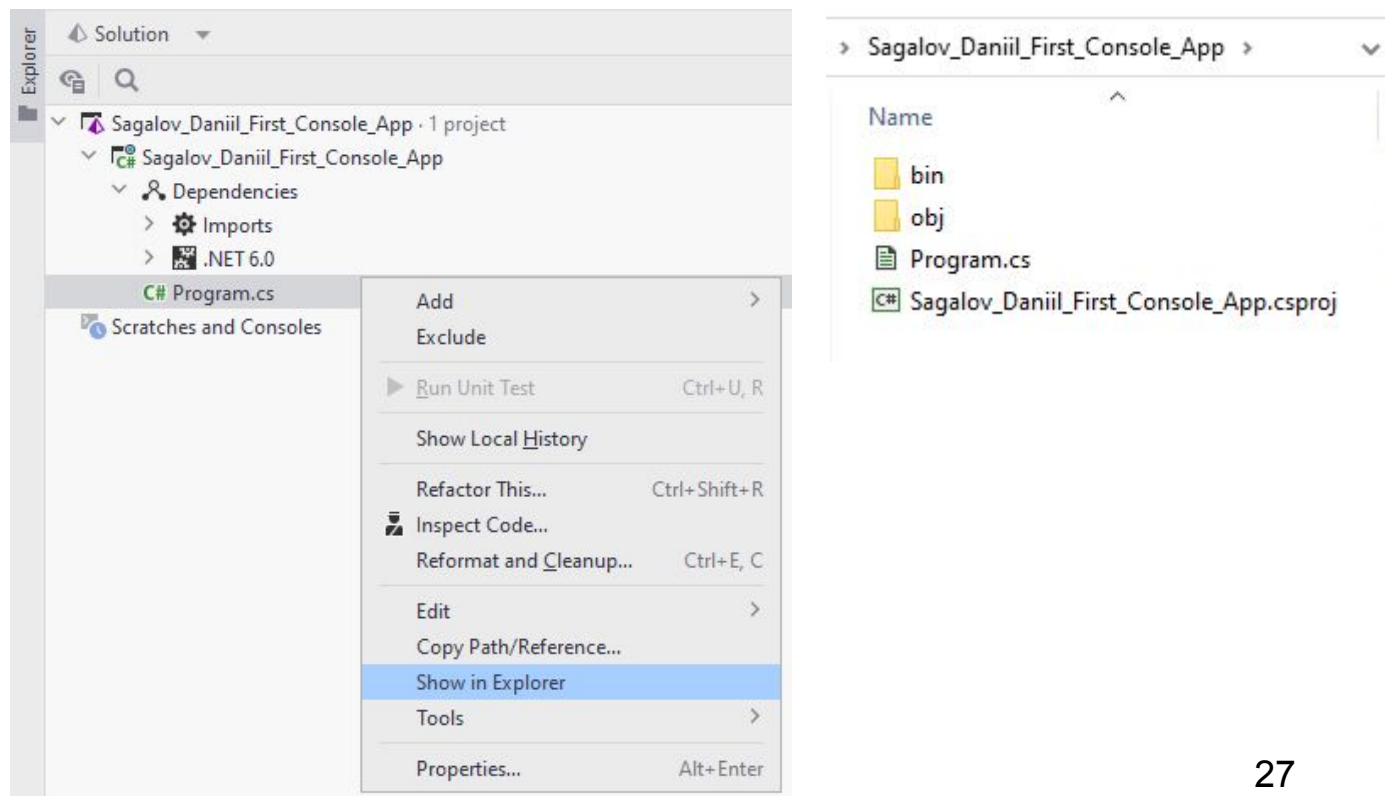
Process finished with exit code 0.
```

On the left side of the window, there are vertical tabs for 'Structure' and 'Bookmarks'. At the bottom of the window, there is a toolbar with icons for 'Version Control', 'Run' (highlighted with a blue box), 'TODO', 'Problems', 'Unit Tests', 'NuGet', and 'Terminal'.

# Запуск Программы вне Rider

Простой способ запустить программу вне Rider – ПКМ по файлу с исходным кодом → перейти с помощью **“Show in Explorer”** в папку, содержащую файл с исходным кодом, затем спуститься в **bin** → **Профиль сборки (по умолчанию Debug)** → **Версия .NET** → ЛКМ по **.exe** файлу.

**Важно:** программа мгновенно завершится, т. к. не ожидает никаких действий со стороны пользователя!



# Модификация Программы

```
using System;

namespace Sagalov_Daniil_First_Console_App
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
            Console.WriteLine("Press any key to exit...");
            Console.ReadKey();
        }
    }
}
```

Теперь программа ожидает нажатия любой клавиши, поэтому не закроется автоматически.

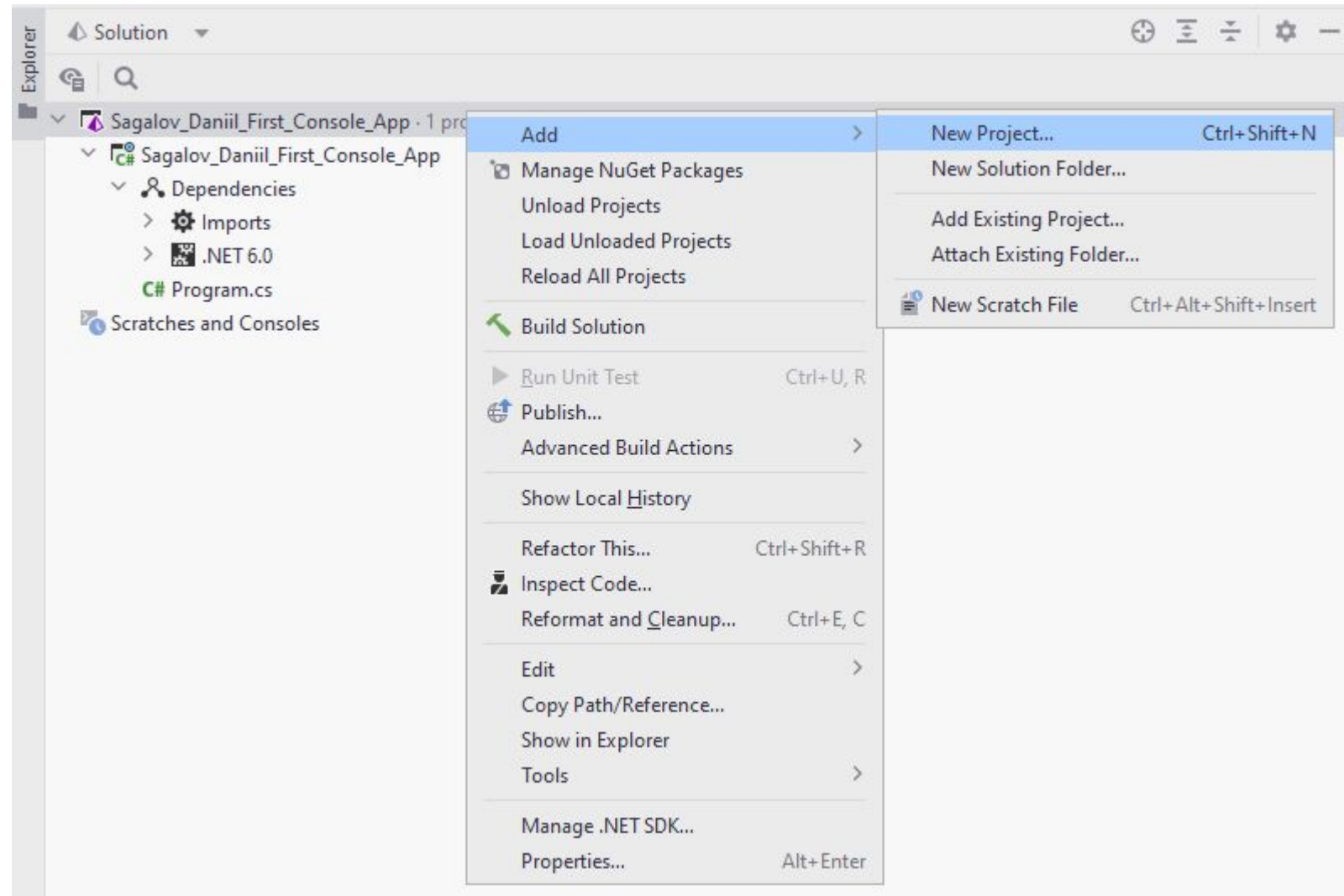
Попробуйте запустить данную программу вне Rider – теперь она не закроется мгновенно.

# Создание Второго Проекта в Rider

Удобно иметь несколько запускаемых приложений в одном решении.

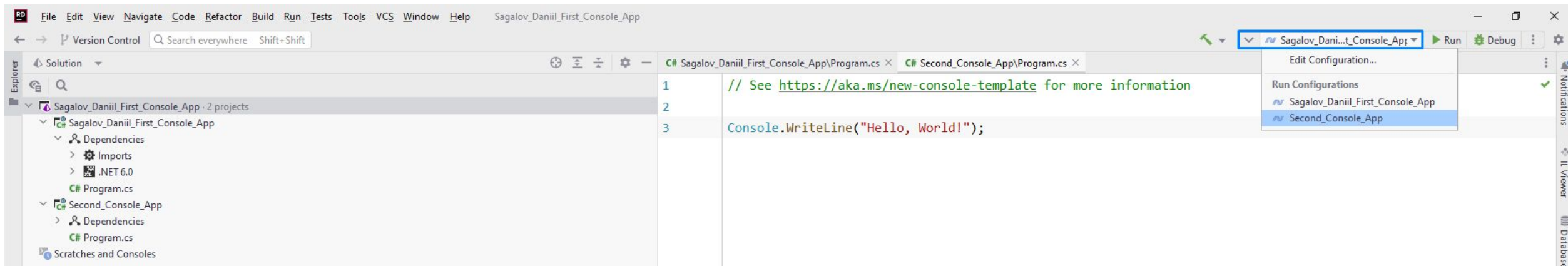
Для этого:

**ПКМ по названию решения (не проекта!)  
→ Add → New Project.**



# Выбор Запускаемого Проекта в Rider

Rider позволяет выбрать запускаемый проект с помощью меню выбора конфигураций в правом верхнем углу интерфейса:



С помощью опции “Edit Configurations” можно изменить настройки запуска проекта.

**На заметку:** для одного проекта можно создать несколько конфигураций.

# Ввод Строк с Консоли

```
using System;

class Program
{
    static void Main()
    {
        string userInput = Console.ReadLine();
        Console.WriteLine(userInput);
    }
}
```

Метод [Console.ReadLine\(\)](#) считывает строку с консоли (до перехода на новую строку) и вернёт её как значение типа string.

# Вывод Строк на Консоль. Простое Форматирование

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter text: ");
        // Рекомендуется описывать переменные максимально близко
        // к месту первого использования в коде.
        string userInput = Console.ReadLine();
        Console.WriteLine("Your input: " + userInput);
        Console.WriteLine("Your input: {0}", userInput);
        Console.WriteLine($"Your input: {userInput}");
    }
}
```

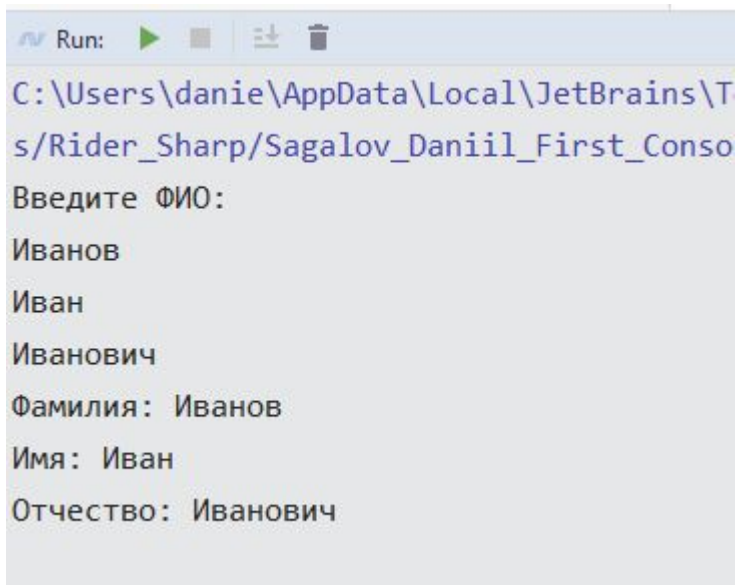
Console.Write() не добавляет переход на новую строку, в отличие от Console.WriteLine().



# Проект 3: Ввод Строк

1) Изменить код программы второго проекта так, чтобы программа запрашивала имя пользователя и здоровалась с ним по этому имени.

Пример результата:



```
Run: [Play] [Close] [Debug] [Exit]
C:\Users\danie\AppData\Local\JetBrains\Tools\Rider_Sharp\Sagalov_Daniil_First_Conso
Введите ФИО:
Иванов
Иван
Иванович
Фамилия: Иванов
Имя: Иван
Отчество: Иванович
```

2) Добавить в решение третий проект с именем Task\_03. Программа последовательно запрашивает у пользователя имя, фамилию и отчество. Сохраняет данные в три разные переменные и выводит на экран на отдельных строках в виде анкетных данных.

**Ввод:**

Иванов

Иван

Иванович

**Пример вывода:**

Фамилия: Иванов

Имя: Иван

Отчество: Иванович

# Проект 4: Арифметические Выражения

1. Создайте в решении проект с именем Task\_04;
2. В методе Main() инициализируйте переменную result:  
`string result = "5 / 3 = " + 5 / 3;`
3. Добавьте вывода значения переменной result;
4. Запустите программу (Ctrl+F5), добавьте строчку кода, изменяющую значение переменной result:  
`result = "5.0 / 3.0 = " + 5.0 / 3.0;`
5. Повторите шаги 3 и 4;
6. Добавьте ещё одну строчку кода, изменяющую значение переменной result:  
`result = 5 / 3;`
7. Повторите шаги 3 и 4;
8. Измените строку, добавленную на шаге 7 на:  
`result = 5 / 3 + " - это 5/3";`
9. Выполните шаг 4;
10. Добавьте команду, приостанавливающую выполнение программы до нажатия Enter;
11. Скомпилируйте программу;
12. Вспомните, как найти и запустить исполняемый файл вне Rider.

# Форматирование Чисел с Помощью ToString()

С помощью вызова ToString() можно форматировать числа:

**<выражение>.ToString("SR"),**

где **выражение**:

- Арифметическая константа;
- Переменная встроенного арифметического типа;
- Арифметическое выражение в круглых скобках.

**S** - спецификатор формата (**D, d, F, f, E, e, G, g,...**)

**R** – натуральное число (количество цифр в изображении числа или его дробной части)

Подробнее о форматировании чисел:

<https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings>

# Проект 5: Форматирование с Помощью ToString()

Создайте в решении проект с именем Task\_05 и посмотрите результат выполнения следующей программы:

```
using System;
```

```
class FirstSeminar
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        Console.WriteLine("2/3 = " + (2 / 3).ToString("D"));
```

```
        Console.WriteLine("2.0/3 = " + (2.0 / 3).ToString("F4"));
```

```
        Console.WriteLine("Press any key to exit...");
```

```
        Console.ReadKey();
```

```
    }
```

```
}
```

**Вывод:**

2/3 = 0

2.0/3 = 0.6667

Press any key to exit...

# Проект 6: Форматирование Таблицы с Числами

Создайте в решении проект с именем Task\_06 и, воспользовавшись возможностями метода ToString(), сформируйте и выведите на экран таблицу:

```
*-----*-----*-----*
| *Выражение* | *Формат* | *** Изображение *** |
|-----|-----|-----|
| (5.0/3.0) | F      | 1,67 |
| (5.0/3.0) | F4     | 1,6667 |
| (5.0/3.0) | E      | 1,666667E+000 |
| (5.0/3.0) | E5     | 1,66667E+000 |
| (5.0/3.0) | G      | 1,666666666666667 |
| (5.0/3.0) | G3     | 1,67 |
| (5.0/3e10) | G3     | 1,67E-10 |
| (5.0/3e-10) | G      | 16666666666,6667 |
| (5.0/3e20) | G      | 1,666666666666667E-10 |
*-----*-----*-----*
```

# Форматирование Таблицы с Числами –

## Результат

```
using System;
class Program {
    static void Main() {
        string top = "|*Выражение*|*Формат*|**** Изображение ****|";
        Console.WriteLine(top);
        string line = "|-----|-----|-----|";
        Console.WriteLine(line);
        string str = "| (5.0/3.0) | F      | " + (5.0 / 3.0).ToString("F");
        Console.WriteLine(str);
        str = "| (5.0/3.0) | F4    | " + (5.0 / 3.0).ToString("F4");
        Console.WriteLine(str);
        str = "| (5.0/3.0) | E      | " + (5.0 / 3.0).ToString("E");
        Console.WriteLine(str);
        str = "| (5.0/3.0) | E5    | " + (5.0 / 3.0).ToString("E5");
        Console.WriteLine(str);
        str = "| (5.0/3.0) | G      | " + (5.0 / 3.0).ToString("G");
        Console.WriteLine(str);
        str = "| (5.0/3.0) | G3    | " + (5.0 / 3.0).ToString("G3");
        Console.WriteLine(str);
        str = "| (5.0/3e10)| G3    | " + (5.0 / 3e10).ToString("G3");
        Console.WriteLine(str);
        str = "|(5.0/3e-10)| G      | " + (5.0 / 3e-10).ToString("G");
        Console.WriteLine(str);
        str = "| (5.0/3e20)| G      | " + (5.0 / 3e10).ToString("G");
        Console.WriteLine(str);
        line = "*-----*";
        Console.WriteLine(line);
    }
}
```

# Проект 7: Преобразование Строк в Целые Числа

Создайте в решении проект с именем Task\_07:

```
using System;

class Program
{
    static void Main()
    {
        int first; // Объявляем целочисленные
переменные.
        Console.Write("Enter an integer: ");
        string userInput = Console.ReadLine(); // Чтение с клавиатуры строки.
        first = int.Parse(userInput); // Преобразование в тип int.

        Console.Write("Enter an integer: ");
        userInput = Console.ReadLine();
        int.TryParse(userInput, out int second);
        Console.WriteLine("Result: " + first + second); // Вывод.
    }
}
```

*Попробуйте:*

1. Ввести вместо первого числа строку;
2. Ввести вместо второго числа строку;
3. Заключить `firstInt + secondInt` в круглые скобки.

# Методы Преобразования Строк – Parse и TryParse

```
int parseResult = int.Parse(str);
```

**<тип> <тип>.Parse(<строка>)**

Преобразует строковое представление числа в эквивалентное ему значение типа <тип>. В случае, если строковое представление не может быть преобразовано к соответствующему типу, возникает ошибка выполнения – исключение.

```
bool parsedSuccessfully = int.TryParse(str, out int parseResult);
```

**bool <тип>.TryParse(<строка>, out <переменная>)**

Преобразует строковое представление числа в эквивалентное ему значение типа <тип> и присваивает значение переменной типа <тип>. В случае, если строковое представление не может быть преобразовано к соответствующему типу, метод возвращает значение **false**, иначе возвращает **true**.



# Проект 8: Операции над Целыми Числами

Создайте в решении проект с именем Task\_08. Введите два целых числа типа `int` (`left`, `right`) и выведите результаты выполнения следующих операций:

```
(left - right)
(left * right)
(left / right)
(left % right)
(left << right)
(left >> right)
```

# Проект 9: Операция Приведения Типов

(<тип>)(выражение)

<http://msdn.microsoft.com/ru-ru/library/ms173104.aspx>

```
double myPi = 3.1418281828;  
int piIntegralPart = (int)myPi;  
Console.WriteLine(piIntegralPart);
```

3

```
char letter = 'A';  
int letterCode = letter;  
Console.WriteLine(letterCode);
```

65

```
int value = 98;  
char letter = value;  
Console.WriteLine(letter);
```

**Ошибка компиляции!!!**

Создайте в решении проект с именем Task\_09. Введите два вещественных числа типа double. Выведите сумму их дробных частей.

# Проекты 10-11

В проекте с именем Task\_10 напишите программу, решающую следующую задачу. Программа запрашивает у пользователя три любых слова, после чего печатает их в одну строку, разделяя вместо пробела символом восклицательный знак.

**Ввод:**

Небо  
Солнце  
Море

**Вывод:**

Небо!Солнце!Море

В проекте с именем Task11 напишите программу, решающую следующую задачу. Программа запрашивает у пользователя три любых слова, после чего печатает их в столбик, обрамляя каждое слово слева и справа знаками минус.

**Ввод:**

Небо  
Солнце  
Море

**Ввод:**

-Небо-  
-Солнце-  
-Море-

# Задачи для самостоятельного решения

Создайте решение с именем `<Имя>_<Фамилия>_HW_1` и добавьте в него 3 проекта:

- **Task01:** осуществить вывод на экран строки “Hello, World!”;
- **Task02:** В текстовом виде выведите в консольное окно изображение вашей фамилии, составленное из выбранных вами символов (\*, x, проч.);
- **ASCIIDecoder:** Пользователем вводится корректное число в диапазоне `symbolCode [32; 127]`. Выведите на экран изображение символа, представленное в таблице кодов **ASCII** кодом `symbolCode`. При выполнении задания используйте только методы преобразования строк в целочисленные типы и операцию приведения типов.

# Задачи для самостоятельного решения

Добавьте в созданное ранее решение ДЗ (см. прошлый слайд) ещё 2 проекта:

- **Task04:** Давайте вспомним физику! Получите от пользователя значения напряжения **U** и сопротивления **R** и вычислите
  - 1) силу тока:  
$$I = U / R$$
  - 2) потребляемую мощность электрической цепи:  
$$P = U^2 / R$$
- **Task05:** На основе ввода пользователем вещественных длин двух катетов, вычислите и выведите на экран длину гипотенузы.