# Программирование на C#

# Семинар №1

Модуль №1

## Тема: «Среда разработки VS Code»

Настройка инфраструктуры разработки.

Распределение кода по файлам.

Командная строка .NET.

Запуск и отладка программ

Задачи на ввод и вывод текстовых данных.

# Задания преподавателя к семинару

1. Установить на личный компьютер интегрированную среду разработки VS Code по инструкциям, указанным на слайде 3. При выборе дополнительных задач установщика – выбрать ВСЕ!

**ССЫЛКА НА ВИДЕО** https://youtu.be/tC6nTO6zBfQ?si=VkBC8ASFZ1l-fpxR

1. Дополнительно установить из запущенного VS Code

- C# (https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp ): позволяет подсвечивать синтаксис, отлаживать и запускать код; для корректной работы проверить установку .Net

Полезные расширения для работы с кодом и тестами:

- C# namespace autocompletion (https://marketplace.visualstudio.com/items?itemName=adrianwilczynski.namespace

- Auto-using for C# (https://marketplace.visualstudio.com/items?itemName=Fudge.auto-using)

- .Net Core Test Explorer (https://marketplace.visualstudio.com/items?itemName=formulahendry.dotnet-test-explorer)

3. Создать проект в VS Code. Выполните задания task01

4. Попробуйте внести изменения в код (error), запустить отладчик и изучить способы диагностики ошибок в среде разработки.

5. Выполнить задания task02- task04.

# Полезные Материалы

## Старт

1. **Get started with C# and .NET in Visual Studio Code**
https://code.visualstudio.com/docs/csharp/get-started

2. **«Гайд по установке VS Code»** – см файл  в папке семинара. Разработчики – учебные ассистенты.

3. **«Горячие клавиши для работы в VS Code»** – см файл в папке семинара

4. **Редактор кода Visual Studio Code. Самый подробный гайд по настройке и установке плагинов для начинающих** https://habr.com/ru/articles/490754/#install

5. **Отладка** https://learn.microsoft.com/ru-ru/dotnet/core/tutorials/debugging-with-visual-studio-code?pivots=dotnet-7-0

## Разработка приложений

**Операторы верхнего уровня — программы без Main методов**
https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/program-structure/top-level-statements

**Учебник. Создание консольного приложения .NET в Visual Studio Code** https://learn.microsoft.com/ru-ru/dotnet/core/tutorials/with-visual-studio-code?pivots=dotnet-7-0

# Настройка Инфраструктуры

**Вид > Макет редактора**

**Меню > Вид > Внешний вид.**

Для увеличения элементов я - **Ctrl+,** для уменьшения — **Ctrl-,** для сброса настроек — **Ctrl+0.**

Полноэкранный режим **- F11.** Скроется верхнее меню, кнопки управления.

**Режим Zen** скрывает все элементы, даже вкладки с файлами, виден только код. Выход - **Escape**.

**Alt + Z** перенос текста

**Цветовая схема** – выбор стандартных установок или загрузка собственных

**«Хлебные крошки» –** элементы управления

**Управление > Тема значков файлов** убрать значки файлов или выбрать

# Управление Параметрами Сохранения

**Управление > Параметры** или **Ctrl+,**

**afterDelay** — файл сохранится после задержки, которую можно настроить в окне ниже.

**onfocusChange —**сохранение файла  при переходе к другому файлу

**onWindowChange —**сохранение файла при переходе к другой программе.

## Форматирование

**Shift + Alt + F –** автоформатирование

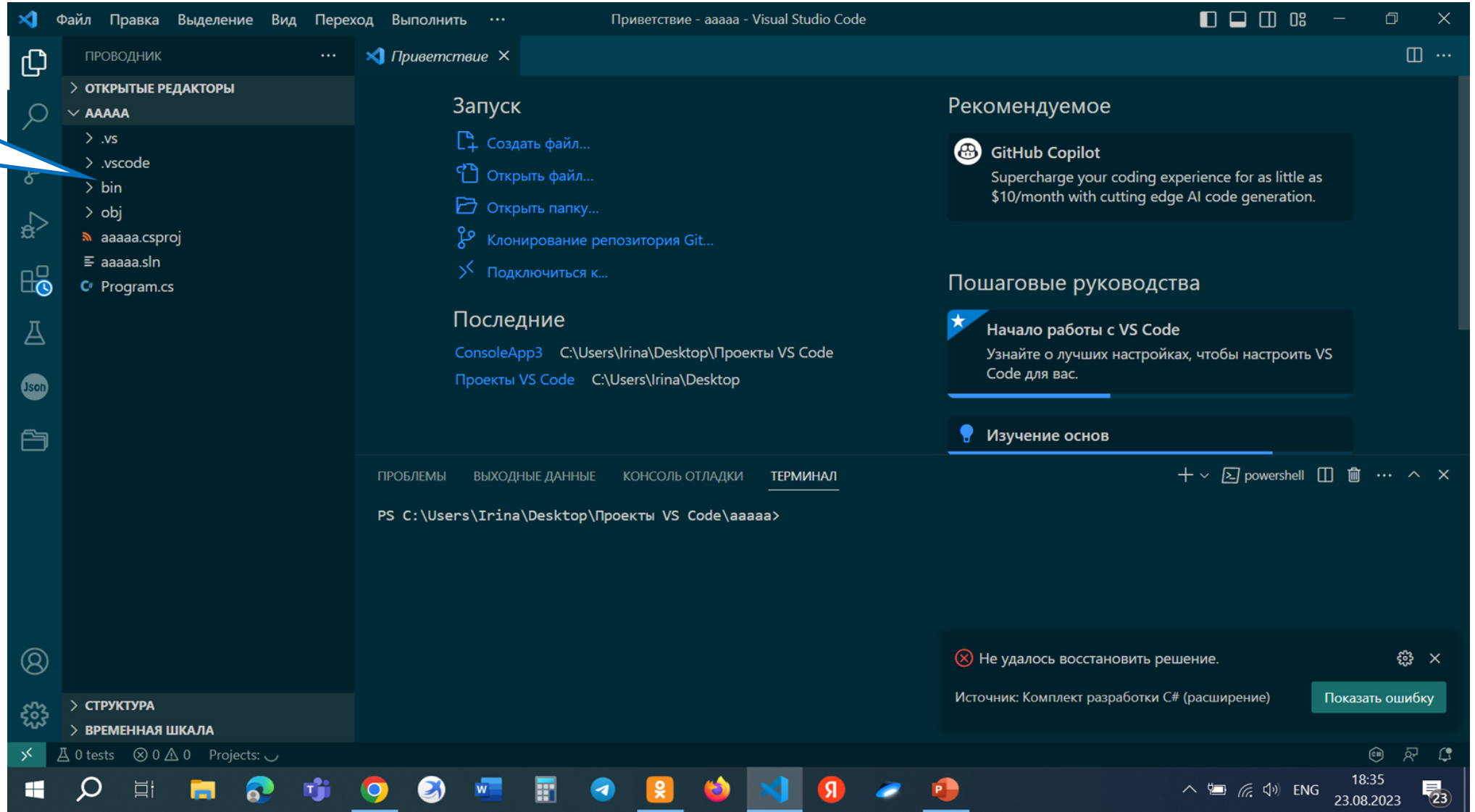
Отмечаем один из пунктов в настройках:

**Format On Paste** — форматирование производится автоматически при вставке кода.
**Format On Save** — форматирование производится в момент сохранения файла.
**Format On Type** — форматирование автоматическое

# Распределение Кода по Файлам

**Файл конфигурирования отладчика** SolutionDir\.vscode\launch.json создается при запуске отладки проекта, содержит конфигурацию по умолчанию

**Раскрываем папки**



.VS

**>aaaa**

**>**DesignTimeBuild

>FileContentIndex

>v17

**>ProjectEvaluation**

aaaa.metadata.v7.bin

aaaa.projects.v7.bin

**>.vscode**

{ } settings.json

**>bin\Debug\net7.0**

{ } aaaa.dll

aaaa.dll

aaaa.exe

aaaa.pdb

aaaa.rumtimeconfig.json

**>obj**

. . . . . .

- **aaaa.csproj**
- **aaaa.sln**
- **Program.cs**

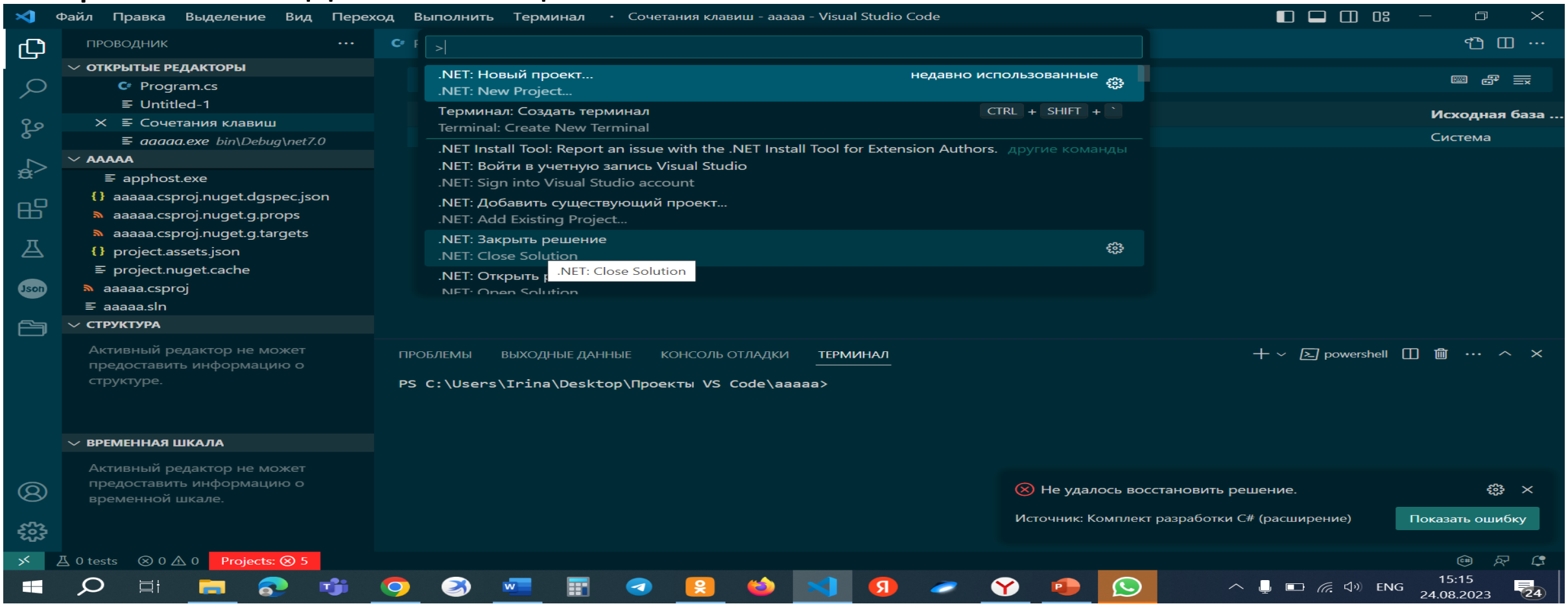
***главный файл проекта, определяет его конфигурацию***

***Файл решения – данные проекта***

***Файл, содержащий код программы***

# Командная Строка .Net

1. Вызов – **Ctrl+Shuft+P.**
2. **ToDo 01:** создайте консольное приложение при помощи командной строки. Выведите сообщение «Hello  C#!»



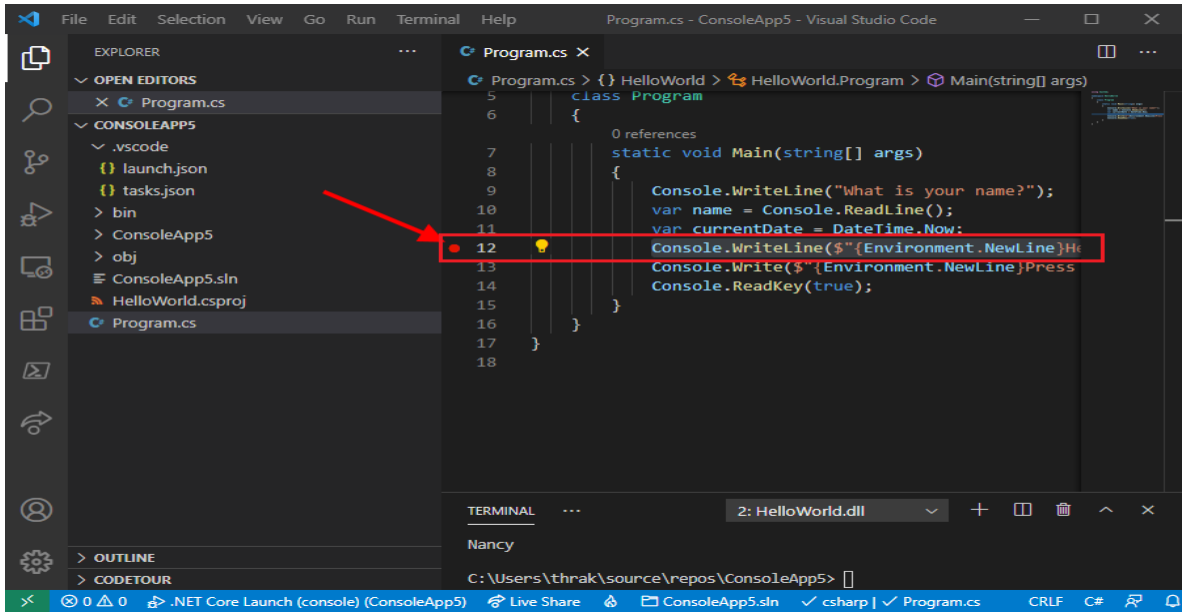
# Запуск и Отладка

## Пакетная отладка

Варианты:
**F5**
**Отладка>Начать с отладки** в меню
зеленая стрелка **Запуск** и имя проекта на панели инструментов

## Отладка с точками останова

1.Откройте файл Program.cs (имя по умолчанию)или другой  файл с модулем Main().
2. Установите **Точку останова** (приостанавливает выполнение  на предшествующей точке инструкции)
3. **Запуск>Перейти к следующей точке останова**  или **F9**




**Запуск>Шаг с заходом** или **F11 –** переход на вызываемую функцию
**Запуск>Шаг с обходом или F10 –** без Захода в функции
**Запуск>Шаг с выходом** или **SHIFT+F11**

# Ввод-Вывод Строк  на Консоль

Ввод данных осуществляется в виде строки символов.

```
string str;  // Описание переменной для сохранения строки.
                    присваивание считанного значения
 str = Console.ReadLine(); //Вызов метода чтения строк из консоли класса Console.

/* Преобразование строкового представления
    в вещественное число  или  в иной встроенный тип данных, кроме строки*/
double r = double.Parse(str);
 r = r/2.;
// Для вывода значений любого типа преобразуем их в строку при помощи метода ToString().
Console.WriteLine(r.ToString());
 // метод вывода строк
Console.WriteLine("Good luck with your studies! "); // Вывод строкового литерала.
```

# ToDo 02. Поиск ошибок в коде

Перед вами код, содержащий ошибку.

```
Console.Write("C# ");
Console.WriteLine("Hello, World!");
Console.WriteLine(23);
Console.WriteLine(a+b);)
```

Сформируйте проект, скопируйте в исходный модуль код и запустите отладку. Определите при помощи отладчика ошибку и включите в текст программы комментарий с соответствующим сообщением (последней строкой)

# Demo. Ввод-Вывод Строк

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

Метод Console.ReadLine() считает строку с консоли (до перехода на новую строку) и вернёт её как значение типа string.

# Вывод Строк  в Консоль

Console.**WriteLine**(r.ToString());  - вывод строки с переходом на новую
Console.**Write**(аргумен_вывода)  позволяет выводить данные любого типа, но не осуществляет сброс на следующую строку:

```
Console.Write(“Hi all”);
```

Для сброса курсора на новую строку добавим Escape-последовательность **‘\n’:**

```
Console.Write(“Hi all\n”);
```

**D** или **d** - целое число
**F** или **f** –  вещественное число в форме F
**E** или **e** – веществ. число в форме E (мантисса и порядок)
**N** или **n** – вещественное число с выделением тысячи
**X** или **x** – целое 16-ричное число
**P** или **p** – процент
**C** или **c –** валютный формат

Полный перечень Esc-последовательностей см.  https://learn.microsoft.com/ru-ru/cpp/c-language/escape-sequences?view=msvc-170

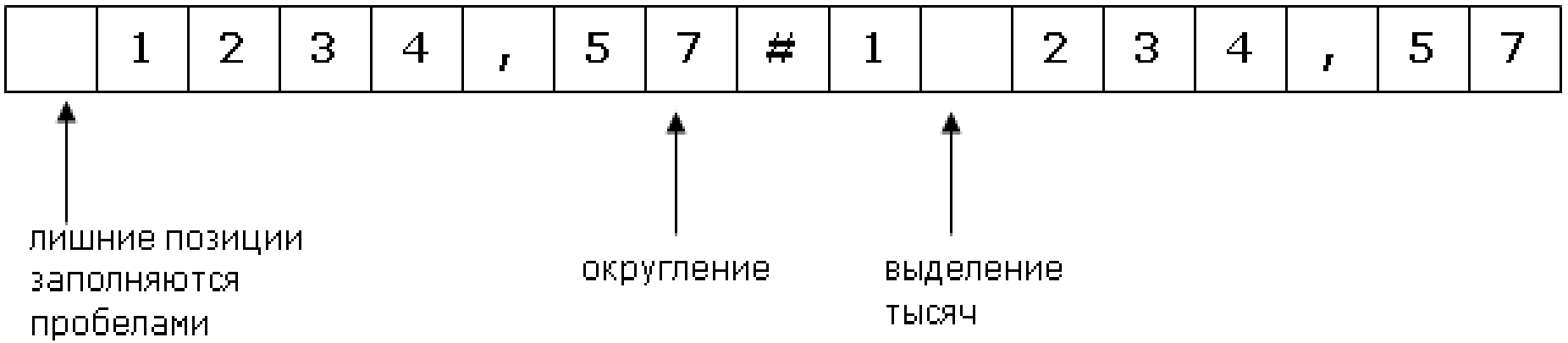
# Вывод Строк  на Консоль. Метод Format

Метод **Format()** класса String. В строковый литерал вывода помимо выводимых на экран символов включается формат вида:

{номер значения<,*ширина поля*><:Вид  *Кол-во знак*ов>}

необязательные поля

```
double a=1234.5678;
long b=3456;
short c=65;
string s1;
s1=String.Format("{0,8:f2}#{1,8:n2}",a,a);
```



# Вывод Строк. Интерполированная строка

**$“<text> {<expr>** [,width][:<format_string>]**} <text>”**

Параметры в квадратных скобках не обязательны.
NB! Между $ и кавычкой “ в начале строкового литерала не может быть пробела.

```
    Console.WriteLine("Интерполированные строки");
    Console.WriteLine();
    Console.WriteLine($"Дата {DateTime.Now,-20:d}");//Вывод системного
времени.

    double x = 22.99;
    double y = 35.88;
    long phone = 79114444332;

    double s = x + y;

    Console.WriteLine($"{x} + {y} ={s,6:f1}"); //58,9 вместо 58,87
    Console.WriteLine($"{x} - {y} ={x - y,6:f3}"); // -12,890

    string str = $"Мой номер телефона: {phone:+#-(###)-###-##-##}";
    Console.WriteLine(str);
```

# Demo. Вывод строк в 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: {0}", userInput);
         Console.WriteLine($"Your input: {userInput}");
         Console.WriteLine("Your input: " + userInput);
    }
}
```

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

Строки можно соединить в одну при помощи знака +

# Self. Ввод-Вывод

**Task01**. Разработайте программу, которая позволяет считать с клавиатуры фамилию, имя и отчество. Вывод фамилии дополнить текстом «Фамилия:», аналогично для имени и отчества.

**Ввод:**
Иванов
Иван
Иванович

**Пример вывода:**
Фамилия: Иванов
Имя: Иван
Отчество: Иванович

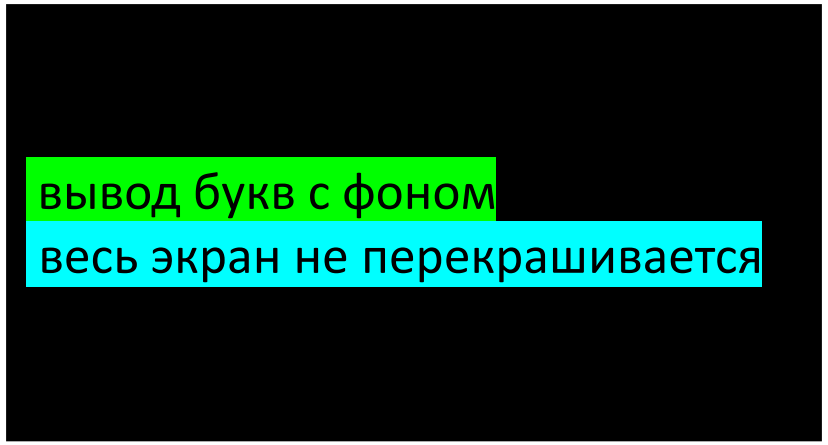
**Task02**. В проекте Task 01 замените вывод на «Фамилия Имя Отчество» в одну строку: Иванов Иван Иванович

**Task03**. Изменить код программы Task01 так, чтобы программа запрашивала имя пользователя и здоровалась с ним по этому имени.

# Self. Ввод-Вывод

**Task04.** Черно-белую консоль можно сделать цветной, изменяя:

- Цвет буквы
- Цвет фона, на котором выводятся буквы
- Цвет консольного окна.




Для заливки окна определенным цветом применяем Console.Clear()

Попробуйте сделать «цветную» консоль из любой выполненной программы.

https://learn.microsoft.com/ru-ru/dotnet/api/system.console.backgroundcolor?view=net-7.0

https://learn.microsoft.com/ru-ru/dotnet/api/system.console.foregroundcolor?view=net-7.0

# Self. Ввод-Вывод

**Task05:** 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:

$$P = U^2 / R$$

**Task06:** На основе ввода пользователем вещественных длин двух катетов, вычислите и выведите на экран длину гипотенузы.