Доклад

По Функционално програмиране

1.Какво означава функционално програмиране

2.Концепции за функционално програмиране в C#

3.Функции - определение/описание/ примерен програмен код

4.Чисти функции - определение/описание/ примерен програмен код

5.Функции като стойности на функция - определение/описание/ примерен програмен код

6.Рекурсия - определение/описание/ примерен програмен код чрез ФП

7.Рекурсивна реализация на цикли - примерен програмен код чрез ФП

8.Опашкова рекурсия - примерен програмен код чрез ФП

9.Рекурсивно обхождане на списък - примерен програмен код чрез ФП

1. Какво означава функционално програмиране

- Функционалното програмиране е начин за съставяне на програми, при който единственото действие е обръщението към функции, единственият начин за разделяне на програмата на части е въвеждането на име на функция и задаването за това име на израз, който пресмята стойността на функцията, а единственото правило за композиция е суперпозицията на функции.

Тук под функция се разбира програмна част, която “връща” резултат (по-точно, във функционалното програмиране се работи с т.нар. строги функции, които не предизвикват никакви странични ефекти, а само връщат стойности). Във функционалното програмиране не се използват оператори за присвояване и за цикъл, блок-схеми, предаване на управлението.

2. Концепции за функционално програмиране в C#

Определен брой концепции и парадигми са специфични за функционалното програмиране и като цяло чужди на императивното програмиране (включително и обектно-ориентираното програмиране). Въпреки това, програмните езици често са хибриди на няколко програмни парадигми, така че програмистите използващи "предимно императивни” езици може и да използват някои от тези понятия.

3. Функции - определение/описание/ примерен програмен код

- Функциите са основни за организацията на кода. Те съществуват във всички езици за програмиране от по-висок ред. Като цяло функционалното програмиране означава използване на функции с най-добър ефект за създаване на чист и поддържан софтуер. По-конкретно, функционалното програмиране е набор от подходи за кодиране, обикновено описвани като парадигма за програмиране.

Зачада: Да се състави функция, в която се въвеждат 2 числа като параметри. Функцията да връща сумата на двете числа.

Да се въведат двете числа, да се изчисли сумата им чрез функцията и да се изведе резултата.

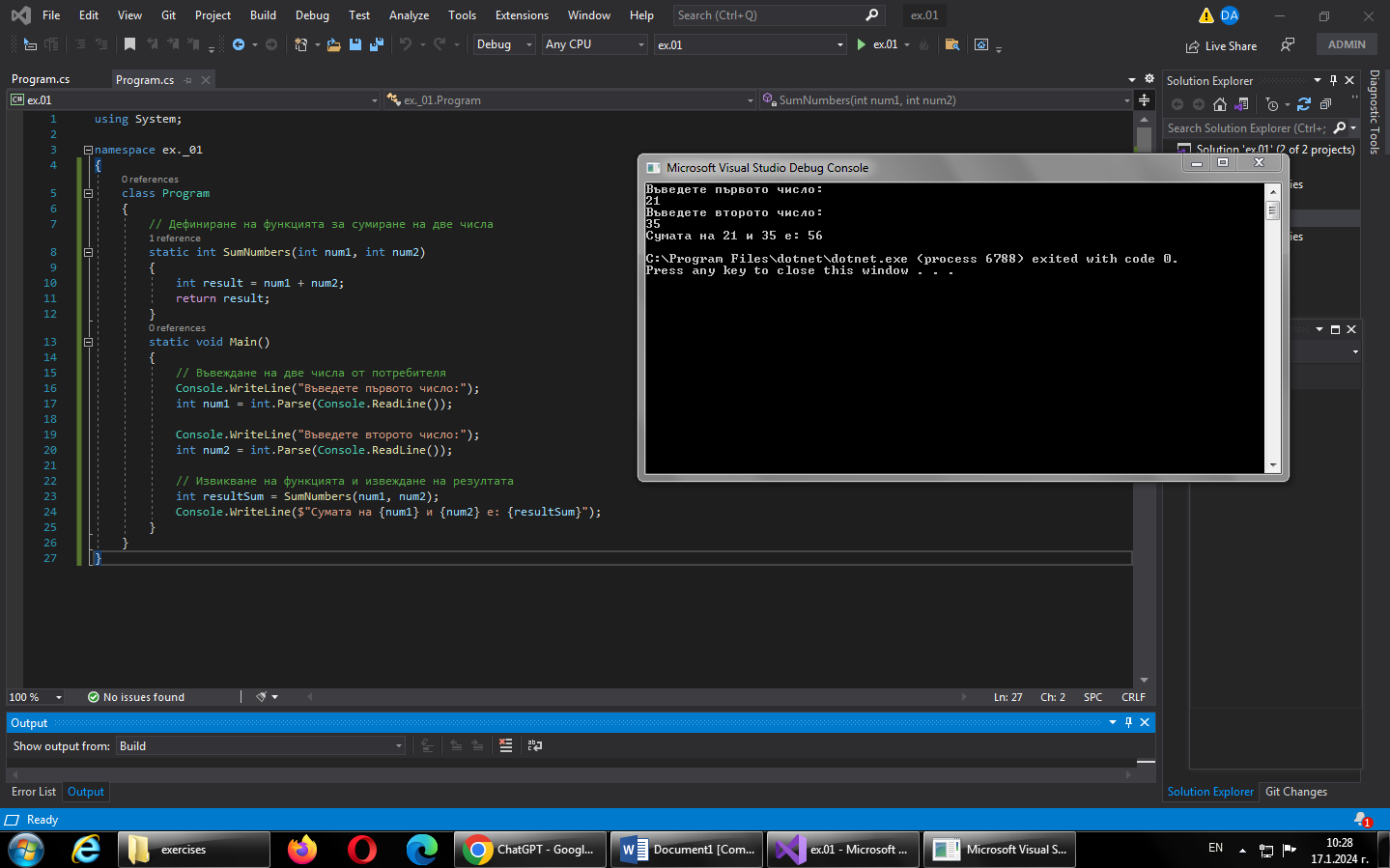
Вход:

21

35

Изход:

56



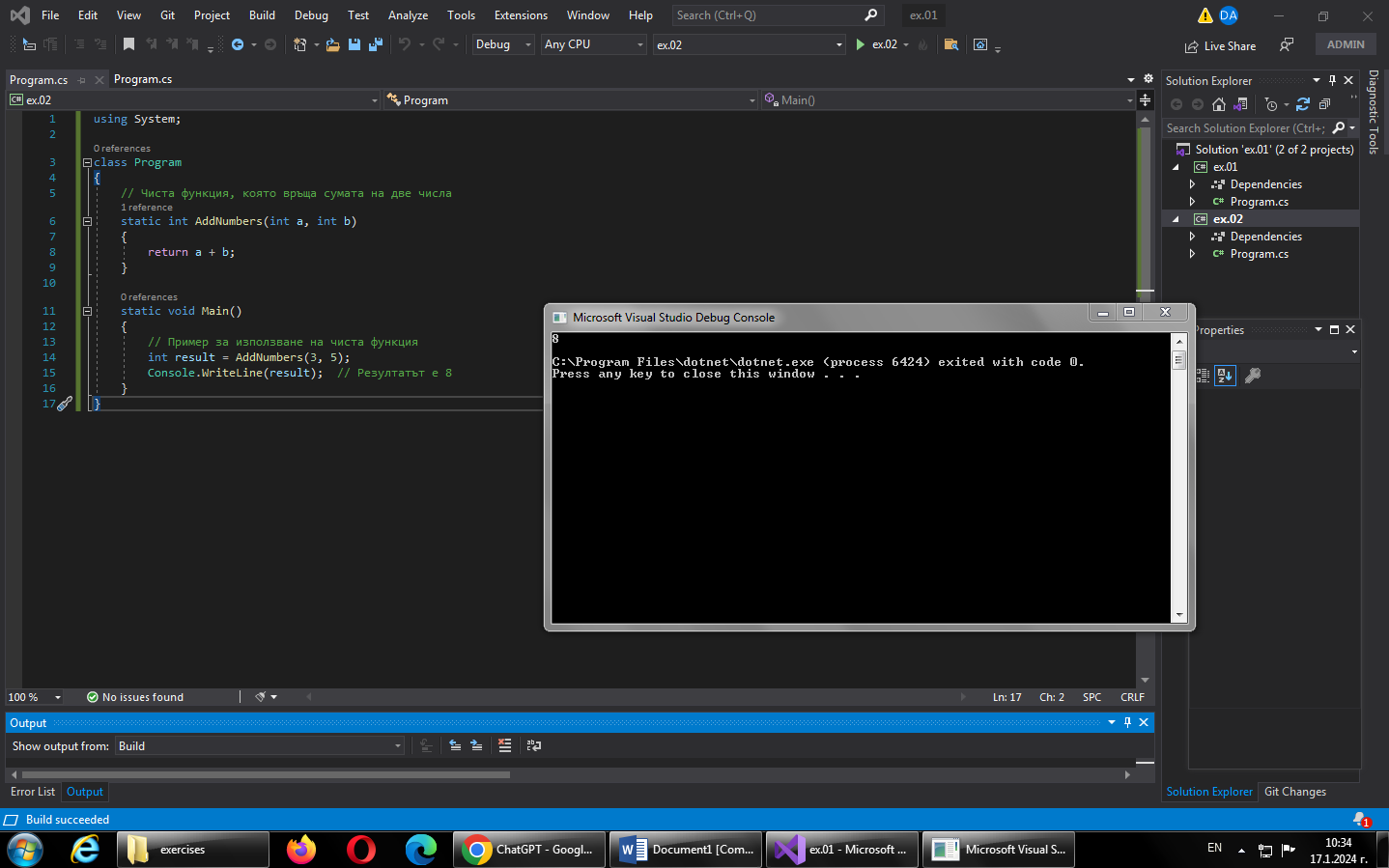
4.Чисти функции - определение/описание/ примерен програмен код

- Чиста функция е тази, чиито резултати зависят само от входните параметри и чиято операция не инициира никакъв страничен ефект, тоест не прави външно въздействие освен връщаната стойност.

Красотата на чистата функция е в нейната архитектурна простота. Тъй като чистата функция се свежда само до аргументите и връщаната стойност (т.е. нейния API), тя може да се разглежда като задънена улица на сложността. Единственото й взаимодействие с външната система, в която работи, е чрез дефинирания API.

Задача: Чиста функция, която връща сумата на две числа

Изход: 8



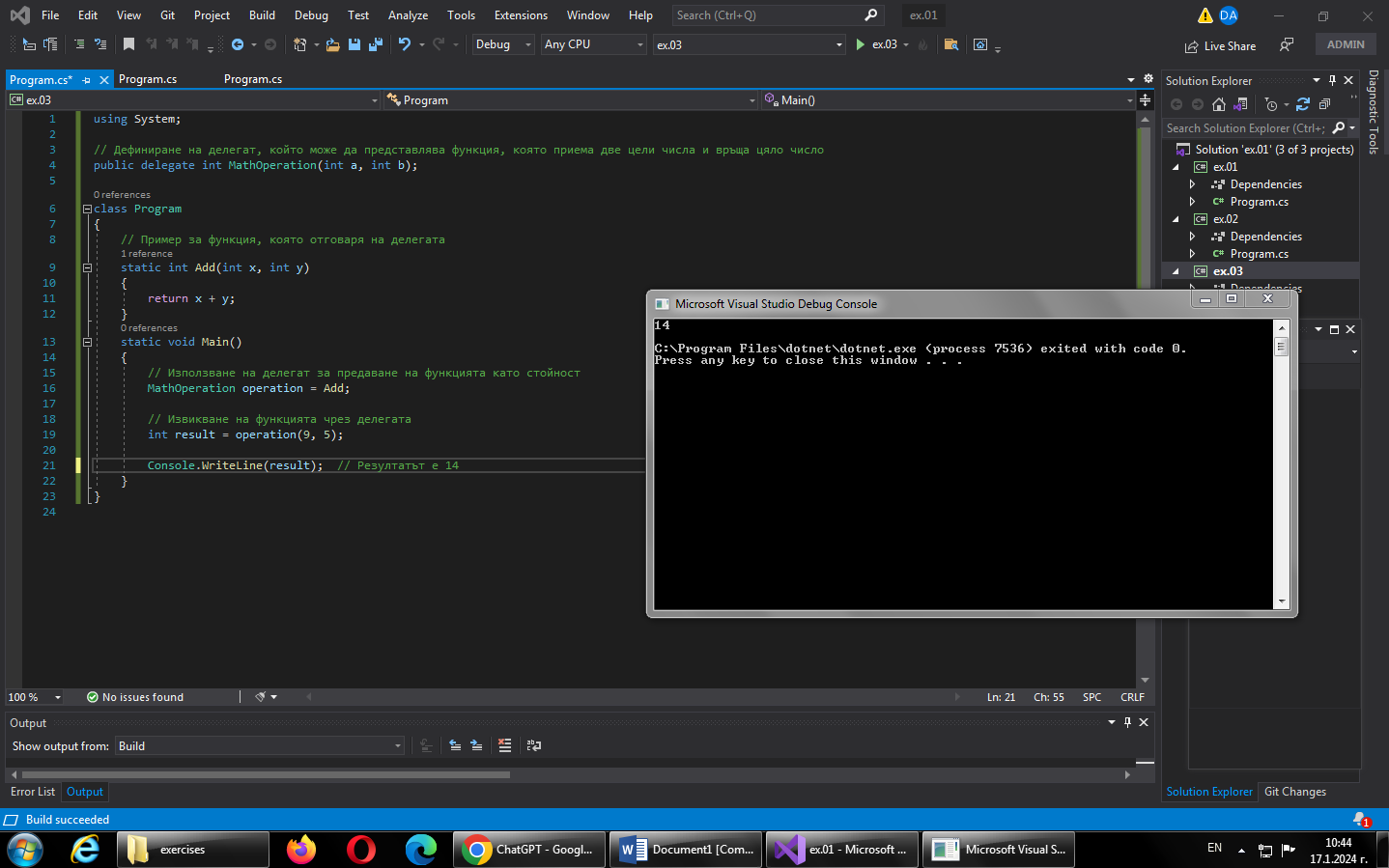
Функцията AddNumbers в този пример е чиста, тъй като връща резултат само въз основа на своите входни параметри a и b, без да променя никакво глобално състояние или променлива. Това е основно условие за функцията да се счита за чиста в парадигмата на функционалното програмиране.

5.Функции като стойности на функция - определение/описание/ примерен програмен код

- В C# функциите не се третират директно като стойности (като функционална програмиране), но можем да използваме делегати, които предоставят сходна функционалност.

Задача: Функция, която илюстрира използването на делегати за предаване на функции като стойности:

Изход: 14



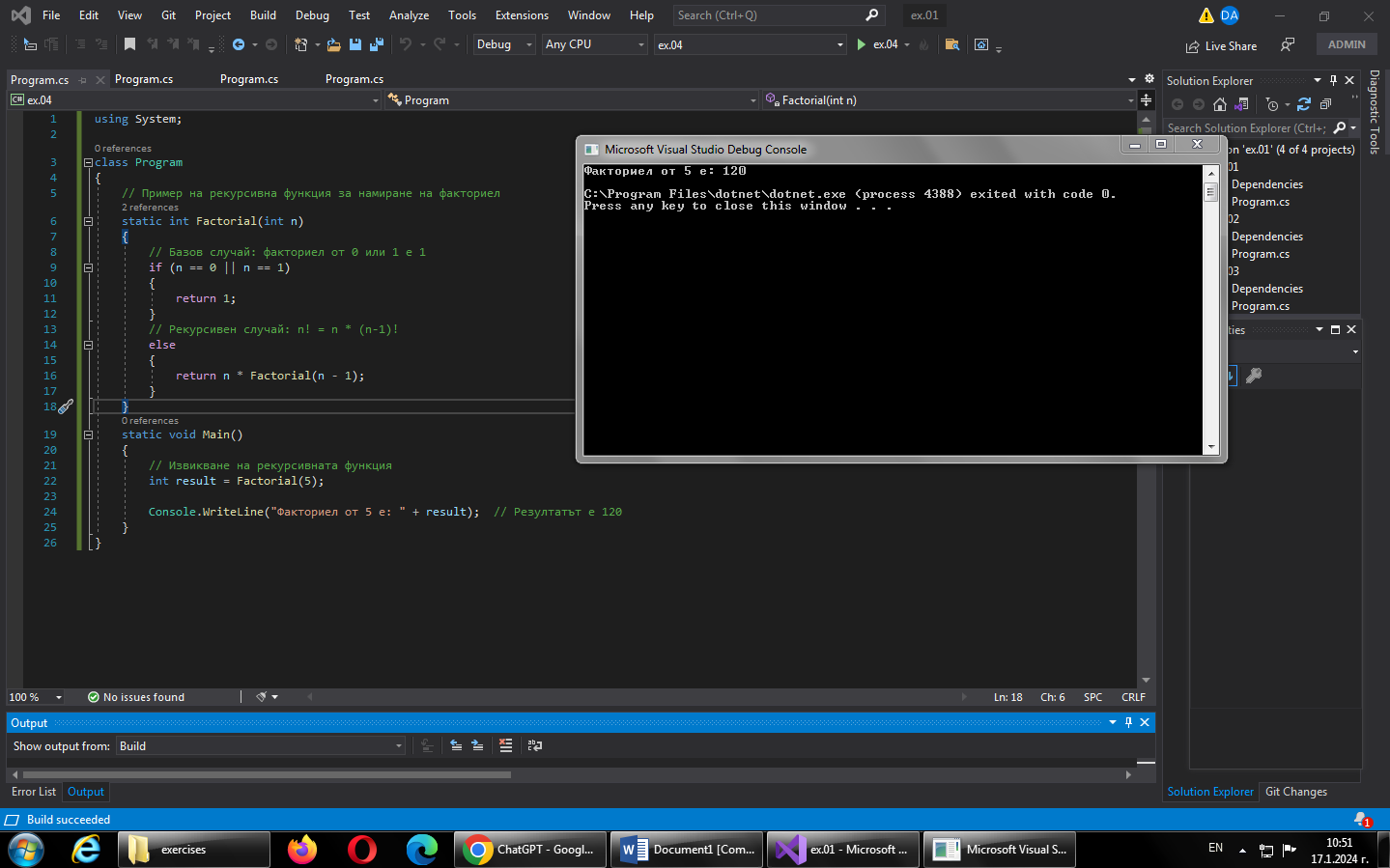
В този пример MathOperation е делегат, който може да представлява функции, приемащи две цели числа и връщащи цяло число. Функцията Add съответства на този делегат. Можем да създадем делегат, който ще съхранява референция към функцията Add и след това да я извикаме чрез делегата. Така можем да използваме функции като стойности.

6.Рекурсия - определение/описание/ примерен програмен код чрез ФП

- Рекурсията в програмирането се отнася до техниката, при която функция се извиква сама, непосредствено или чрез други функции, които от своя страна я извикват. Рекурсията е често използвана техника в функционалното програмиране, където функциите са първокласни обекти, и те могат да бъдат подавани като аргументи на други функции.

Задача: Пример на рекурсивна функция в C#:

Изход: Факториел от 5 е: 120



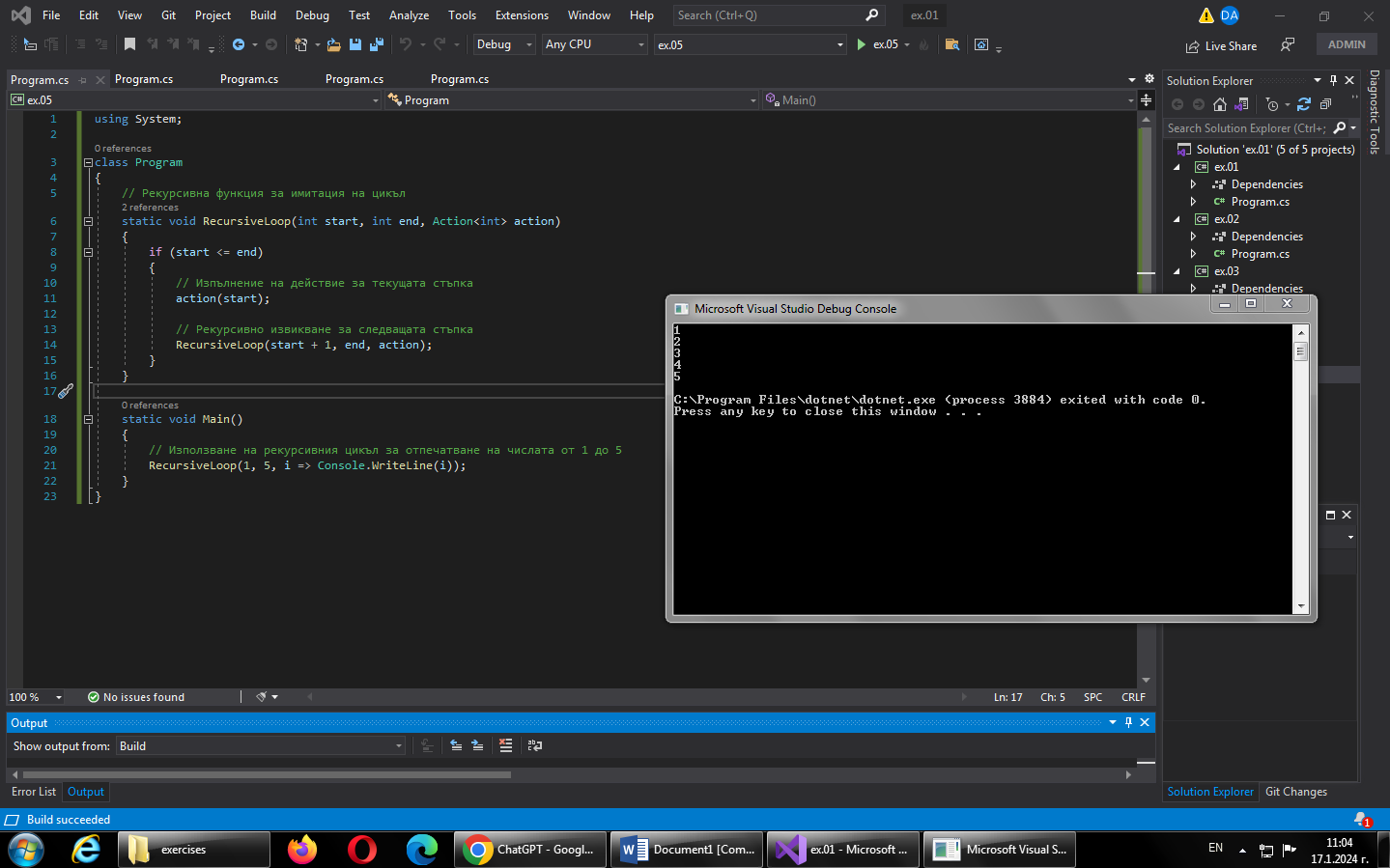
В този пример функцията Factorial изчислява факториел от дадено число n. Тя използва рекурсия, където стъпката за n! е n умножено по (n-1)!, и така нататък, докато се достигне базовия случай (когато n стане 0 или 1). Рекурсивните функции често изискват базов случай, за да предотвратят безкрайната рекурсия.

7.Рекурсивна реализация на цикли - примерен програмен код чрез ФП

- Рекурсивна реализация на цикъл е вид рекурсия, която се използва за имитиране на поведението на цикъл (например, for или while цикъл) чрез рекурсивни извиквания на функции. Вместо използването на итеративни конструкции за повторение на определени действия, рекурсивният цикъл постига същия ефект чрез рекурсивни извиквания.

Задача: Пример на рекурсивна реализация на цикъл в C#:

Изход: 1 2 3 4 5

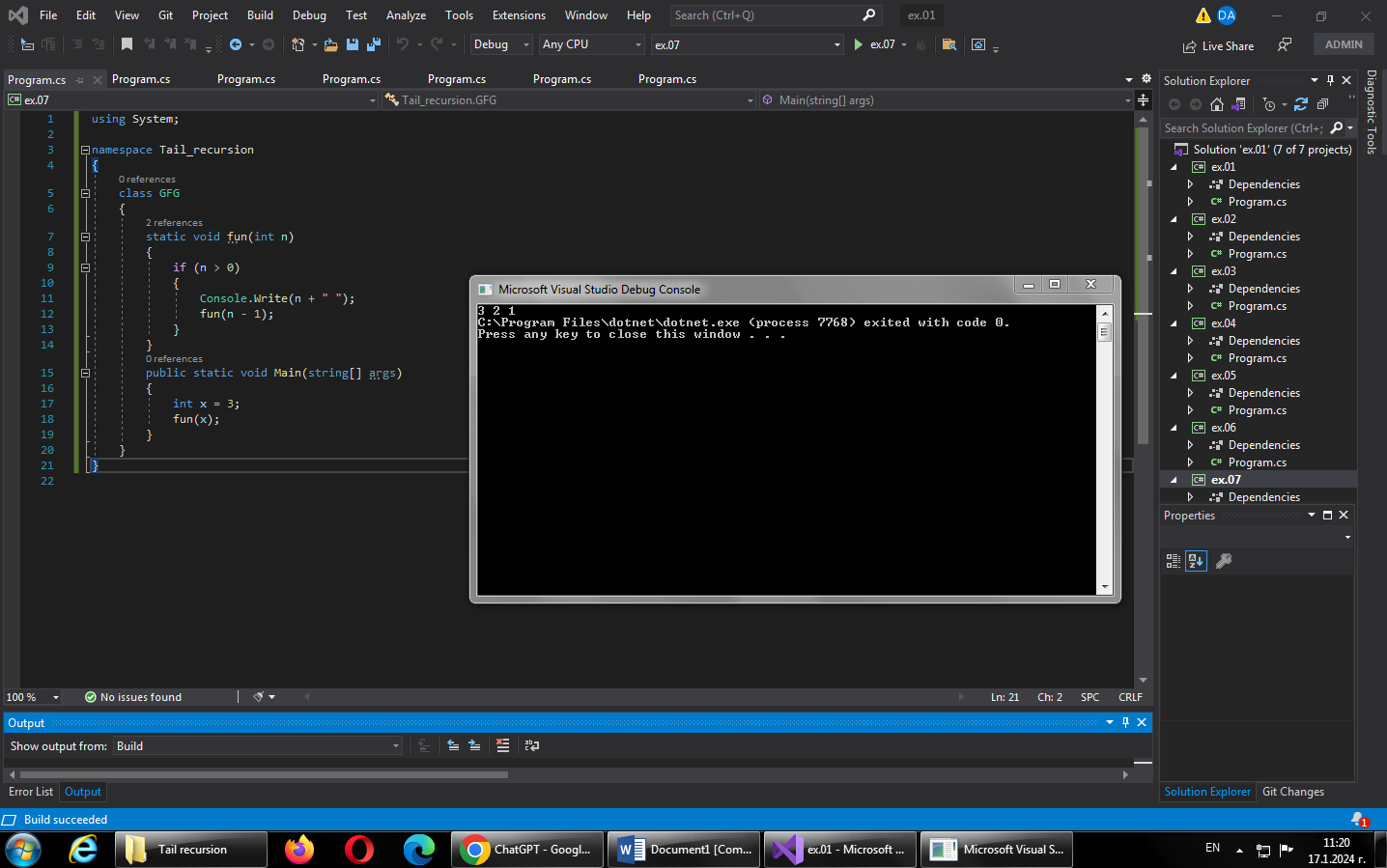


8.Опашкова рекурсия - примерен програмен код чрез ФП

- В C# езикът по подразбиране не поддържа опашкова рекурсия на същата степен, както функционалните езици като Haskell. Но въпреки това, можем да напишем код, който използва подобен подход, използвайки допълнителен параметър за аккумулатор.

Задача: Пример на опашкова рекурсия в C#

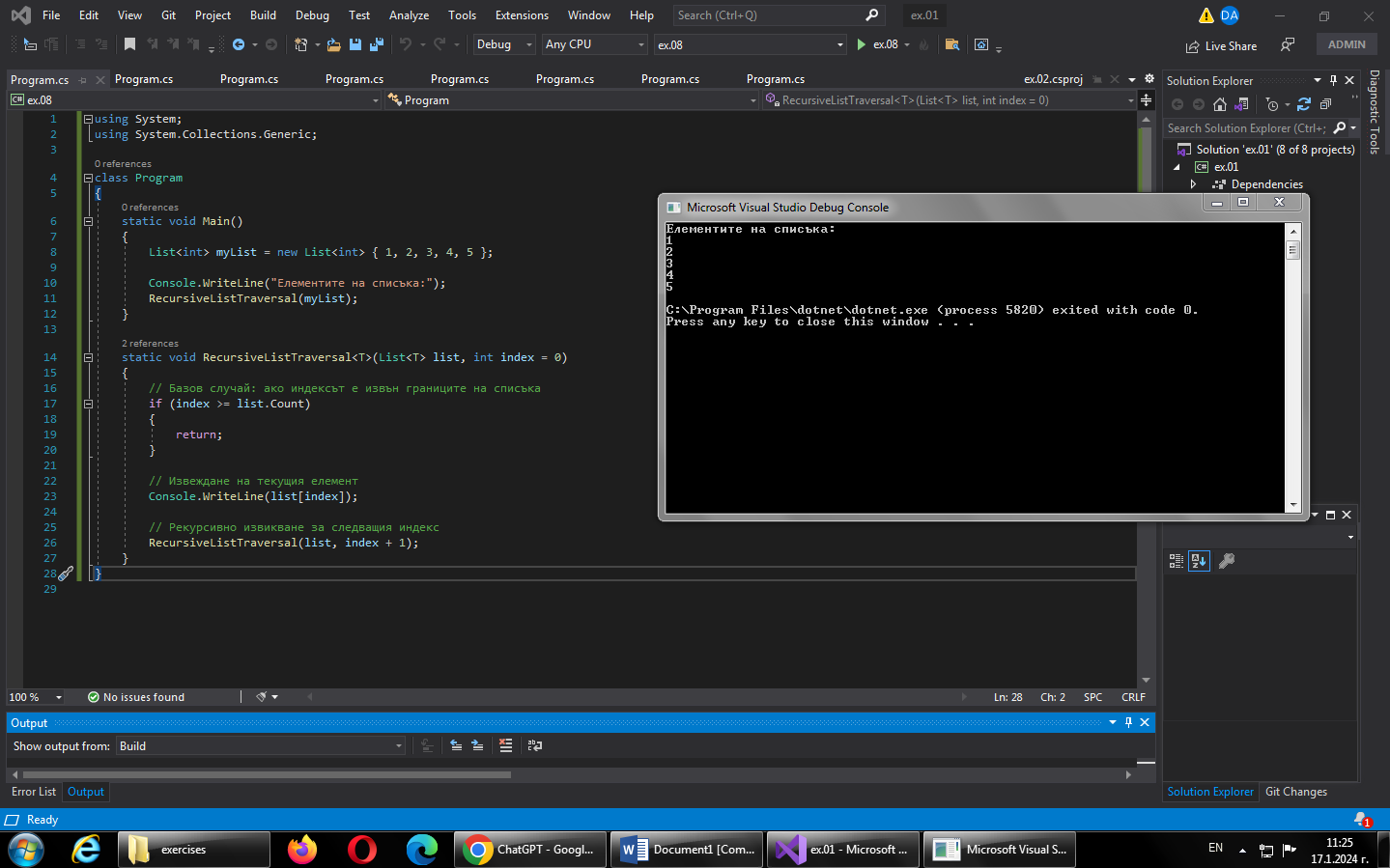
Изход: 3 2 1



9.Рекурсивно обхождане на списък - примерен програмен код чрез ФП

Задача: В следния пример е показана рекурсивна функция за обхождане на елементите на списък в C#:

Изход: Елементите на списъка: 1 2 3 4 5



В този пример RecursiveListTraversal е функция, която обхожда списъка рекурсивно. Ако индексът е валиден (по-малък от броя на елементите в списъка), функцията извежда елемента на текущата позиция и рекурсивно извиква себе си с увеличен индекс. Този процес продължава докато не бъдат обходени всички елементи на списъка.

Източници:

<https://vinss.files.wordpress.com/2010/11/19-1-funkcionalnoprogramiranerevisedbynisheva.pdf>

<https://bg.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5>

<https://skillception.com/blog/%D0%BF%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE-%D1%80%D1%8A%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-%D0%BF%D0%BE-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD/>

<https://azprogramiram.bg/%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B8-%D0%BF%D0%BE-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8/>

Изготвил: Валентин Кьосев 12 ,,a’’ клас