Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Звіт**

з лабораторної роботи № 6 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження рекурсивних алгоритмів»   
Варіант 22

Виконав студент ІП-13, Музичук Віталій Андрійович

(шифр, прізвище, ім'я, по батькові)

Перевірила Вєчерковська Анастасія Сергіївна

( прізвище, ім'я, по батькові)

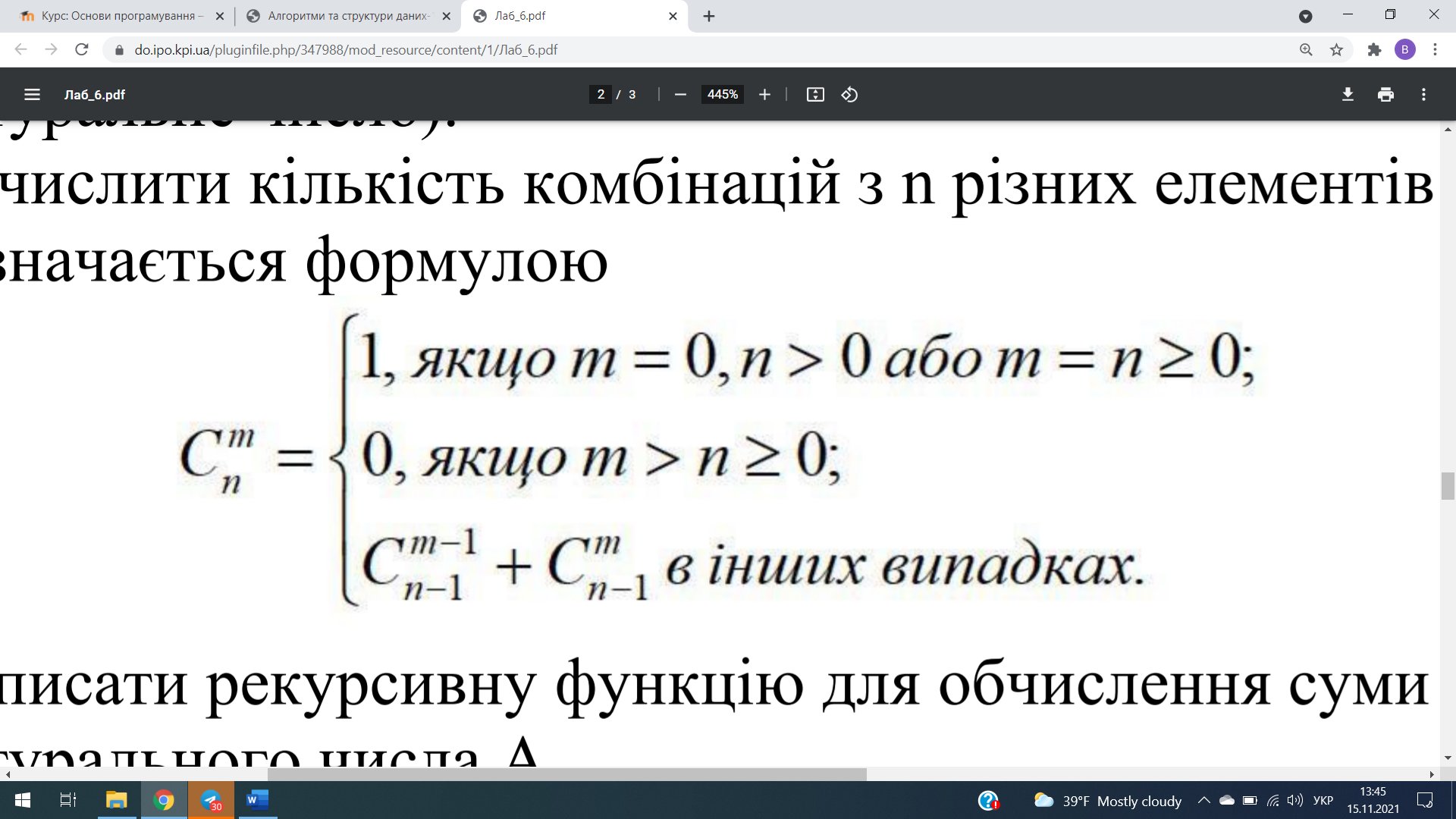
Київ 2021

**Лабораторна робота 6  
Дослідження рекурсивних алгоритмів**

**Мета** – дослідити особливості роботи рекурсивних алгоритмів та набути практичних навичок їх використання під час складання програмних специфікацій підпрограм.

**Варіант 22**

Обчислити кількість комбінацій з n різних елементів по m. Кількість комбінацій визначається формулою:



**Постановка задачі**

За допомогою рекурсивної функції обрахувати значення комбінацій перестановки n елементів по m елементів. Якщо m = 0, n > 0 або m = n ≥ 0, то функція повертає значення 1, або якщо m > n ≥ 0 то функція повертає 0.

Результатом роботи програми є кількість можливих комбінацій перестановки n елементів по m елементів

**Побудова математичної моделі**

Складемо таблицю змінних

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Кількість вибраних елементів | Цілий | *m* | Початкове значення |
| Загальна кількість елементів | Цілий | *n* | Початкове значення |
| Результат | Цілий | *result* | Кінцеве значення |

Розв’язок задачі зводиться до реалізації рекурсивного алгоритму описаного в постановці задачі.

**Combination** – підпрограма, яка обчислює значення комбінацій перестановки n елементів по m елементів;

Дана підпрограма приймає 2 значення: n (загальна кількість елементів) та m (загальна кількість елементів). В алгоритмі присутні дві термінальні та одна рекурсивна гілка. Спочатку перевіряємо термінальні умови якщо **m** = 0, **n** > 0 або **m** = **n** ≥ 0, то функція повертає значення 1, або якщо **m** > **n** ≥ 0 то функція повертає 0. Спочатку робимо перевірку на ці умови, і якщо вони обидві не виконуються, тоді викликаємо суму цих функцій із зменшеними значеннями: **combination**(m - 1, n - 1) + **combination**(m, n - 1). Далі алгоритм повторюється поки не досягне одну із термінальних гілок.

Повернуте значення підпрограми записуємо в змінну **result** і виводимо.

**Розв’язання**

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

*Крок 1*. Визначаємо основні дії  
*Крок 2.* Деталізація дії рекурсивного алгоритму

**Псевдокод**

*крок 1***початок  
введення m, n**деталізація дії рекурсивного алгоритму  
**виведення result**  
**кінець**

*крок 2*

**початок  
введення m, n**result = combination(m, n)  
**виведення result**  
**кінець**

**Псевдокод підпрограми:**

**combination (m, n)**

**якщо** **m** > **n** **І** **n** >= 0  
 **то  
 повернути** 0  
 **інакше**  
 **якщо** (**m** ==0 **І** **n** > 0) **АБО** (**m** == **n І n** >= 0) **то  
 повернути 1  
 інакше  
 повернути** combination(m-1,n-1) + combination(m, n-1)

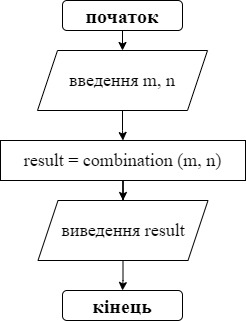
**все якщо  
 все якщо**

**Блок-схема**

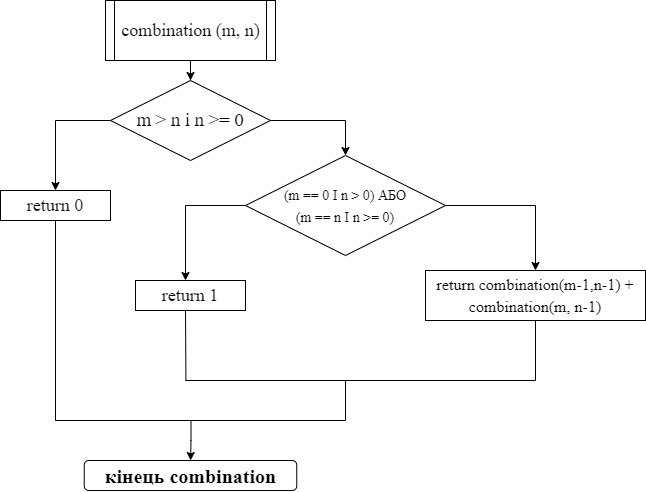
*Крок 1*



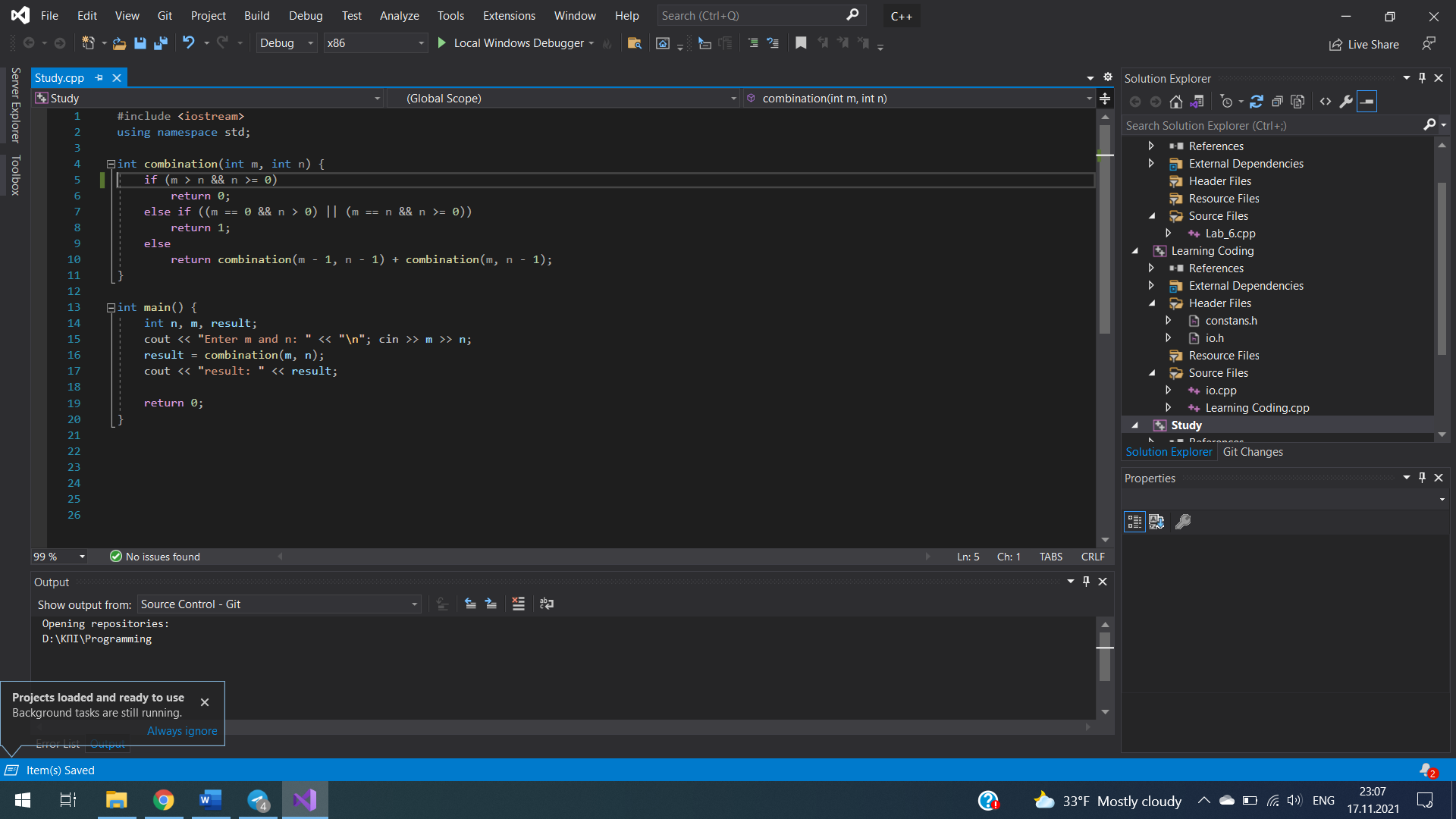
*Крок 2*

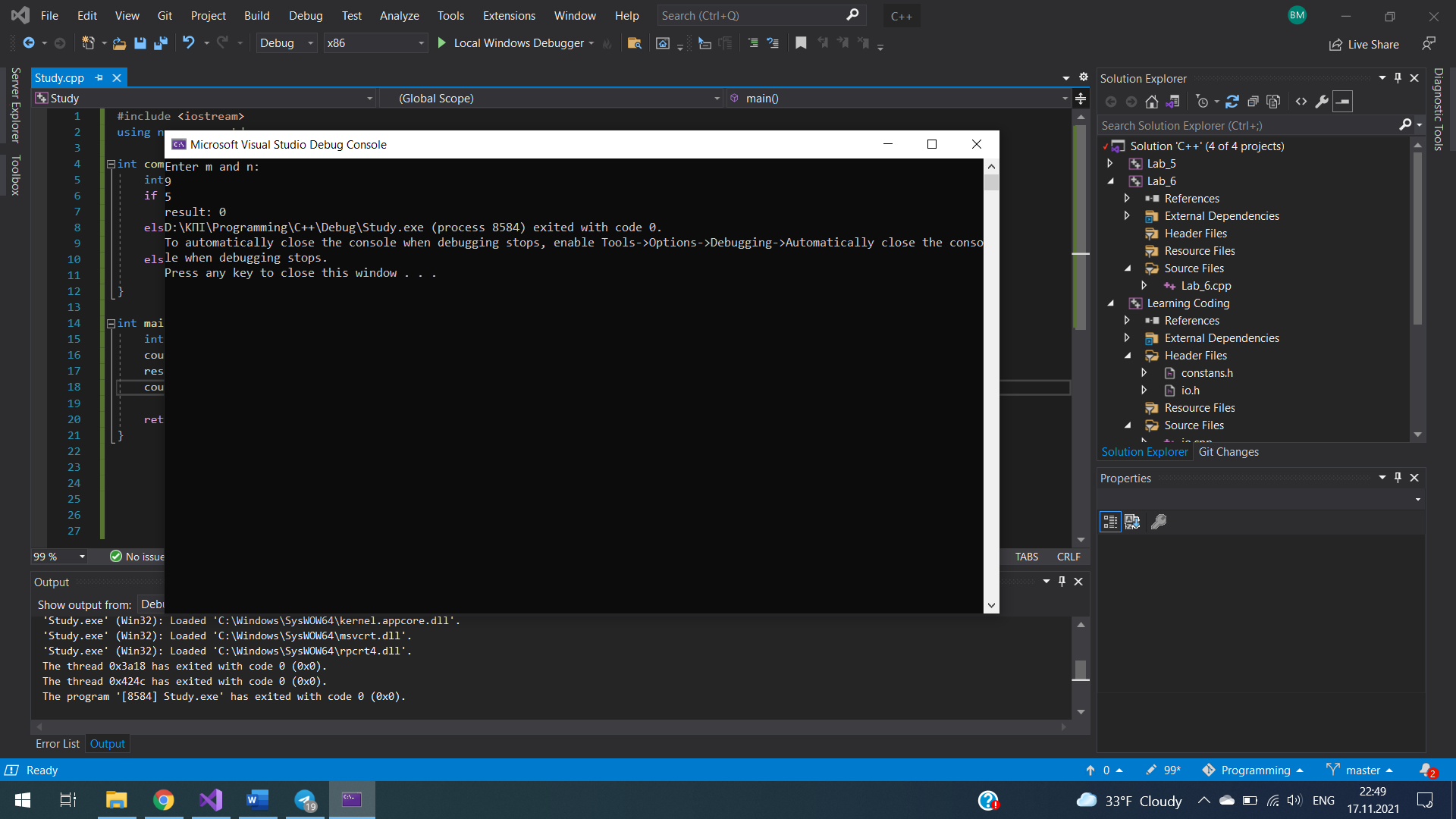


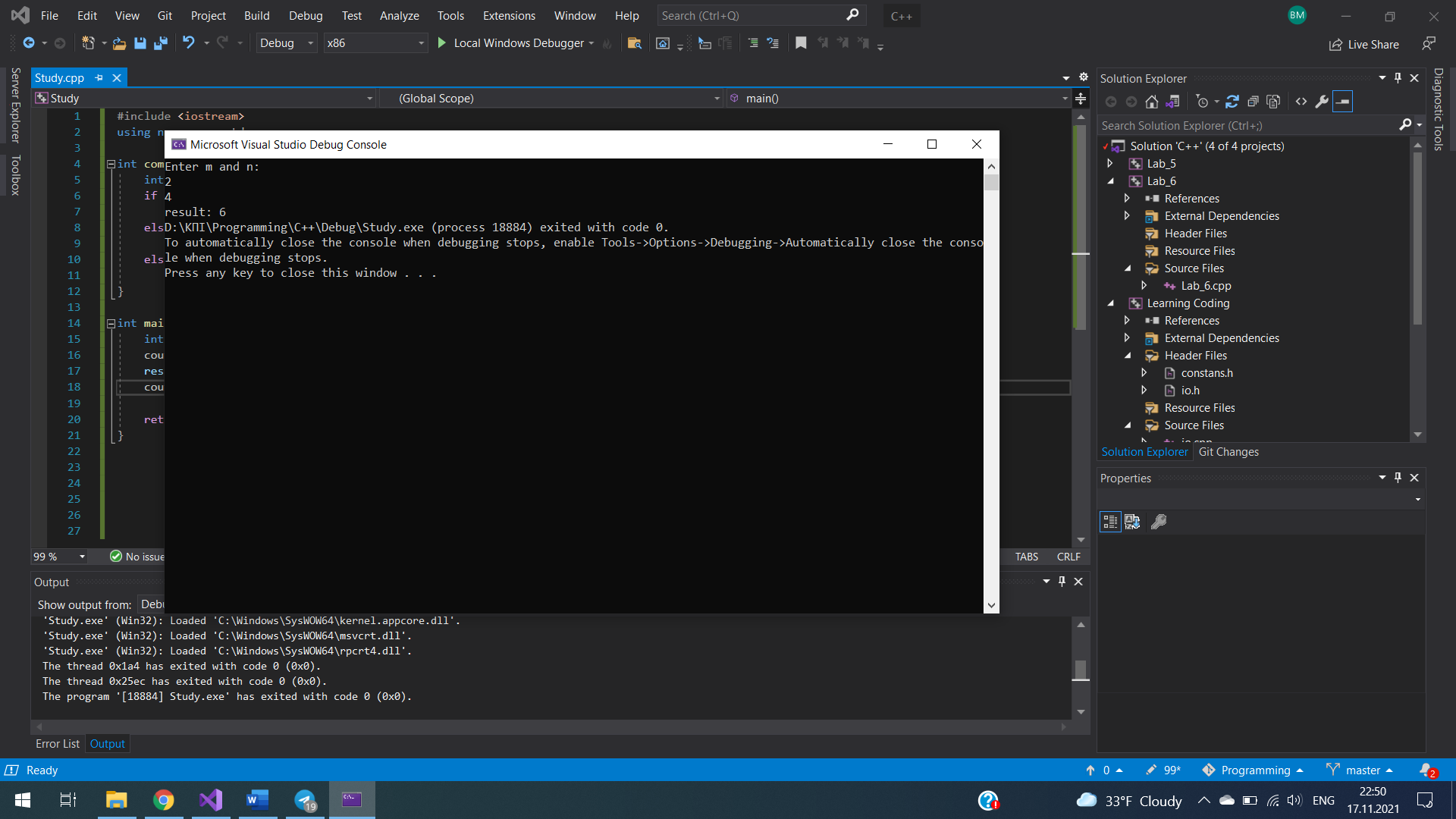
*Блок схема підпрограми:*



Код програми:







**Випробування алгоритму:**

Умовні позначення

|  |  |
| --- | --- |
|  | Межі виконання підпрограми |
|  | Процес виконання другого виклику підпрограми |
|  | Процес виконання третього виклику підпрограми |
|  | Процес виконання четвертого виклику підпрограми |

|  |  |
| --- | --- |
| **Блок** | **Дія** |
|  | Початок |
| 1 | Введення: m = 2, n = 4; |
| 2 | combination (2, 4) |
| 3 | (2 > 4 і 4 >= 0) = false |
| 4 | (2 == 0 i 4 > 0) АБО (2 == 4 i 4 >= 0) = false |
| 5 | **return** combination (1, 3) + combination(2, 3) |
| 6 | combination (1, 3) |
| 7 | (1 > 3 і 3 >= 0) = false |
| 8 | (1 == 0 i 3 > 0) АБО (1 == 3 i 3 >= 0) = false |
| 9 | **return** combination (0, 2) + combination(1, 2) |
| 10 | combination (0, 2) |
| 11 | (0 > 2 і 2 >= 0) = false |
| 12 | (0 == 0 i 2 > 0) АБО (0 == 2 i 2 >= 0) = true |
| 13 | **return 1** |
| 14 | combination (1, 2) |
| 15 | (1 > 2 і 2 >= 0) = false |
| 16 | (1 == 0 i 2 > 0) АБО (1 == 2 i 2 >= 0) = false |
| 17 | **return** combination (0, 1) + combination(1, 1) |
| 18 | combination (0, 1) |
| 19 | (0 > 1 і 1 >= 0) = false |
| 20 | (0 == 0 i 1 > 0) АБО (0 == 1 i 1 >= 0) = true |
| 21 | **return** **1** |
| 22 | combination (1, 1) |
| 23 | (1 > 1 і 1 >= 0) = false |
| 24 | (1 == 0 i 1 > 0) АБО (1 == 1 i 1 >= 0) = true |
| 25 | **return** **1** |
| 26 | combination (2, 3) |
| 27 | (2 > 3 і 3 >= 0) = false |
| 28 | (2 == 0 i 3 > 0) АБО (2 == 3 i 3 >= 0) = false |
| 29 | **return** combination(1, 2) + combination (2, 2) |
| 30 | combination(1, 2) |
| 31 | **return 2** – з попередніх обрахувань |
| 32 | combination (2, 2) |
| 33 | (2 > 2 і 2 >= 0) = false |
| 34 | (2 == 0 i 2 > 0) АБО (2 == 2 i 2 >= 0) = true |
| 35 | **return** **1** |
| 36 | combination (2, 4) |
| 37 | **return** 3 + 3 = 6 |
| 38 | result = 6 |
|  | виведення 6 |
|  | **Кінець** |

**Висновки:**

На цій практичній ми дослідили особливості роботи рекурсивних алгоритмів та набули практичних навичок їх під час складання програмних специфікацій підпрограм. В майбутньому це дозволить нам реалізовувати більш складні математичні задачі за допомогою різних мов програмування.