

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)

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:

$$C_n^m = \begin{cases} 1, & \text{якщо } m = 0, n > 0 \text{ або } m = n \geq 0; \\ 0, & \text{якщо } m > n \geq 0; \\ C_{n-1}^{m-1} + C_{n-1}^m & \text{в інших випадках.} \end{cases}$$

Постановка задачі

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

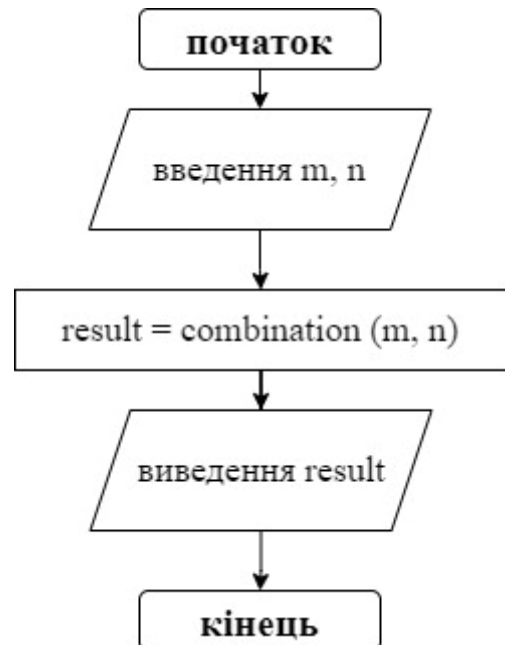
все якщо
все якщо
```

Блок-схема

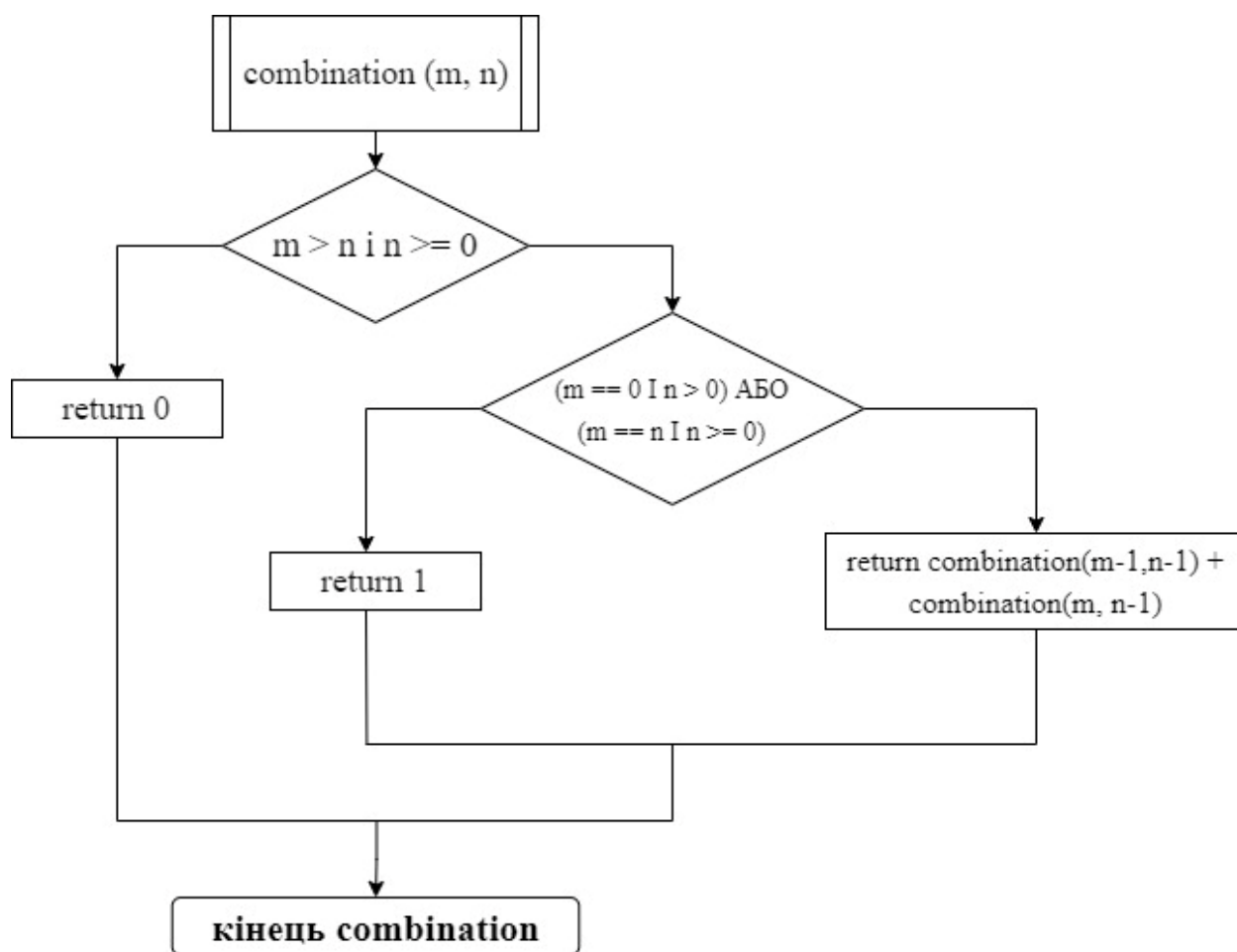
Крок 1



Крок 2



Блок схема підпрограми:




Код програми:

```
#include <iostream>
using namespace std;


int combination(int m, int n) {
    if (m > n && n >= 0)
        return 0;
    else if ((m == 0 && n > 0) || (m == n && n >= 0))
        return 1;
    else
        return combination(m - 1, n - 1) + combination(m, n - 1);
}

int main() {
    int n, m, result;
    cout << "Enter m and n: " << "\n"; cin >> m >> n;
    result = combination(m, n);
    cout << "result: " << result;

    return 0;
}
```

 Microsoft Visual Studio Debug Console

```
Enter m and n:  
9  
5  
result: 0  
D:\КПІ\Programming\C++\Debug\Study.exe (process  
To automatically close the console when debugg  
le when debugging stops.  
Press any key to close this window . . .
```

 Microsoft Visual Studio Debug Console

```
Enter m and n:  
2  
4  
result: 6  
D:\КПІ\Programming\C++\Debug\Study.exe (p  
To automatically close the console when c  
le when debugging stops.  
Press any key to close this window . . .
```

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	$(2 > 4 \text{ і } 4 \geq 0) = \text{false}$
4	$(2 == 0 \text{ і } 4 > 0) \text{ АБО } (2 == 4 \text{ і } 4 \geq 0) = \text{false}$
5	return combination (1, 3) + combination(2, 3)
6	combination (1, 3)
7	$(1 > 3 \text{ і } 3 \geq 0) = \text{false}$
8	$(1 == 0 \text{ і } 3 > 0) \text{ АБО } (1 == 3 \text{ і } 3 \geq 0) = \text{false}$
9	return combination (0, 2) + combination(1, 2)
10	combination (0, 2)
11	$(0 > 2 \text{ і } 2 \geq 0) = \text{false}$
12	$(0 == 0 \text{ і } 2 > 0) \text{ АБО } (0 == 2 \text{ і } 2 \geq 0) = \text{true}$
13	return 1
14	combination (1, 2)
15	$(1 > 2 \text{ і } 2 \geq 0) = \text{false}$
16	$(1 == 0 \text{ і } 2 > 0) \text{ АБО } (1 == 2 \text{ і } 2 \geq 0) = \text{false}$
17	return combination (0, 1) + combination(1, 1)
18	combination (0, 1)
19	$(0 > 1 \text{ і } 1 \geq 0) = \text{false}$
20	$(0 == 0 \text{ і } 1 > 0) \text{ АБО } (0 == 1 \text{ і } 1 \geq 0) = \text{true}$
21	return 1
22	combination (1, 1)
23	$(1 > 1 \text{ і } 1 \geq 0) = \text{false}$
24	$(1 == 0 \text{ і } 1 > 0) \text{ АБО } (1 == 1 \text{ і } 1 \geq 0) = \text{true}$
25	return 1
26	combination (2, 3)
27	$(2 > 3 \text{ і } 3 \geq 0) = \text{false}$
28	$(2 == 0 \text{ і } 3 > 0) \text{ АБО } (2 == 3 \text{ і } 3 \geq 0) = \text{false}$
29	return combination(1, 2) + combination (2, 2)
30	combination(1, 2)
31	return 2 – з попередніх обрахунків

32	combination (2, 2)
33	(2 > 2 і 2 >= 0) = false
34	(2 == 0 і 2 > 0) АБО (2 == 2 і 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.