

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 \_\_\_\_\_ Мєшков Андрій Ігорович \_\_\_\_\_  
(шифр, прізвище, ім'я, по батькові)

Перевірив \_\_\_\_\_ 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

**Задача.** Обчислити кількість комбінацій з  $n$  різних елементів по  $m$ . Кількість комбінацій визначається формулою

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

**Постанова задачі.** З клавіатури вводиться два числа. Числа перевіряються на невід'ємне значення та на один з трьох випадків формули для вибору подальших математичних дій. Результат буде реалізований рекурентно за допомогою двох підпрограм у вигляді функцій для обчислення кількості комбінацій та для обчислення факторіалу.

**Побудова математичної моделі:** для більшої наочності складемо таблицю імен змінних.

Змінна	Тип	Ім'я	Призначення
Кількість елементів більшої групи	Цілий/Невід'ємний	$n$	Початкові дані
Кількість елементів меншої групи	Цілий/Невід'ємний	$m$	Початкові дані
Перший параметр першої функції	Цілий/Невід'ємний	$n_0$	Проміжні дані
Другий параметр першої функції	Цілий/Невід'ємний	$m_0$	Проміжні дані
Результат першої функції	Цілий/Невід'ємний	$c$	Проміжні дані
Параметр другої функції	Цілий/Невід'ємний	$f$	Проміжні дані
Лічильник циклу	Цілий/Невід'ємний	$i$	Проміжні дані
Результат другої функції	Цілий/Невід'ємний	FACT	Проміжні дані
Кількість комбінацій з $n$ різних елементів по $m$	Цілий/Натуральний	$Cnm$	Результат

Числа перевіряються на невід'ємність – кількість елементів не може бути за менше нуль.

Якщо  $n < 0$  або  $m < 0$  на екран виводиться текст: «Числа повинні бути додатніми».

Далі перевіряються випадки:

1. Якщо  $m=0$ ,  $n>0$  або  $m=n>0$ , то  $C_{nm}=1$ ;
2. Якщо  $m>n>0$ , то  $C_{nm}=0$ ;
3. У інших випадках –  $C_{nm}=C(n-1,m-1)+ C(n-1,m)$ , де  $C(n0,m0)$  – підпрограма, в якій обчислюється результат  $c$  за формулою  $c = fact(n0) / (fact(n0-m0) * fact(m0))$ ,  $fact(f)$  – друга підпрограма, в якій обчислюється факторіал чисел FACT у арифметичному циклі(за таких умов є вірогідність, що не буде обчислений факторіал при занадто великому числі, в залежності від мови програмування і місткості типу змінної).

Кінцевий результат виводиться на екран.

*Розв'язання.* Програмні специфікації запишемо у псевдокоді та у графічній формі у вигляді блок-схеми.

*Крок 1.* Визначимо основні дії.

*Крок 2.* Деталізуємо дію перевірки чисел на від'ємність.

*Крок 3.* Деталізуємо дію перевірки на перший випадок системи.

*Крок 4.* Деталізуємо дію перевірки на другий та третій випадки системи.

## **Псевдокод**

### **Програма**

*Крок 1*

**Початок**

**введення n, m**

**Перевірка**

**чисел на**

**від'ємність та**

**обчислення**

**Кінець**

*Крок 2*

**Початок**

**введення n, m**

**якщо**  $n < 0 \parallel m < 0$

**то**

**виведення:** «Числа повинні бути додатнimi»

**інакше**

**Перевірка на перший випадок системи**

**Кінець**

*Крок 3*

**Початок**

**введення n, m**

**якщо**  $n < 0 \parallel m < 0$

**то**

**виведення:** «Числа повинні бути додатними»

**інакше якщо** ( $m == 0 \&\& n > 0$ )

$\parallel (m == n \&\& n >= 0)$

**то**

$C_{nm} := 1$

**виведення:**  $C_{nm}$

**інакше**

**Перевірка на другий та третій випадки системи**

**Кінець**

*Крок 4*

**Початок**

**введення n, m**

**якщо**  $n < 0 \parallel m < 0$

**то**

**виведення:** «Числа повинні бути додатними»

**інакше якщо** ( $m == 0 \&\& n > 0$ )  $\parallel$

$(m == n \&\& n >= 0)$

**то**

$C_{nm} := 1$

**виведення:**  $C_{nm}$

**інакше якщо**  $m > n \&\& n >= 0$

**то**

$C_{nm} := 0$

**виведення:**  $C_{nm}$

**інакше**

$C_{nm} := C(n-1, m-1) + C(n-1, m)$

**виведення:**  $C_{nm}$

**Кінець**

## Підпрограми

**C(n0, m0):**

**C(n0, m0)**

c:=fact(n0) / (fact(n0-m0)  
\* fact(m0))

**повернути c**

**fact(f):**

**fact(f)**

FACT:=1

**повторити**

**для i від 1 до f**

FACT:=FACT\*i

**все повторити**

**повернути FACT**

## Блок-схема Програма:

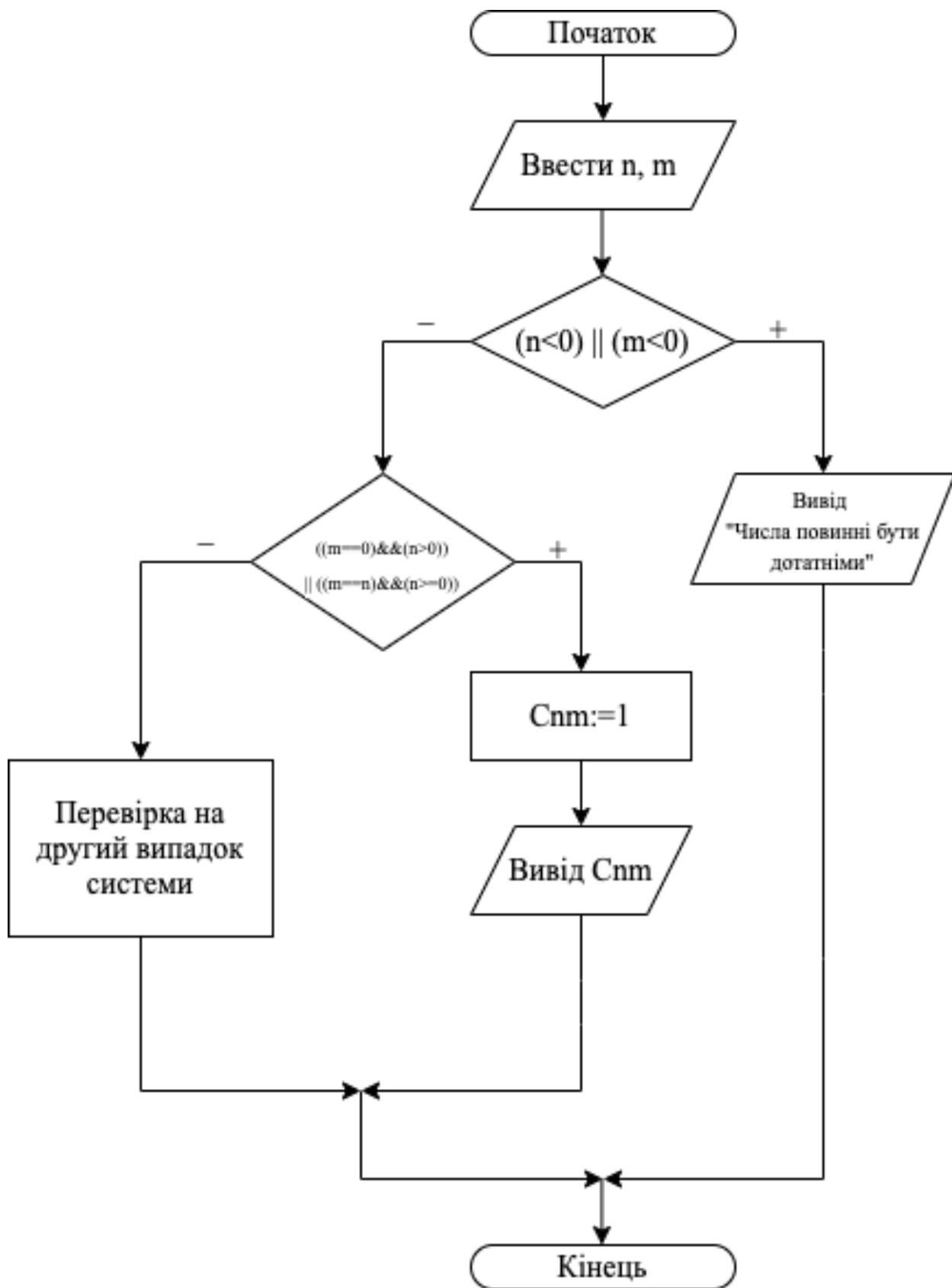
Крок 1



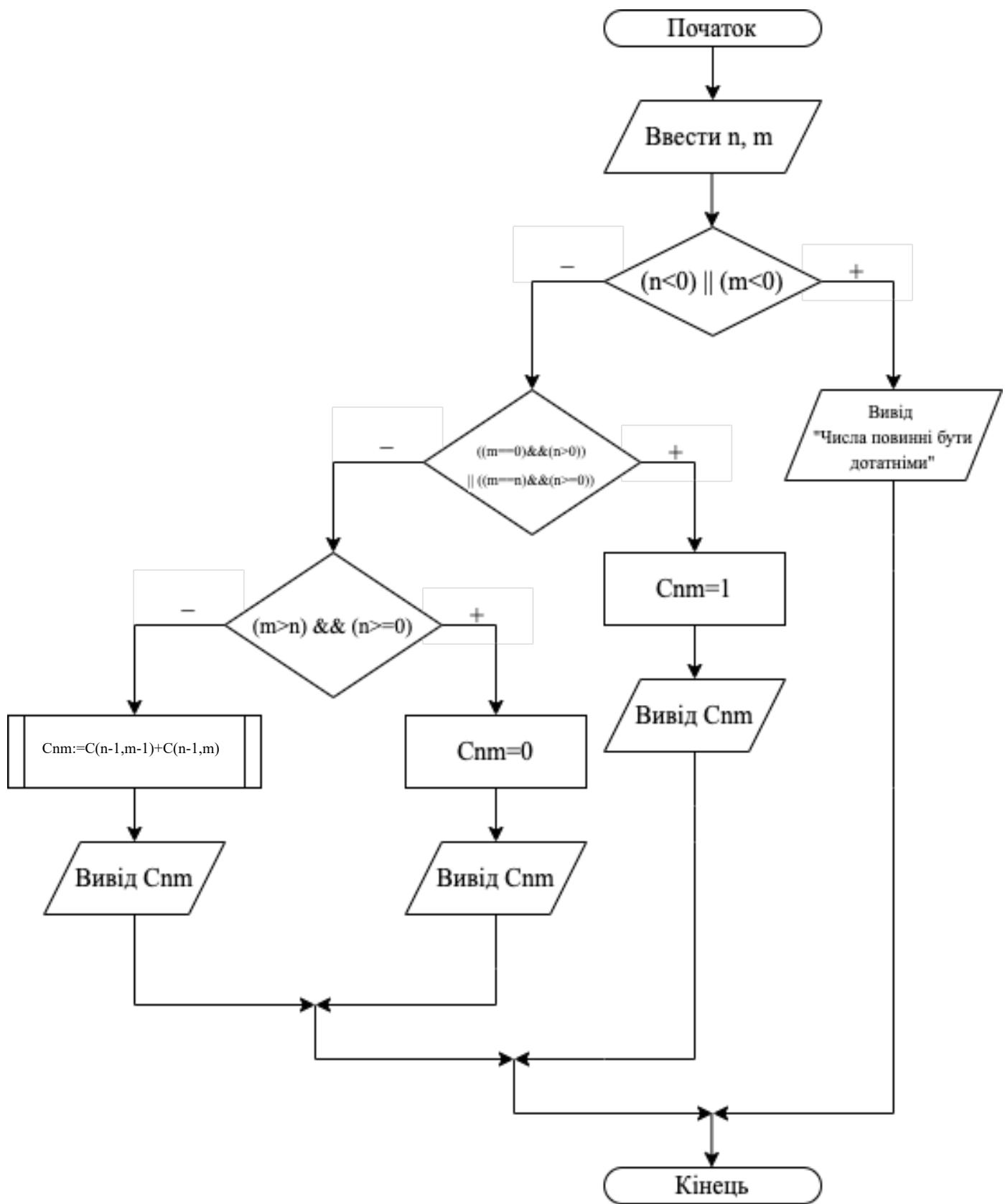
Крок 2



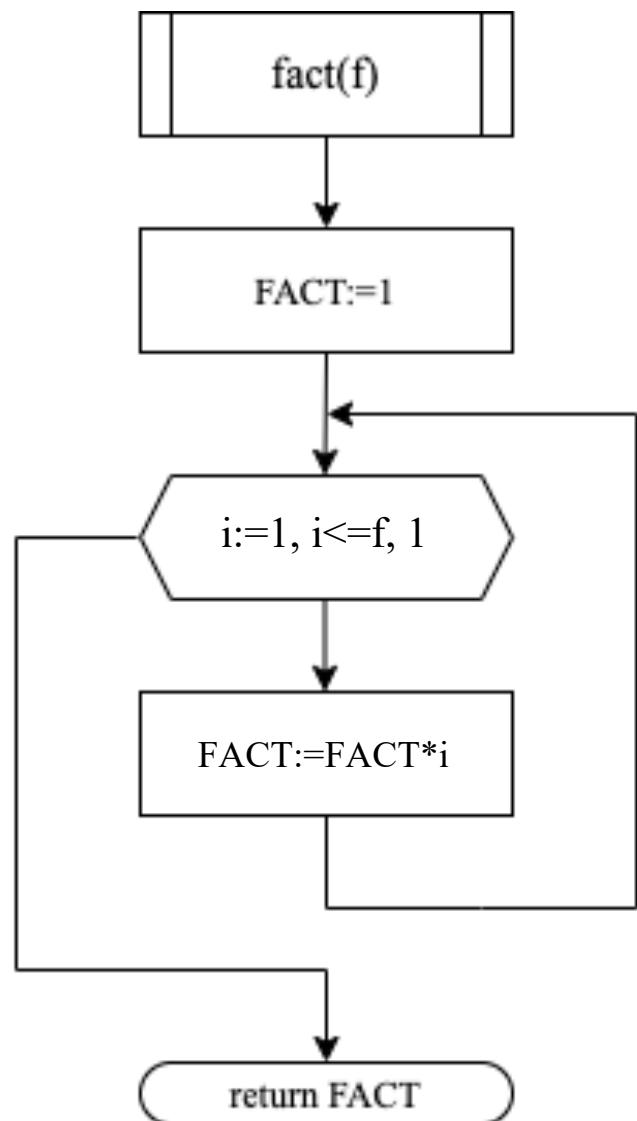
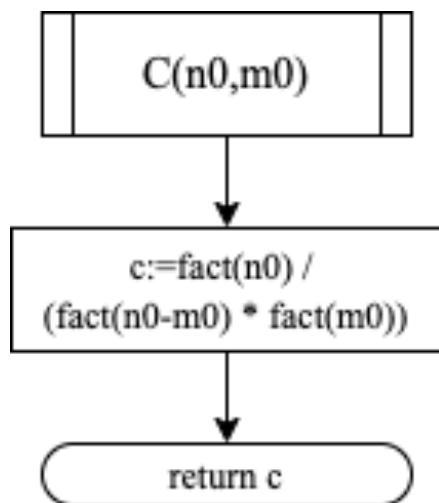
### Крок 3



#### Крок 4



## Підпрограми:



**Випробування алгоритму:** перевіримо правильність алгоритму на довільних конкретних значеннях початкових даних.

### Тест №1

Блок	Дія
1	<b>Початок</b>
2	Введення n=10, m=5
3	$(10 < 0) \parallel (5 < 0)$ - <b>false</b>
4	$((5 == 0) \&\& (10 > 0)) \parallel ((5 == 10) \&\& (10 >= 0))$ - <b>false</b>
5	$(5 > 10) \&\& (10 >= 0)$ – <b>false</b>
6	$C_{nm} := C(10-1, 5-1) + C(10-1, 5) = C(9, 4) + C(9, 5)$
7	Ініціалізація підпрограми C(9, 4)
8	$c := fact(9) / (fact(9-4) * fact(4))$
9	Ініціалізація підпрограми fact(9)
10	FACT:=1
11	<b>Початок арифм. циклу. i=1; i&lt;=9; i++</b>
12	FACT:=1*1=1
13	FACT:=1*2=2
14	FACT:=2*3=6
15	FACT:=6*4=24
16	FACT:=24*5=120
17	FACT:=120*6=720
18	FACT:=1*7=5040
19	FACT:=1*8=40320
20	FACT:=1*9=362880
21	<b>Вихід з арифм. циклу – return 362880</b>
...	...
...	$c := 362880 / (120 * 24) = 126$ – return 126
...	...
...	$C_{nm} := 126 + 126 = 252$
...	Виведення: $C_{nm} = 252$
...	<b>Кінець</b>

## Код программы:

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 long long fact(long long f){
5     long long FACT=1;
6     int m;
7     for(m=1; m <= f; m++){
8         FACT*=m;
9     }
10    return FACT;
11 }
12 long long C(int n0, int m0){
13     long long c;
14     c = fact(n0)/(fact(n0-m0)*fact(m0));
15     return c;
16 }
17 int main ()
18 {
19     int n, m;
20     long long Cnm;
21     cout << "n = ";
22     cin >> n;
23     cout << "m = ";
24     cin >> m;
25
26     if(n<0 || m<0){
27         cout << "n and m must be positive." << endl;
28     }
29
30     else if(((m==0)&&(n>0)) || ((m==n)&&(n>=0))) {
31         Cnm = 1;
32         cout << "Cnm = " << Cnm << endl;
33     }
34
35     else if((m>n)&&(n>=0)){
36         Cnm = 0;
37         cout << "Cnm = " << Cnm << endl;
38     }
39
40     else{
41         Cnm = C(n-1,m-1) + C(n-1,m);
42         cout << "Cnm = " << Cnm << endl;
43     }
44 }
45 }
```

## Tect№1(10, 5)

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 long long fact(long long f){
5     long long FACT=1;
6     int m;
7     for(m=1; m <= f; m++){
8         FACT*=m;
9     }
10    return FACT;
11 }
12 long long C(int n0, int m0){
13     long long c;
14     c = fact(n0)/(fact(n0-m0)*fact(m0));
15     return c;
16 }
17 int main ()
18 {
19     int n, m;
20     long long Cnm;
21     cout << "n = ";
22     cin >> n;
23     cout << "m = ";
24     cin >> m;
25
26     if(n<0 | | m<0){
27         cout << "n and m must be positive." << endl;
28     }
29
30     else if(((m==0)&&(n>0)) | | ((m==n)&&(n>=0))) {
31         Cnm = 1;
32         cout << "Cnm = " << Cnm << endl;
33     }
```

```
n = 10
m = 5
Cnm = 252
Program ended with exit code: 0
```

### Tect№2(-5, 4)

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 long long fact(long long f){
5     long long FACT=1;
6     int m;
7     for(m=1; m <= f; m++){
8         FACT*=m;
9     }
10    return FACT;
11 }
12 long long C(int n0, int m0){
13     long long c;
14     c = fact(n0)/(fact(n0-m0)*fact(m0));
15     return c;
```

n = -5  
m = 4  
n and m must be positive.  
Program ended with exit code: 0

### Tect№3(5, 0)

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 long long fact(long long f){
5     long long FACT=1;
6     int m;
7     for(m=1; m <= f; m++){
8         FACT*=m;
9     }
10    return FACT;
```

n = 5  
m = 0  
 $C_{nm} = 1$   
Program ended with exit code: 0

## Тест№4(0, 6)

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 long long fact(long long f){
5     long long FACT=1;
6     int m;
7     for(m=1; m <= f; m++){
8         FACT*=m;
9     }
10    return FACT;
11 }
```

```
n = 0
m = 6
Cnm = 0
Program ended with exit code: 0
```

**Висновок:** отже, за допомогою підпрограми(рекурсивна функція) було організовано знаходження кількості комбінацій з  $n$  різних елементів по  $m$ . Було досліджено рекурсивні алгоритми, проаналізовано подане завдання, декомпозовано та виконано. Також були розроблені псевдокод, код програми та блок-схема поставленого алгоритму.