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	n0	Проміжні дані
Другий параметр першої функції	Цілий/Невід'ємний	m0	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 **Cnm**=1;
2. Якщо **m**>**n**>=0, то **Cnm**=0;
3. У інших випадках – **Cnm**=*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 || m<0
то
виведення: «Числа повинні бути додатніми»
інакше
Перевірка на перший випадок системи
Кінець

Крок 3

Початок
введення n, m
якщо n<0 || m<0
то
виведення: «Числа повинні бути додатними»
інакше якщо (m==0 && n>0) || (m==n && n>=0)
то
Cnm:=1
виведення: Cnm
інакше
Перевірка на другий та третій випадки системи
Кінець

Крок 4

Початок
введення n, m
якщо n<0 || m<0
то
виведення: «Числа повинні бути додатними»
інакше якщо (m==0 && n>0) || (m==n && n>=0)
то
Cnm:=1
виведення: Cnm
інакше якщо m>n && n>=0
то
Cnm:=0
виведення: Cnm
інакше
Cnm:=C(n-1,m-1)+C(n-1,m)
виведення: Cnm
Кінець

Підпрограми

C(n0, m0):

```
C(n0, m0)
   c:=fact(n0) / (fact(n0-m0)
   * fact(m0))
повернути c
```

fact(f):

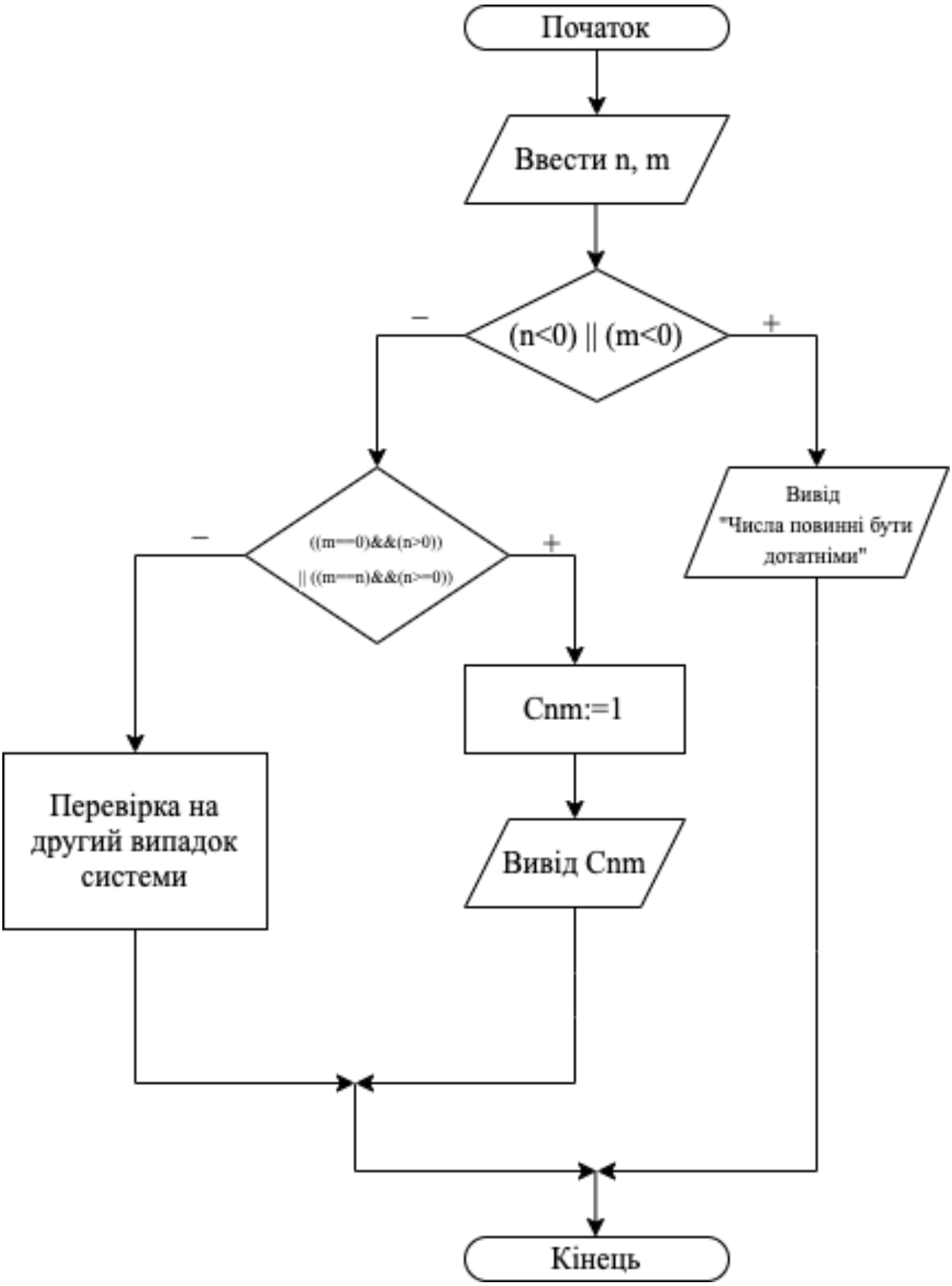
```
fact(f)
   FACT:=1
   повторити
      для і від 1 до f
         FACT:=FACT*i
   все повторити
повернути FACT
```

Блок-схема
Програма:

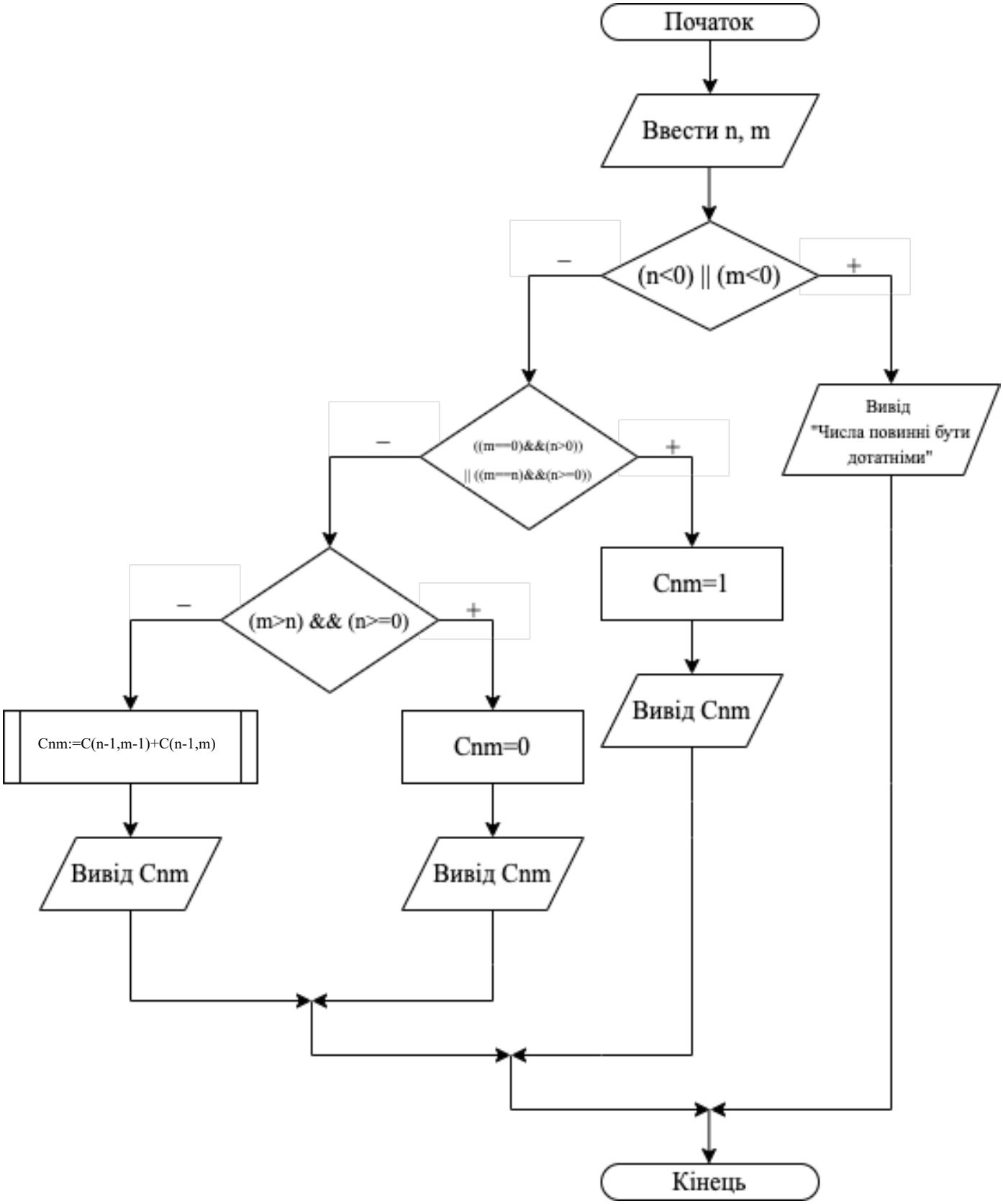
Крок 1

Крок 2

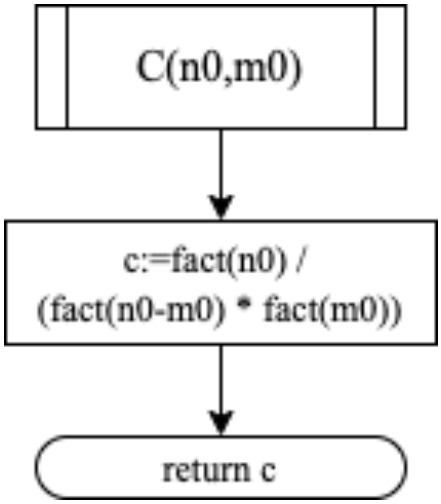
Крок 3

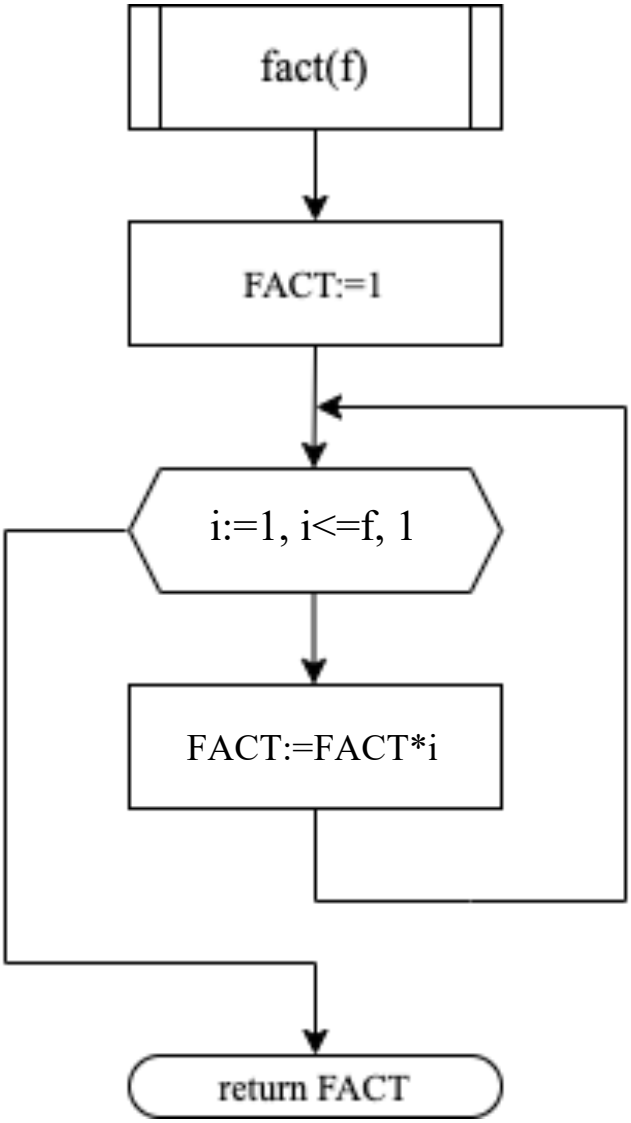

Початок
Ввести n, m
(n<0) || (m<0)
–
+
((m==0)&&(n>0))
|| ((m==n)&&(n>=0))
–
+
Вивід
"Числа повинні бути
дотатніми"
Cnm:=1
Перевірка на
другий випадок
системи
Вивід Cnm
Кінець

Крок 4


Початок
Ввести n, m
(n<0) || (m<0)
–
+
Вивід
"Числа повинні бути
дотатніми"
((m==0)&&(n>0))
|| ((m==n)&&(n>=0))
–
+
Cnm=1
(m>n) && (n>=0)
–
+
Вивід Cnm
Cnm:=C(n-1,m-1)+C(n-1,m)
Cnm=0
Вивід Cnm
Вивід Cnm
Кінець

Підпрограми:





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

Тест№1

Блок	Дія
1	**Початок**
2	Введення n=10, m=5
3	(10<0) || (5<0) - **false**
4	((5==0)&&(10>0)) || ((5==10)&&(10>=0)) - **false**
5	(5>10)&&(10>=0) – **false**
6	Cnm:=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	**Початок арифм. циклу. i=1; i<=9; i++**
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	**Вихід з арифм. циклу –** return 362880
…	…
…	c:=362880/(120*4)=126 – return 126
…	…
…	Cnm:=126 + 126= 252
…	Виведення: Cnm=252
…	**Кінець**

Код програми:

```cpp
#include <iostream>
#include <cmath>
using namespace std;
long long fact(long long f){
    long long FACT=1;
    int m;
    for(m=1; m <= f; m++){
        FACT*=m;
    }
    return FACT;
}
long long C(int n0, int m0){
    long long c;
    c = fact(n0)/(fact(n0-m0)*fact(m0));
    return c;
}
int main ()
{
    int n, m;
    long long Cnm;
    cout << "n = ";
    cin >> n;
    cout << "m = ";
    cin >> m;

    if(n<0||m<0){
        cout << "n and m must be positive." << endl;
    }

    else if(((m==0)&&(n>0))||((m==n)&&(n>=0))) {
        Cnm = 1;
        cout << "Cnm = " << Cnm << endl;
    }

    else if((m>n)&&(n>=0)){
        Cnm = 0;
        cout << "Cnm = " << Cnm << endl;
    }

    else{
        Cnm = C(n-1,m-1) + C(n-1,m);
        cout << "Cnm = " << Cnm << endl;
    }
}

```

Тест№1(10, 5)

```cpp
#include <iostream>
#include <cmath>
using namespace std;
long long fact(long long f){
    long long FACT=1;
    int m;
    for(m=1; m <= f; m++){
        FACT*=m;
    }
    return FACT;
}
long long C(int n0, int m0){
    long long c;
    c = fact(n0)/(fact(n0-m0)*fact(m0));
    return c;
}
int main ()
{
    int n, m;
    long long Cnm;
    cout << "n = ";
    cin >> n;
    cout << "m = ";
    cin >> m;

    if(n<0||m<0){
        cout << "n and m must be positive." << endl;
    }

    else if(((m==0)&&(n>0))||((m==n)&&(n>=0))) {
        Cnm = 1;
        cout << "Cnm = " << Cnm << endl;
    }
```

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

Тест№2(-5, 4)

```cpp
#include <iostream>
#include <cmath>
using namespace std;
long long fact(long long f){
    long long FACT=1;
    int m;
    for(m=1; m <= f; m++){
        FACT*=m;
    }
    return FACT;
}
long long C(int n0, int m0){
    long long c;
    c = fact(n0)/(fact(n0-m0)*fact(m0));
    return c;
}
```

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

Тест№3(5, 0)

```cpp
#include <iostream>
#include <cmath>
using namespace std;
long long fact(long long f){
    long long FACT=1;
    int m;
    for(m=1; m <= f; m++){
        FACT*=m;
    }
    return FACT;
```

```
n = 5
m = 0
Cnm = 1
Program ended with exit code: 0
```

Тест№4(0, 6)

```cpp
#include <iostream>
#include <cmath>
using namespace std;
long long fact(long long f){
    long long FACT=1;
    int m;
    for(m=1; m <= f; m++){
        FACT*=m;
    }
    return FACT;
}
```

```
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.