

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 26

Виконав студент ПІ-11 Рябов Юрій Ігорович
(шифр, прізвище, ім'я, по батькові)

Перевірів Мартінова Оксана Петрівна
(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.

Індивідуальне завдання:

Варіант 26

26. Задано натуральне n . Обчислити

$$\sum_{k=m}^n \frac{(-1)^k}{k!} \left(\frac{a_k + 2}{3} \right)^k \quad a_0 = 1, a_k = \sqrt{|4a_{k-1} + 2|}$$

Постановка задачі

Необхідно за допомогою підпрограм з обчислення частин формули обчислити суму для k між заданими цілими невід’ємними числами. Вхідних даних достатньо, результатом виконання алгоритму є значення суми.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім’я	Призначення
Нижня границя суми	цілий	n	Вхідне дане
Верхня границя суми	цілий	m	Вхідне дане
Сума	дійсний	sum	Результат
Ітератор	цілий	k	Проміжне дане
Формальний параметр підпрограми факторіалу	цілий	num	Проміжне дане
Формальний параметр підпрограми a з індексом k	дійсний	a	Проміжне дане

Підпрограми обчислення факторіалу та a з індексом k реалізуємо за допомогою рекурсії, після чого з їх допомогою обчислимо суму за формулою,

використовуючи арифметичну форму оператора повторення.

Розв'язання

Програмні специфікації запишемо у псевдокодi та графічній формi у вигляді блок-схеми.

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо обчислення a з індексом k

Крок 3. Деталізуємо обчислення факторіалу

Псевдокод

Основна програма

Start

repeat

input m, n

while $n < m$ **or** $m < 1$ **or** $n < 1$

end repeat

$sum = 0$

repeat for k **from** m **to** n

$sum += (-1)^k * (recursiveRoot(k) + 2) / 3^k / factorial(k)$

end repeat

output "The sum is ", sum

End

Підпрограми

recursiveRoot(a)

if $a == 0$

then

$result = 1$

else

$result = \sqrt{4 * recursiveRoot(a - 1) + 2}$

end if

return $result$

End

factorial(num)

if num == 0 **or** num == 1

then

result = 1

else

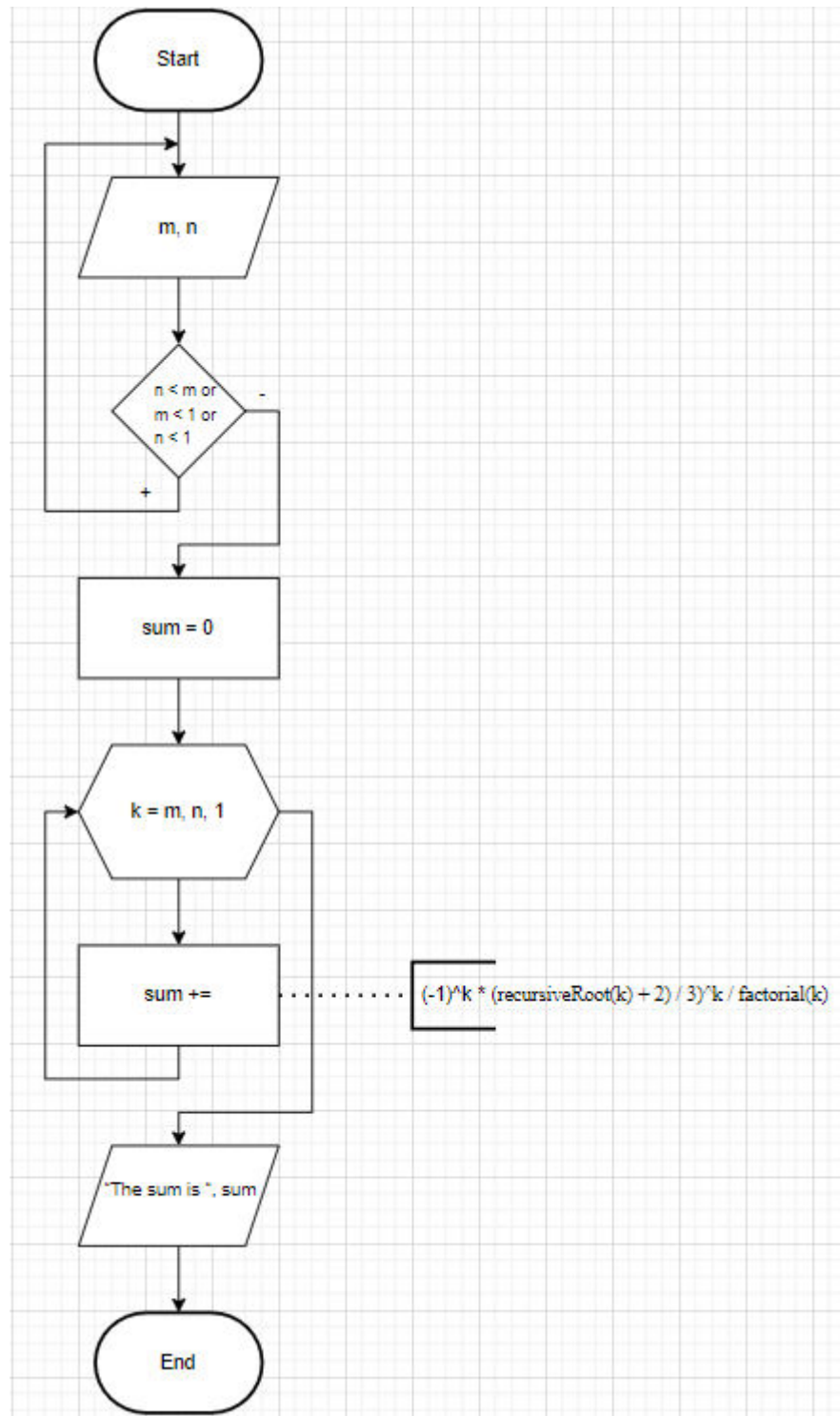
result = num * factorial(num - 1)

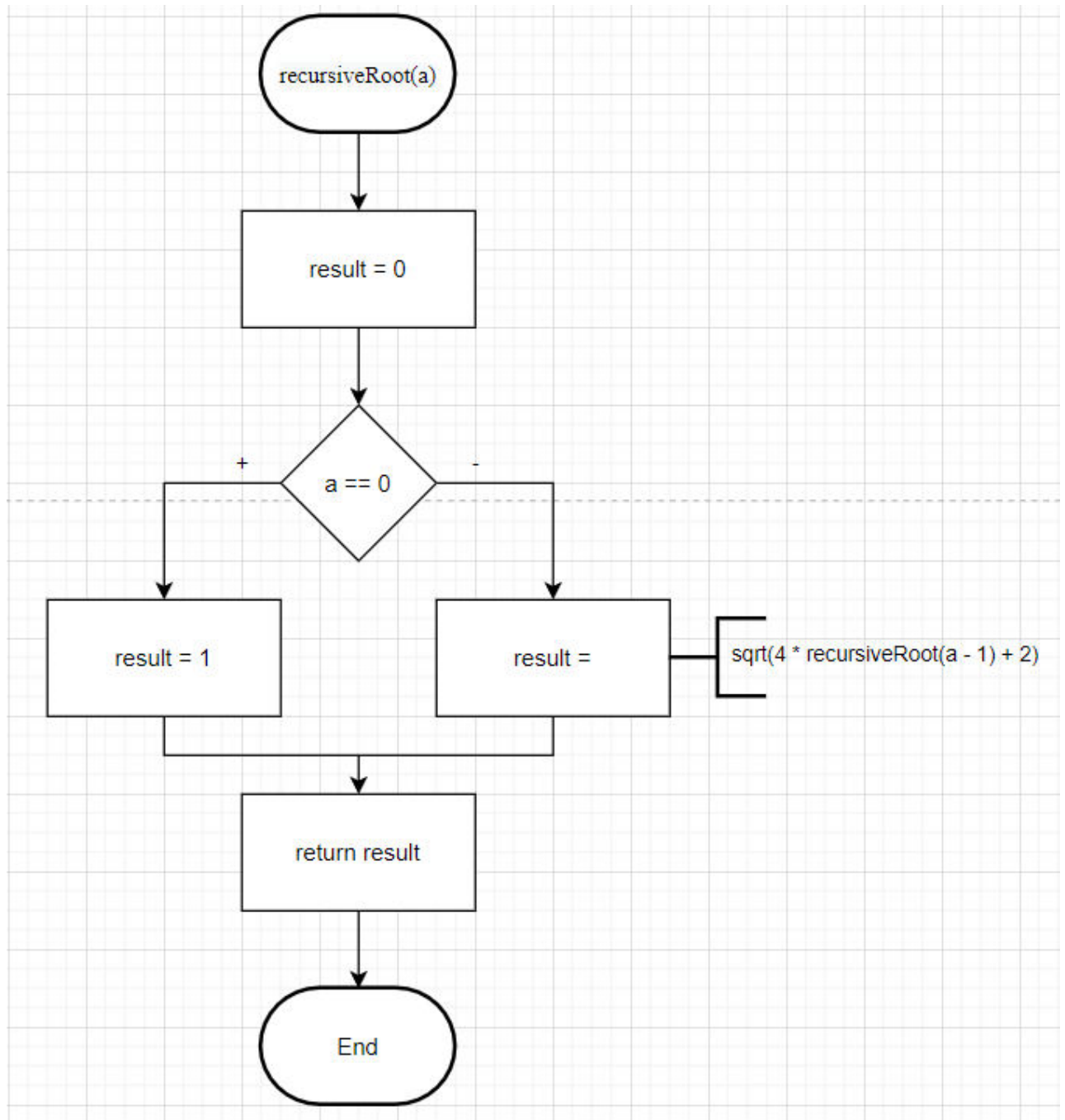
end if

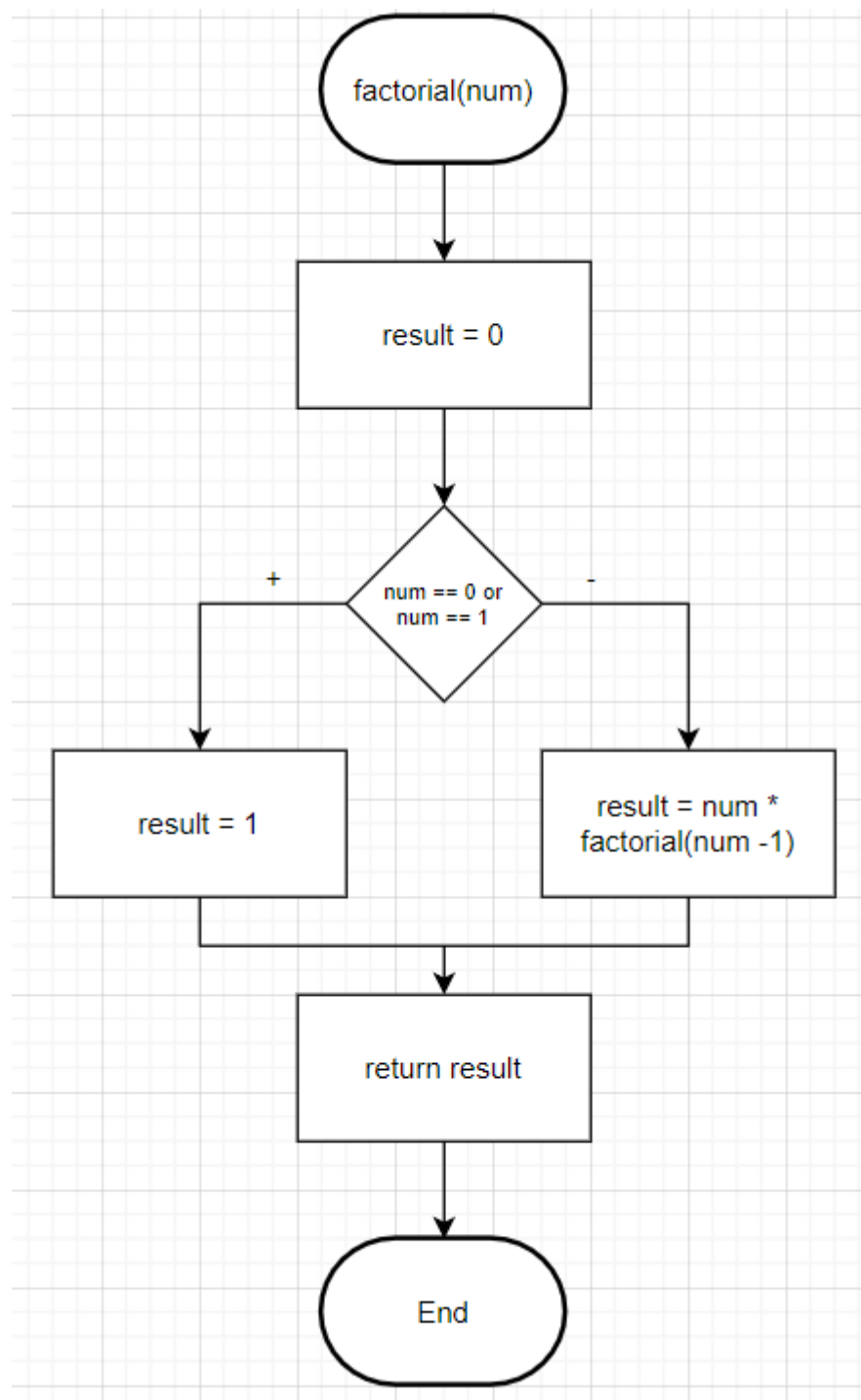
return result

End

Блок-схема







Програма на мові C++

```
#include <iostream>
#include <cmath>

long double recursiveRoot(int a);
long double factorial(int num);

int main()
{
    int m; // Lower bound of the sum, input
    int n; // Upper bound of the sum, input
    long double sum = 0; // Sum, result

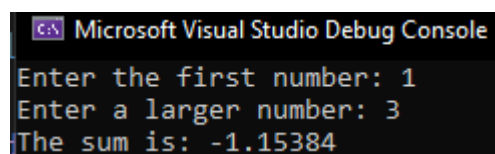
    do {
        std::cout << "Enter the first number: ";
        std::cin >> m;
        std::cout << "Enter a larger number: ";
        std::cin >> n;
    } while (n < m || n < 1 || m < 0);

    for (int k = m; k <= n; k++) {
        sum += pow(-1, k) * pow((recursiveRoot(k) + 2) / 3, k) / factorial(k);
    }

    std::cout << "The sum is: " << sum;
}

// Function which calculates a root which diverges to 2 + sqrt(6) using recursion
long double recursiveRoot(int a)
{
    long double result;
    if (a == 0) {
        result = 1;
    }
    else {
        result = sqrt(4 * recursiveRoot(a - 1) + 2);
    }
    return result;
}

// Function which calculates factorial via recursion
long double factorial(int num)
{
    long double result;
    if (num == 1 || num == 0) {
        result = 1.0;
    }
    else {
        result = (long double)(num * factorial(num - 1));
    }
    return result;
}
```



C:\ Microsoft Visual Studio Debug Console

```
Enter the first number: 1
Enter a larger number: 3
The sum is: -1.15384
```


Перевірка

Блок	Дія
	Початок
1	$m = 1, n = 3$
2	Після 1 ітерації: $\text{recursiveRoot}(1) = 2.449$ $\text{factorial}(1) = 1$ $\text{sum} = -1.483$
3	Після 2 ітерації: $\text{recursiveRoot}(2) = 3.435$ $\text{factorial}(2) = 2$ $\text{sum} = 0.158$
4	Після 3 ітерації: $\text{recursiveRoot}(3) = 3.967$ $\text{factorial}(3) = 6$ $\text{sum} = -1.154$
5	Виведення "The sum is -1.154"
	Кінець

Висновок

Отже, ми дослідили особливості роботи рекурсивних алгоритмів та набули практичних навичок їх використання, склавши алгоритм з обчислення суми з використанням двох підпрограм, що визивають самі себе.