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 складних рекурсивних алгоритмів» «Дослідження складних циклічних алгоритмів»

Варіант 18

Виконав студент: ІП-15 Лазьов Кирило Владиславович

Перевірив: Вечерковська Анастасія Сергіївна

Лабораторна робота 6

Дослідження складних рекурсивних алгоритмів

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

Варіант 18

Постановка задачі

Задано натуральне n. Обчислити $\sum_{k=1}^{n} \frac{a_k - b_k}{k!}$ $a_1 = 1,\ a_k = 0.5\left(\sqrt{b_{k-1}} + 5\sqrt{a_{k-1}}\right)$

$b_1 = 1,\ b_k = 2a_{k-1}^2 + b_{k-1}.$

Змінна	**Тип**	**Ім’я**	**Призначення**
Верхня границя суми	Натуральне	n	Вхідні дані
Індекс суми	Натуральне	k	Проміжні дані
Перший член виразу	Дійсне	a	Проміжні дані
Другий член виразу	Дійсне	b	Проміжні дані
Попереднє значення першого члена	Дійсне	a_prev	Проміжні дані
Попереднє значення другого члена	Дійсне	b_prev	Проміжні дані
Факторіал	Натуральне	fact	Проміжні дані
Лічильник циклу	Натуральне	i	Проміжні дані
Змінна, яку повертає функція	Дійсне	result	Проміжні дані
Сума членів послідовності	Дійсне	Sum	Результат

Розв’язання

n задається користувачем. Значення n, k, a_prev, b_prev - передаються у однойменні параметри, а змінна Sum у параметр result функції Suma, що рахує суму. Термінальна гілка функції виконується коли k>n, тоді функція повертає значення змінної result у змінну Sum. Інакше виконується рекурсивна гілка, яка рахує значення змінних a, b, fact і використовує їх у знаходженні значення змінної result. В кінці рекурсивної гілки значення k збільшується на 1, що в решті призведе до того, що k буду більше за n, тоді виконується термінальна гілка і функція повертає значення result, і основна програма записує його у змінну Sum і виводить її.

Корінь з числа обчислюємо за допомогою функції sqrt. Модуль числа за допомогою функції pow

Крок 1. Визначаємо основні дії

Крок 2. Обчислення Sum за допомогою функції Suma

Псевдокод

Основна програма

Крок 1

Початок

Введення n

k = 1

Sum = 0

a_prev = 0

b_prev = 0

Обчислення Sum за допомогою функції Suma

Виведення Sum

Кінець

Крок 2

Введення n

k = 1

Sum = 0

a_prev = 0

b_prev = 0

Sum = Suma(n, k, a_prev, b_prev, Sum)

Виведення Sum

Кінець

Підпрограма

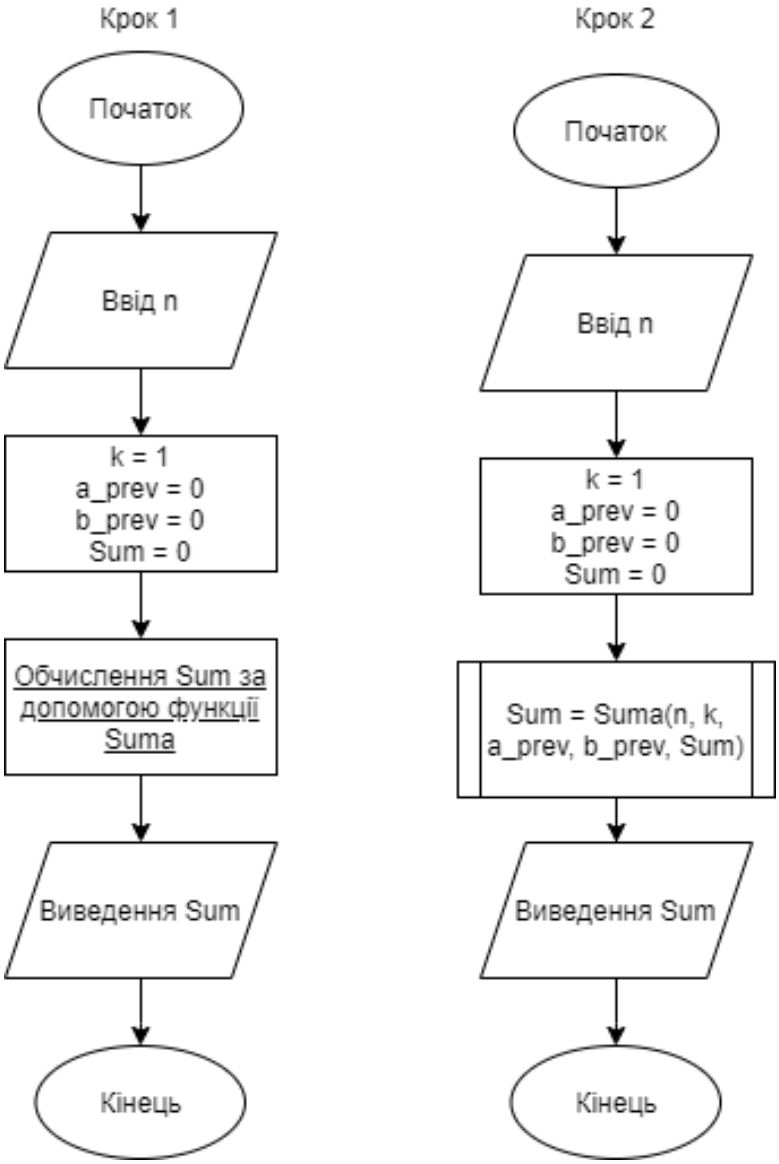
Suma(n, k, a_prev, b_prev, Sum)

Початок

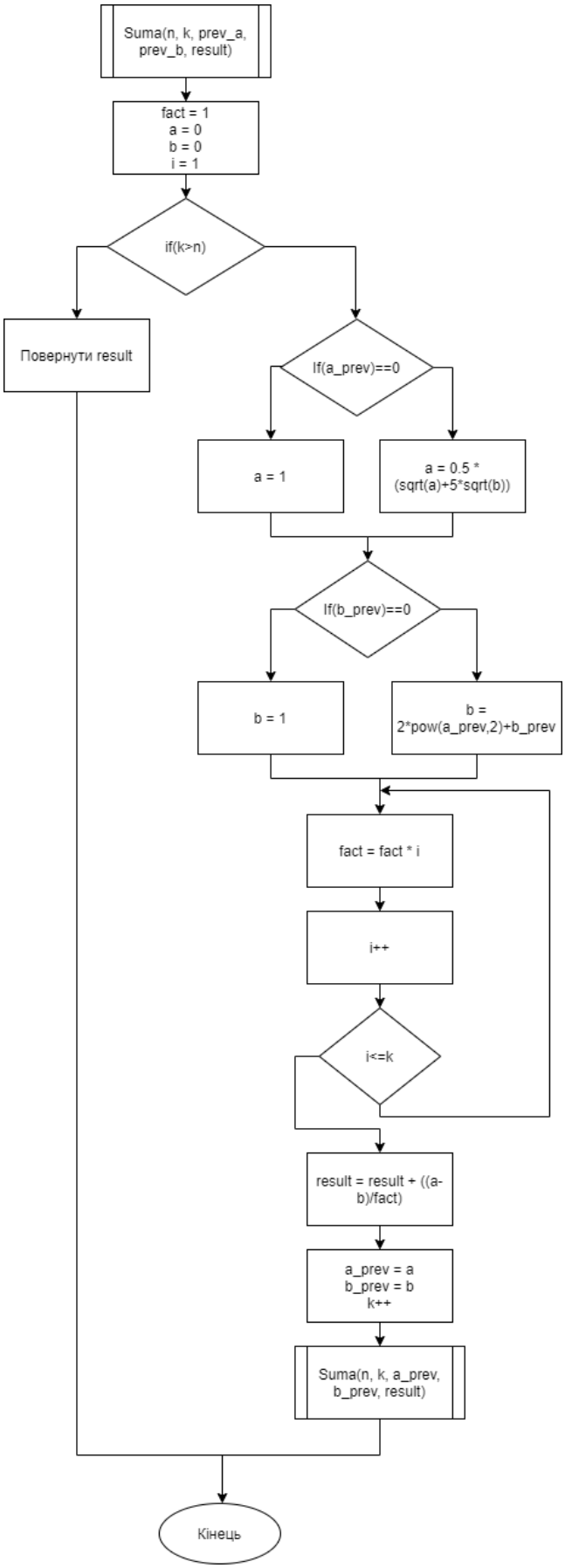
```
fact = 1
a = 0
b = 0
якщо k>n
        то
                повернути result
інакше
                якщо a_prev == 0
                        то
                                a = 1
                        Інакше
                                a  = 0.5*(sqrt(a_prev)+5*(sqrt(b_prev))
                якщо и_prev == 0
                        то
                                b = 1
                        Інакше
                                b = 2* pow(a_prev, 2) + b_prev
                Повторити
                        fact = fact*i
                        i++
                        поки i<=k
                result = result + ((a-b)/fact)
                a_prev = a
                b_prev  = b
                k++
                Suma(n, k, a_prev, b_prev, Sum)
```

Блок-схема

Основна програма



Підпрограма



Код

```cpp
#include <iostream>
#include <math.h>

using namespace std;

float Suma(float n, float k, float a_prev, float b_prev, float result) {
    float fact = 1;
    float a;
    float b;
    if (k > n) {
        return result;
    }
    else {
        if (a_prev == 0) {
            a = 1;
        }
        else {
            a = 0.5 * (sqrt(a_prev) + 5 * sqrt(b_prev));
        }
        if (b_prev == 0) {
            b = 1;
        }
        else {
            b = 2 * pow(a_prev, 2) + b_prev;
        }
        for (int i = 1; i <= k; i++) {
            fact = fact * i;
        }
        result +=  (a - b)/fact;
        a_prev = a;
        b_prev = b;
        k++;
        Suma(n, k, a_prev, b_prev, result);
    }
}

int main() {
    int k = 1;
    float a_prev = 0;
    float b_prev = 0;
    float Sum = 0;
    int n;
    cin >> n;
    Sum = Suma(n, k, a_prev, b_prev, Sum);
    cout << Sum;
}
```

Тестування програми

```
6
Sum = -10.3054

C:\Users\User\Source\Repos\ASD_Lab6\Debug\ASD_Lab6.exe
To automatically close the console when debugging stop
Press any key to close this window . . .
```

Висновок

Ми дослідили особливості роботи рекурсивних алгоритмів та набули практичних навичок їх використання. В результаті виконання лабороторної роботи, був отриман алгоритм, що використовує рекурсивну підпрограму для знаходження суми.