

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Катедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження рекурсивних алгоритмів»

Варіант 18

Виконав студент ІП-11 Лесів Владислав Ігорович

Перевірив Мартинова О.П.

Київ 2021

Лабораторна робота 6

Дослідження рекурсивних алгоритмів

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

Варіант №18.

18. Задано натуральне n . Обчислити $\sum_{k=1}^n \frac{a_k - b_k}{k!}$ $a_1 = 1, a_k = 0.5(\sqrt{b_{k-1}} + 5\sqrt{a_{k-1}}),$
 $b_1 = 1, b_k = 2a_{k-1}^2 + b_{k-1}.$

Постановка задачі. Результатом розв'язку є сума, обрахована за заданою формулою. Для визначення результату повинне бути задане натуральне число n . Інших початкових даних для розв'язку не потрібно.

Для знаходження суми скористаємося підпрограмою. У тілі підпрограми буде відбуватися рекурсія. Умова виходу з рекурсії: лічильник добіг до свого кінця, з умови $k==n$.

Побудова математичної моделі. Складемо таблицю імен змінних.

Змінна	Тип	Ім'я	Призначення
Натуральне число n	Цілий	n	Початкове дане
Змінна факторіалу (у функції)	Цілий	f	Проміжний результат
Обчислена сума	Дійсний	s	Результат

Математичне формулювання задачі зводиться до знаходження суми за заданою формулою шляхом обчислення наступних елементів за допомогою попередніх, підставлення їх у формулу, та додавання підрахованого значення до загальної суми. Коли сума обчислена, виводимо її.

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

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію знаходження заданої в умові задачі суми за допомогою підпрограми.

Крок 3. Деталізуємо дію знаходження факторіалу числа за допомогою підпрограми.

Псевдокод

крок 1

початок

введення n

знаходження суми за допомогою підпрограми

виведення s

кінець

крок 2

початок

введення n

s:=summ(1, n, 1, 1);

виведення s

кінець

підпрограма summ(k, n, a, b)

f:= факторіал (k)

якщо k==n

то

повернути (a - b) / f

інакше

повернути ((a - b) / f) + summ(k + 1, n, 0.5 * (sqrt(b) + 5 * sqrt(a)), 2 * a *
*a + b)

кінець підпрограми

Легенда псевдокоду: факторіал(k) – знаходження факторіалу числа за допомогою підпрограми

крок 3

початок

введення n

s:=summ(1, n, 1, 1);

виведення s

кінець

підпрограма summ(k, n, a, b)

f:= fact(k)

якщо k==n

то

повернути (a - b) / f

інакше

повернути ((a - b) / f) + summ(k + 1, n, 0.5 * (sqrt(b) + 5 * sqrt(a)), 2 * a *
*a + b)

кінець підпрограми

підпрограма fact(c)

якщо c>1

то

повернути c*fact(c-1)

інакше

повернути 1

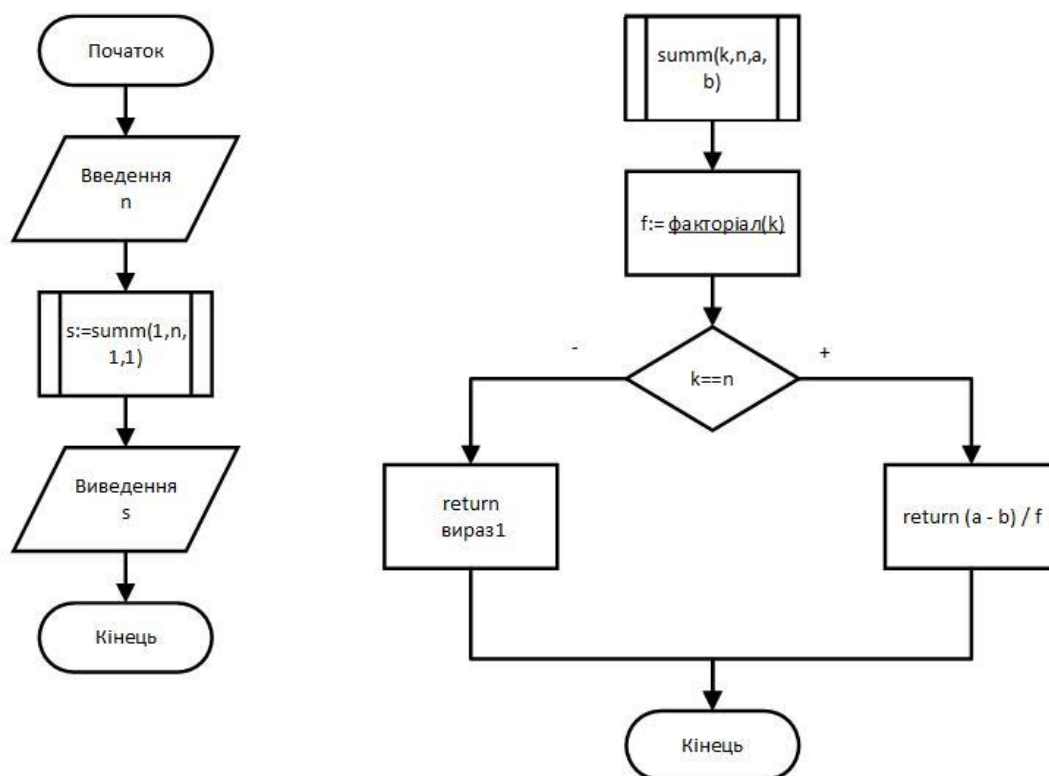
кінець підпрограми

Блок-схема

Крок 1



Крок 2

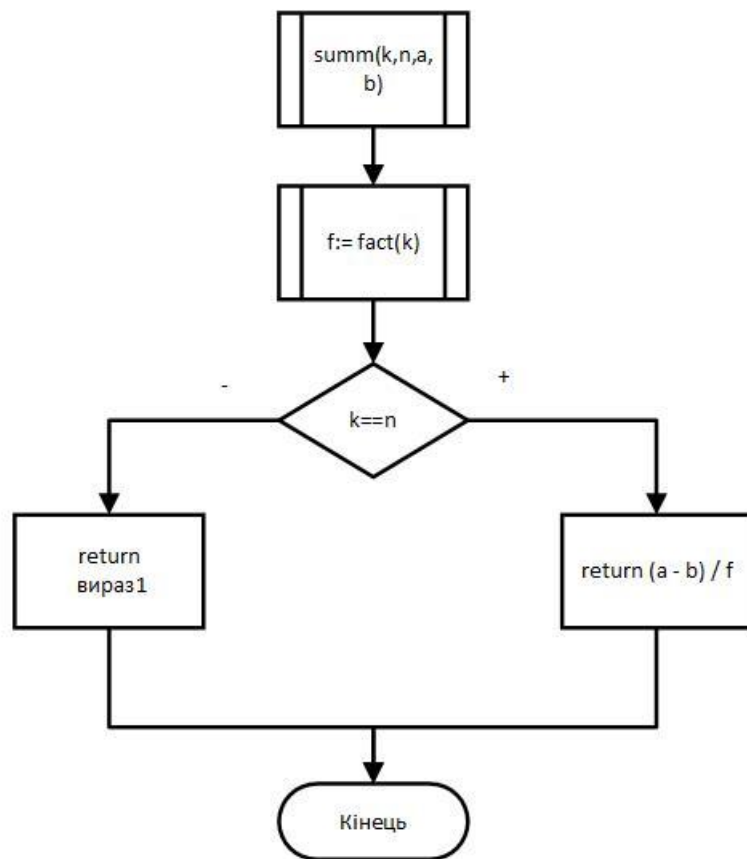
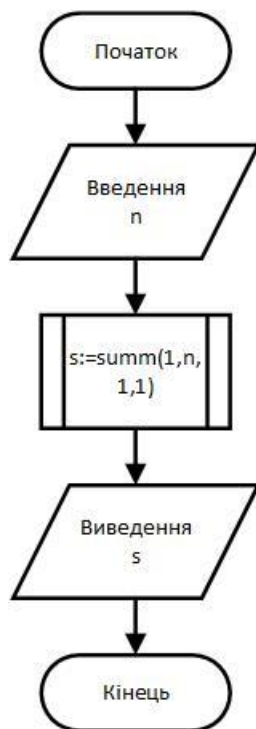


Легенда блок-схеми:

факторіал(k) - знаходження факторіалу числа за допомогою підпрограми

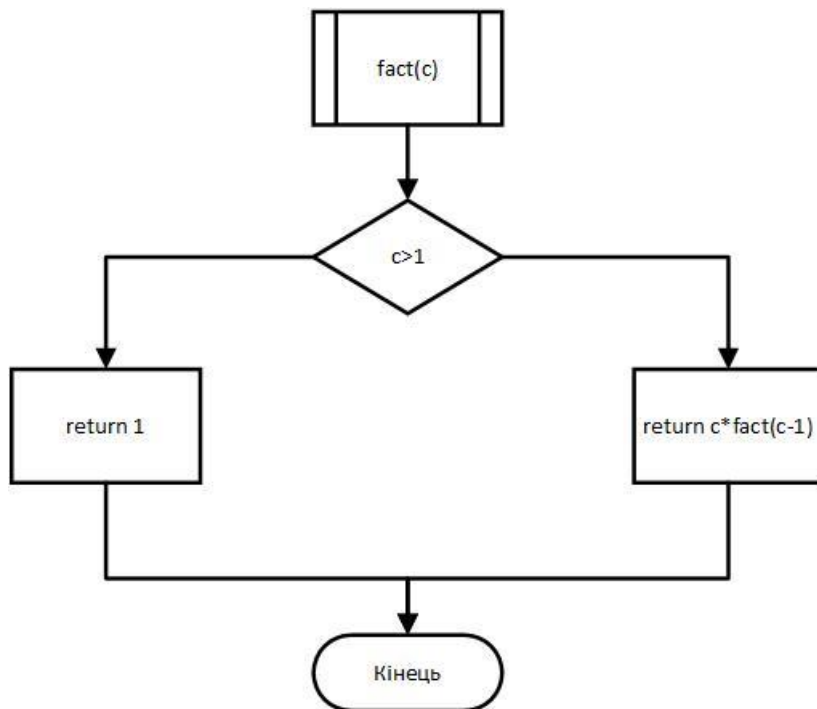
вираз1 - $((a - b) / f) + \text{summ}(k + 1, n, 0.5 * (\text{sqrt}(b) + 5 * \text{sqrt}(a)), 2 * a * a + b)$

Крок 3



Легенда блок-схеми:

вираз1 - $((a - b) / f) + \text{summ}(k + 1, n, 0.5 * (\text{sqrt}(b) + 5 * \text{sqrt}(a)), 2 * a * a + b)$



Виконання мовою C++.

Код програми:

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

using namespace std;

int fact(int); //Прототипи функцій: знаходження факторіалу
double summ(int, int, double, double); //Знаходження суми

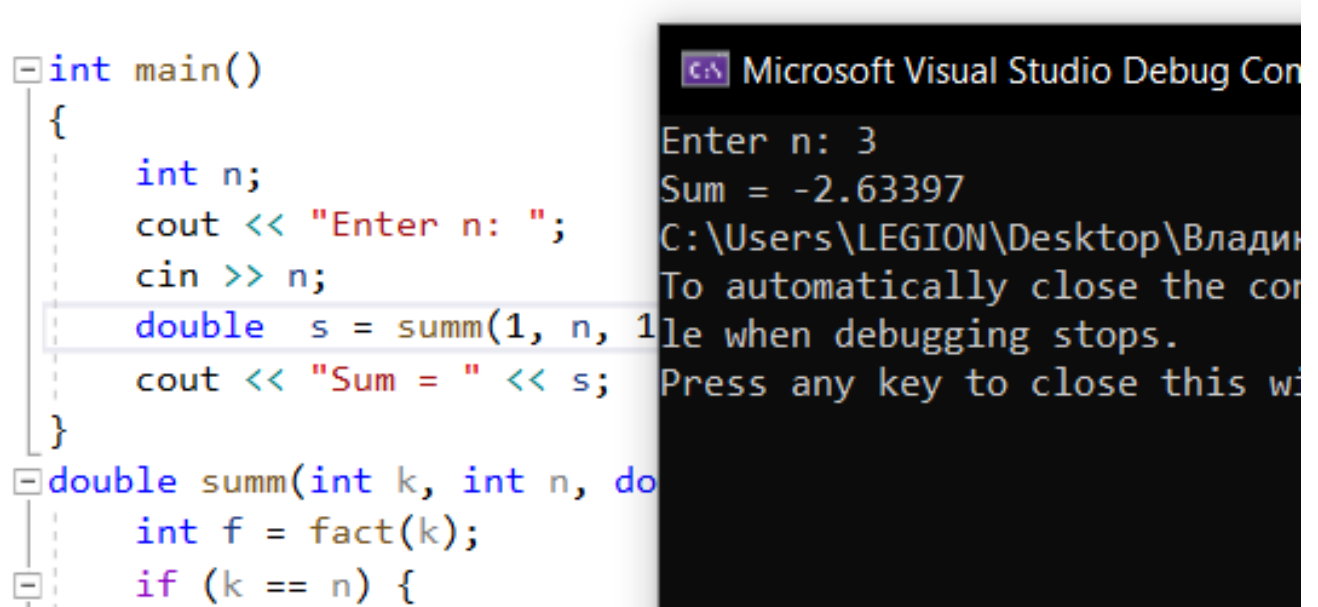
int main()
{
    int n;
    cout << "Enter n: ";
    cin >> n;
    double s = summ(1, n, 1, 1);
    cout << "Sum = " << s;
}

double summ(int k, int n, double a, double b) {
    int f = fact(k);
    if (k == n) { //Термінальна гілка:
        return (a - b) / f; //Повертаємо останній доданок суми
    }
    else { //Рекурсивна гілка
        return ((a - b) / f) + summ(k + 1, n, 0.5 * (sqrt(b) + 5 * sqrt(a)), 2 * a * a + b); //Повертаємо поточний + сума вже без нього
    }
}

int fact(int c) {
    if (c > 1) { //Рекурсивна гілка
        return c * fact(c - 1);
    }
    else { //Термінальна гілка
        return 1;
    }
}
```

Випробування алгоритму.

```
double summ(int, int, double, double); //Знаходже
```



```
int main()
{
    int n;
    cout << "Enter n: ";
    cin >> n;
    double s = summ(1, n, 1, 1);
    cout << "Sum = " << s;
}

double summ(int k, int n, do
    int f = fact(k);
    if (k == n) {
```

Microsoft Visual Studio Debug Console

Enter n: 3
Sum = -2.63397
C:\Users\LEGION\Desktop\Владимир
To automatically close the console window when debugging stops.
Press any key to close this window.

Перевірка алгоритму.

Блок	Дія
	Початок
1	Введення $n=3$
2	$s := \text{summ}(1, n, 1, 1) -$ $\text{return } 0 + \text{summ}(2, 1, 3, 3)$
3	$\text{summ}(2, 1, 3, 3) - \text{return } 0 + \text{summ}(3, 1, 0.5 * 6\sqrt{3}, 21)$
4	$\text{summ}(3, 1, 0.5 * 6\sqrt{3}, 21) - \text{return } -2.63397$
5	Виведення $s = -2.63397$
	Кінець

Висновок. Отже, у цій роботі я дослідив особливості роботи рекурсивних алгоритмів та набув практичних навичок їх використання під час складання програмних специфікацій підпрограм. У результаті лабораторної роботи було розроблено математичну модель, що відповідає постановці задачі; псевдокод та блок-схеми, які пояснюють логіку алгоритму. Використовуючи дві підпрограми з рекурсивними алгоритмами, отримуємо коректний результат.