

Міністерство освіти і науки України
Київський національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

3bit

з лабораторної роботи №6 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження рекурсивних алгоритмів»

Вариант 18

Виконав студент ІП-12 Кушнір Ганна Вікторівна
(шифр, прізвище, ім'я, по батькові)

Перевірів _____
(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 18

Задача. Задано натуральне n . Обчислити $\sum_{k=1}^n \frac{a_k - b_k}{k!}$

$$a_1 = 1, \quad a_k = 0,5(\sqrt{b_{k-1}} + 5\sqrt{a_{k-1}}),$$

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

- Постановка задачі.** Початковими даними є натуральне число n , яке вводиться користувачем з клавіатури, та дійсні числа a_1 і b_1 , які за умовою мають початкове значення 1. Результатом розв'язку є сума S n членів послідовності, які задаються формулою k -го члена $(a_k - b_k)/k!$.
- Побудова математичної моделі.** Складемо таблицю імен змінних.

Змінна	Тип	Ім'я	Призначення
Натуральне число n	Цілий, додатній (натуральне число)	n	Початкове дане
Копія числа n	Цілий, додатній (натуральне число)	m	Допоміжна змінна
Перший член послідовності (a_n)	Дійсний	a_1	Початкове дане
Перший член послідовності (b_n)	Дійсний	b_1	Початкове дане
k -1-й член послідовності (a_n)	Дійсний	a_{k-1}	Допоміжна змінна
k -1-й член послідовності (b_n)	Дійсний	b_{k-1}	Допоміжна змінна
k -й член послідовності (a_n)	Дійсний	a_k	Допоміжна змінна
k -й член послідовності (b_n)	Дійсний	b_k	Допоміжна змінна
Лічильник ітераційного циклу пошуку факторіала	Цілий, додатній	i	Лічильник
Факторіал числа	Цілий, додатній	f	Допоміжна змінна

Обчислення суми n членів послідовності, заданих формулою $(a_k - b_k)/k!$	Функція	Sum()	Допоміжний алгоритм
Обчислення факторіалу числа	Функція	factorial()	Допоміжний алгоритм
Обчислення квадратного кореня з числа	Вбудована функція	sqrt()	Допоміжна функція
Сума n членів послідовності	Дійсний	S	Результат

Таким чином, математичне формулювання задачі зводиться до присвоєння початкових значень змінним $a_1 := 1$, $b_1 := 1$, $m := n$ та обчислення суми S n членів послідовності з використанням допоміжної функції Sum() за формулою: $S := \text{Sum}(a_1, b_1, n, m)$.

✓ Sum(a_{k-1}, b_{k-1}, n, m) – функція, яка використовує рекурсію та виконує наступні дії:

- 1) Обчислення члена послідовності, який відповідає номеру $k-1$, з використанням допоміжної функції factorial(), та присвоєння його значення змінній S :
 $S := (a_{k-1} - b_{k-1}) / \text{factorial}(m - n + 1)$.
- 2) Обчислення k -го члена послідовності (a_n) за формулою:
 $a_k := 0.5 * (\text{sqrt}(b_{k-1}) + 5 * \text{sqrt}(a_{k-1}))$.
- 3) Обчислення k -го члена послідовності (b_n) за формулою:
 $b_k := 2 * a_{k-1} * a_{k-1} + b_{k-1}$.
- 4) За допомогою альтернативної форми оператора вибору (умова: $n \geq 1$) виконання таких дій:
 - * $n := n - 1$ та $S := S + \text{Sum}(a_k, b_k, n, m)$ (рекурсія) – якщо умова «істинна»;
 - * $S := 0$ – якщо умова «хибна».
- 5) Повернення значення S як значення функції Sum().

✓ factorial(i) – функція, яка виконує наступні дії:

- 1) Присвоєння початкового значення змінній f :
 $f := 1$.
- 2) Задання ітераційного циклу з передумовою, умовою якого є $i \geq 1$, та у випадку істинності умови – виконання таких дій:
 $f * = i$ та $i--$ (скорочене множення та віднімання; інакше: $f := f * i$ та $i := i - 1$).
- 3) Повернення значення f як значення функції factorial().

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

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію присвоєння початкових значень змінним a_1 , b_1 та m .

Крок 3. Деталізуємо дію обчислення суми S n членів послідовності.

Крок 4. Деталізуємо функцію $\text{Sum}()$.

Крок 5. Деталізуємо функцію $\text{factorial}()$.

3. Псевдокод алгоритму.

Крок 1

початок

введення n

присвоєння початкових
значень змінним

обчислення суми S n
членів послідовності

виведення S

кінець

Крок 2

початок

введення n

$a_1 := 1$

$b_1 := 1$

$m := n$

обчислення суми S n
членів послідовності

виведення S

кінець

Крок 3

початок

введення n

$a_1 := 1$

$b_1 := 1$

$m := n$

$S := \text{Sum}(a_1, b_1, m, n)$

виведення S

кінець

3.1. Псевдокод допоміжних алгоритмів (функцій).

Крок 4

початок $\text{Sum}(a_k, b_k, m, n)$

якщо $(n \geq 1)$

то

$S := (a_k - b_k) / \text{factorial}(m - n + 1)$

$a_k := 0.5 * (\text{sqrt}(b_k) + 5 * \text{sqrt}(a_k))$

$b_k := 2 * a_k * a_k + b_k$

$n := n - 1$

$S := S + \text{Sum}(a_k, b_k, n, m)$

інакше

$S := 0$

все якщо

повернути S

кінець Sum

Крок 5

початок $\text{factorial}(i)$

$f := 1$

повторити

поки $(i \geq 1)$

$f * = i$

$i--$

все повторити

повернути f

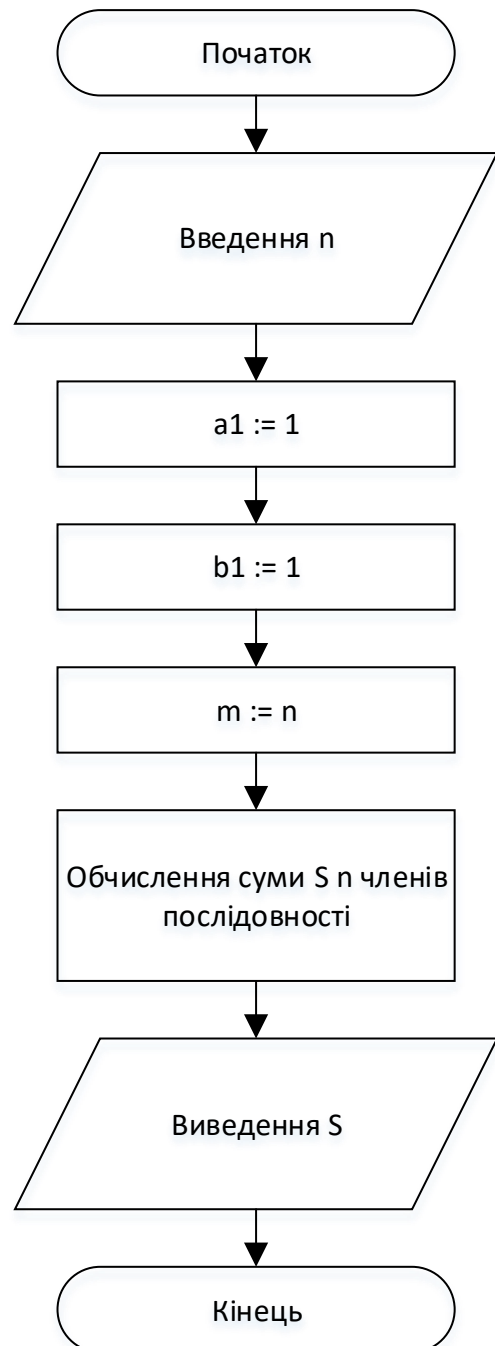
кінець factorial

4. Блок-схема алгоритму.

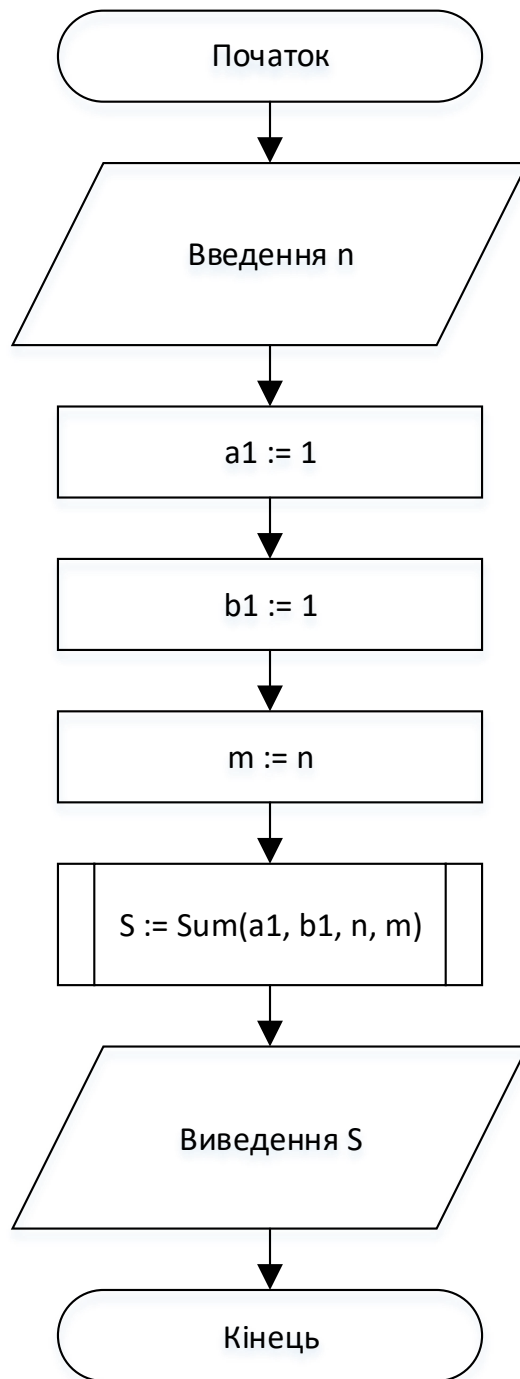
Крок 1



Крок 2

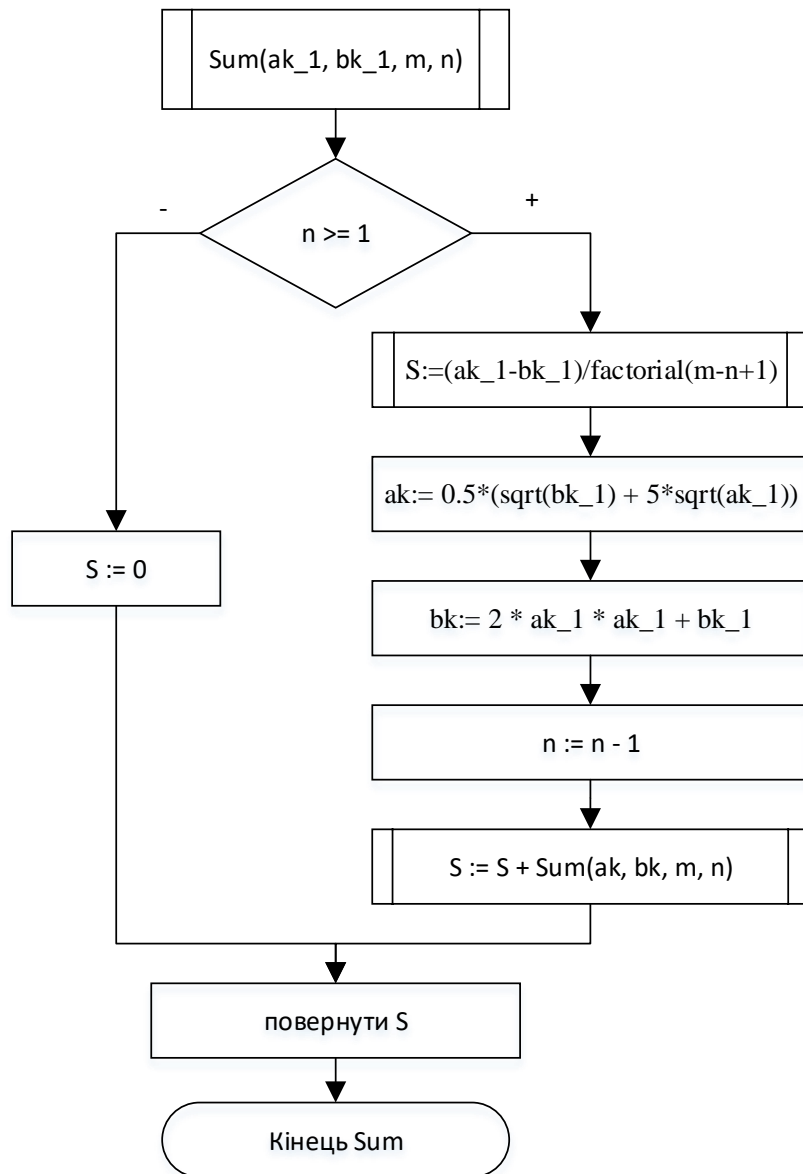


Крок 3

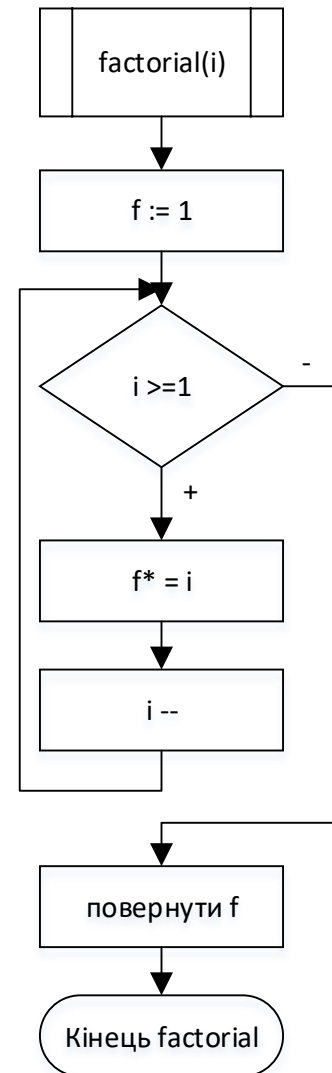


4.1. Блок-схеми допоміжних алгоритмів (функцій).

Крок 4



Крок 5



5. Код програми (на мові програмування C++).

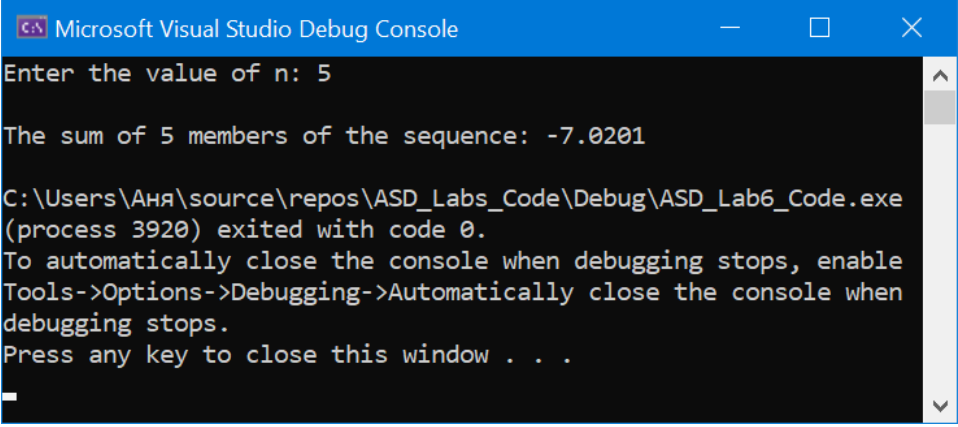
```
#include <iostream>
using namespace std;

long long factorial(long i)
{
    long long f=1;
    while (i >= 1)
    {
        f *= i;
        i--;
    }
    return f;
}

double Sum(double ak_1, double bk_1, long n, long m)
{
    double ak, bk, S;
    if (n >= 1)
    {
        S = (ak_1 - bk_1) / factorial(m - n + 1);
        ak = 0.5 * (sqrt(bk_1) + 5 * sqrt(ak_1));
        bk = 2 * ak_1 * ak_1 + bk_1;
        n = n - 1;
        S = S + Sum(ak, bk, n, m);
    }
    else S = 0;
    return S;
}

int main()
{
    long n, m;
    double a1, b1, S;
    cout << "Enter the value of n: "; cin >> n;
    cout << endl;
    a1 = 1;
    b1 = 1;
    m = n;
    S = Sum(a1, b1, n, m);
    cout << "The sum of " << n << " members of the sequence: " << S << endl;
}
```

6. Тестування програми.



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar is blue with the Visual Studio logo and the text 'Microsoft Visual Studio Debug Console'. The console output is as follows:

```
Enter the value of n: 5

The sum of 5 members of the sequence: -7.0201

C:\Users\Аня\source\repos\ASD_Labs_Code\Debug\ASD_Lab6_Code.exe
(process 3920) exited with code 0.
To automatically close the console when debugging stops, enable
Tools->Options->Debugging->Automatically close the console when
debugging stops.
Press any key to close this window . . .
```


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

Блок	Дія
	Початок
1	Введення $n = 5$
2	$a1 = 1$ $b1 = 1$ $m = 5$
3	$S = \text{Sum}(1, 1, 5, 5)$
3.1	Sum(1,1,5,5): $n = 5 \geq 1$ – істина $S = (1-1) / \text{factorial}(5-5+1)$ factorial(1): $f = 1$ $*i = 1 \geq 1$ – істина $f = 1 * 1 = 1$ $i = 1 - 1 = 0$ $*i = 0 \geq 1$ – хибність Вихід з циклу factorial(1) = 1 $S = 0 / 1 = 0$ $ak = 0.5 * (\text{sqrt}(1) + 5 * \text{sqrt}(1)) = 3$ $bk = 2 * 1 * 1 + 1 = 3$ $n = 5 - 1 = 4$ $S = 0 + \text{Sum}(3, 3, 5, 4)$
3.2	Sum(3,3,5,4): $n = 4 \geq 1$ – істина $S = (3-3) / \text{factorial}(5-4+1)$ factorial(2): $f = 1$ $*i = 2 \geq 1$ – істина $f = 1 * 2 = 2$ $i = 2 - 1 = 1$ $*i = 1 \geq 1$ – істина $f = 2 * 1 = 2$ $i = 1 - 1 = 0$ $*i = 0 \geq 1$ – хибність Вихід з циклу factorial(2) = 2 $S = 0 / 2 = 0$ $ak = 0.5 * (\text{sqrt}(3) + 5 * \text{sqrt}(3)) = 5.19615 \sim 5.196$ $bk = 2 * 3 * 3 + 3 = 21$ $n = 4 - 1 = 3$ $S = 0 + \text{Sum}(5.196, 21, 5, 3)$
3.3	Sum(5.196,21,5,3): $n = 3 \geq 1$ – істина $S = (5.196 - 21) / \text{factorial}(5-3+1)$ factorial(3):

	(...) factorial(3) = 6 $S = -15.804 / 6 = -2.63398 \sim -2.634$ $ak = 0.5 * (\text{sqrt}(21) + 5 * \text{sqrt}(5.196)) = 7.99$ $bk = 2 * 5.196 * 5.196 + 21 = 75$ $n = 3 - 1 = 2$ $S = -2.634 + \text{Sum}(7.99, 75, 5, 2)$
3.4	Sum(7.99, 75, 5, 2): $n = 2 \geq 1 - \text{істина}$ $S = (7.99 - 75) / \text{factorial}(5 - 2 + 1)$ factorial(4): (...) factorial(4) = 24 $S = -67 / 24 = -2.7917 \sim -2.792$ $ak = 0.5 * (\text{sqrt}(75) + 5 * \text{sqrt}(7.99)) = 11.4$ $bk = 2 * 7.99 * 7.99 + 75 = 202.68$ $n = 2 - 1 = 1$ $S = -2.7917 + \text{Sum}(11.4, 202.68, 5, 1)$
3.5	Sum(11.4, 203, 5, 1): $n = 1 \geq 1 - \text{істина}$ $S = (11.4 - 202.68) / \text{factorial}(5 - 1 + 1)$ factorial(5): (...) factorial(5) = 120 $S = -191.28 / 120 = -1.594$ $ak = 0.5 * (\text{sqrt}(202.68) + 5 * \text{sqrt}(11.4)) = 15.56$ $bk = 2 * 11.4 * 11.4 + 203 = 462.92 \sim 462.6$ $n = 1 - 1 = 0$ $S = -1.594 + \text{Sum}(15.56, 462.6, 5, 0)$
3.6	Sum(15.56, 462.6, 5, 0): $n = 0 \geq 1 - \text{хибність}$ $S = 0$ Sum(15.56, 462.6, 5, 0) = 0
*3.5	$S = -1.594 + 0 = -1.594$ Sum(11.4, 202.68, 5, 1) = -1.594
*3.4	$S = -2.792 + (-1.594) = -4.386$ Sum(7.99, 75, 5, 2) = -4.386
*3.3	$S = -2.634 + (-4.386) = -7.02$ Sum(5.196, 21, 5, 3) = -7.02
*3.2	$S = 0 + (-7.02) = -7.02$ Sum(3, 3, 5, 4) = -7.02
*3.1	$S = 0 + (-7.02) = -7.02$ Sum(1, 1, 5, 5) = -7.02
*3	$S = -7.02$
4	Виведення $S = -7.02$
	Кінець

8. *Висновки.* На цій лабораторній роботі було досліджено особливості роботи рекурсивних алгоритмів та було набуто практичних навичок їх використання під час складання програмних специфікацій.

Готовий алгоритм та код програми, побудований за цим алгоритмом, було випробувано на довільному конкретному значенні початкового даного: $n = 5$.

Під час випробування алгоритму (за допомогою калькулятора та систематизації проміжних результатів обчислень у таблиці) було отримано наступне: $S = -7.02$.

Під час виконання коду було отримано наступне: $S = -7.0201$.

Бачимо, що після випробування алгоритму та коду програми було отримано майже однаковий результат. Похибка під час обрахунку результатів на калькуляторі була отримана через округлення деяких дробових чисел; ця похибка становить 0.0001.

Отже, отриманий алгоритм працює правильно.