Додаток 1

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 лінійних алгоритмів»

Варіант_18__

Виконав студент __ ІП-13 Король Валентин Олегович_
(шифр, прізвище, ім'я, по батькові)

Перевірив _____Вечерковська Анастасія Сергіївна_____
(прізвище, ім'я, по батькові)

Київ 202_1_

Лабораторна робота 6

Дослідження складних циклічних алгоритмів

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

Індивідуальне завдання

Варіант 18

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

Завдання

1. **Ввести натуральне число n. Як результат вивести суму членів від k до n.**
2. **Постановка задачі**

   Заданий алгоритм повинен приймати на вводі натуральне число n та виводити суму всіх n членів послідовності
3. **Математична модель**

Побудуємо таблицю імен змінних:

Змінна	***Тип***	***Ім'я***	***Призначення***
Задане число	Натуральне	n	Початкові дані
Аргумент підпрограми, що відповідає рекурсивній функції факторіалу	Натуральне	r	Аргумент рекурсивної функції
Аргумент підпрограми, що відповідає рекурсивній функції знаходження a	Дійсний	n	Аргумент рекурсивної функції
Аргумент підпрограми, що відповідає рекурсивній функції знаходження b	Дійсний	n	Аргумент рекурсивної функції
Результат	Дійсний	S	Вихідні дані

Розв’язання

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо дію знаходження S

Крок 3. Деталізуємо рекурсивні ф-ї

Псевдокод

Крок 1.

Функція multiplication (first_number, step, last_number)

Реалізація рекурсії

Все функція

Початок

Введення даних

Виклик функцій

Виведення даних

Кінець

Крок 2.

Функція factorial(r)

r := 1;

якщо (r == 0 || r == 1)

return 1

все якщо

інакше

return r * factorial(r - 1);

все інакше

Функція adifference(n)

якщо (n == 1)

return 1;

все якщо

інакше return 0.5 * (sqrt(bdifference(n - 1)) + 5 * sqrt(adifference(n - 1)));

Функція bdifference(n)

якщо (n == 1)

return 1;

все якщо

інакше return 2 * pow(adifference(n - 1), 2) + bdifference(n - 1);

Все функція

Початок Введення даних

Виклик функції

Виведення даних

Кінець

Крок 3

Функція factorial(r)

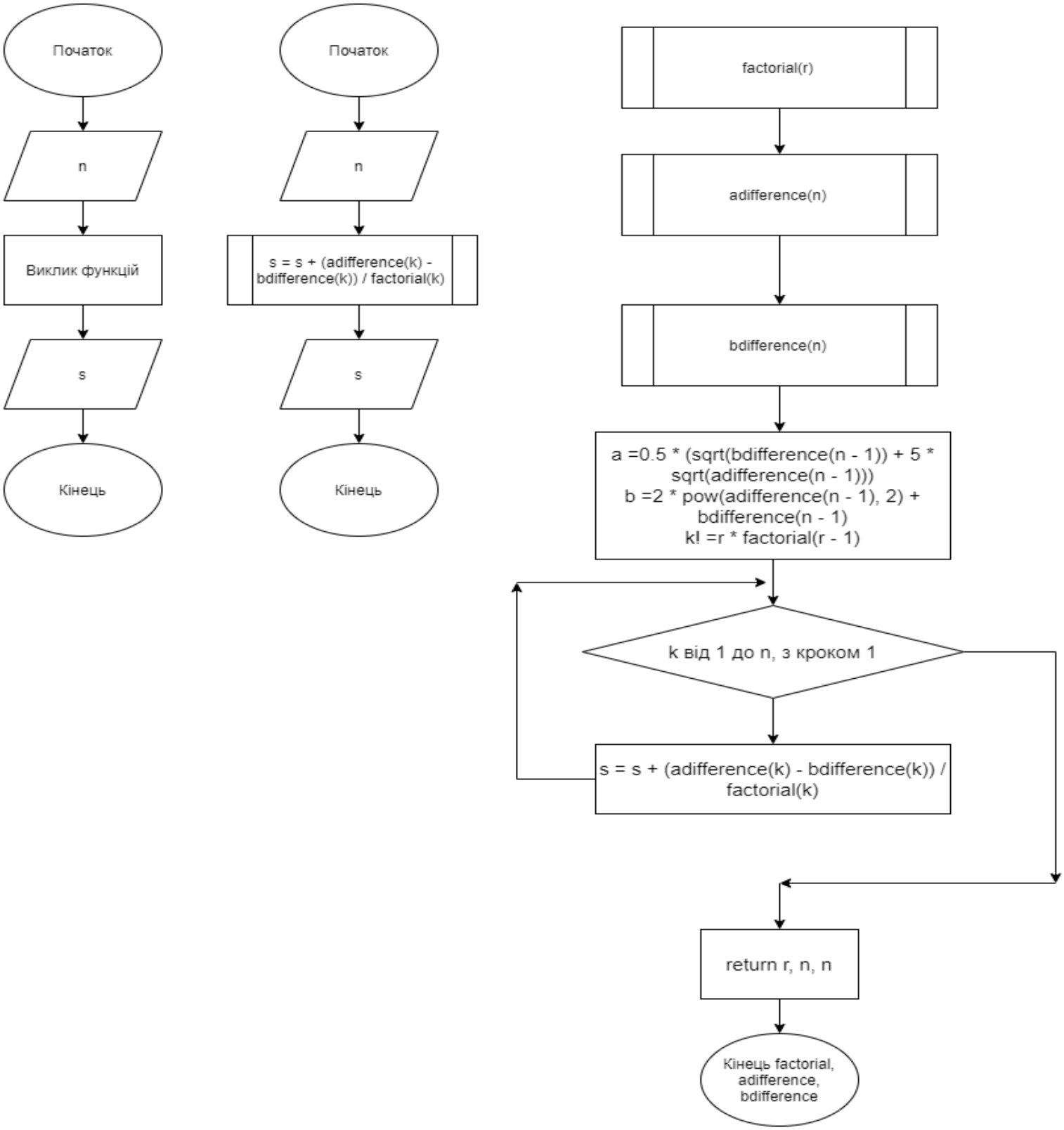
```
        r := 1;
            якщо (r == 0 || r == 1)
                return 1
        все якщо
    інакше
    return r * factorial(r - 1);
        все інакше
    Функція adifference(n)
        якщо  (n == 1)
            return 1;
все якщо
    інакше  return 0.5 * (sqrt(bdifference(n - 1)) + 5 * sqrt(adifference(n - 1)));

    Функція bdifference(n)
        якщо  (n == 1)
            return 1;
все якщо
     інакше  return 2 * pow(adifference(n - 1), 2) + bdifference(n - 1);

початок
        Введеня n
          Для k  Від 1 до n, з кроком 1
                s = s + (adifference(k) - bdifference(k)) / factorial(k);
                виведення s
                        Все повторити

кінець
```

Блок-схем


Початок
n
Виклик функцій
s
Кінець
Початок
n
s = s + (adifference(k) - bdifference(k)) / factorial(k)
s
Кінець
factorial(r)
adifference(n)
bdifference(n)
a =0.5 * (sqrt(bdifference(n - 1)) + 5 * sqrt(adifference(n - 1)))
b =2 * pow(adifference(n - 1), 2) + bdifference(n - 1)
k! =r * factorial(r - 1)
k від 1 до n, з кроком 1
s = s + (adifference(k) - bdifference(k)) / factorial(k)
return r, n, n
Кінець factorial, adifference, bdifference

```cpp
#include <iostream>
#include <math.h>
 using namespace std;
 double bdifference(int n);

double factorial(int r) //рекурсивна ф-я розрахунку факторіала
{
    if (r == 0 || r == 1)
        return 1;
    else
        return r * factorial(r - 1);
}
double adifference(int n)
{
    if (n == 1)
        return 1;

    else
        return 0.5 * (sqrt(bdifference(n - 1)) + 5 * sqrt(adifference(n - 1)));
}

double bdifference(int n)
{
    if (n == 1)
        return 1;
    else
        return 2 * pow(adifference(n - 1), 2) + bdifference(n - 1);
}

    int main()
{
    double a = 1, b = 1, s = 0;
    int n;
    cout << "n="; cin >> n;
    for(int k = 1; k <= n; k++)
```

```cpp
    {
        s = s + (adifference(k) - bdifference(k)) / factorial(k);
    }
    cout << s;
}
```

Висновок

Під час виконання лабораторної роботи я дослідив особливості роботи рекурсивних алгоритмів, набув практичних навичок їх використання під час складання програмних специфікацій підпрограм. Розробив алгоритм для розв’язання поставленої задачі, побудував математичну модель, псевдокод, блок-схему. Написав код програми. Протестував алгоритм та довів його правильність.