МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра систем управління літальних апаратів

**Лабораторна робота № 5**

з дисципліни «Алгоритмізація та програмування»

на тему «Реалізація циклічних алгоритмів мовою С ++

Виконав студент гр. \_\_\_\_\_\_319\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Микита Тiтов\_\_\_\_\_\_\_*

(підпис, дата) (П.І.Б.)

Перевірив

\_\_\_\_\_\_\_\_\_ к.т.н., доц. Олена ГАВРИЛЕНКО

(підпис, дата) (П.І.Б.)

2023

# МЕТА РОБОТИ

Вивчити теоретичний матеріал із синтаксису мовою С ++ і поданням у вигляді UML діаграм циклічних алгоритмів і реалізувати алгоритми з використанням інструкцій циклу з передумовою, циклу з післяумовою і параметризованого циклу мовою C ++ в середовищі Visual Studio.

# ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Дано дійсні числа (xi, yi), i = 1,2, ... n, – координати точок на площині. Визначити кількість точок, що потрапляють в фігуру заданого кольору (або групу фігур).

Завдання 2. Дано дійсне число х і натуральне число n. Необхідно:

a) Обчислити значення виразу при заданих x і n для виразу з табл.2.

b) Вивести: для парних варіантів – значення кожного третього елемента, для непарних – значення кожного четвертого елемента.

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу обчислення суми прийняти у вигляді: | un | < е або | un | > g, де е – мала величина для переривання циклу обчислення суми збіжного ряду (е = 10-5 ... 10-20); g – величина для переривання циклу обчислення суми розбіжного ряду (g = 102 ... 105 ).

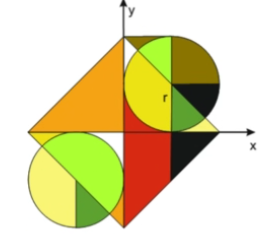
Завдання 4. Організувати меню в командному вікні для багаторазового виконання завдань \*та для перевірки вхідних даних на коректність описати функції, що повертають логічне значення (true – в разі коректного значення переданих параметрів і false – в іншому випадку).

# ВИКОНАННЯ РОБОТИ

Вирішення задачі: варiант 16; варiант 18; варiант 9

Вхідні дані (ім’я, опис, тип, обмеження):

Варiант 16



Варiант 18



Варiант 9



Вихідні дані (ім’я, опис, тип):

Пояснення коду:

Перевірка збіжності/розбіжності ряду з умовами 𝜖 ϵ і 𝑔 g. Обчислення суми ряду за формулою, показаною на зображенні. Виведення кожного третього або четвертого члена ряду в залежності від парності 𝑛 n. 1. Перевірка збіжності/розбіжності ряду

Мета: Обчислити ряд 𝑢 𝑛 = 𝑛 2 + 5 2 𝑛 u n ​ = 2 N n 2 +5 ​ і перевірити, чи збігається він за заданими умовами.

Основна логіка:

Кожен член ряду обчислюється за формулою:

𝑢 𝑛 = 𝑛 2 +5 2 𝑛 . u n ​ = 2 N n 2 +5 ​ .

Умови завершення:

Збіжність: Якщо ∣ 𝑢 𝑛 ∣ <𝜖 ∣u N ​ ∣<ϵ, цикл завершується.

Розбіжність: Якщо

∣𝑢 𝑛 ∣ 𝑔 ∣u N ​ ∣>g, цикл завершується. Член додається до суми, доки не виконається одна з умов.

Виведення:

Якщо ряд збігся, виводиться повідомлення: "Ряд збігається за умовою |u\_n| < epsilon."Якщо ряд розбігся, виводиться: "Ряд розбігається за умовою |u\_n| > g." Додається сума ряду та кількість обчислених членів.

2. Обчислення другого ряду

Мета: Обчислити значення ряду за формулою:

∑ 𝑘 =0 𝑛 1 5 ( 2𝑘 1 ) 𝑥 2𝑘 + 1 4 6 ( 2 𝑘 ) ( 2 𝑘 + 1 ) . k=0 ∑ N ​ 2 4 6(2k) (2k+1) 1 3 5(2k−1) x 2k+1 ​ .

Формула:

Чисельник:

1 3 5 ( 2 𝑘 ) . 1 3 5(2k−1). Для його обчислення використовується функція factorialOdd, яка множить всі непарні числа до 2 𝑘 1 2k−1.

Знаменник:

2 4 6 ( 2 𝑘 ) ( 2 𝑘 + 1 ) . 2 4 6(2k) (2k+1). Для його обчислення використовується функція factorialEven, яка множить всі парні числа до 2 𝑘 2k, а потім множить на 2 𝑘 +1 2k+1.

Поточний член:

Term 𝑘 = Чисельник 2 𝑘 + 1 Знаменник . term k ​ = Знаменник чисельник x 2k+1 ​ .

Цикл:

Обчислення кожного члена та додавання його до суми. Виведення кожного третього члена для парного 𝑛 n або кожного четвертого члена для непарного 𝑛 n.

Виведення:

Сума ряду. Значення кожного третього або четвертого елемента. . Особливості об'єднання

Загальний алгоритм:

Користувач вводить

𝜖 ϵ і g для першого завдання (збіжність/розбіжність).

Розраховується перший ряд 𝑢 𝑛 u n ​ , і перевіряється збіжність/розбіжність.

Користувач вводить 𝑥 x і 𝑛 n для другого ряду (формула з факторіалами).

Розраховується сума другого ряду, а також виводяться вибрані елементи (третій або четвертий залежно від 𝑛 n).

Структура:

Вхідні параметри:

𝜖, 𝑔 ϵ,g — для збіжності/розбіжності.

𝑥, 𝑛 x,n — для другого завдання.

Логічно об'єднані цикли:

Цикл для першого ряду (до збіжності або розбіжності).

Цикл для другого ряду (обчислення суми і вибраних елементів).

Алгоритм вирішення

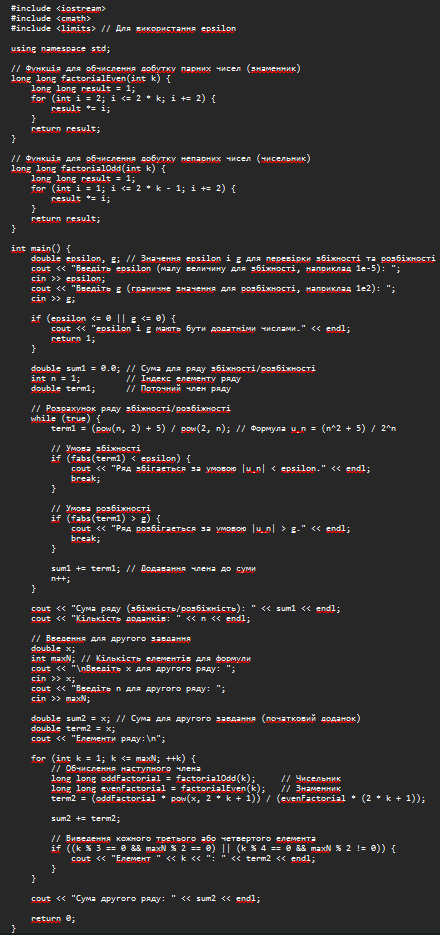


Рисунок 1 – Алгоритм вирiшення

Лістинг коду вирішення задачі:

#include <iostream>

#include <cmath>

#include <limits> // Для використання epsilon

using namespace std;

// Функція для обчислення добутку парних чисел (знаменник)

long long factorialEven(int k) {

long long result = 1;

for (int i = 2; i <= 2 \* k; i += 2) {

result \*= i;

}

return result;

}

// Функція для обчислення добутку непарних чисел (чисельник)

long long factorialOdd(int k) {

long long result = 1;

for (int i = 1; i <= 2 \* k - 1; i += 2) {

result \*= i;

}

return result;

}

int main() {

double epsilon, g; // Значення epsilon і g для перевірки збіжності та розбіжності

cout << "Введіть epsilon (малу величину для збіжності, наприклад 1e-5): ";

cin >> epsilon;

cout << "Введіть g (граничне значення для розбіжності, наприклад 1e2): ";

cin >> g;

if (epsilon <= 0 || g <= 0) {

cout << "epsilon і g мають бути додатніми числами." << endl;

return 1;

}

double sum1 = 0.0; // Сума для ряду збіжності/розбіжності

int n = 1; // Індекс елементу ряду

double term1; // Поточний член ряду

// Розрахунок ряду збіжності/розбіжності

while (true) {

term1 = (pow(n, 2) + 5) / pow(2, n); // Формула u\_n = (n^2 + 5) / 2^n

// Умова збіжності

if (fabs(term1) < epsilon) {

cout << "Ряд збігається за умовою |u\_n| < epsilon." << endl;

break;

}

// Умова розбіжності

if (fabs(term1) > g) {

cout << "Ряд розбігається за умовою |u\_n| > g." << endl;

break;

}

sum1 += term1; // Додавання члена до суми

n++;

}

cout << "Сума ряду (збіжність/розбіжність): " << sum1 << endl;

cout << "Кількість доданків: " << n << endl;

// Введення для другого завдання

double x;

int maxN; // Кількість елементів для формули

cout << "\nВведіть x для другого ряду: ";

cin >> x;

cout << "Введіть n для другого ряду: ";

cin >> maxN;

double sum2 = x; // Сума для другого завдання (початковий доданок)

double term2 = x;

cout << "Елементи ряду:\n";

for (int k = 1; k <= maxN; ++k) {

// Обчислення наступного члена

long long oddFactorial = factorialOdd(k); // Чисельник

long long evenFactorial = factorialEven(k); // Знаменник

term2 = (oddFactorial \* pow(x, 2 \* k + 1)) / (evenFactorial \* (2 \* k + 1));

sum2 += term2;

// Виведення кожного третього або четвертого елемента

if ((k % 3 == 0 && maxN % 2 == 0) || (k % 4 == 0 && maxN % 2 != 0)) {

cout << "Елемент " << k << ": " << term2 << endl;

}

}

cout << "Сума другого ряду: " << sum2 << endl;

return 0;

}

Екран роботи програми показаний на рис.

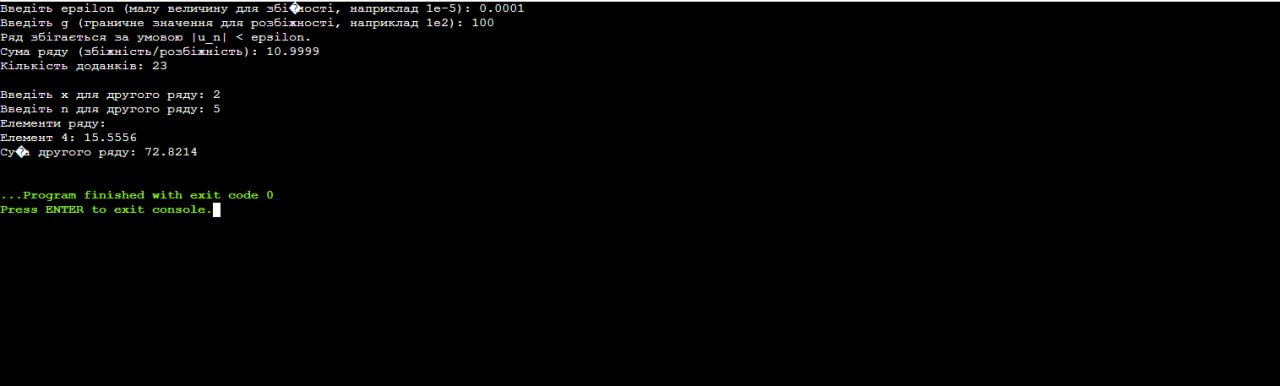


Рисунок 2. Екран роботи програми

# ВИСНОВКИ

Пiд час роботи вивчили теоретичний матеріал із синтаксису мовою С ++ і поданням у вигляді UML діаграм циклічних алгоритмів і реалізували алгоритми з використанням інструкцій циклу з передумовою, циклу з післяумовою і параметризованого циклу мовою C ++ в середовищі Visual Studio.

ДОДАТОК А

Лістинг коду програми

#include <iostream>

#include <cmath>

#include <limits> // Для використання epsilon

using namespace std;

// Функція для обчислення добутку парних чисел (знаменник)

long long factorialEven(int k) {

long long result = 1;

for (int i = 2; i <= 2 \* k; i += 2) {

result \*= i;

}

return result;

}

// Функція для обчислення добутку непарних чисел (чисельник)

long long factorialOdd(int k) {

long long result = 1;

for (int i = 1; i <= 2 \* k - 1; i += 2) {

result \*= i;

}

return result;

}

int main() {

double epsilon, g; // Значення epsilon і g для перевірки збіжності та розбіжності

cout << "Введіть epsilon (малу величину для збіжності, наприклад 1e-5): ";

cin >> epsilon;

cout << "Введіть g (граничне значення для розбіжності, наприклад 1e2): ";

cin >> g;

if (epsilon <= 0 || g <= 0) {

cout << "epsilon і g мають бути додатніми числами." << endl;

return 1;

}

double sum1 = 0.0; // Сума для ряду збіжності/розбіжності

int n = 1; // Індекс елементу ряду

double term1; // Поточний член ряду

// Розрахунок ряду збіжності/розбіжності

while (true) {

term1 = (pow(n, 2) + 5) / pow(2, n); // Формула u\_n = (n^2 + 5) / 2^n

// Умова збіжності

if (fabs(term1) < epsilon) {

cout << "Ряд збігається за умовою |u\_n| < epsilon." << endl;

break;

}

// Умова розбіжності

if (fabs(term1) > g) {

cout << "Ряд розбігається за умовою |u\_n| > g." << endl;

break;

}

sum1 += term1; // Додавання члена до суми

n++;

}

cout << "Сума ряду (збіжність/розбіжність): " << sum1 << endl;

cout << "Кількість доданків: " << n << endl;

// Введення для другого завдання

double x;

int maxN; // Кількість елементів для формули

cout << "\nВведіть x для другого ряду: ";

cin >> x;

cout << "Введіть n для другого ряду: ";

cin >> maxN;

double sum2 = x; // Сума для другого завдання (початковий доданок)

double term2 = x;

cout << "Елементи ряду:\n";

for (int k = 1; k <= maxN; ++k) {

// Обчислення наступного члена

long long oddFactorial = factorialOdd(k); // Чисельник

long long evenFactorial = factorialEven(k); // Знаменник

term2 = (oddFactorial \* pow(x, 2 \* k + 1)) / (evenFactorial \* (2 \* k + 1));

sum2 += term2;

// Виведення кожного третього або четвертого елемента

if ((k % 3 == 0 && maxN % 2 == 0) || (k % 4 == 0 && maxN % 2 != 0)) {

cout << "Елемент " << k << ": " << term2 << endl;

}

}

cout << "Сума другого ряду: " << sum2 << endl;

return 0;

}

ДОДАТОК Б

Скрін-шоти вікна виконання програми

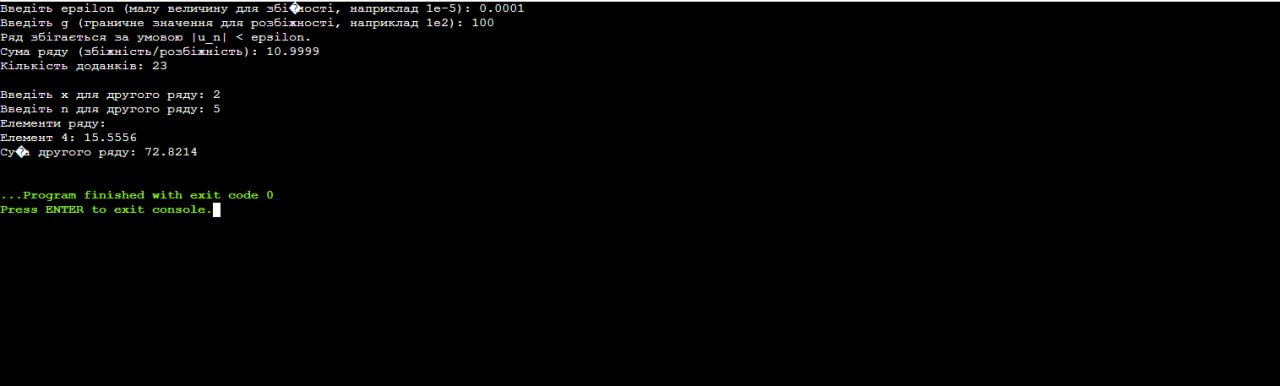


Рисунок Б.1 – Екран виконання програми для вирішення завдання