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»

Тема: "Реалізація циклічних алгоритмів мовою С++"

ХАІ.301.175.318.15 ЛР

Виконав студент гр. № 318

Мальківська В. А

(підпис, дата) (П.І.Б.)

Перевірив к.т.н., доцент

( вчена ступінь, вчене звання)

  Гавриленко О. В

(підпис, дата) (П.І.Б.)

2024

# 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).

Варіант фігури представлено на рис.1 (варіант 32).

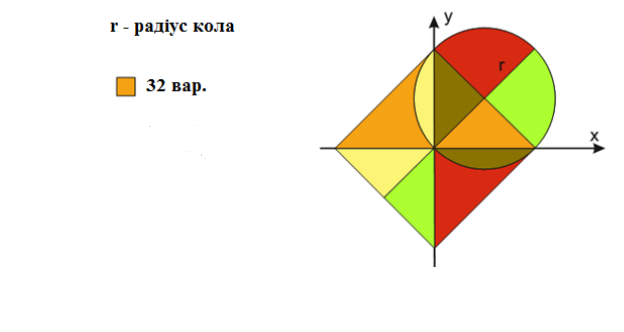


Рисунок 1 – Фігура із завдання 1

**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 (варіант 15).

Варіант виразу представлено на рис.2.

b) Вивести: для парних варіантів – значення кожного третього

елемента, для непарних – значення кожного четвертого елемента.

****

Рисунок 2 – вираз для завдання 2

**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).

Варіант представлено в табл.3. рис. 3 (варіант 37)



Рисунок 3 – вираз для завдання 3

**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

**Завдання 1.**

Вирішення задачі (варіант 32).

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

1. Ім'я: r (радіус кола);

Тип: Дійсне число (double);

Обмеження: r > 0;

2. Ім'я: n (кількість точок);

Тип: Ціле число (int);

Обмеження: n > 0;

3. Ім'я: x, y (координати точок);

Тип: Дійсні числа (double);

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

Ім'я: count (кількість точок у помаранчевій області);

Тип: Ціле число (int);

Опис: результат роботи програми – кількість точок, які належать помаранчевій області;

Діапазон: 0 ≤ count ≤;

Алгоритм

1. Ввести радіус r і перевірити, чи r > 0;

2. Ввести кількість точок n і перевірити, чи n > 0;

3. Перевірити належність точки до помаранчевої області;

4. Підрахувати кількість точок у фігурі (count);

5. Вивести результат;

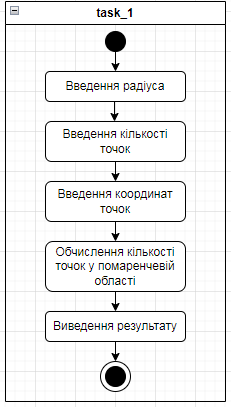


Рисунок 1 — Діаграма активності для завдання 1

Лістинг коду вирішення завдання 1 наведено в дод. А (стор. 11-14)

Екран роботи програми показаний на рис. Б.1 (дод. Б, стор.15)

**Завдання 2.**

Вирішення задачі (варіант 15).

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

Ім'я: double x; int n;

Опис: double x — значення аргумента виразу, в межах [-1, 1];

int n — кількість елементів послідовності, натуральне ціле число;

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

Ім'я: double numerator; double donominator; double result; int i;

Опис: double numerator — чисельник поточного члена ряду;

double donominator — знаменник поточного члена ряду;

double result — значення виразу, сума всіх елементів ряду;

int i — індекс поточного елемента;

Обмеження:

x в межах [-1, 1];

n — натуральне число;

Алгоритм:

1. Ввести x і перевірити, чи в межах [-1, 1];

2. Ввести n і перевірити, чи є натуральним числом;

3. Обчислити кожен елемент ряду, додавати до результату;

4. Вивести значення виразу;

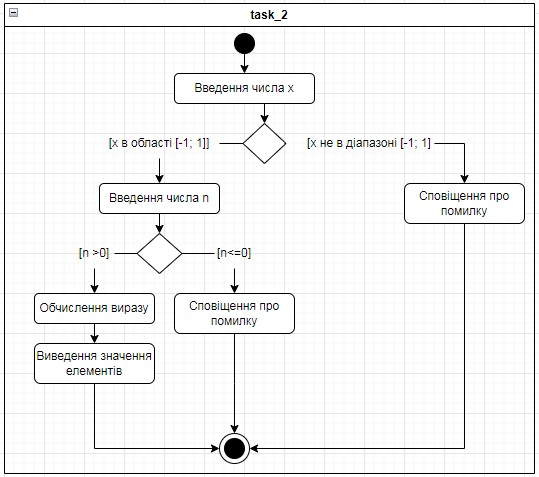


Рисунок 2 — Діаграма активності для завдання 2

Лістинг коду вирішення завдання 1 наведено в дод. А (стор. 11-14)

Екран роботи програми показаний на рис. Б.1 (дод. Б, стор. 15)

**Завдання 3.**

Вирішення задачі (варіант 37).

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

Ім'я: int n; const double e; const double g; const int M; const int maxIterations;

Опис: int n — номер члена ряду, натуральне число, починається з 2;

const double e — точність для визначення збіжності ряду (1e-7);

const double g — межа для визначення розбіжності ряду (1e3);

const int M — крок виведення результатів (100);

const int maxIterations — максимальна кількість ітерацій (1,000,000);

Вихідні дані:

Ім'я: double sum; int n;

Опис:double sum — поточна сума всіх елементів ряду на кожній ітерації;

int n — кількість ітерацій, після яких ряд збігся або розбіжний;

Обмеження: максимальна кількість ітерацій обмежена 1,000,000;

Алгоритм:

1. Ініціалізувати змінні для точності та межі розбіжності;

2. Для кожного члена ряду обчислити його значення;

3. Перевірити, чи досягло значення члена ряду точності e (ряд збігається);

4. Перевірити, чи перевищує значення члена ряду межу g (ряд розбігається);

5. Вивести поточну суму ряду через кожні M ітерацій;

6. Якщо ряд збігається або розбігається, вивести відповідний результат і завершити обчислення;

7. Підсумувати суму і вивести результат та кількість ітерацій;

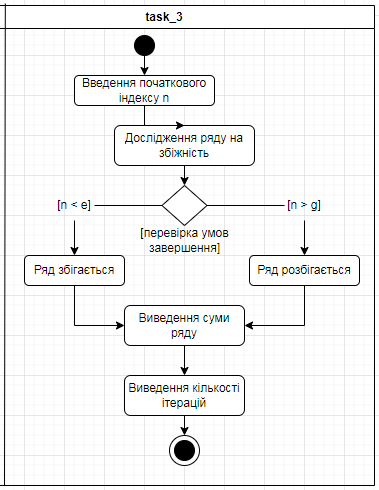


Рисунок 3 — Діаграма активності для завдання 3

Лістинг коду вирішення завдання 1 наведено в дод. А (стор. 11-14)

Екран роботи програми показаний на рис. Б.1 (дод. Б, стор. 15)

**Завдання 4.**

Вирішення задачі (Mеню).

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

Ім'я: int choice;

Опис: int choice — вибір користувача для виконання одного з завдань:

1 — Завдання 1 (точки у фігурі);

2 — Завдання 2 (обчислення виразу);

3 — Завдання 3 (дослідження ряду);

0 — Вихід з програми;

Вихідні дані: task\_1, task\_2 task\_3

Виконання відповідної функції (task\_1, task\_2, task\_3) в залежності від вибору користувача.

Алгоритм:

1. Вивести меню для вибору користувачем завдання;

2. Введення користувачем номер варіанту:

• Якщо вибрано 1, викликається функція task1(), яка обчислює кількість точок у фігурі;

• Якщо вибрано 2, викликається функція task2(), яка обчислює значення виразу;

• Якщо вибрано 3, викликається функція task3(), яка досліджує ряд на збіжність;

• Якщо вибрано 0, програма завершується;

3. Якщо вибір не входить у допустимі варіанти, вивести повідомлення про помилку і запитати ще раз;

4. Програма працює в циклі до тих пір, поки користувач не вибере 0 для виходу;

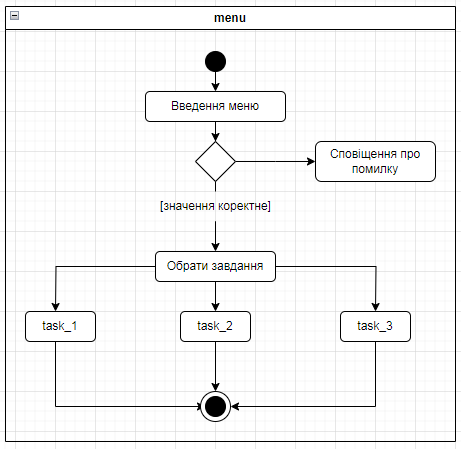


Рисунок 4 — Діаграма активності для завдання 4

Лістинг коду вирішення завдання 1 наведено в дод. А (стор. 11-14)

Екран роботи програми показаний на рис. Б.1 (дод. Б, стор. 15)

ВИСНОВКИ

Під час виконання лабораторної роботи було вивчено синтаксис мови програмування C++ та розроблено UML-діаграми для представлення алгоритмів. Також були реалізовані алгоритми з використанням різних типів циклів: циклів з передумовою, циклів з післямовою та параметризованих циклів, у середовищі Visual Studio.

ДОДАТОК А

Лістинг до коду

#include <iostream>

#include <cmath>

#include <iomanip>

#include <limits>

using namespace std;

// Перевірка входження точки в помаранчеву область (завдання 1, варіант 32)

bool isInOrangeFigure(double x, double y, double r) {

// Помаранчева область: ліва половина кола (x <= 0) + частина правого квадрата (x > 0 && y >= 0 && x <= r && y <= r)

double distanceToCenter = sqrt(x \* x + y \* y);

return (x <= 0 && distanceToCenter <= r) || (x > 0 && y >= 0 && x <= r && y <= r);

}

// Завдання 1: Обчислення кількості точок у фігурі

void task1() {

double r;

int n;

cout << "Введіть радіус кола: ";

cin >> r;

if (r <= 0) {

cout << "Помилка! Радіус має бути додатнім числом." << endl;

return;

}

cout << "Введіть кількість точок: ";

cin >> n;

if (n <= 0) {

cout << "Помилка! Кількість точок має бути натуральним числом." << endl;

return;

}

int count = 0;

for (int i = 0; i < n; ++i) {

double x, y;

cout << "Введіть координати точки " << (i + 1) << " (x, y): ";

cin >> x >> y;

if (isInOrangeFigure(x, y, r)) {

count++;

}

}

cout << "Кількість точок у помаранчевій області: " << count << endl;

}

// Завдання 2: Обчислення значення виразу з табл. 2 (варіант 15)

void task2() {

double x;

int n;

cout << "Введіть значення x: ";

cin >> x;

if (x < -1 || x > 1) {

cout << "Помилка! x має бути в межах [-1, 1]." << endl;

return;

}

cout << "Введіть натуральне число n: ";

cin >> n;

if (n <= 0) {

cout << "Помилка! n має бути натуральним числом." << endl;

return;

}

double result = 0.0;

double numerator = 1.0;

double denominator = 1.0;

cout << fixed << setprecision(15);

cout << "Елементи послідовності:\n";

for (int i = 1; i <= n; ++i) {

numerator \*= (2 \* i - 1);

denominator \*= (2 \* i);

double term = (pow(-1, i + 1) \* numerator \* pow(x, 2 \* i - 1)) / denominator;

result += term;

// Виведення кожного третього для парних n або кожного четвертого для непарних

if ((n % 2 == 0 && i % 3 == 0) || (n % 2 != 0 && i % 4 == 0)) {

cout << "Елемент " << i << ": " << term << endl;

}

}

cout << "Значення виразу: " << result << endl;

}

// Завдання 3: Дослідження ряду на збіжність (варіант 37)

double seriesTerm(int n) {

return pow(-1, n + 1) / (n \* log(n + 1));

}

void task3() {

const double e = 1e-7; // точність

const double g = 1e3; // межа розбіжності

const int M = 100; // крок виведення результатів

const int maxIterations = 1000000; // максимальна кількість ітерацій

int n = 2; // Початок з другого члена ряду

double sum = 0.0;

cout << fixed << setprecision(15);

while (n <= maxIterations) {

double currentTerm = seriesTerm(n);

if (fabs(currentTerm) < e) {

cout << "Ряд збігається на " << n << "-й ітерації.\n";

break;

}

if (fabs(currentTerm) > g) {

cout << "Ряд розбігається на " << n << "-й ітерації.\n";

break;

}

sum += currentTerm;

if (n % M == 0) {

cout << "Ітерація " << n << ", поточна сума: " << sum << "\n";

}

n++;

}

cout << "Кінцева сума ряду: " << sum << endl;

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

}

// Меню програми

int main() {

int choice;

do {

cout << "\nОберіть завдання:\n";

cout << "1 - Завдання 1 (точки у фігурі)\n";

cout << "2 - Завдання 2 (обчислення виразу)\n";

cout << "3 - Завдання 3 (дослідження ряду)\n";

cout << "0 - Вихід\n";

cout << "Ваш вибір: ";

cin >> choice;

switch (choice) {

case 1:

task1();

break;

case 2:

task2();

break;

case 3:

task3();

break;

case 0:

cout << "Вихід з програми...\n";

break;

default:

cout << "Помилка! Оберіть завдання з меню." << endl;

}

} while (choice != 0);

return 0;

}

ДОДАТОК Б

Скрін-шоти вікна виконання програми

На рис. Б.1 показаний приклад виконання завдання 1.

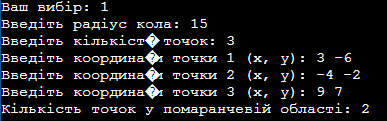


Рисунок Б.1 — Екран виконання програми для вирішення завдання 1

На рис. Б.2 показаний приклад виконання завдання 2.

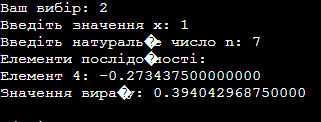


Рисунок Б.2 — Екран виконання програми для вирішення завдання 2

На рис. Б.1 показаний приклад виконання завдання 3.



Рисунок Б.3 — Екран виконання програми для вирішення завдання 3