МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №10**

**з дисципліни «Основи програмування»**

**на тему: «Рекурсія та ітерація»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Нежуміра О. І.

Дніпро, 2020

**Лабораторна робота №10**

**Тема.** Рекурсія та ітерація.

**Мета.** Навчитися описувати повторюванні обчислювальні процеси у рекурсивноїта ітераційної формах.

**Постановка задачи**

1.Загальна постановка: Розробити програму обчислення значення функції для заданого індивідуального завдання і обраного рівня складності.

2.Індивідуальна постановка:

Рівень II (75-89 балів): за допомогою звичайної і хвостової рекурсивних функцій та функції на основі ітерації для заданої кількості членів ряду.

|  |  |
| --- | --- |
| Варіант | Завдання |
| 18. |  |

3.Вимоги до програми:

- дані для обробки вводяться з консолі

- валідація вхідних даних;

- результати роботи програми виводяться на консоль;

- обчислення функції рекурсивним методом;

- обчислення функції ітераційним методом;

- обчислення функції за допомогою функцій математичної бібліотеки, якщо це можливо.

**4.Вимоги до тексту програми:**

- специфікації програми і її функцій;

- самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних.

**Зовнішні специфікації**

1. Вхідні дані

Вхідні дані: n, x

Формат вхідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1. | Число n | n | Ціле число, n > 0 | 10 |
| 2. | Число x | x | -1 < x < 1 | 0.5 |

2. Вихідні дані

Формат вихідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1. | Результат зв. рекурсії | result | Дійсне число | 1.09861 |
| 2. | Результат хв. рекурсії | - | Дійсне число | 1.09861 |
| 3. | Результат ітерації | result | Дійсне число | 1.09861 |
| 4. | Результат мат. функцій | - | Дійсне число | 1.09861 |

**3. Функціональні вимоги до програми**

Програма повинна реалізувати такі дії:

-введення даних у програму

-перевірку введених даних

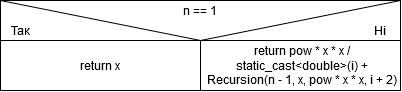
-підрахування прикладу за допомогою рекурсії(звичайної, хвостової) та ітерацієї

-підрахування прикладу за допомогою математичних функцій

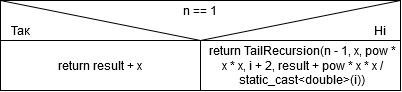
-вивід результату

**Алгоритм розв’язання завдання**

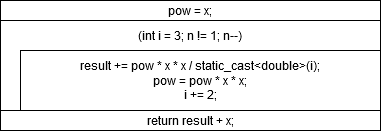
Звичайна рекурсія:



Хвостова рекурсія:

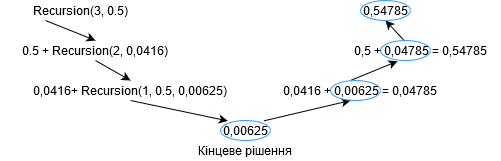


Ітераційний метод:

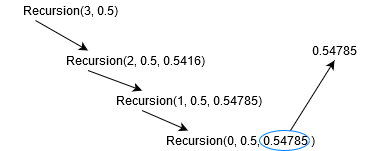


**Схема виконання рекурсивної функції**

Звичайна рекурсія:



Хвостова рекурсія:



**Текст програми**

/\*Вимоги до програми:

-дані для обробки вводяться з консолі;

-валідація вхідних даних;

-результати роботи програми виводяться на консоль;

-обчислення функції рекурсивним методом;

-обчислення функції ітераційним методом;

-обчислення функції за допомогою функцій математичної бібліотеки, якщо це можливо.\*/

#include <iostream>

#include <cmath>

#include "Windows.h"

double Recursion(int n, double x, double pow, int i = 3) { //звичайна рекурсія

if (n == 1)

return x;

else

return pow \* x \* x / static\_cast<double>(i) + Recursion(n - 1, x, pow \* x \* x, i + 2);

}

double TailRecursion(int n, double x, double pow, int i = 3, double result = 0) { //хвостова рекурсія

if (n == 1)

return result + x;

else

return TailRecursion(n - 1, x, pow \* x \* x, i + 2, result + pow \* x \* x / static\_cast<double>(i));

}

double IterationMethod(int n, double x, double result = 0) { //ітераційний метод обчислення

double pow = x;

for (int i = 3; n != 1; n--) {

result += pow \* x \* x / static\_cast<double>(i);

pow = pow \* x \* x;

i += 2;

}

return result + x;

}

int Input\_n() {

system("cls");

int n;

while (true) {

std::cout << "Введіть n: ";

std::cin >> n;

if ((std::cin.peek() != '\n') || !(n > 0)) {

std::cin.clear();

std::cin.ignore(32767, '\n');

std::cout << "На жаль, цей введення неправильне. Число n повино бути більше за 0. Повторіть знову." << std::endl;

}

else

return n;

}

}

double Input\_x() {

double x;

system("cls");

while (true) {

std::cout << "Введіть x: ";

std::cin >> x;

if ((std::cin.peek() != '\n') || !(x < 1) || !(x > -1)) {

std::cin.clear();

std::cin.ignore(32767, '\n');

std::cout << "На жаль, цей введення неправильне. Число x повино вадповідати діапозано (-1<x<1). Повторіть знову." << std::endl;

}

else

return x;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n;

double x;

n = Input\_n();

x = Input\_x();

system("cls");

std::cout << "Число n = " << n << std::endl;

std::cout << "Число x = " << x << std::endl;

std::cout << "Рекурсивний метод обчислення(звичайна): " << 2 \* Recursion(n, x, x) << std::endl; // вызов рекурсивной функции

std::cout << "Рекурсивний метод обчислення(хвостова): " << 2 \* TailRecursion(n, x, x) << std::endl; // вызов рекурсивной функции

std::cout << "Обчислення функції за допомогою функцій математичної бібліотеки: " << log((1 + x)/(1 - x)) << std::endl;

std::cout << "Ітераційний метод обчислення: " << 2 \* IterationMethod(n, x) << std::endl;

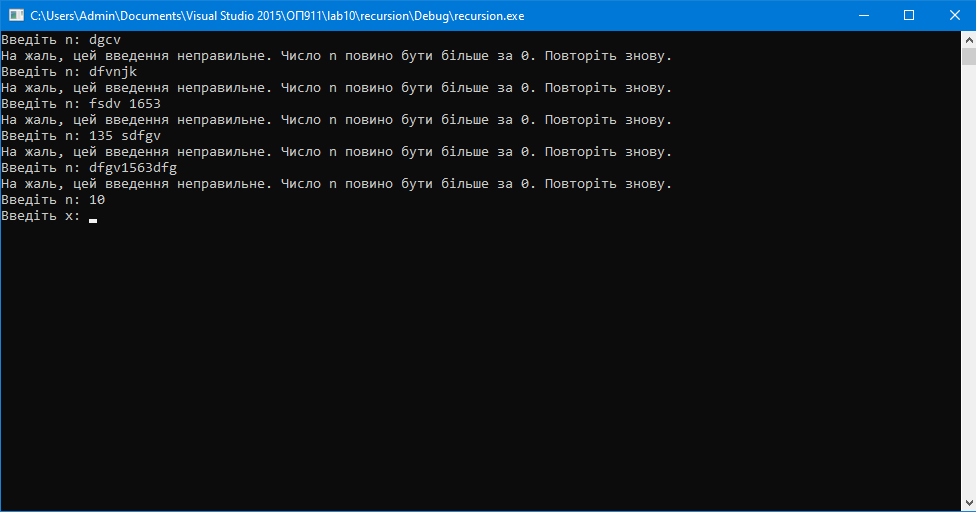
system("pause");

return 0;

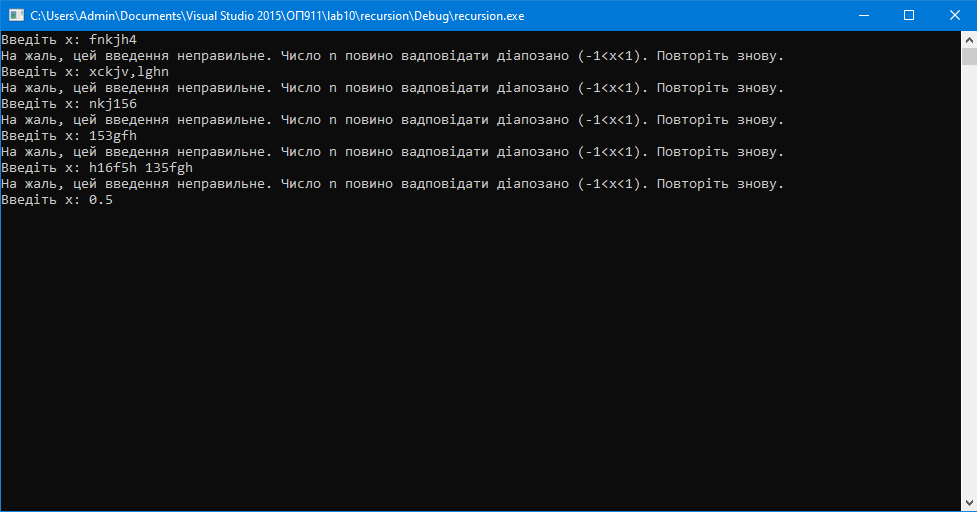
}

Результати тестування програми

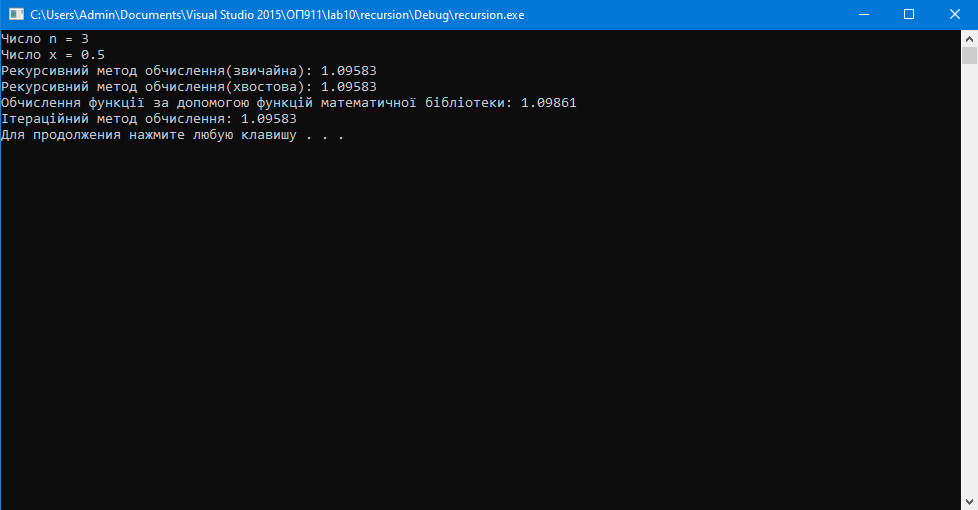
1)Введення числа n



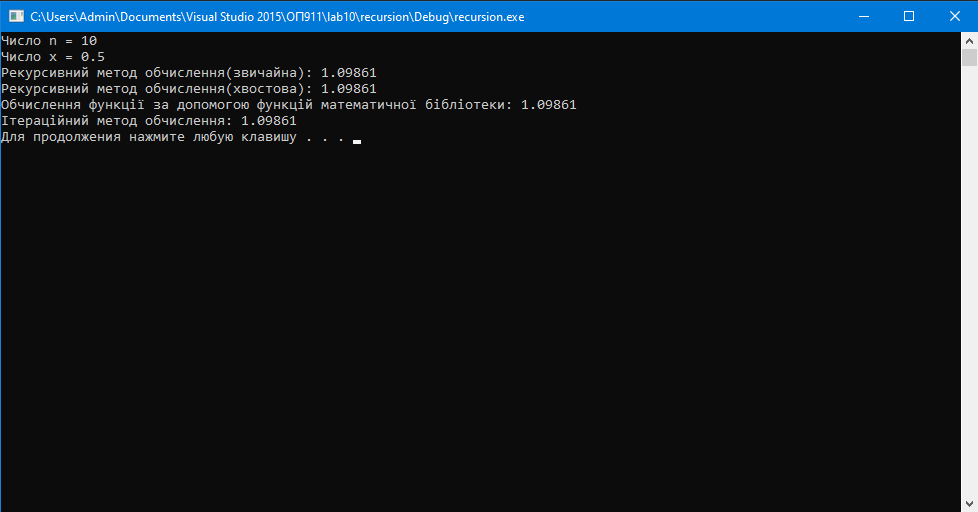
2)Введення числа x



3)Невелика кількість членів ряду (n = 3)



4) Більша кількість членів ряду (n = 10)



Висновок: Рекурсію часто порівнюють з ітерацією. Організація циклічного процесу за допомогою рекурсії має свої переваги і недоліки.

Можна виділити наступні взаємопов'язані переваги рекурсії:

* природність (натуральність) представлення складних, на перший погляд, алгоритмів;
* рекурсивний алгоритм більш читабельний в порівнянні з ітераційним;
* для багатьох поширених завдань рекурсією легше реалізувати ніж ітерацію. Рекурсія добре підходить для реалізації алгоритмів обходу списків, дерев, аналізаторів виразів, комбінаторних задач і т.д.

Недоліки рекурсії полягають у наступному:

* в порівнянні з ітерацією багаторазовий виклик рекурсивної функції вимагає більше часу. Це пов'язано з тим, що при виклику рекурсивного методу його параметри копіюються в стек. Після завершення дзвінка сеанс рекурсивної функції попередні значення параметрів витягуються з стека, що призводить до зайвих операцій. Ітераційний алгоритм для тієї ж завдання працює швидше;
* якщо рекурсивна функція містить великі обсяги локальних внутрішніх змінних і велика кількість параметрів, то використання рекурсії не є ефективним. Це пов'язано з тим, що для кожного рекурсивного виклику потрібно робити копії цих змінних і параметрів. При великій кількості рекурсивних викликів це призведе до надмірного використання пам'яті.