**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **КНІТ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 2

**На тему:** *“Програмування циклічних процесів в С”*

**З дисципліни:** *“Основи програмування”*

**Лектор:**

ст.викл. каф. ПЗ

Муха Т.О.

**Виконав:**

ст. гр. ПЗ-15

Марущак А.С.

**Прийняла:**

асист. каф. ПЗ

Заводська Н.О.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2021 р.

∑= \_\_\_\_ .

Львів – 2021

**Тема роботи:** Програмування циклічних процесів в С.

**Мета роботи:** навчитися організовувати програми циклічної структури, які дозволяють повторювати певну групу операторів задану кількість разів.

**Теоретичні відомості**

**Поняття про цикли**

Алгоритм називається *циклічним*, якщо певна послідовність однотипних дій (тіло циклу) виконується багато разів. Однократне виконання тіла циклу називається *ітерацією*. В залежності від постановки задачі розрізняють *арифметичні* цикли (тобто цикли з наперед відомою кількістю ітерацій) та *ітераційні* цикли (число ітерацій такого циклу наперед невідомо). В арифметичних циклах кількість повторень відома до його початку і задається за допомогою лічильника повторень. В ітераційних циклах кількість ітерацій задається неявно,

наприклад досягненням якоїсь змінної заданої точності, або заданого значення. Отже, цикли дозволяють багатократно виконувати сукупність однотипних операцій. Це означає, що програмний код за рахунок циклів виглядатиме компактнішим. Умовою виходу циклу або ж умовою продовження циклу є вираз, від істинності якого залежить, буде виконуватися тіло циклу чи ні. Цикл може мати лічильник (можливо, не один), що зберігає номер ітерації циклу. Значення лічильника водночас може бути і умовою виходу з циклу. У мові С використовуються оператори циклу **while**, **do…while** та **for**. Усі три оператори циклу в С можуть використовуватися для реалізації як арифметичних, так й ітераційних циклічних процесів. Цикл while є циклом *з передумовою*, а цикл do…while – *з післяумовою*. В циклах з передумовою *спочатку* перевіряється умова, а *тоді*, залежно від того, істинна вона чи хибна,  виконується або не виконується тіло циклу. У циклах з післяумовою *спочатку* виконується тіло циклу, а *тоді* перевіряється умова, від істинності якого залежить виконання *наступної* ітерації циклу. Одне з типових застосувань циклу while – перевірка, чи не досягнуто кінець файлу. Цикл for, як правило, застосовується при заздалегідь відомій кількості ітерацій – наприклад, для опрацювання всіх елементів масиву.

**Конструкція while**

Оператор while – це оператор циклу з передумовою. Синтаксис:

**while(<умова>) <оператор>;**

Цикл виконується, поки умова є істинною. Якщо умова початково є хибною (вираз, що задає умову має нульове значення), цикл не виконається жодного разу. У ролі <оператор>, який і є тілом циклу, може бути будь-який допустимий оператор мови С. Єдине обмеження, яке накладає тут мова С полягає в тому, що, структурно, тіло циклу має мати вигляд одного єдиного оператора, що змушує нас використовувати операторні дужки тоді, коли нам потрібно задати тіло циклу за допомогою декількох операторів (аналогічно як було в операторі умови if).

**Конструкція for** Цикл for управляється змінною, що називається лічильником циклу. Величину зміни лічильника називають кроком. Синтаксис циклу є наступний: for(; ; ) ; Спочатку виконується початкова інструкція та перевіряється умова. Якщо вона істинна, то виконуються інструкції з тіла циклу, а тоді обчислюється вираз та управління передається в початок циклу з тією різницею, що початкова інструкція вже не виконується.

**Вкладені цикли** У мові С існують вкладені цикли. Тобто, один цикл (внутрішній) може знаходитися в тілі іншого циклу (зовнішнього). При цьому глибина вкладення (кількість вкладених циклів) не обмежується. Типове застосування вкладених циклів – для обробки матриць (двовимірних масивів). Цикл for може знаходитися у тілі циклу while (чи do…while), і навпаки.

**Лабораторне завдання**

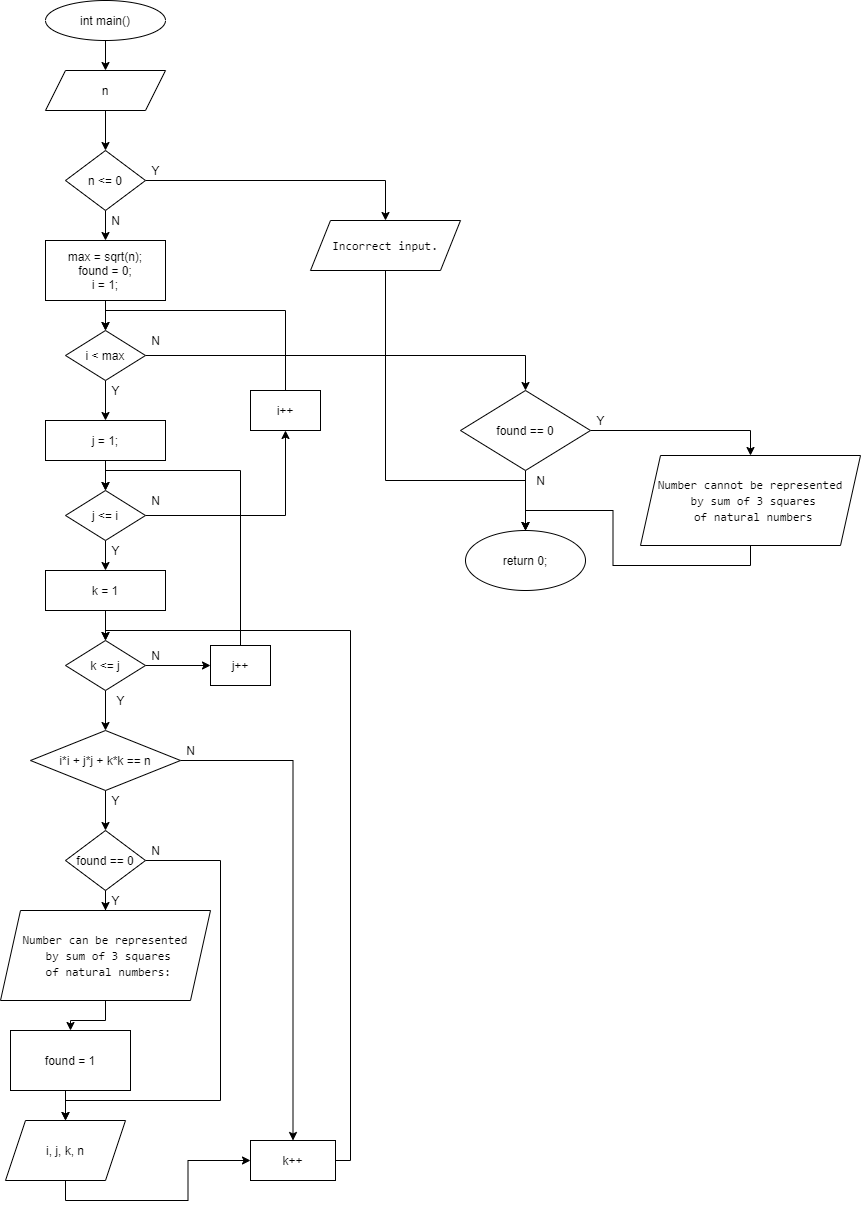
1. Ознайомитися з теоретичним матеріалом викладеним вище в даній інструкції і виконати приклади програм.
2. Одержати індивідуальне завдання з Додатку 1.
3. Розробити алгоритм розв’язання індивідуального завдання і подати його у вигляді блоксхеми.
4. Скласти програму на мові С у відповідності з розробленим алгоритмом.
5. Виконати обчислення по програмі.
6. Одержати індивідуальне завдання з Додатку 2.
7. Розробити алгоритм розв’язання індивідуального завдання і подати його у вигляді блоксхеми.
8. Скласти програму на мові С у відповідності з розробленим алгоритмом.
9. Виконати обчислення по програмі при різних значеннях точності і порівняти отримані результати.
10. Підготувати та здати звіт про виконання лабораторної роботи.

**Індивідуальне завдання**

**№1**

20. Скласти програму, яка читає натуральне число і визначає чи воно дорівнює сумі квадратів яких-небудь 3 натуральних чисел.

**Блок-схема:**



**Код програми: (назва файлу Lab02\_01.c):**

#include <stdio.h>

#include <math.h>

int main()

{

    long long int n;

    scanf("%lli", &n);

    if(n <= 0){

        printf("Incorrect input.");

        return 0;

    }

    long long int max = (int)sqrt(n);

    int found = 0;

    for (long long int i = 1; i < max; i++)

    {

        for(long long int j = 1; j <= i; j++)

        {

            for (long long int k = 1; k <= j; k++)

            {

                if(i\*i + j\*j + k\*k == n)

                {

                    if(!found)

                    {

                        printf("Number can be represented by sum of 3 squares of natural numbers:\n");

                        found = 1;

                    }

                    printf("%lli^2 + %lli^2 + %lli^2 = %lli\n", i, j, k, n);

                }

            }

        }

    }

    if(!found){

        printf("Number cannot be represented by sum of 3 squares of natural numbers.\n");

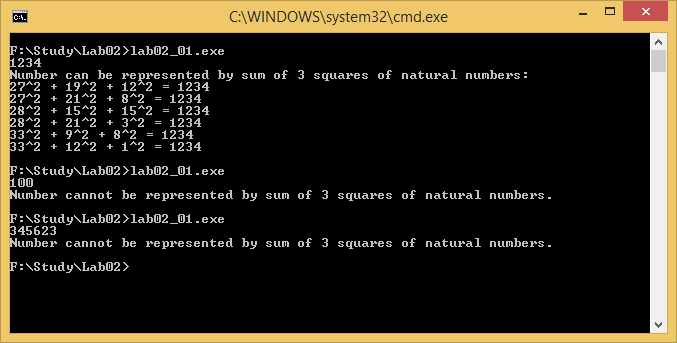
    }

    return 0;

}

**Протокол роботи:**

Програма отримує число і за допомогою трьох циклів перебирає всі можливі комбінації сум і виводить ті, що підходять.



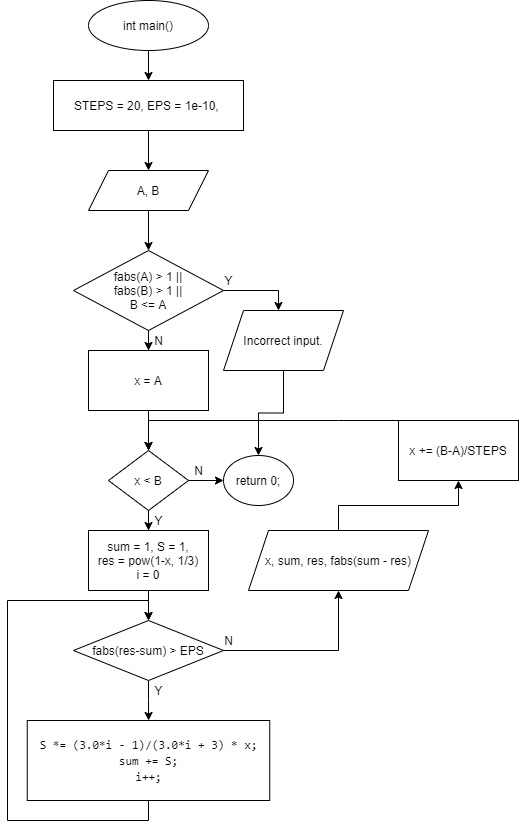
І дійсно, число 1234 можна розкласти декількома способами, а число 100 неможливо.

**Висновок:** ми написали програму, яка перевіряє, чи є число сумою трьох квадратів і виводить всі можливі комбінації, якщо це так. При написанні цієї програми ми навчились використовувати цикл for. Особливу увагу довелось звернути на механізм вкладення циклів.

**№2**

З допомогою операторів циклу, протабулювати на відрізку від А до В з області визначення функцію, задану розкладом у ряд Тейлора. Для порівняння обчислити також у кожній точці табуляції значення функції задане формулою. Результати подати у виді таблиці з коментарями.

**Блок-схема:**



**Код програми: (назва файлу Lab02\_02.c):**

#include <stdio.h>

#include <math.h>

int main(){

    const int STEPS = 20;

    const double EPS = 1e-10;

    double x = 0, sum = 1, S = 1, res = 0, A = 0, B = 0;

    scanf("%lf %lf", &A, &B);

    if(fabs(A) > 1 || fabs(B) > 1 || B <= A){

        printf("Incorrect input.");

        return 0;

    }

    printf("X\t\tf(x)\t\t\tpow(1-x, 1/3)\t\tdifference\n");

    for(x = A; x < B; x += (B-A)/STEPS){

        sum = 1;

        S = 1;

        res = pow(1-x, 1.0/3);

        int i = 0;

        while(fabs(res - sum) > EPS){

            S \*= (3.0\*i - 1)/(3.0\*i + 3) \* x;

            sum += S;

            i++;

        }

        printf("% lf\t%.18lf\t%.18lf\t%.18lf\n", x, sum, res, fabs(res-sum));

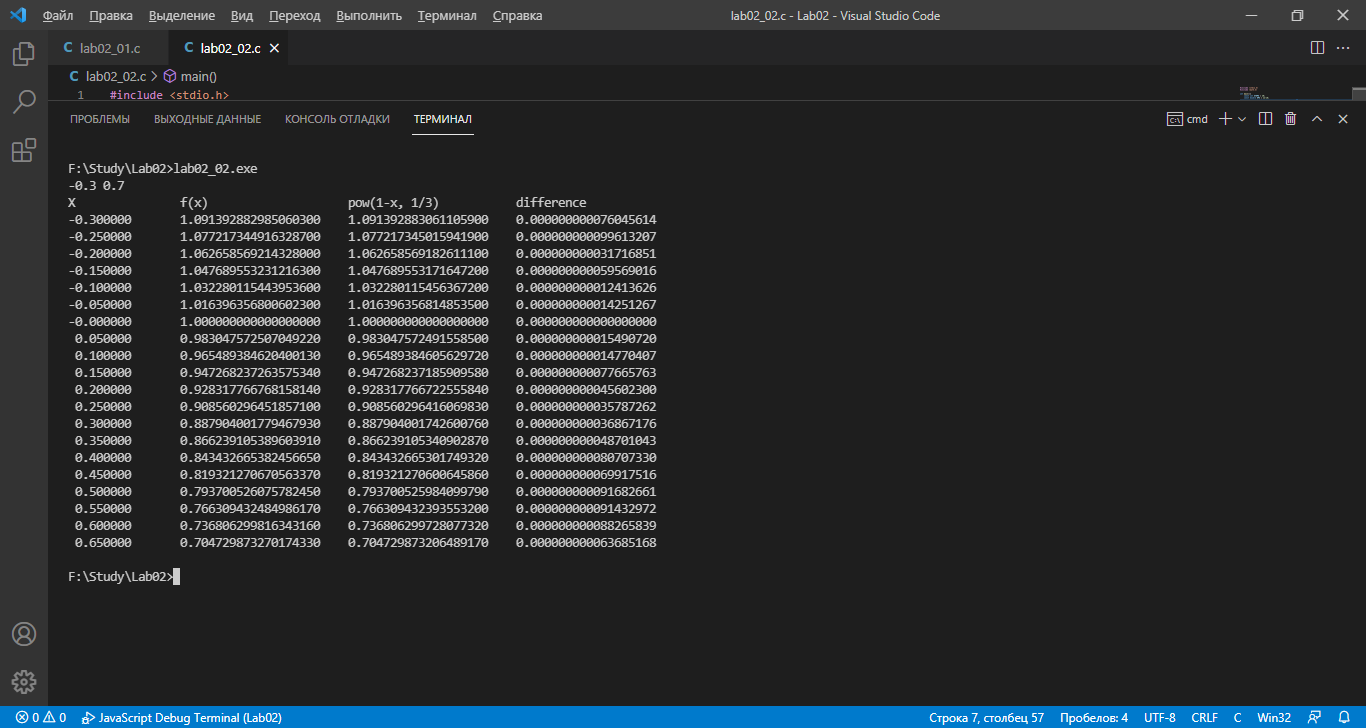
    }

    return 0;

}

**Протокол роботи:**

Програма зчитує 2 числа і табулює функцію в заданому цими числами діапазоні.



Як видно з скріншоту, наша функція змогла досягти точності порядку , яку можна збільшити за рахунок збільшення часу обчислень.

**Висновок:** ми написали програму для обчислення виразу за допомогою ряду Тейлора. Це допомогло нам навчитися використовувати цикл while і закріпило знання про цикл for.