**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-14

Губик А. С.

**Прийняв:**

асист. каф. ПЗ

Кутельмах Р. К.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_ .

Львів – 2022

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

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

**Індивідуальне завдання**

Використовуючи цикл:

Додаток 1. Задано натуральне число. Надрукувати усі прості дільники цього числа.

Додаток 2. 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.

**Теоретичні відомості**

**Конструкція do…while**

Цикл do…while відрізняється від циклу while тим, що умова його виконання перевіряється

після виконання тіла циклу. Це означає, що тіло циклу виконається принаймні один раз.

Синтаксис цього оператоа:

do <тіло циклу> while(<умова>)

Інша відмінність оператора циклу do...while полягає в тому, що тіло циклу може складатися з

довільної кількості операторів, тобто використання операторних дужок не є обов’язковим.

**Конструкція for**

Цикл for управляється змінною, що називається лічильником циклу. Величину зміни

лічильника називають кроком. Синтаксис циклу є наступний:

for(<початкова\_інструкція>; <умова>; <вираз>)

<тіло циклу>;

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.

**Використані функції**



**Код програми**

Додаток 1.

Назва файлу: **uglyprimes.c**

#include <stdio.h>

int main(){

int n;

int DividerIsPrime = 1, nIsPrime = 1;

printf("Enter the number: ");

if(!scanf("%d", &n)){

printf("Wrong input!\n");

return 1;

}

printf("Prime dividers of your number are: ");

for(int i = 2; i < n / 2 + 1; i++)

if (n % i == 0){

for (int j = 2; j < i / 2 + 1; j++)

if (i % j == 0)

DividerIsPrime = 0;

if (DividerIsPrime)

printf("%d ", i);

DividerIsPrime = 1;

nIsPrime = 0;

}

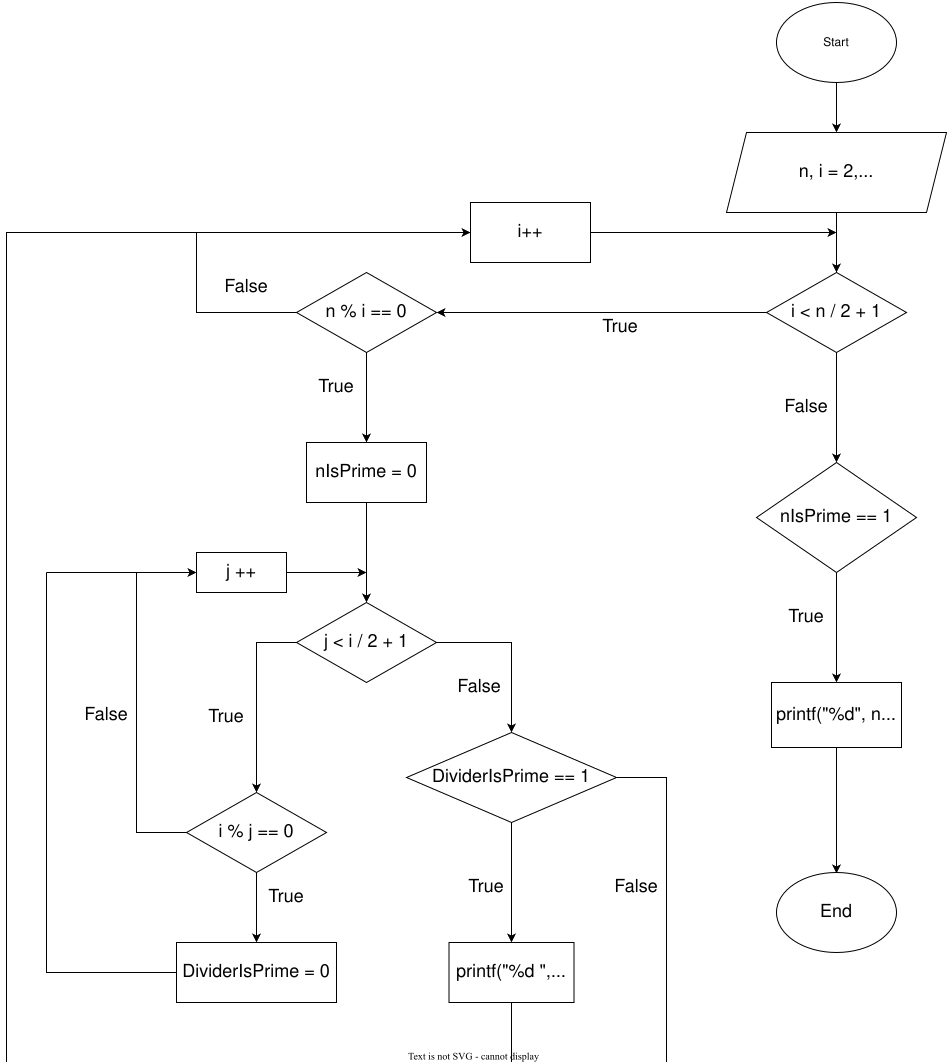
if (nIsPrime)

printf("%d", n);

printf("\n");

return 0;

}



**Додаток 2.**

Назва файлу: **tailor.c**

#include <stdio.h>

#include <math.h>

void DrawLine(int n){

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

printf("-");

printf("\n");

}

int main(){

double a, b, step;

double Accuracy;

const int LineLength = 56;

printf("Enter the limits of tabulation and its step: ");

scanf("%lf %lf %lf", &a, &b, &step);

printf("Enter the accuracy of Taylor series(0.01, 0.1E-5 etc.): ");

scanf("%lf", &Accuracy);

DrawLine(LineLength);

printf("| x | ln(x) | Taylor series |\n");

DrawLine(LineLength);

for(double i = a; i <= b; i += step){

double tln = 0;

if (i <= 1){

printf("| %06.3f | X should be more then 1! |\n", i);

DrawLine(LineLength);

}else{

int j = 0;

double elem;

do{

elem = (1.0 / (2 \* j + 1)) \* pow((i - 1) / (i + 1), 2 \* j + 1);

tln += elem;

++j;

}while(elem > Accuracy);

tln \*= 2;

printf("| %06.3f | %0.16lf | %0.16lf |\n",

i, log(i), tln);

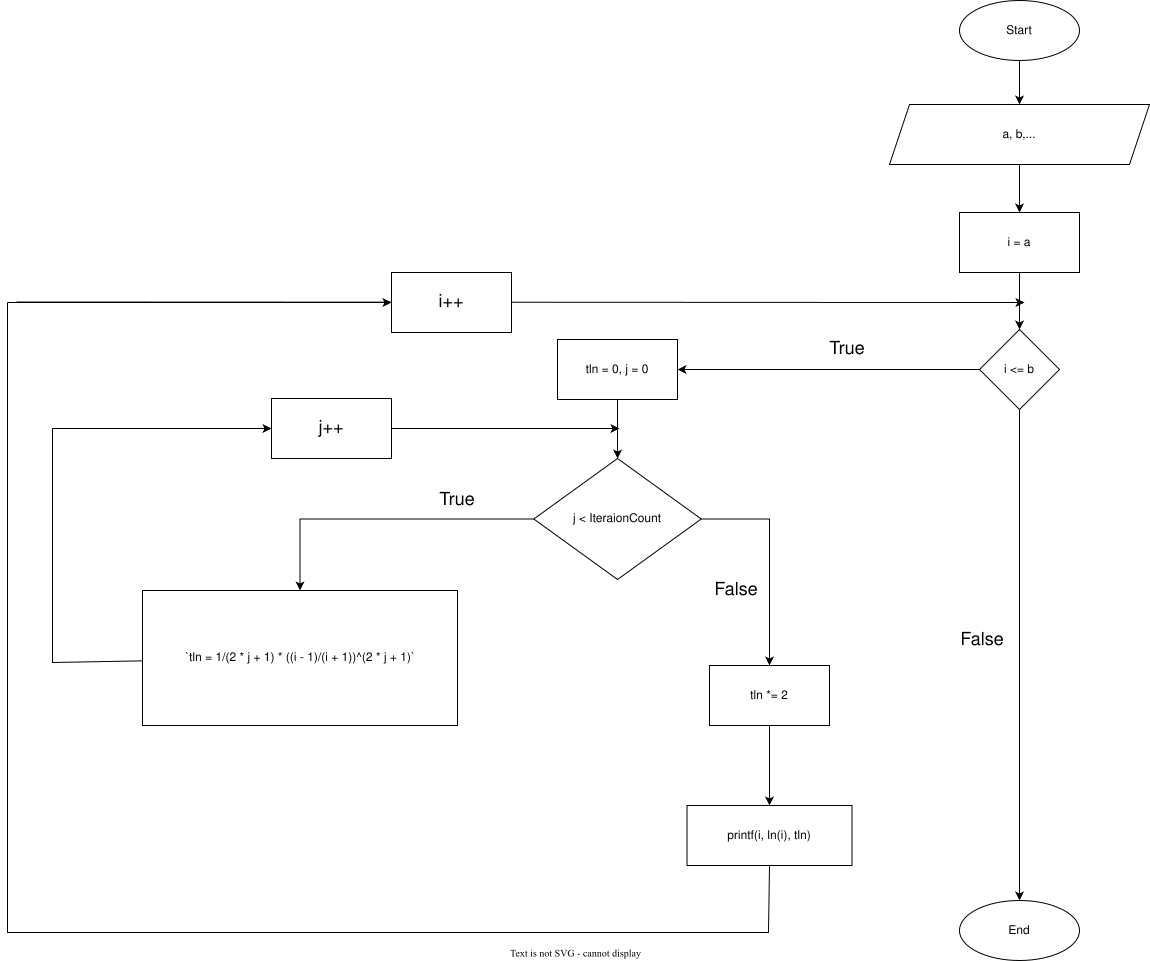
}

}

DrawLine(LineLength);

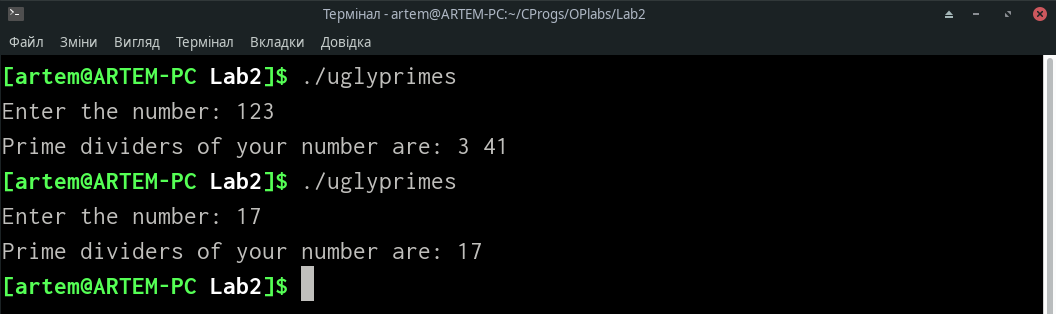
return 0;

}

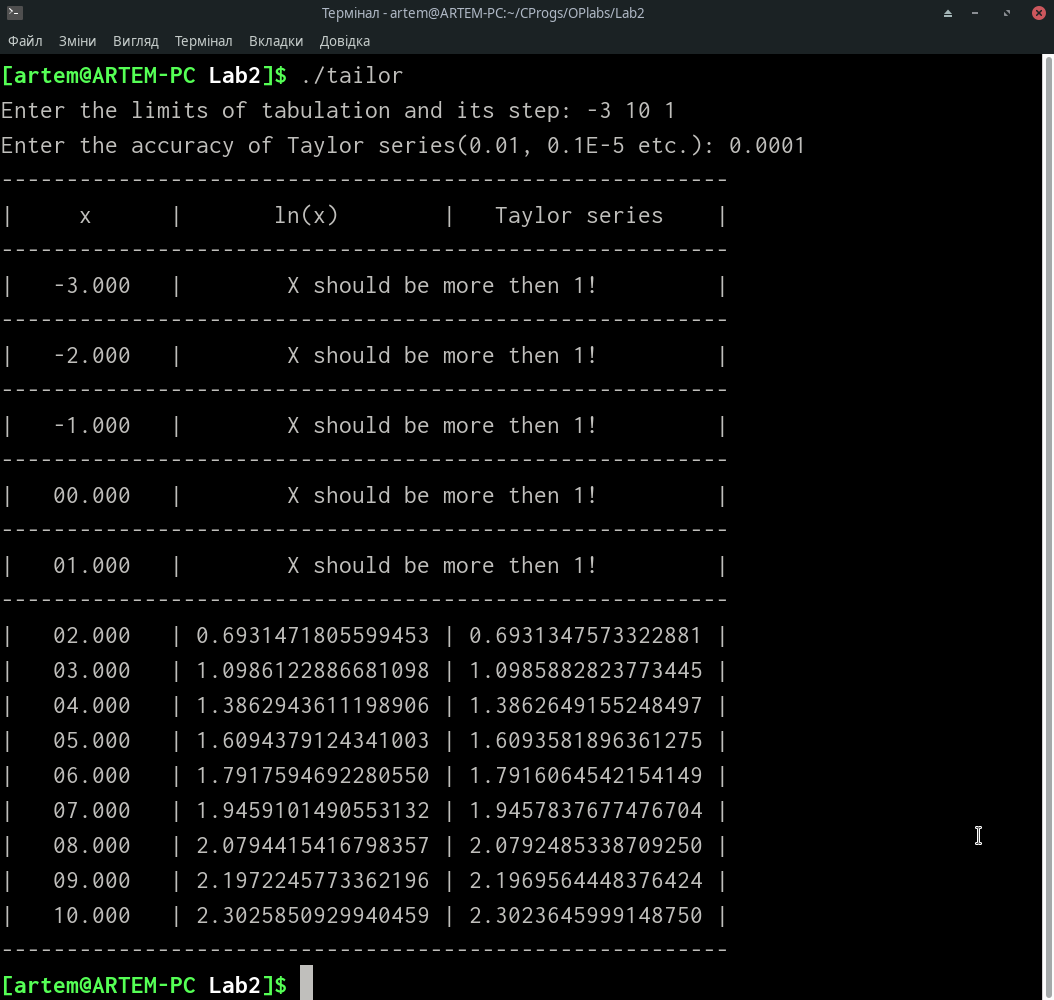


**Протокол роботи**

**Додаток 1: uglyprimes.c**



**Додаток 2: tailor.c :**



**Висновки**

За допомогою різних операторів циклу та їх вкладення один в одного можна реалізувати алгоритми для задач, що потребують багато подібних та повторюваних дій. Таким чином ми зменшуємо об’єм коду, тим самим покращуючи його читабельність і прискорюючи процес виправлення помилок в коді.