МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



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

Кафедра КИТ

**Звіт  
по навчальній практиці**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Дніпро, 2019

**Постановка задачі**

Ознайомитись з методами визначення мінімуму функції однієї або декількох змінних, методами пошуку нуля (кореня) функції. Провести програмну реалізацію методів у відповідності до обраного рівня складності та оформити звіт.

Програма повинна забезпечувати введення наступних даних: границь відрізку на якому розглядається функція (або початкового наближення для методів багатовимірної оптимізації), точності обчислень, вибір функції для розрахунків та метода розрахунку. Відповідь координати точки в якій досягається мінімум або нуль функції, значення мінімуму функції. Значення виводяться з точністю до шести знаків. В доповнення може будуватись графік функції з позначенням отриманого результату.

|  |  |
| --- | --- |
| Вимоги до оформлення програми | Вимоги до змісту завдання |
| Консольний додаток з контролем вхідної інформації і реалізацією функцій в окремому файлі | Набір мінімум з трьох різних функцій. Комбінація методів: Один простий і один середньої складності або всі прості методи Метод визначення нуля функції |

**Зовнішні специфікації**

Формат вхідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Нижня межа інтегрування | a | Дійсне число з області визначення функції | 2 або 2,56 |
| 2 | Верхня межа інтегрування | b | Дійсне число з області визначення функції | 4 або 2,56 |
| 3 | Метод рішення | number\_metod | Ціле позитивне число в діапазоні від 1 до 3 | 1 |
| 4 | Номер функції | number\_fun | Ціле позитивне число в діапазоні від 1 до 3 | 2 |

При завданні виду підінтегральної функції можуть використовуватися знаки ариметичних операцій (+, -, \*, /), операція піднесення до степеня ^, круглі дужки, позначення елементарних математичних функцій (sin, cos, ln і т.д.). В якості аргументу функції використовується х. Складати функцію необхідно з урахуванням правил записи математичних виразів і пріоритету операцій.

Формат вихідних даних

|  |  |  |
| --- | --- | --- |
| № | Вхідні дані | Тестове повідомлення |
| 1 | Вивід відповіді | Відповідь: -5, -148 |

**Функціональні вимоги до програми**

-Вычисление значение определенного интеграла с заданной точностью

-Обеспечивать ввод пользователем исходных данных для вычислений, с контролем входной информации

-Выбор метода численного интегрирования−Построение графика подынтегральной функции

-Сохранение результатов вычислений в соответствии с форматом

-Предоставлять пользователю справочную информацию об использованных методах и порядке работы с программой.

**Вибір методу рішення завдання**

1)Метод золотого перерізу.

### Алгоритм:

1)На першій ітерації заданий відрізок ділиться двома симетричними відносно центру точками і розраховуються значення в цих точках.

2)Після чого той з кінців відрізка, до якого серед двох знову поставлених точок ближче виявилася та, значення в якій максимальне (для випадку пошуку мінімуму), відкидають.

3)На наступній ітерації в силу показаній вище властивості золотого перетину вже треба шукати лише одну нову точку.

4)Процедура триває до тих пір, поки не буде досягнута задана точність.

**Формалізація:**

**Крок 1.** Задаються початкові межі відрізка *а , b* і точність *eps*.

**Крок 2.**Розраховуються початкові точки поділу

х1 = b – ((b - a) / *ф*) (*ф* = 1.618)

х2 = a + ((b - a) / *ф*) (*ф* = 1.618)

і значення в них цільової функції: y1 = *f* (х1) , y2 = *f* (х2).

**Якщо** y1 > y2 то a = х1

**Інакше** b = х2

**Крок 3.** Якщо |b - a| < eps , то x = (a + b) / 2 і зупинитися.

Інакше повернутися до кроку 2.

2) Метод дотичних.

Призначений для наближеного знаходження нулів функції.

На першому кроці потрібно вибрати початкове наближення х0 кореня. Зазвичай це один з кінців відрізка. Початкове наближення має задовольняти наступні умови: *f(х0) \* f “(х0) > 0*. (1)

Далі знайдемо першу і другу похідні функції. Після чого підставляємо в формулу (1) границі відрізка на якому шукаются нулі функції.

На другому кроці нас чекає дорога до кореня. Кожне наступне наближення кореня xn+1 розраховується на підставі попередніх даних за допомогою наступної рекурентної формули: *xn+1= xn – (( f(xn) )/( f ’(xn) ))*

Процес завершується при виконанні умови *( f(xn) )/( f ’(xn) ) < eps* , де *eps* заздалегідь задана точність обчислень. В результаті за наближене значення кореня приймається «енну» наближення: *x\* = xn*

3) Метод параболи.

На першому кроці знаходиться крок *h = |b – a / 10|* та перша точка від якої ми будемо рухатися по графіку *x0 = (a + b) / 2*.

Якщо f(x0) < f(x0 + h) то ми будемо рухатися у зворотному напрямку.

На другому кроці обираються перші три точки за допомогою яких буде знаходитися мінімум.

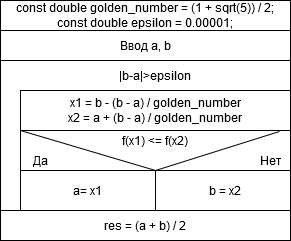
На третьому кроці знаходяться *d1 = f(x1) - f(x2)* та *d2 = f(x3) - f(x2)*

Якщо не виконується умова *d1>= 0* та *d2 >= 0* та *d1 + d2 >0*, то ми переходимо до наступних 3-х точок , та скорочуємо крок у два рази.

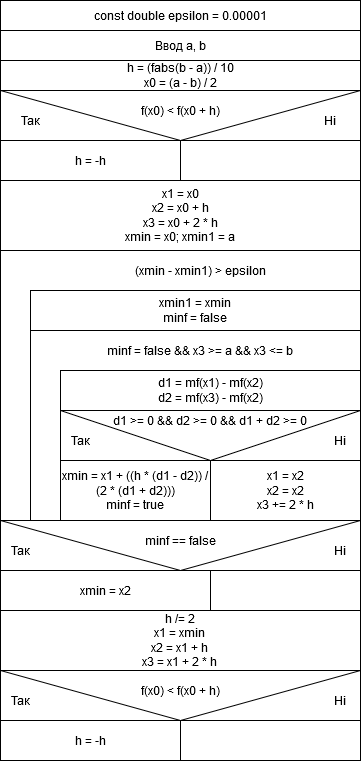
Якщо xmin > eps , то значення цієї точки і є мінмумом функції, якщо ні, то повертаємося до третього кроку.

**Алгоритм програми**

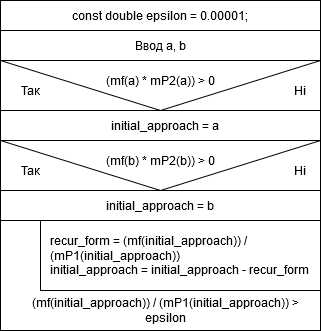
1)Метод залотого перетину



2)Метод параболи



3)Нулі функції



**Текст програми**

metod.h

#ifndef metodH

#define metodH

typedef double(\*pointFunction)(double);

typedef double(\*pointfunction\_derivative1)(double);

typedef double(\*pointfunction\_derivative2)(double);

double function1(double x);

double function\_derivative1(double x);

double function\_derivative2(double x);

double function2(double x);

double function2\_derivative1(double x);

double function2\_derivative2(double x);

double function3(double x);

double function3\_derivative1(double x);

double function3\_derivative2(double x);

//double function3(double x);

int input\_a();

int input\_b();

double metod\_golden(double a, double b, pointFunction mf);

double metod\_parabol(double a, double b, pointFunction mf);

double function\_root(double a, double b, pointFunction mf, pointfunction\_derivative1 mP1, pointfunction\_derivative2 mP2);

#endif

metod.cpp

#include <iostream>

#include "metod.h"

#include <math.h>

const double golden\_number = (1 + sqrt(5)) / 2;

const double epsilon = 0.000001;

double function1(double x)

{

return 10 \* (x \* x \* x) + 5 \* (x \* x) + 10;

}

double function\_derivative1(double x)

{

return 30 \* (x \* x) + 10;

}

double function\_derivative2(double x)

{

return 60 \* x;

}

double function2(double x)

{

return ((x \* x \* x) + (4 \* x) - 3);

}

double function2\_derivative1(double x)

{

return ((3 \* x \* x) + 4);

}

double function2\_derivative2(double x)

{

return (6 \* x);

}

double function3(double x) {

return 11 \* (x \* x \* x) - 5 \* x + 9;

}

double function3\_derivative1(double x) {

return 33 \* (x \* x) + 5;

}

double function3\_derivative2(double x) {

return 66 \* x;

}

double metod\_golden(double a, double b, pointFunction mf)

{

double x1, x2;

double res;

while (fabs(b - a) > epsilon)

{

x1 = b - (b - a) / golden\_number;

x2 = a + (b - a) / golden\_number;

if (mf(x1) <= mf(x2)) // Условие для поиска максимума

a= x1;

else

b = x2;

}

res = (a + b) / 2;

return res;

}

double metod\_parabol(double a, double b, pointFunction mf)

{

double h;

double x0, x1, x2, x3;

double xmin, xmin1;

double d1, d2;

bool minf;

h = (fabs(b - a)) / 10;

x0 = (a - b) / 2;

if (mf(x0) < mf(x0 + h)) {

h = -h;

}

x1 = x0;

x2 = x0 + h;

x3 = x0 + 2 \* h;

xmin = x0; xmin1 = a;

while ((xmin - xmin1) > epsilon)

{

xmin1 = xmin;

minf = false;

while (minf = false && x3 >= a && x3 <= b)

{

d1 = mf(x1) - mf(x2);

d2 = mf(x3) - mf(x2);

if (d1 >= 0 && d2 >= 0 && d1 + d2 >= 0) {

xmin = x1 + ((h \* (d1 - d2)) / (2 \* (d1 + d2)));

minf = true;

}

else {

x1 = x2;

x2 = x2;

x3 += 2 \* h;

}

}

if (minf == false)

{

xmin = x2;

}

h /= 2;

x1 = xmin;

x2 = x1 + h;

x3 = x1 + 2 \* h;

if (mf(x0) < mf(x0 + h))

{

h = -h;

}

}

return xmin;

}

double function\_root(double a, double b, pointFunction mf, pointfunction\_derivative1 mP1, pointfunction\_derivative2 mP2)

{

double initial\_approach;

double recur\_form;

if ((mf(a) \* mP2(a)) > 0)

{

initial\_approach = a;

}

if ((mf(b) \* mP2(b)) > 0)

{

initial\_approach = b;

}

do

{

recur\_form = (mf(initial\_approach)) / (mP1(initial\_approach));

initial\_approach = initial\_approach - recur\_form;

} while ((mf(initial\_approach)) / (mP1(initial\_approach)) > epsilon);

return initial\_approach;

}

int input\_a()

{

int a;

std::cout << "Введіть нижню границю: ";

while (!(std::cin >> a) || (std::cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

std::cin.clear();

while (std::cin.get() != '\n');

std::cerr << "Помилка вводу! Повторить ввод. n повино бути цілим" << std::endl;

std::cout << std::endl;

std::cout << "Введіть нижню границю: ";

}

return a;

}

int input\_b()

{

int b;

std::cout << "Введіть верхня границю: ";

while (!(std::cin >> b) || (std::cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

std::cin.clear();

while (std::cin.get() != '\n');

std::cerr << "Помилка вводу! Повторить ввод. n повино бути цілим" << std::endl;

std::cout << std::endl;

std::cout << "Введіть верхня границю: ";

}

return b;

}

source.cpp

#include <iostream>

#include <Windows.h>

#include <cmath>

#include <math.h>

#include "metod.h"

const double golden\_number = (1 + sqrt(5)) / 2;

const double epsilon = 0.000001;

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int number\_fun, number\_metod;

double a, b, o, f\_initial\_approach;

pointFunction mf; pointFunction mP1;pointFunction mP2;

mf = 0;

do

{

system("cls");

std::cout << "1. 10\*(x\*x\*x) + 5\*(x\*x) + 10" << std::endl;

std::cout << "2. (x\*x\*x\*) + (4\*x)-3" << std::endl;

std::cout << "3. 11\*(x\*x\*x) - 5\*x + 9" << std::endl;

std::cout << "4. Вихід" << std::endl;

std::cin >> number\_fun;

switch (number\_fun)

{

case 1:

{

mf = function1;

mP1 = function\_derivative1;

mP2 = function\_derivative2;

}

break;

case 2:

{

mf = function2;

mP1 = function2\_derivative1;

mP2 = function2\_derivative2;

}

break;

case 3:

{

mf = function3;

mP1 = function3\_derivative1;

mP2 = function3\_derivative2;

}

break;

default:

break;

}

if(number\_fun != 4)

{

do

{

system("cls");

std::cout << "1. Метод золотого сечения" << std::endl;

std::cout << "2. Метод парабол" << std::endl;

std::cout << "3. Mетод дотичних" << std::endl;

std::cout << "4. Вихід" << std::endl;

std::cin >> number\_metod;

switch (number\_metod)

{

case 1:

{

system("cls");

a = input\_a();

b = input\_b();

o = metod\_golden(a, b, mf);

std::cout << "Відповідь: " << o << ", " << mf(o) << std::endl;

system("pause");

}

break;

case 2:

{

system("cls");

a = input\_a();

b = input\_b();

o = metod\_parabol(a, b, mf);

std::cout << "Відповідь: " << o << ", " << mf(o) << std::endl;

system("pause");

}

break;

case 3:

{

system("cls");

a = input\_a();

b = input\_b();

f\_initial\_approach = function\_root(a, b, mf, mP1, mP2);

std::cout << "Відповідь: " << f\_initial\_approach << std::endl;

system("pause");

}

break;

default:

break;

}

} while (number\_metod != 4);

}

} while (number\_fun != 4);

return;

}

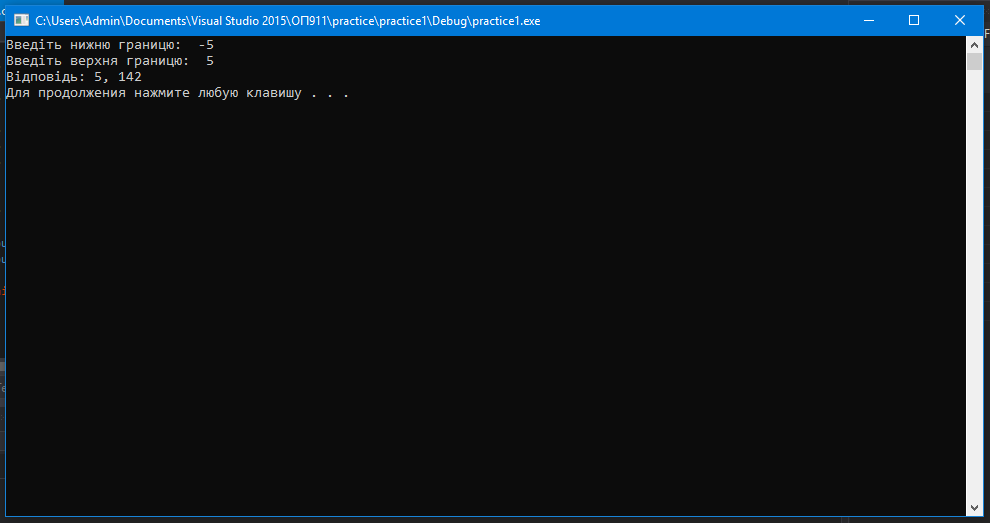
Розробка тестів

Для всіх тестів значення a = -5, b = 5

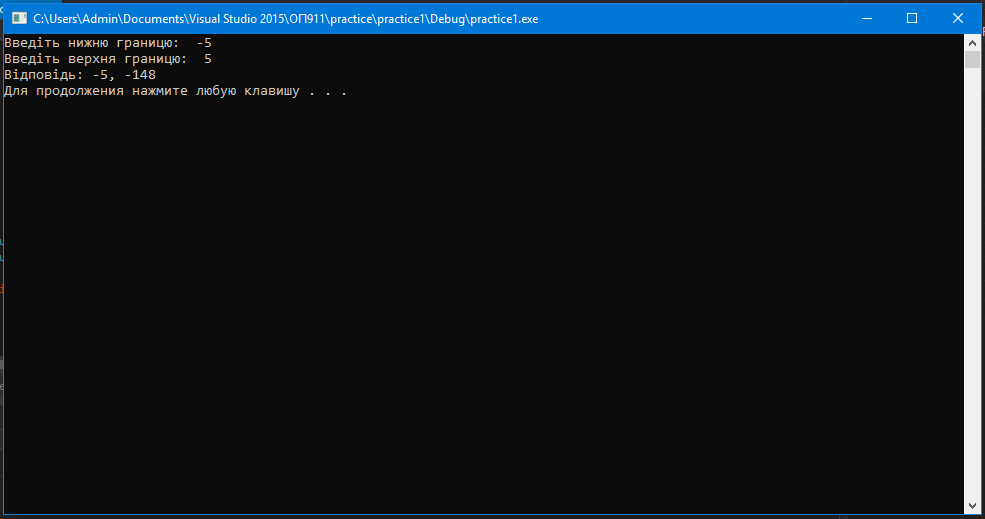
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Функція | Золотій перетин | Метод парабол | Метод дотичних |
| 1 | x^3 + 4\*x - 3 | (5; 142) | (-5; 148) | 0.673597 |
| 2 | 10\*x^3 + 5\*x^2 + 10 | (5; 1389) | (-5; -1115) | -1.19742 |

1)

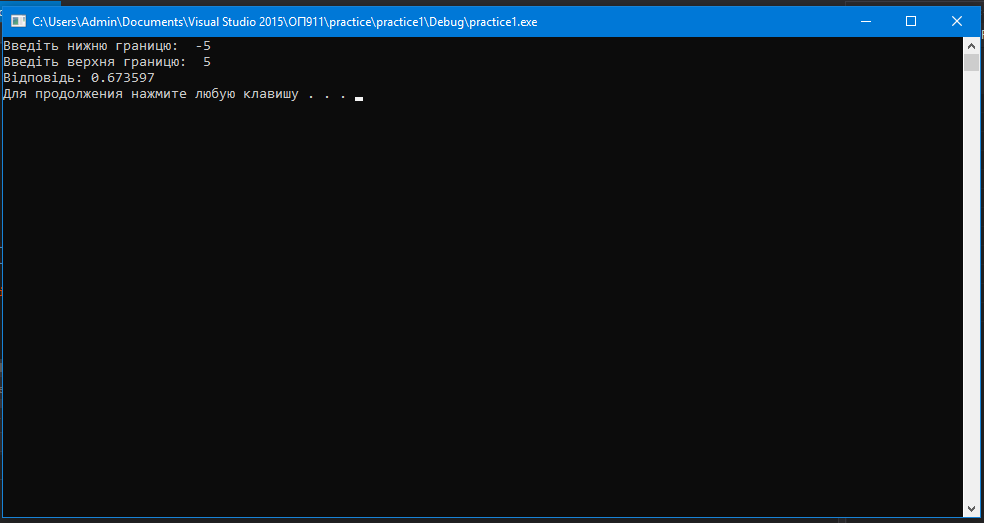
Метод золотого перетину:



Метод перебол:

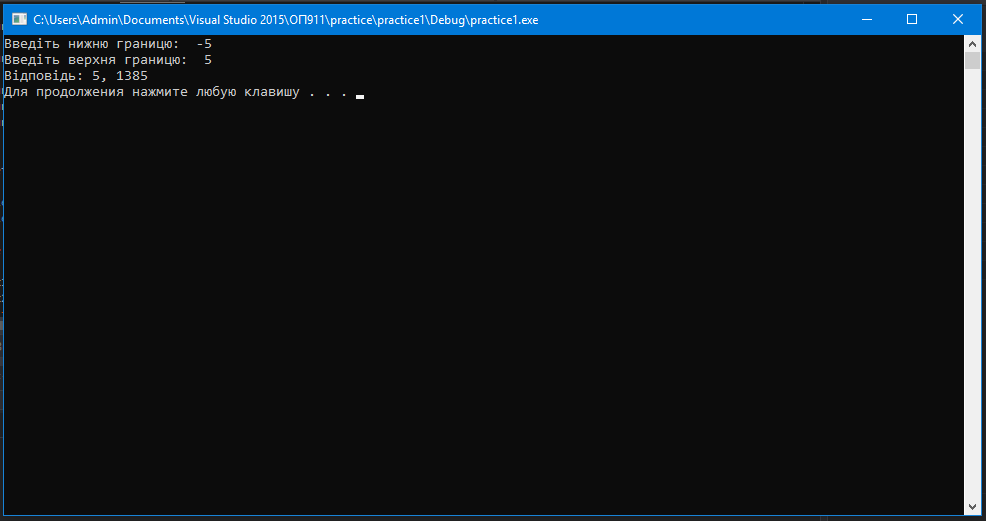


Метод дотичних:

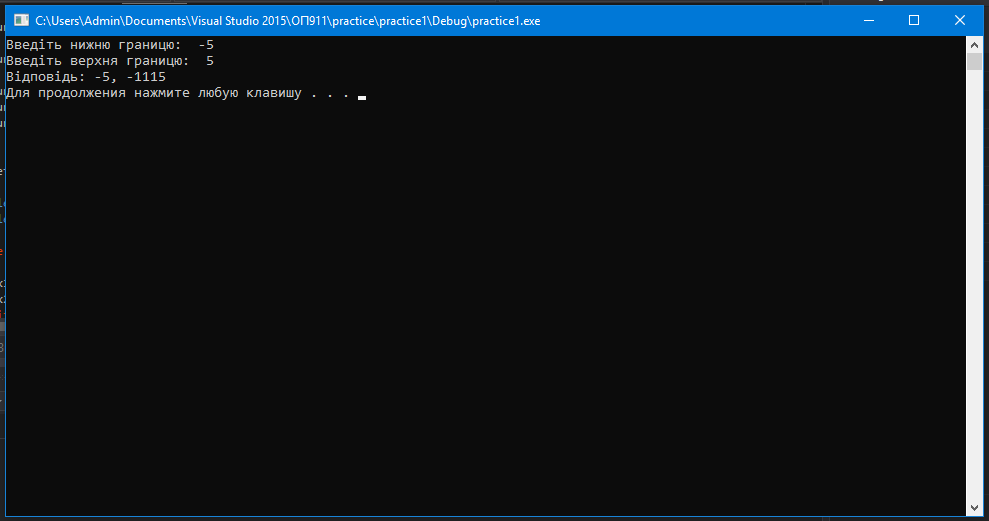


2)

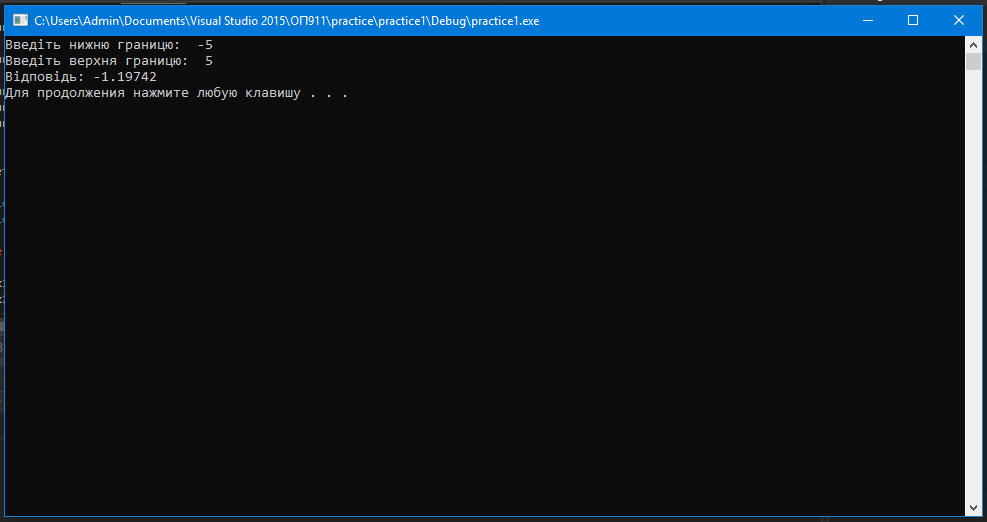
Метод золотого перетину:



Метод парабол:



Метод дотичних:



Висновок: було розроблено програму, що працює з функціями та знаходить точи мінімуму графіків функцій, а також точки перетину з віссю оХ. Пошук мінімумів відбувався методами золотого перерізу та методом парабол, точки перетину були знайдені методом дотичних. При початку роботи програми з’являється меню в якому є запропоновані для вибору функції, після вибору однієї із них користувач може вибрати метод роботи з функцією. Після введення необхідних даних на екрані з’явиться відповідь.

Контекстне меню було реалізовано за допомогою операторів switch case, функції та методи були реалізовані в окремому .срр файлі, прототипи функцій були винесені в файл .h.