НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра Автоматизованих Систем Обробки Інформації і Управління

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з предмету «Основи програмування»

(назва дисципліни)

на тему: Пошук екстремумів функції однієї змінної

Студента 1 курсу ІП-61 групи

напряму 6.050103 Програмна інженерія

спеціальності 121 Інженерія програмного

забезпечення Кушки М.О.

(прізвище та ініціали)

Керівник Муха І. П.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ - 2017 рік

Національний технічний університет України “КПІ”

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Програмна інженерія"

Курс 1 Група ІП-61 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Кушки Михайла Олександровича |

(прізвище, ім’я, по батькові)

|  |  |
| --- | --- |
| 1. Тема роботи | Пошук екстремумів функції однієї змінної |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи |  |

|  |  |
| --- | --- |
| 3. Вихідні дані до роботи | Функція однієї змінної, область пошуку, точність обчислень. Методи |
| розв’язання: метод «золотого» перетину, метод рівномірного пошуку, метод діхотомії. | |
|  | |
|  | |

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

|  |
| --- |
|  |
|  |
|  |
|  |

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )

|  |
| --- |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| 6. Дата видачі завдання |  |

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи |  |  |
| 2. | Підготовка ТЗ |  |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи |  |  |
| 4. | Розробка алгоритму вирішення задачі |  |  |
| 6. | Узгодження алгоритму з керівником |  |  |
| 5. | Розробка сценарію роботи програми |  |  |
| 6. | Узгодження сценарію роботи програми з керівником |  |  |
| 7. | Узгодження з керівником інтерфейсу користувача |  |  |
| 8. | Розробка програмного забезпечення |  |  |
| 9. | Налагодження розрахункової частини програми |  |  |
| 10. | Розробка та налагодження інтерфейсної частини програми |  |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу |  |  |
| 12. | Тестування програми |  |  |
| 13. | Підготовка пояснювальної записки |  |  |
| 14. | Здача курсової роботи на перевірку |  |  |
| 15. | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  |  |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | Муха І. П. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Анотація

Пояснювальна записка до курсової роботи: 66 сторінок, 22 рисунки, 3 таблиці, 3 посилання.

Об’єкт дослідження: задача пошуку екстремумів функції на відрізку.

Мета роботи: дослідження методів пошуків максимуму і мінімуму функції на відрізку, створення програмного забезпечення для реалізації цієї задачі наступними методами: «золотого» перетину, рівномірного пошуку та діхотомії.

Виконана програмна реалізація методів «золотого» перетину, рівномірного пошуку та діхотомії для пошуку мінімуму і максимуму функції на відрізку. Також програма будує графік заданої функції для наочності та за вимогою користувача зберігає результати роботи програми у окремий файл.

Зміст

[Вступ 7](#_Toc483256352)

[1 Постановка задачі 8](#_Toc483256353)

[2 Теоретичні викладки 9](#_Toc483256354)

[2.1 Метод «золотого» перетину отступ 1, 25 9](#_Toc483256355)

[2.2 Метод рівномірного пошуку 10](#_Toc483256356)

[2.3 Метод діхотомії або Метод половинного ділення 11](#_Toc483256357)

[3 Опис алгоритмів 12](#_Toc483256358)

[3.1 Загальний алгоритм 12](#_Toc483256359)

[3.2 Метод «золотого» перетину 13](#_Toc483256360)

[3.3 Метод рівномірного пошуку 13](#_Toc483256361)

[3.4 Метод діхотомії 14](#_Toc483256362)

[4 Опис програмного забезпечення 15](#_Toc483256363)

[4.1 Опис функціональної структури програмного забезпечення 15](#_Toc483256364)

[4.2 Опис функцій частин програмного забезпечення 15](#_Toc483256365)

[4.2.1 Користувацькі функції 15](#_Toc483256366)

[4.2.2 Стандартні функції 21](#_Toc483256367)

[5 Тестування програмного забезпечення 24](#_Toc483256368)

[5.1 План тестування 24](#_Toc483256369)

[5.2 Приклади тестування тут помимо рисунков еще таблички должны быть как в примере хотя рисунки не особо и нужны)) 24](#_Toc483256370)

[6 Інструкція користувача 32](#_Toc483256371)

[6.1 Призначення програми 32](#_Toc483256372)

[6.2 Вимоги до системи 32](#_Toc483256373)

[6.3 Інструкція по роботі з програмою 32](#_Toc483256374)

[7 Аналіз результатів 34](#_Toc483256375)

[Висновки 38](#_Toc483256376)

[Перелік посилань 39](#_Toc483256377)

[Додатки 40](#_Toc483256378)

[Додаток А 40](#_Toc483256379)

[Додаток Б 43](#_Toc483256380)

[Б.1 MyForm.h 44](#_Toc483256381)

[Б.2 MyForm.cpp 52](#_Toc483256382)

[Б.3 stdafx.h 52](#_Toc483256383)

[Б.4 GoldenSectionSearch.h 53](#_Toc483256384)

[Б.5 GoldenSectionSearch.cpp 53](#_Toc483256385)

[Б.6 UniformSearch.h 55](#_Toc483256386)

[Б.7 UniformSearch.cpp 55](#_Toc483256387)

[Б.8 Dichotomy.h 56](#_Toc483256388)

[Б.9 Dichotomy.cpp 57](#_Toc483256389)

[Б.10 functions.h 58](#_Toc483256390)

[Б.11 functions.cpp 60](#_Toc483256391)

# Вступ

Завданням даної курсової роботи є пошук екстремумів функції однієї змінної на заданому відрізку. Дана задача була розв’язана наступними методами: методом «золотого» перетину [1], рівномірного пошуку [2] та діхотомії [1],.

У випадку наявності кількох локальних максимумів чи мінімумів функції на заданому її проміжку програма видає лише один з них, оскільки вищеописані методи працюють лише для проміжків, на яких наявний лише один максимум та мінімум.

Дане програмне забезпечення може широко використовуватись для тестування швидкості роботи кожного з методів в залежності від введених параметрів та порівняння цих методів між собою за критеріями ефективність та швидкість роботи.

# Постановка задачі

Розробити програмне забезпечення для пошуку екстремумів функції однієї змінної. Програма буде знаходити екстремуми наступними методами:

* метод "золотого" перетину;
* метод рівномірного пошуку;
* метод діхотомії (половинного ділення)

Вхідними даними є сама функція, область пошуку її екстремумів та точність обчислень. Ці дані вводяться користувачем з клавіатури.

Вихідними даними даної програми є вивід на екран двох значень: мінімуму і максимуму даної функції на заданому проміжку, графік даної функції, збереження результату у файл (за бажанням користувача), виведення кількості ітерацій алгоритму та час його роботи. В разі, якщо мінімум і максимум функції співпадають (наприклад: y = 1) програма виводить відповідне повідомлення та лише одне значення, що є мінімумом і максимумом функції одночасно.

# Теоретичні викладки

## Метод «золотого» перетину

Метод золотого перетинуЗолотим перетином відрізка [a; b] називається поділ його точкою Метод золотого перетину на дві нерівні частини таким чином, щоб відношення усього відрізка до більшої частини дорівнювало відношенню більшої частини до меншої, тобто (число r називають золотим відношенням, значення якого є відомим: .

Рисунок 2.1 - Ілюстрація методу золотого перетину

Опишемо алгоритм мінімізації функції однієї змінної f(x), на відрізку [a;b]  використовуючи метод золотого перетину. Для цього, початковий відрізок [a;b]  ділимо точками c (перша точка) і d (друга точка) за правилом золотого перетину.

Далі, обчислюємо значення функцій f(c) і f(d). Порівняння цих значень дозволяє відкинути інтервал [a;c], при умові, що f(c)>f(d), або інтервал [d;b], якщо f(d)>f©. Довжина відрізка, що залишився, зменшиться у r разів.

Після цього процес повторюємо, тобто на інтервалі, що залишився уже є одна точка, що робить його золотий перетин: c є друга точка золотого перетину відрізка [c;b], а d — перша точка золотого перетину відрізка [c;b]. Знаючи одну з точок золотого перетину, іншу можна знайти за однією із вищезгаданих формул та обчислити значення f(x) у знову знайденій точці (значення в іншій точці вже обчислено на попередньому кроці).

Таким чином, на кожному кроці, починаючи з другого, потрібно лише одне обчислення функції f(x), і інтервал невизначеності зменшується в r разів: .

Після виконання Метод золотого перетину кроків довжина інтервалу невизначеності дорівнюватиме:

Процес обчислень за методом золотого перетину продовжуємо до тих пір, поки довжина інтервалу невизначеності не стане меншою деякого заданого числа Ɛ (тобто, до тих пір, поки для деякого n не буде виконуватись умова ).

## Метод рівномірного пошуку

Нехай потрібно розв'язати задачу на відшукання безумовного мінімум унімодальної функції однієї змінної f(x), тобто знайти таку точку , що . Нагадаємо, що функція f(x) називається унімодальною на інтервалі , якщо на даному інтервалі вона досягає свого глобального мінімуму в єдиній точці , причому ліворуч від  ця функція строго спадає, а праворуч — строго зростає. Для розв'язку поставленої задачі на практиці, як правило, застосовують наближені методи. Вони дозволяють знайти рішення цієї задачі з необхідною точністю в результаті визначення кінцевого числа значень функції f(x) і її похідних в деяких точках відрізка . Методи, які використовують тільки значення функції і які не потребують обчислення її похідних, називаються прямими методами мінімізації.

Великою перевагою прямих методів є те, що від цільової функції не потрібно диференційовності і, більше того, вона може бути не задана в аналітичному вигляді. Єдине, на чому грунтуються алгоритми прямих методів мінімізації, це можливість визначення значень функції f(x) в заданих точках. До найбільш поширени прямих методів можна віднести метод рівномірного пошуку, метод ділення відрізка навпіл, метод золотого перетину та метод Фібоначі. Сьогодні розглянемо перший з них.

Отже, основна ідея методу рівномірного пошуку (є найпростішим серед перерахованих вище методів) полягає в тому, що інтервал невизначеності  розбивається на n рівновіддалених точок, та знаходиться значення функції в них. Після цього, шляхом порівняння величин   знаходиться деяка точку , в якій значення функції найменше. Дане значення вказує на те, що шукана точка мінімуму   належить інтервалу ,.

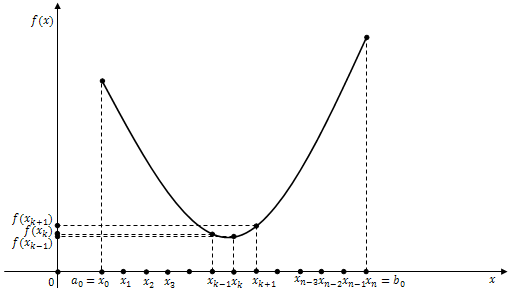


Рисунок 2.2 - Графічне представлення методу рівномірного пошуку

Далі, на основі вищесказаного сформулюємо більш детальний алгоритм розглядуваного методу, після чого застосуємо його для мінімізації деякої функції на заданому інтервалі.

## Метод діхотомії або Метод половинного ділення

Нехай дано рівняння f(x)=0. Необхідно знайти його корінь з деякою точністю на відрізку [a,b], на якому функція безперервна і у кінцях має значення різних знаків, тобто f(a)×f(b)<0. Таким чином, на цьому відрізку існує хоча б один розв’язок рівняння.

Знаходиться середина відрізку [a,b] точка с. Корінь може опинитись на відрізку [a,с] або на [с,b], чи співпасти з с. В останньому випадку метод припиняє роботу, інакше за допомогою перевірки виконання умов f(a)×f(c)<0 і f(c)×f(b)<0 з’ясовується, на якій частині відрізку залишився корінь. Далі процедура повторюється для тієї половини відрізку, на якій є корінь, доки відрізок не зменшиться настільки, що його довжина буде менше від заданої похибки.

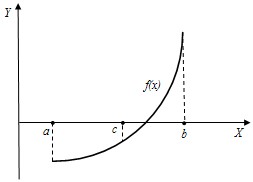


Рисунок 2.3 - Метод половинного ділення

# Опис алгоритмів

## Загальний алгоритм

1. ПОЧАТОК
2. Зчитати функцію, діапазон пошуку екстремумів, точність обрахунків і метод розв’язання.
   1. Зчитати введену функцію як рядок.
   2. Зчитати нижню межу діапазону пошуку екстремумів.
      1. ЯКЩО нижня межа діапазону – число, ТО записати його у відповідну комірку пам’яті. ІНАКШЕ видати повідомлення про помилку і пропозицією ввести число ще раз.
   3. Зчитати верхню межу діапазону пошуку екстремумів.
      1. ЯКЩО верхня межа діапазону – число, ТО записати його у відповідну комірку пам’яті. ІНАКШЕ видати повідомлення про помилку і пропозицією ввести число ще раз.  
         2.4. Зчитати точність обчислень.
   4. Зчитати точність обчислень.
      1. ЯКЩО точність обчислень – число, ТО записати його у відповідну комірку пам’яті. ІНАКШЕ видати повідомлення про помилку і пропозицією ввести число ще раз.
   5. Зчитати метод розв’язання.
      1. ЯКЩО обраний метод «золотого» перетину, ТО знайти мінімальне і максимальне значення функції на проміжку методом «золотого» перетину.
      2. ЯКЩО обраний метод рівномірного пошуку, ТО знайти мінімальне і максимальне значення функції на проміжку методом рівномірного пошуку.
      3. ЯКЩО обраний метод діхотомії, ТО знайти мінімальне і максимальне значення функції на проміжку методом діхотомії.
3. Вивести результати.
   1. ЯКЩО мінімум функції відмінний від її максимума, ТО вивести мінімум і максимум функції. ІНАКШЕ вивести повідомлення, що вони співпадають та вивести відповідне значення функції.
   2. Вивести кількість ітерацій методу, обраного користувачем, та час його роботи.
   3. ЯКЩО користувач вибрав опцію "зберегти у файл", ТО зберегти результати у файл.
4. КІНЕЦЬ

## Метод «золотого» перетину

1. ПОЧАТОК
2. КРОК 1
   1. Задати початкові межі відрізка a, b та точність ε.
3. КРОК 2
   1. Розрахувати початкові точки ділення: , , де , і значення в них цільової функції: , .
   2. ЯКЩО (або для пошуку максимуму – ), то . ІНАКШЕ .
4. КРОК 3
   1. ЯКЩО , то і зупинитися. ІНАКШЕ повернутися до кроку 2.
5. КІНЕЦЬ

## Метод рівномірного пошуку

1. ПОЧАТОК
2. КРОК 1
   1. Задати початкові межі відрізка a, b та точність ε.
   2. Обчислити кількість ітерацій алгоритму: .
3. КРОК 2
   1. Розбити відрізок на рівних частин точками ділення:
   2. Обчисливши значення цільової функції f(x) в точках , знайти шляхом порівняння точку , m – число від 1 до n таку, що ().
4. КІНЕЦЬ

## Метод діхотомії

1. ПОЧАТОК
2. КРОК 1
   1. Задати початкові межі відрізка a, b та точність ε.
3. КРОК 2
   1. Розбити відрізок навпіл та взяти дві точки, симетричні відносно центру: та , де – деяке число в інтервалі .
   2. Обчислити значення цільової функції f(x) в цих точках: , .
   3. ЯКЩО (або для пошуку максимуму – ), то . ІНАКШЕ .
4. КРОК 3
   1. ЯКЩО , то і зупинитися. ІНАКШЕ повернутися до кроку 2.
5. КІНЕЦЬ

# Опис програмного забезпечення

## Опис функціональної структури програмного забезпечення

Проект містить головний файл, 4 файли реалізації, 5 заголовних файлів та заголовний файл для реалізації інтерфейсу.

Функціональна структура програмного забезпечення представлена на рисунку 4.1.

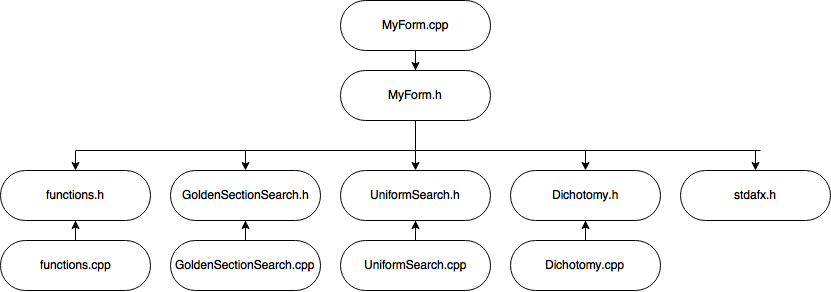


Рисунок 4.1 - Функціональна структура програмного забезпечення

## Опис функцій частин програмного забезпечення

### Користувацькі функції

Особливості та деталі кожної функції наведені у таблиці 4.1.

Таблиця 4.1 - Специфікація функцій

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Наз.  к-су | Назва функції | Призначення функції | Опис вхідних парам-в | Опис вихідних параметрів | Заголовний файл |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 |  | GoldenSectionSearch | Знаходити мінімум і максимум функції на відрізку методом «золотого» перетину. | s – вхідна функція; a - нижня межа діапазону задання функції; b- верхня межа | Рядок, що містить точки, в яких приймає мінімальне і максимальне значення та значення | GoldenSectionSearch.h |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  |  |  | діапазону задання функції; acc - точність обчислень | функції в них; кількість ітерацій алгоритму та час його роботи. |  |
| 2 |  | UniformSearch | Знаходити мінімум і максимум функції на відрізку методом рівномірного пошуку. | s – вхідна функція; a - нижня межа діапазону задання функції; b - верхня межа діапазону задання функції; acc - точність обчислень. | Рядок, що містить точки, в яких функція приймає мінімальне і максимальне значення та значення функції в них; кількість ітерацій алгоритму та час його роботи. | UniformSearch.h |
| 3 |  | Dichotomy |  | s – вхідна функція; a - нижня межа діапазону задання функції; b - верхня межа діапазону задання функції; acc - точність обчислень | Рядок, що містить точки, в яких функція приймає мінімальне і максимальне значення та значення функції в них; кількість ітерацій алгоритму та час його роботи. | Dichotomy.h |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 4 |  | PowerReplace | Замінює усі x в степені на добуток x, прибирає усі пробіли в вхідному рядку. | s – вхідна функція | Рядок, що не містить x в степені та пробілів. | functions.h |
| 5 |  | SaveToFile | Зберігає даний рядок до файлу "extremums.txt". | s – даний рядок |  | functions.h |
| 6 |  | ReplaceAll | Замінює в рядку всі входження одного підрядка на інші. | str – початковий рядок; from – підрядок, що потрібно замінити; to – підрядок, на який потрібно замінити | Рядок, в якому всі входження підрядка from замінені на підрядок to. | functions.h |
| 7 |  | ToString | Перетворює дійсне число на рядок. | a – дане число; n – кількість знаків після коми | Рядок, що утворений з дійсного числа. | functions.h |
| 8 |  | FindMaxValue | Знаходить максимальне значення серед елементів масиву. | arr – даний масив; n – довжина масиву | Максимальне значення серед елементів масиву. | functions.h |
| 9 |  | FindMinValue | Знаходить мінімальне значення серед елементів масиву. | arr – даний масив; n – довжина масиву | Мінімальне значення серед елементів масиву. | functions.h |
| 10 |  | ShuntingAlgorithm | Допомагає обчислити введену як рядок функцію в певній точці. | s – даний рядок; x – точка, в якій необхідно обчислити | Список з чисел і операторів. | functions.h |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  |  |  | значення функції. |  |  |
| 11 |  | ReversePolish | Рахує список з чисел і операторів, що повертає функція ShuntingAlgorithm. | queue – список з чисел і операторів | Значення порахованого списку. | functions.h |
| 12 |  | f | Знаходить значення функції, що задана рядком у певній точці. | s – дана функція; x – точка, в якій необхідно порахувати значення функції | Значення функції у точці. | functions.h |
| 13 |  | RoundTo | Округлює дане число до певної кількості цифр після коми. | a – дане число; n – кількість знаків після коми | Округлене число. | functions.h |
| 14 |  | CheckFunction | Перевіряє чи вхідна функція має коректний формат. | s – дана функція | Коректний формат чи ні. | functions.h |
| 15 |  | IsNumber | Перевіряє чи даний рядок є числом. | s – даний рядок | Є числом чи ні. | functions.h |

### Стандартні функції

Стандартні функції, що використані у проекті наведені у таблиці 4.2.

Таблица 4.2 – Специфікація функцій

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Бібліотека |
| 1 | 2 | 3 | 4 |
| 1 | msclr::interop::marshal\_as<std::string>() | Перетворює String^ у std::string. | msclr\marshal\_cppstd.h |
| 2 | Convert::ToDouble() | Перетворює String^ у double. |  |
| 1 | 2 | 3 | 4 |
| 3 | msclr::interop::marshal\_as<String^> | Перетворює string у String^. | msclr\marshal\_cppstd.h |
| 4 | pow(a, b) | Підносить a до степеня b. | cmath |
| 5 | clock() | Отримує поточний час. | ctime |
| 6 | abs(a) | Отримує модуль числа a. | cmath |
| 7 | double | Перетворює ціле чило у дійсне. |  |
| 8 | std::to\_string | Перетворює число будь-якого типу у std::string. | string |
| 9 | arr.push\_back(a) | Додає на кінець масиву arr елемент a. | vector |
| 10 | a.length() | Визначає довжину рядка a. | string |
| 11 | std::find() | Шукає у певних межах масиву деяке значення. | vector |
| 12 | arr.begin() | Повертає перший елемент масиву arr. | vector |
| 13 | arr.end() | Повертає останній елемент масиву arr. | vector |
| 14 | isdigit(a) | Перевіряє чи є цифрою певний символ a. |  |
| 15 | std::ofstream a | Зв’язує файловий потік з деякою змінною a. | fstream |
| 16 | a.close() | Закриває файл | fstream |
| 17 | std::round(a) | Округлює число a до цілих. | cmath |
| 18 | a.substr(b, c) | З рядка a з позиції b до c “вирізає” підрядок. | string |
| 19 | a.replace(b, c, d) | В рядку a замінює підрядок з позиції b до c на рядок d. | string |
| 20 | a.erase(b, c) | В рядку a очищує підрядок з позиції b до c. | string |
| 21 | a.insert(b, c) | Вставляє в рядок до перед позицією b підрядок c. | string |
| 22 | a.pop\_back() | Видаляє останній елемент списку. | list |
| 23 | a.empty() | Перевіряє чи є порожнім список a. | list |

# Тестування програмного забезпечення

## План тестування

Тестування програми відбувається після її розробки для виявлення помилок в логіці програми та її реалізації. Наведений нижче план тестування перевіряє усі “слабкі місця” програми.

1. Тестування правильності введених значень.
   1. Тестування на введення некоректних символів.
   2. Тестування на введення замалих та завеликих значень.
   3. Тестування на введення нульового проміжку зміни функції.
2. Тестування коректної роботи при введені функції, що не залежить від x (паралельна до осі Ox).
3. Тестування коректності роботи методів “золотого” перетину, рівномірного пошуку та діхотомії.
   1. Перевірка коректності роботи методу “золотого” перетину.
   2. Перевірка коректності роботи методу рівномірного пошуку.
   3. Перевірка коректності роботи методу діхотомії.
4. Тестування коректності роботи методів “золотого” перетину, рівномірного пошуку та діхотомії з дробовим початком та кінцем проміжку задання функції.
   1. Перевірка правильності результатів методу.
   2. Перевірка, що час роботи методу не залежить від того, чи початок та кінець проміжку задання функції є цілим або дробовим числом.
5. Тестування побудови графіків.

## Приклади тестування

Тестування програмного забезпечення було проведено згідно з планом, наведеним у підрозділі 5.1. Були отримані наступні результати.

1. Тестування правильності введених значень.

При введенні в поля для вводу даних некоректних символів, завеликих чи замалих значень та нульового проміжку зміни функції були отримані відповідні повідомлення про помилки. Результати роботи програми наведені у таблицях 5.1, 5.2 та 5.3. Також на малюнках 5.1, 5.2 та 5.3 можна побачити відповідні скріншоти.

Таблиця 5.1 - Тестування на введення некоректних символів

|  |  |
| --- | --- |
| Мета тесту | Тестування на введення некоректних символів |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x - n)^2 |
| Схема проведення тесту | Введення символів, що не є елементами функції |
| Очікуваний результат | Повідомлення про помилку  формату даних |
| Стан програми після проведення випробувань | Видано помилку «Input function has incorrect format, try enter it again.» |

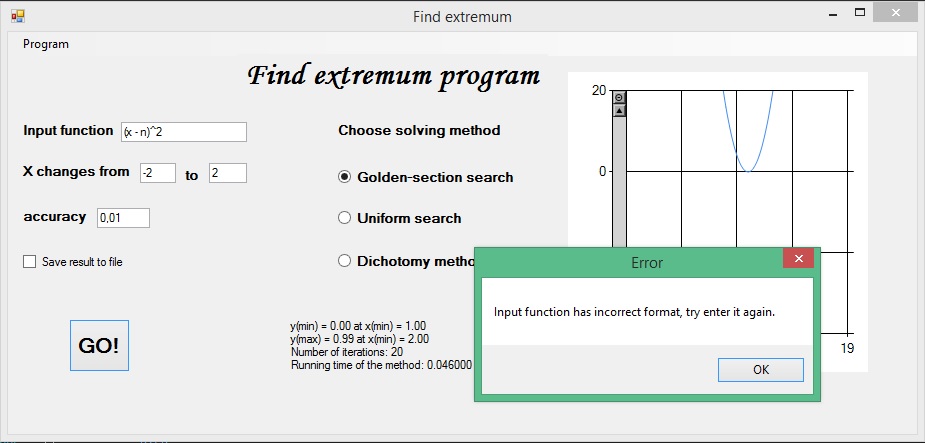


Рисунок 5.1 - Тестування на введення некоректних символів

Таблиця 5.2 - Тестування на введення завеликих значень

|  |  |
| --- | --- |
| Мета тесту | Тестування на завеликих значень |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | -2000 |
| Схема проведення тесту | Введення значень, що можуть призвести до надто довгої роботи програми |
| Очікуваний результат | Повідомлення про введення завеликого числа |
| Стан програми після проведення випробувань | Видано помилку «X-range is too big, enter another range.» |

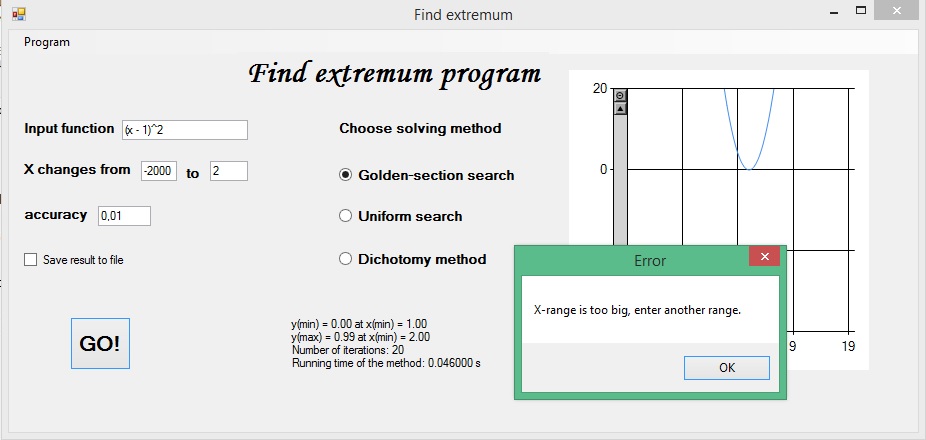


Рисунок 5.2 - Тестування на введення завеликих значень

Таблиця 5.3 - Тестування на введення нульового проміжку зміни функції

|  |  |
| --- | --- |
| Мета тесту | Тестування на введення нульового проміжку зміни функції |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 1 та 1 |
| Схема проведення тесту | Введення рівних початку і кінця відрізку задання функції |
| Очікуваний результат | Повідомнення про введення нульового проміжку зміни функції |
| Стан програми після проведення випробувань | Видано помилку «X-range can’t be equal to zero, enter different range.» |

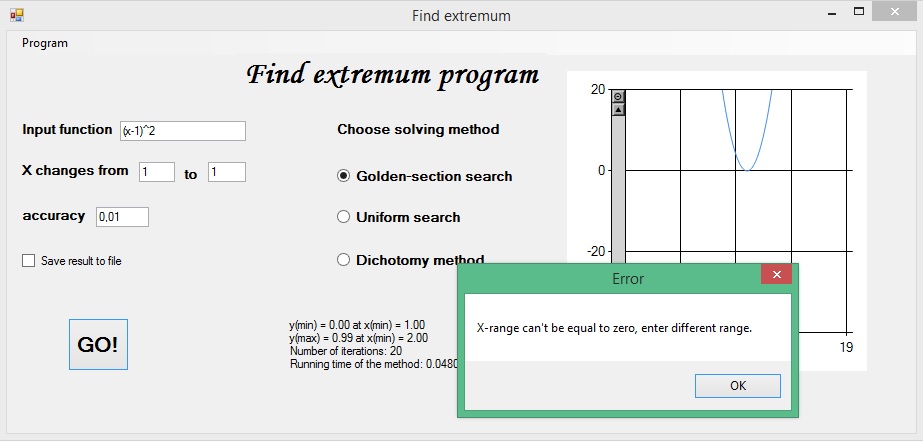


Рисунок 5.3 - Тестування на введення нульового проміжку зміни функції

1. Тестування коректної роботи при введені функції, що не залежить від x наведено на рисунку 5.4 та у таблиці 5.4.

Таблиця 5.4 - Введення функції, що не залежить від аргументу

|  |  |
| --- | --- |
| Мета тесту | Введення функції, що не залежить від аргументу |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 8 |
| Схема проведення тесту | Введення функції, що містить лише числа |
| Очікуваний результат | Виведення повідомлення про співпадання максимуму та мімімуму функції на відрізку та виведення відповідного значення. |
| Стан програми після проведення випробувань | Виведено результат «Function is independent of x. So f(min) = f(max) = 8.00» |

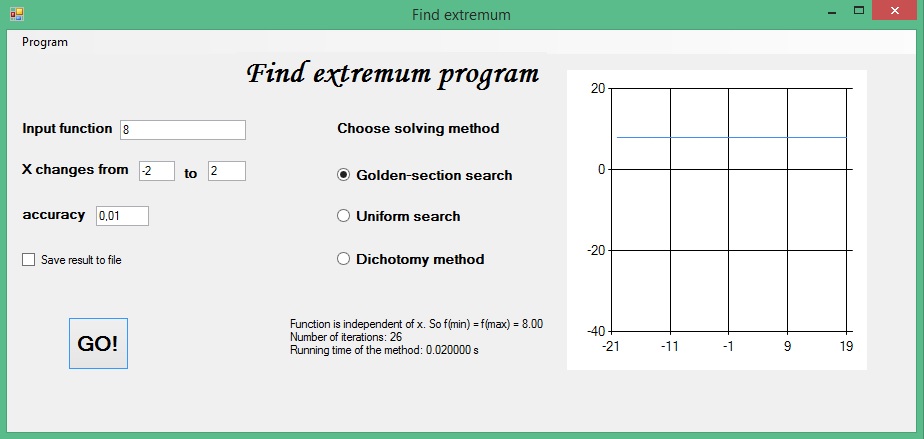


Рисунок 5.4 - Введення функції, що не залежить від аргументу

1. Тестування коректності роботи методів “золотого” перетину, рівномірного пошуку та діхотомії – рисунки 5.5, 5.6, 5.7 та таблиці 5.5, 5.6, 5.7 відповідно.

Таблиця 5.5 - Метод "золотого" перетину

|  |  |
| --- | --- |
| Мета тесту | Переконатися в коректності роботи методу «золотого» перетину |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x-1)^2, -2 та 2, 0.01 |
| Схема проведення тесту | Введення деякої функції, проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Виведено результат «y(min) = 0.00 at x(min) = 1.00; y(max) = 8.98 at x(min) = -2.00» |

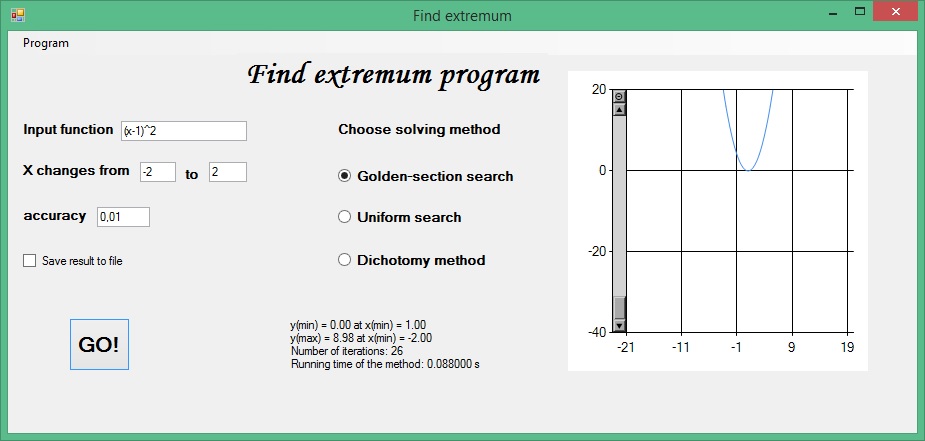


Рисунок 5.5 - Метод "золотого" перетину

Таблиця 5.6 - Метод рівномірного пошуку

|  |  |
| --- | --- |
| Мета тесту | Переконатися в коректності роботи методу рівномірного пошуку |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x-1)^2, -2 та 2, 0.01 |
| Схема проведення тесту | Введення деякої функції, проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Виведено результат «y(min) = 0.00 at x(min) = 1.00; y(max) = 8.94 at x(min) = -1.99» |

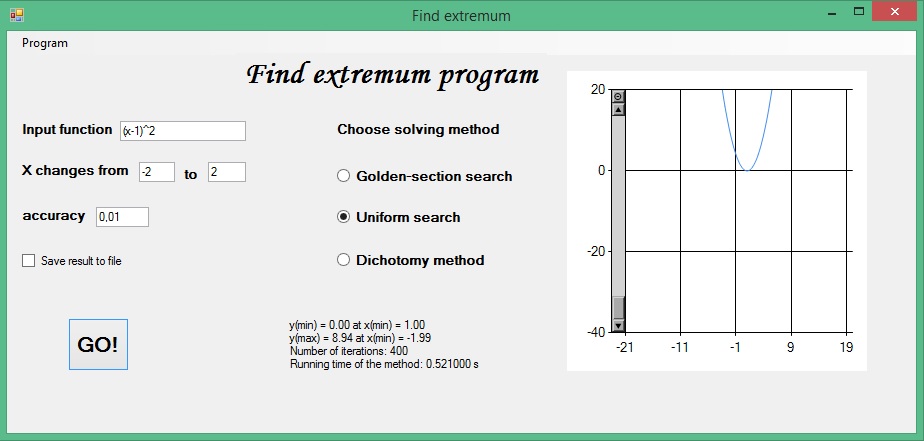


Рисунок 5.6 - Метод рівномірного пошуку

Таблиця 5.7 - Метод діхотомії

|  |  |
| --- | --- |
| Мета тесту | Переконатися в коректності роботи методу діхотомії |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x-1)^2, -2 та 2, 0.01 |
| Схема проведення тесту | Введення деякої функції, проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Виведено результат «y(min) = 0.00 at x(min) = 1.00; y(max) = 8.98 at x(min) = -2.00» |

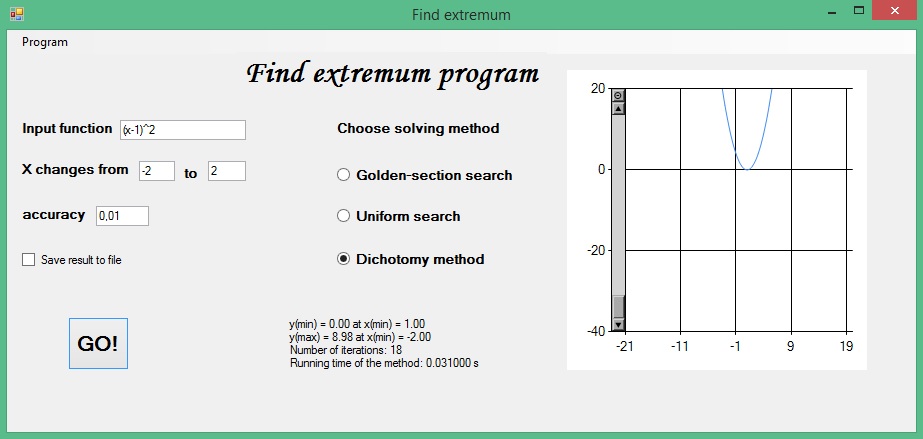


Рисунок 5.7 - Метод діхотомії

1. Тестування коректності роботи методів “золотого” перетину, рівномірного пошуку та діхотомії з дробовим початком та кінцем проміжку задання функції (рисунки 5.8 - 5.12 та таблиці 5.8 - 5.12).

Таблиця 5.8 - Метод "золотого" перетину

|  |  |
| --- | --- |
| Мета тесту | Переконатися в коректності роботи методу «золотого» перетину |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x-1)^2, 0.7 та 2.5, 0.01 |
| Схема проведення тесту | Введення деякої функції, дробового проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Виведено результат «y(min) = 0.00 at x(min) = 1.00; y(max) = 2.24 at x(min) = 2.50» |

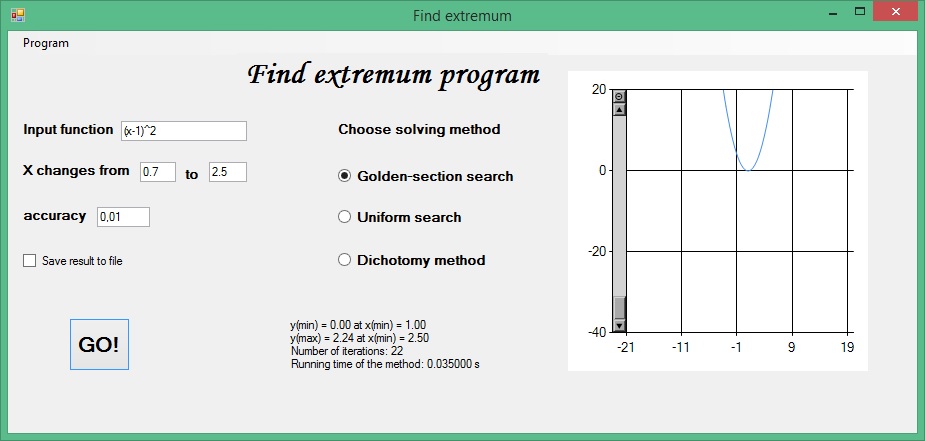


Рисунок 5.8 - Метод "золотого" перетину

Таблиця 5.9 - Метод рівномірного пошуку

|  |  |
| --- | --- |
| Мета тесту | Переконатися в коректності роботи методу рівномірного пошуку |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x-1)^2, 0.7 та 2.5, 0.01 |
| Схема проведення тесту | Введення деякої функції, дробового проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Виведено результат «y(min) = 0.00 at x(min) = 1.00; y(max) = 2.22 at x(min) = 2.49» |

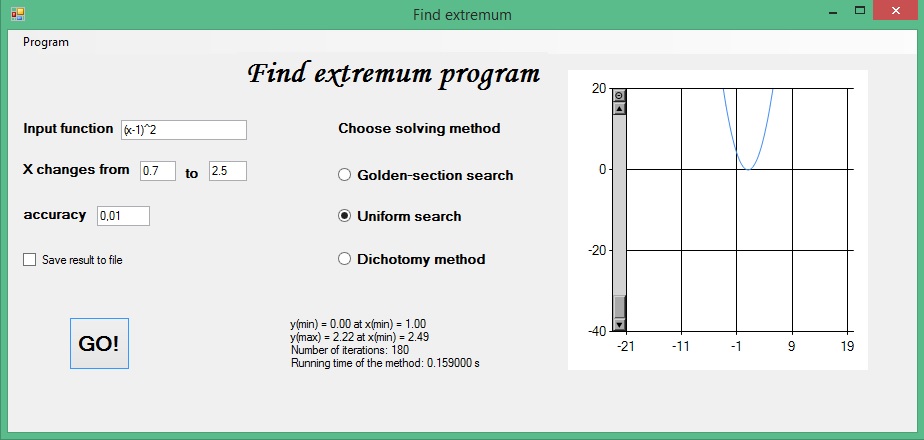


Рисунок 5.9 - Метод рівномірного пошуку

Таблиця 5.10 - Метод діхотомії

|  |  |
| --- | --- |
| Мета тесту | Переконатися в коректності роботи методу діхотомії |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x-1)^2, 0.7 та 2.5, 0.01 |
| Схема проведення тесту | Введення деякої функції, дробового проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Виведено результат «y(min) = 0.00 at x(min) = 1.00; y(max) = 2.22 at x(min) = 2.49» |

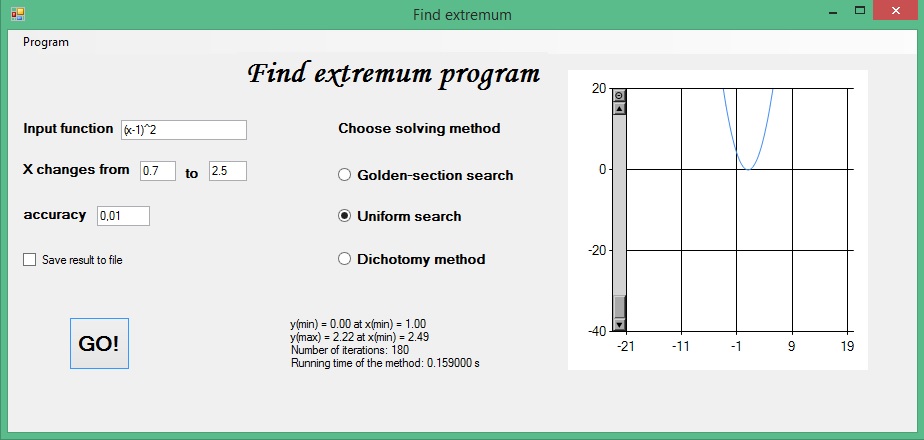


Рисунок 5.10 - Метод діхотомії

Таблиця 5.11 - Робота методу при введенні проміжку з дробових чисел

|  |  |
| --- | --- |
| Мета тесту | Переконатися в коректності роботи методу діхотомії при введенні проміжку з дробових чисел |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x-1)^2, -2.0 та 2.0, 0.01 |
| Схема проведення тесту | Введення деякої функції, дробового проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Виведено результат «y(min) = 0.00 at x(min) = 1.00; y(max) = 8.98 at x(min) = -2.00» |

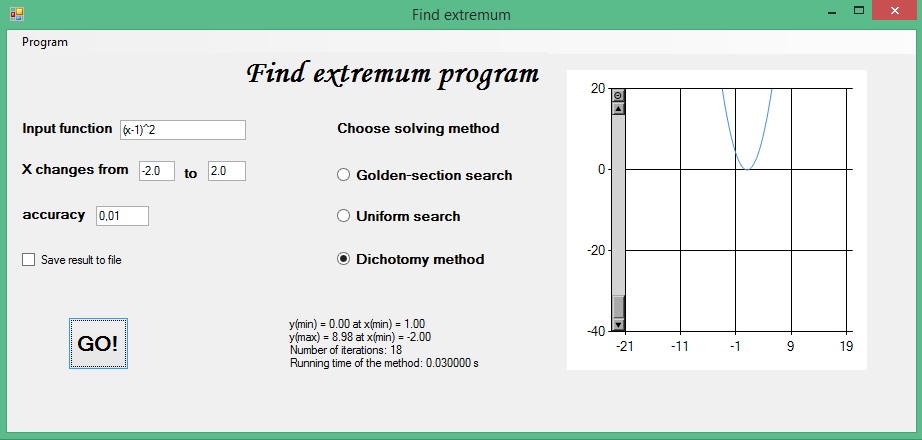


Рисунок 5.11 - Робота методу при введенні проміжку з дробових чисел

Таблиця 5.12 - Робота методу при введенні проміжку з цілих чисел

|  |  |
| --- | --- |
| Мета тесту | Переконатися в незмінності результату роботи методу при введенні дробового та цілого проміжку зміни функції |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | (x-1)^2, -2 та 2, 0.01 |
| Схема проведення тесту | Введення деякої функції, проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Виведено результат «y(min) = 0.00 at x(min) = 1.00; y(max) = 8.98 at x(min) = -2.00» |

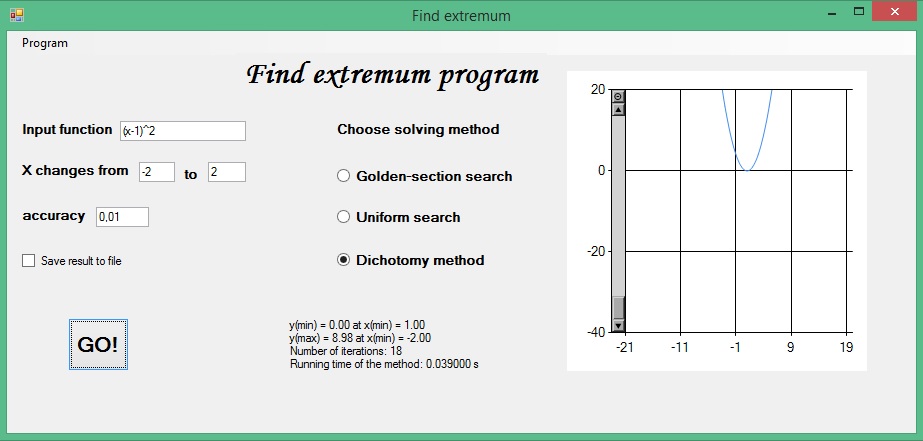


Рисунок 5.12 - Робота методу при введенні проміжку з цілих чисел

1. Тестування побудови графіків.

Програма правильно будує будь-які типи графіків функцій, що залежить від однієї змінної у разі введення коректних символів (проходження тесту 1). Результат наведений на рисунку 5.13 та у таблиці 5.13.

Таблиця 5.13 - Тестування побудови графіків функцій

|  |  |
| --- | --- |
| Мета тесту | Переконатися в правильності побудови вхідної функції |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | x^3, -2 та 2, 0.01 |
| Схема проведення тесту | Введення деякої функції, проміжку її задання та точності обчислень. |
| Очікуваний результат | Виведення коректних значень мінімуму та максимуму функції на відрізку |
| Стан програми після проведення випробувань | Побудовано правильний графік функції y = x^3 |

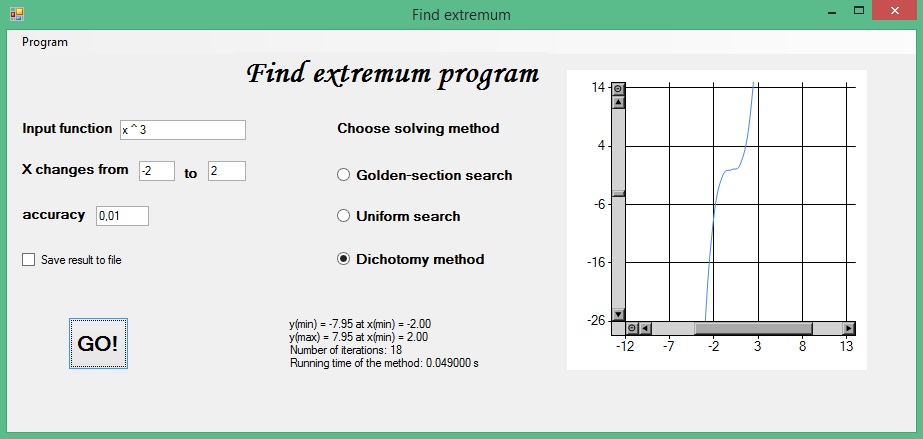


Рисунок 5.13 - Тестування побудови графіків функцій

# Інструкція користувача

## Призначення програми

Передусім призначенням програми є знаходження екстремумів функції у певному діапазоні. Також дану програму можна використовувати для вивчення методів “золотого” перетину, рівномірного пошуку та діхотомії, оскільки програма видає швидкість роботи алгоритму та кількість його ітерацій, що дає змогу порівнювати алгоритми за критерієм швидкості та ефективності розв’язання поставленої задачі.

## Вимоги до системи

* Intel® Core 2 або AMD Athlon® 64 процесор; частота процесора 2 GHz або швидше
* Microsoft Windows 7 with Service Pack 1, Windows 8.1, or Windows 10
* 2 GB або більше RAM (8 GB рекомендовано)
* 2 GB або більше вільного місця на диску
* 1024 x 768 дисплей (1280x800 рекомендовано)
* .NET Framework

## Інструкція по роботі з програмою

Інтерфейс програми наведений на рисунку 6.1.

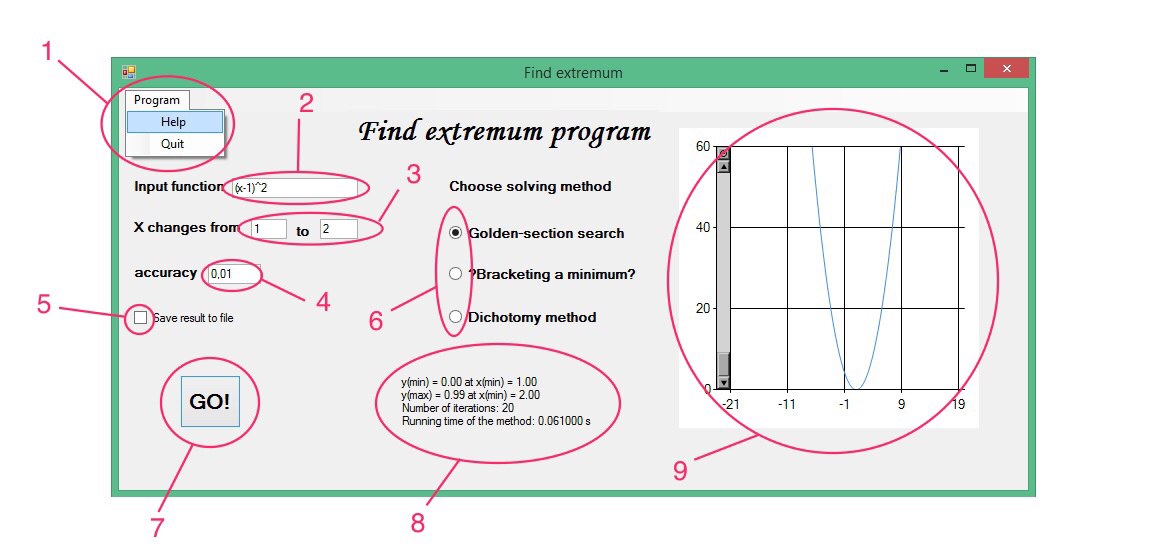


Рисунок 6.1 - Інтерфейс програми

1 – меню програми

2 – поле для введення функції

3 – межі функції

4 – точність обчислень

5 – необхідно поставити «галочку» для збереження результатів до файлу

6 – вибір методу для знаходження екстремумів функції

7 – запуск програми

8 – отримані результати та додаткова інформація щодо роботи програми

9 – поле для побудови графіку функції з пункту 2

Для використання програми користувачеві необхідно задати функцію, межі, в яких будуть шукатися екстремуми, точність обчислення методу та сам метод пошуку екстремумів. Після цього натиснути клавішу «Go». Програма виведе мінімальне та максимальне значення функції на даному відрізку, значення аргументу в цих точках, а також, як додаткову інформацію, кількість ітерацій обраного методу та час його роботи.

# Аналіз результатів

Головною задачею курсової роботи була реалізація програми для знаходження мінімального і максимального значення функції на заданому відрізку наступними методами: «золотого» перетину, рівномірного пошуку та діхотомії.

Оскільки точність обчислень вводиться користувачем, то вона може варіюватися в деяких межах, проте результати роботи програми ніколи не будуть ідеальними.

Критичні ситуації у роботі програми виявлені не були. Проте, в разі введення користувачем надто великої точності або дуже великого діапазону визначення функції програма може працювати дуже довго чи навіть може «зависнути», що необхідно враховувати при введенні відповідних даних.

Складність методів «золотого» перетину, рівномірного пошуку та діхотомії залежить від введеної точності, тому її не можна підрахувати загально для методу.

Перевірка коректності роботи алгоритмів була виконана за допомогою програми Microsoft Excel, де була використана функція «Пошук рішення» для підбору екстремумів функції. При цьому функція мала вигляд , проміжок задання функції був від -2 до 2, а точність мала значення 0,001. Результати роботи програми Microsoft Excel наведені на рисунку 7.1.

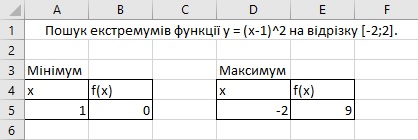


Рисунок 7.1 - Пошук екстремумів функції на відрізку за допомогою Microsoft Excel

А на рисунках 7.2, 7.3 та 7.4 можна побачити результат роботи розробленої програми методами “золотого” перетину, рівномірного пошуку та діхотомії відповідно.

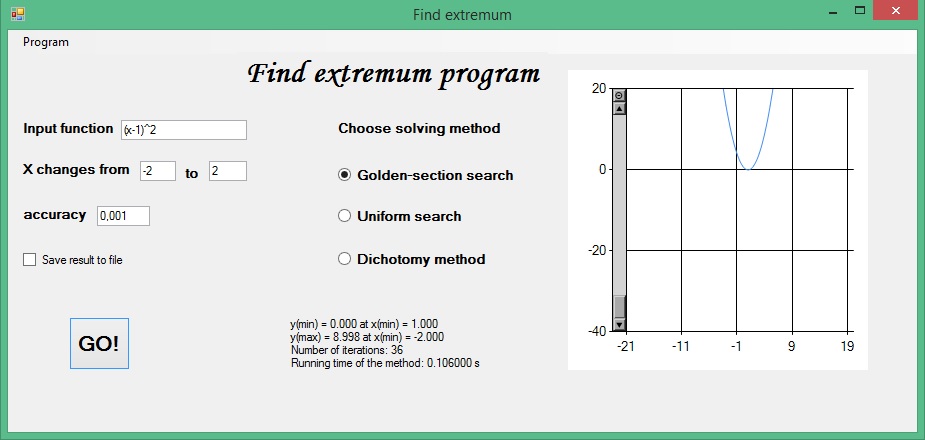


Рисунок 7.2 - Метод "золотого" перетину

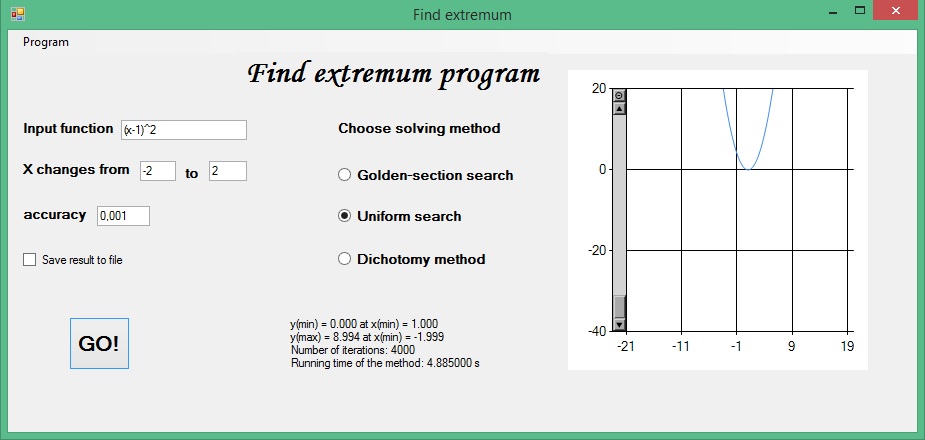


Рисунок 7.3 - Метод рівномірного пошуку

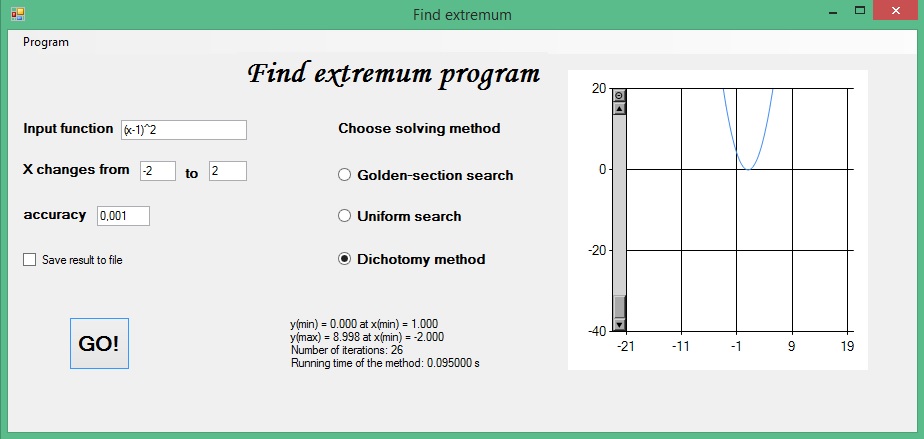


Рисунок 7.4 - Метод діхотомії

Результати тестування ефективності методів пошуку екстремумів функції на проміжку [-2;2] наведені у таблиці 7.1.

Таблиця 7.1 - Тестування методів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Точність обчислень | Параметри тестування | Метод | | |
| «Золотого» перетину | Рівномірного пошуку | Діхотомії |
| 0.1 | Кількість ітерацій | 16 | 40 | 8 |
| Кількість елементарних операцій | 384 | 240 | 144 |
| Час виконання методу (с) | 0,034 | 0,083 | 0,026 |
| 0.01 | Кількість ітерацій | 26 | 400 | 18 |
| Кількість елементарних операцій | 624 | 2400 | 216 |
| Час виконання методу (мс) | 0,054 | 0,456 | 0,038 |
| 0.001 | Кількість ітерацій | 36 | 4000 | 24 |
| Кількість елементарних операцій | 864 | 24000 | 288 |
| Час виконання методу (мс) | 0,059 | 1,229 | 0,057 |
| 0.0001 | Кількість ітерацій | 46 | 40000 | 32 |
| Кількість елементарних операцій | 1104 | 240000 | 384 |
| Час виконання методу (мс) | 0,074 | 36,341 | 0,063 |

# Висновки

На аналітичному етапі розробки програмного забезпечення були детально розглянуті та вивчені методи розв’язку даної задачі: метод «золотого» перетину, метод рівномірного пошуку та метод діхотомії.

Згодом, при проектуванні програмного забезпечення була «розібрана» структура проекту та його графічний інтерфейс [3], що дало змогу чітко розуміти особливості реалізації даного програмного забезпечення для його подальшої програмної реалізації.

Після завершення написання програми був проведений аналіз отриманих результатів на їх достовірність за допомогою сторонньої програми Microsoft Excel, що дало змогу переконатися у коректності роботи програмного забезпечення.

Оскільки метою моєї роботи був пошук екстремумів функції однієї змінної методами «золотого» перетину, рівномірного пошуку та діхотомії, то я вважаю, що досяг мети курсової роботи. У ході її виконання виявилося, що методи «золотого» перетину та діхотомії мають суттєво меншу кількість ітерацій та набагато швидше працюють за метод рівномірного пошуку. Проте перші два методи, на відміну від останнього можуть «застрягати» у локальних максимумах чи мінімумах на заданому відрізку, що призводить до некоректності роботи програми при в разі наявності кількох максимумів чи мінімумів функції на заданому відрізку та вибору методу «золотого» перетину чи діхотомії. Ця особливість не є помилкою програмної реалізації, оскільки згадані вище «швидкі» методи призначені для пошуку максимуму чи мінімуму функції на відрізку лише у разі наявності тільки одного максимуму та одного мінімуму.

# Перелік посилань

1. Лемешко Б.Ю. Методы оптимизации – Новосибирск: НГТУ, 2009. – 126 с.
2. Кокуев А. Г. Решение задач оптимизации в среде Matlab – Астрахань, 2009. – 29 с.
3. Hong K. C++ tutorial - UI application using Visual Studio 2015 [Електронний ресурс] – Режим доступу: http://nuovolabs.fauser.edu/~marfer/docs/2014/4BI/grafica\_c++/C++%20Tutorial\_%20UI%20Application%20using%20visual%20studio%20-%202015.pdf

# Додатки

## Додаток А

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 р.

Виконавець:

Студент Кушка Михайло Олександрович

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: Пошук екстремумів функції однієї змінної

з дисципліни:

«Основи програмування»

Київ 2017

1.1 Мета: Метою курсової роботи є розробка комплексу програм для пошуку мінімального і максимального значення функції на відрізку.

1.2 Найменування та галузь застосування об'єкта розробки:Дана робота присвячена розробці програмного забезпечення для пошуку мінімального і максимального значення функції на відрізку за допомогою методів «золотого» перетину, рівномірного пошуку та діхотомії, що може використовуватись в навчанні для вивчення швидкості та ефективності даних методів.

1.3 Підстава для проведення робіт:Підставою для розробки програмного забезпечення є навчальний план спеціальності 6.050301 «Програмна інженерія», робоча програма дисципліни „Основи програмування”, індивідуальне завдання.

1.4 Дата початку роботи: «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2017 р.

1.5 Дата закінчення роботи: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2017 р.

1.6 Призначення розробки: Функціональним призначенням програми є автоматизація процесу пошуку мінімального та максимального значення функції на відрізку з наперед заданою точністю. Програма може експлуатуватися в області математики та програмування для вивчення методів «золотого» перетину, рівномірного пошуку та діхотомії, для порівняння швидкості роботи та ефективності цих методів.

1.7 Вимоги до програми та програмної документації: Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ИСО 5807 - 85 ГОСТ на розробку програмних документів, схем алгоритмів програм, даних та систем.

ГОСТ 19.781 - 74 - Вимоги до розробки програмного забезпечення.

ГОСТ 19.101-77 (СТ СЭВ 1626 - 79) - Держстандарт на розробку програмної документації, видів програм та програмних документів.

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

1.8 Стадії та етапи розробки:

1. Аналіз методів вирішення поставленої задачі (до \_\_.\_\_.2017 р.)

2. Розробка сценарію роботи програми (до \_\_.\_\_.2017 р.)

3. Розробка алгоритмічного забезпечення (до \_\_.\_\_.2017 р.)

4. Розробка програмного забезпечення (до \_\_.\_\_.2017 р.)

5. Розробка інтерфейсу, планування, тестування розробленої програми (до \_\_.\_\_.2017 р.)

6. Розробка пояснювальної записки (до \_\_.\_\_.2017 р.).

7. Захист курсової роботи (до \_\_.\_\_.2017 р.).

1.9 Порядок контролю та приймання. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв її оцінювання.

## Додаток Б

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення вирішення задачі пошуку екстремумів функції однієї змінної*

(Вид носія даних)

*CD-R*

(Обсяг програми (документа), арк., Кб)

*23 арк, Кб*

*студента групи ІП-61 І курсу*

Кушки М.О.

1. MyForm.h

#pragma once

#include "stdafx.h"

#include "functions.h"

#include "GoldenSectionSearch.h"

#include "UniformSearch.h"

#include "Dichotomy.h"

namespace CourseWork {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ button1;

protected:

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::RadioButton^ radioButton1;

private: System::Windows::Forms::RadioButton^ radioButton2;

private: System::Windows::Forms::RadioButton^ radioButton3;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::CheckBox^ checkBox1;

private: System::Windows::Forms::DataVisualization::Charting::Chart^ graph;

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::ToolStripMenuItem^ programToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ helpToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ quitToolStripMenuItem;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

System::Windows::Forms::DataVisualization::Charting::ChartArea^ chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());

System::Windows::Forms::DataVisualization::Charting::Legend^ legend1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Legend());

System::Windows::Forms::DataVisualization::Charting::Series^ series1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

this->button1 = (gcnew System::Windows::Forms::Button());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->label6 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

this->radioButton1 = (gcnew System::Windows::Forms::RadioButton());

this->radioButton2 = (gcnew System::Windows::Forms::RadioButton());

this->radioButton3 = (gcnew System::Windows::Forms::RadioButton());

this->label2 = (gcnew System::Windows::Forms::Label());

this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());

this->graph = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->programToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->helpToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->quitToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->graph))->BeginInit();

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// button1

//

this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->Location = System::Drawing::Point(61, 287);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(61, 53);

this->button1->TabIndex = 0;

this->button1->Text = L"GO!";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(113, 90);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(126, 20);

this->textBox1->TabIndex = 1;

this->textBox1->Text = L"(x-1)^2";

this->textBox1->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox1\_TextChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Monotype Corsiva", 24, static\_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold | System::Drawing::FontStyle::Italic)),

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(229, 22);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(311, 39);

this->label1->TabIndex = 2;

this->label1->Text = L"Find extremum program";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(12, 90);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(98, 16);

this->label3->TabIndex = 4;

this->label3->Text = L"Input function";

this->label3->Click += gcnew System::EventHandler(this, &MyForm::label3\_Click);

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label4->Location = System::Drawing::Point(12, 131);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(114, 16);

this->label4->TabIndex = 5;

this->label4->Text = L"X changes from";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label5->Location = System::Drawing::Point(174, 135);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(21, 16);

this->label5->TabIndex = 6;

this->label5->Text = L"to";

//

// textBox2

//

this->textBox2->Location = System::Drawing::Point(132, 131);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(36, 20);

this->textBox2->TabIndex = 7;

this->textBox2->Text = L"1";

//

// textBox3

//

this->textBox3->Location = System::Drawing::Point(201, 131);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(38, 20);

this->textBox3->TabIndex = 8;

this->textBox3->Text = L"2";

//

// textBox4

//

this->textBox4->Location = System::Drawing::Point(89, 176);

this->textBox4->Name = L"textBox4";

this->textBox4->Size = System::Drawing::Size(53, 20);

this->textBox4->TabIndex = 9;

this->textBox4->Text = L"0,01";

//

// label6

//

this->label6->AutoSize = true;

this->label6->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label6->Location = System::Drawing::Point(12, 176);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(71, 16);

this->label6->TabIndex = 10;

this->label6->Text = L"accuracy";

//

// label7

//

this->label7->AutoSize = true;

this->label7->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label7->Location = System::Drawing::Point(327, 90);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(170, 16);

this->label7->TabIndex = 11;

this->label7->Text = L"Choose solving method";

//

// radioButton1

//

this->radioButton1->AutoSize = true;

this->radioButton1->Checked = true;

this->radioButton1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->radioButton1->Location = System::Drawing::Point(330, 135);

this->radioButton1->Name = L"radioButton1";

this->radioButton1->Size = System::Drawing::Size(182, 20);

this->radioButton1->TabIndex = 12;

this->radioButton1->TabStop = true;

this->radioButton1->Text = L"Golden-section search";

this->radioButton1->UseVisualStyleBackColor = true;

//

// radioButton2

//

this->radioButton2->AutoSize = true;

this->radioButton2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->radioButton2->Location = System::Drawing::Point(330, 176);

this->radioButton2->Name = L"radioButton2";

this->radioButton2->Size = System::Drawing::Size(130, 20);

this->radioButton2->TabIndex = 13;

this->radioButton2->Text = L"Uniform search";

this->radioButton2->UseVisualStyleBackColor = true;

//

// radioButton3

//

this->radioButton3->AutoSize = true;

this->radioButton3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->radioButton3->Location = System::Drawing::Point(330, 219);

this->radioButton3->Name = L"radioButton3";

this->radioButton3->Size = System::Drawing::Size(154, 20);

this->radioButton3->TabIndex = 14;

this->radioButton3->Text = L"Dichotomy method";

this->radioButton3->UseVisualStyleBackColor = true;

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(280, 287);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(107, 13);

this->label2->TabIndex = 15;

this->label2->Text = L"Answer will be here...";

//

// checkBox1

//

this->checkBox1->AutoSize = true;

this->checkBox1->Location = System::Drawing::Point(15, 222);

this->checkBox1->Name = L"checkBox1";

this->checkBox1->Size = System::Drawing::Size(107, 17);

this->checkBox1->TabIndex = 16;

this->checkBox1->Text = L"Save result to file";

this->checkBox1->UseVisualStyleBackColor = true;

//

// graph

//

chartArea1->AxisX->IsStartedFromZero = false;

chartArea1->AxisX->ScaleView->SizeType = System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Number;

chartArea1->Name = L"ChartArea1";

this->graph->ChartAreas->Add(chartArea1);

legend1->Enabled = false;

legend1->Name = L"Legend1";

this->graph->Legends->Add(legend1);

this->graph->Location = System::Drawing::Point(560, 40);

this->graph->Name = L"graph";

series1->ChartArea = L"ChartArea1";

series1->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline;

series1->Legend = L"Legend1";

series1->Name = L"Series1";

this->graph->Series->Add(series1);

this->graph->Size = System::Drawing::Size(300, 300);

this->graph->TabIndex = 17;

this->graph->Text = L"chart1";

//

// menuStrip1

//

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->programToolStripMenuItem });

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(909, 24);

this->menuStrip1->TabIndex = 18;

this->menuStrip1->Text = L"menuStrip1";

//

// programToolStripMenuItem

//

this->programToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->helpToolStripMenuItem,

this->quitToolStripMenuItem

});

this->programToolStripMenuItem->Name = L"programToolStripMenuItem";

this->programToolStripMenuItem->Size = System::Drawing::Size(65, 20);

this->programToolStripMenuItem->Text = L"Program";

//

// helpToolStripMenuItem

//

this->helpToolStripMenuItem->Name = L"helpToolStripMenuItem";

this->helpToolStripMenuItem->Size = System::Drawing::Size(99, 22);

this->helpToolStripMenuItem->Text = L"Help";

this->helpToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::helpToolStripMenuItem\_Click);

//

// quitToolStripMenuItem

//

this->quitToolStripMenuItem->Name = L"quitToolStripMenuItem";

this->quitToolStripMenuItem->Size = System::Drawing::Size(99, 22);

this->quitToolStripMenuItem->Text = L"Quit";

this->quitToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::quitToolStripMenuItem\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(909, 402);

this->Controls->Add(this->graph);

this->Controls->Add(this->checkBox1);

this->Controls->Add(this->label2);

this->Controls->Add(this->radioButton3);

this->Controls->Add(this->radioButton2);

this->Controls->Add(this->radioButton1);

this->Controls->Add(this->label7);

this->Controls->Add(this->label6);

this->Controls->Add(this->textBox4);

this->Controls->Add(this->textBox3);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->label1);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->button1);

this->Controls->Add(this->menuStrip1);

this->Name = L"MyForm";

this->Text = L"Find extremum";

this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->graph))->EndInit();

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

// axis X

graph->ChartAreas[0]->AxisX->ScaleView->Zoom(-20, 20);

graph->ChartAreas[0]->CursorX->IsUserEnabled = true;

graph->ChartAreas[0]->CursorX->IsUserSelectionEnabled = true;

graph->ChartAreas[0]->AxisX->ScaleView->Zoomable = true;

graph->ChartAreas[0]->AxisX->ScrollBar->IsPositionedInside = true;

// axis Y

graph->ChartAreas[0]->AxisY->ScaleView->Zoom(-40, 20);

graph->ChartAreas[0]->CursorY->IsUserEnabled = true;

graph->ChartAreas[0]->CursorY->IsUserSelectionEnabled = true;

graph->ChartAreas[0]->AxisY->ScaleView->Zoomable = true;

graph->ChartAreas[0]->AxisY->ScrollBar->IsPositionedInside = true;

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ func = textBox1->Text;

std::string ff = msclr::interop::marshal\_as<std::string>(func);

// input function has incorrect format

if (!CheckFunction(ff))

MessageBox::Show("Input function has incorrect format, try enter it again.", "Error");

else

{

// input x range and accuracy

// input a

bool error = false;

double a = 1, b = 2, acc = 0.01;

std::string help\_str = msclr::interop::marshal\_as<std::string>(textBox2->Text);

help\_str = ReplaceAll(help\_str, ".", ","); // replace '.' by ',' (3.14 (NaN) -> 3,14(Number))

if (IsNumber(help\_str))

{

a = Convert::ToDouble(msclr::interop::marshal\_as<String^>(help\_str));

error = false;

}

else

{

MessageBox::Show("Bottom x border is incorrect, try enter again.", "Error");

error = true;

}

// input b

help\_str = msclr::interop::marshal\_as<std::string>(textBox3->Text);

help\_str = ReplaceAll(help\_str, ".", ",");

if (IsNumber(help\_str))

{

b = Convert::ToDouble(msclr::interop::marshal\_as<String^>(help\_str));

error = false;

}

else

{

MessageBox::Show("Up x border is incorrect, try enter again.", "Error");

error = true;

}

// input acc

help\_str = msclr::interop::marshal\_as<std::string>(textBox4->Text);

help\_str = ReplaceAll(help\_str, ".", ",");

if (IsNumber(help\_str))

{

acc = Convert::ToDouble(msclr::interop::marshal\_as<String^>(help\_str));

error = false;

}

else

{

MessageBox::Show("Accuracy is incorrect, try enter again.", "Error");

error = true;

}

if (!error)

{

if (a == b)

MessageBox::Show("X-range can't be equal to zero, enter different range.", "Error");

else if (std::abs(a - b) > 1000)

MessageBox::Show("X-range is too big, enter another range.", "Error");

else if (std::abs(acc) > 0.1)

MessageBox::Show("Accuracy must be no more than 0.1, enter another accuracy.", "Error");

else

{

// uncorrect function boundaries

if (a > b)

{

double c = a;

a = b;

b = c;

}

// replace x^3 by x\*x\*x and so on

PowerReplace(ff);

// draw graph

graph->Series[0]->Points->Clear();

for (int i = -20; i < 20; ++i)

graph->Series[0]->Points->AddXY(i, f(ff, i));

// choose method

std::string result = "";

if (radioButton1->Checked)

result = GoldenSection::GoldenSectionSearch(ff, a, b, acc);

if (radioButton2->Checked)

result = Uniform::UniformSearch(ff, a, b, acc);

if (radioButton3->Checked)

result = Dichotomy::DichotomyMethod(ff, a, b, acc);

// save result to file

if (checkBox1->Checked)

{

SaveToFile(result);

MessageBox::Show("File \"extremums.txt\" was successfully created.", "Success");

}

label2->Text = msclr::interop::marshal\_as<String^>(result);

}

}

}

}

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label3\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void quitToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Application::Exit();

}

private: System::Void helpToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ info = "Info.";

MessageBox::Show(info, "Program information");

}

};

}

1. MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(array<String^>^ arg)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

CourseWork::MyForm form;

Application::Run(%form);

}

1. stdafx.h

/\*\*

\* Used to connect libraries that are used in the project.

\*/

#pragma once

#include <cmath>

#include <string>

#include <list>

#include <vector>

#include <map>

#include <fstream>

#include <ctime>

#include <msclr\marshal\_cppstd.h>

1. GoldenSectionSearch.h

/\*\*

\* Used to describe the prototype of the GoldenSectionSearch() function.

\*/

#pragma once

#include "stdafx.h"

#include "functions.h"

namespace GoldenSection

{

/\*\*

\* Golden selection search method, which finds min and max

\* value of the function on a range.

\*

\* @param s Given function to calculate.

\* @param a Lower bound of the function.

\* @param b Higher bound of the function.

\* @param acc Accuracy of the calculations.

\* @returns String, which contents result and information

\* about calculations.

\*/

std::string GoldenSectionSearch(std::string, double, double, double);

}

1. GoldenSectionSearch.cpp

#include "GoldenSectionSearch.h"

/\*\*

\* Golden selection search method, which finds min and max

\* value of the function on a range.

\*

\* @param s Given function to calculate.

\* @param a Lower bound of the function.

\* @param b Higher bound of the function.

\* @param acc Accuracy of the calculations.

\* @returns String, which contents result and information

\* about calculations.

\*/

std::string GoldenSection::GoldenSectionSearch(std::string s, double a, double b, double acc)

{

const double fi = (1 + pow(5, 0.5)) / 2; // golden section proportion

unsigned int iterations = 0, // number of iterations

start\_time = clock(); // start time of working method

double x1, // first x-value for golden section search

x2, // second x-value for golden section search

y1, // first f(x)-value for golden section search

y2, // second f(x)-value for golden section search

x\_min, // min x value

x\_max, // max x value

a\_original, // initial start of the range

b\_original, // initial end of the range

work\_time = 0; // method's work time

a\_original = a;

b\_original = b;

// find mininum of the function

do

{

x1 = b - (b - a) / fi;

x2 = a + (b - a) / fi;

y1 = f(s, x1);

y2 = f(s, x2);

if (y1 >= y2)

{

a = x1;

x1 = x2;

x2 = a + (b - a) / fi;

}

else

{

b = x2;

x2 = x1;

x1 = b - (b - a) / fi;

}

iterations++;

} while (abs(b - a) > acc);

x\_min = (a + b) / 2;

// find maximum of the function

a = a\_original;

b = b\_original;

do

{

x1 = b - (b - a) / fi;

x2 = a + (b - a) / fi;

y1 = f(s, x1);

y2 = f(s, x2);

if (y1 <= y2)

{

a = x1;

x1 = x2;

x2 = a + (b - a) / fi;

}

else

{

b = x2;

x2 = x1;

x1 = b - (b - a) / fi;

}

iterations++;

} while (abs(b - a) > acc);

x\_max = (a + b) / 2;

work\_time = double(clock() - start\_time) / CLK\_TCK;

// output result

int n = 0; // number of digits after comma

std::string result = ""; // result of method's work

while (acc < 1)

{

acc \*= 10;

n++;

}

if (x\_min == x\_max)

result = "Function is independent of x. So f(min) = f(max) = " + ToString(RoundTo(f(s, x\_max), n), n);

else

{

result = "y(min) = " + ToString(RoundTo(f(s, x\_min), n), n) + " at x(min) = " + ToString(RoundTo(x\_min, n), n);

result += "\ny(max) = " + ToString(RoundTo(f(s, x\_max), n), n) + " at x(min) = " + ToString(RoundTo(x\_max, n), n);

}

result += "\nNumber of iterations: " + std::to\_string(iterations);

result += "\nRunning time of the method: " + std::to\_string(work\_time) + " s";

return result;

}

1. UniformSearch.h

/\*\*

\* Used to describe the prototype of the UniformSearch() function.

\*/

#pragma once

#include "stdafx.h"

#include "functions.h"

namespace Uniform

{

/\*\*

\* Uniform search method, which finds min and max

\* value of the function on a range.

\*

\* @param s Given function to calculate.

\* @param a Lower bound of the function.

\* @param b Higher bound of the function.

\* @param acc Accuracy of the calculations.

\* @returns String, which contents result and information

\* about calculations.

\*/

std::string UniformSearch(std::string, double, double, double);

}

1. UniformSearch.cpp

#include "UniformSearch.h"

/\*\*

\* Uniform search method, which finds min and max

\* value of the function on a range.

\*

\* @param s Given function to calculate.

\* @param a Lower bound of the function.

\* @param b Higher bound of the function.

\* @param acc Accuracy of the calculations.

\* @returns String, which contents result and information

\* about calculations.

\*/

std::string Uniform::UniformSearch(std::string s, double a, double b, double acc)

{

unsigned int start\_time = clock(); // start time of working method

unsigned long n = (b - a) / acc, // number of iterations

x\_min\_index, // index of min x value

x\_max\_index; // index of max x value

double work\_time = 0, // method's work time

x\_min, // min x value

x\_max; // max x value

std::vector <double> x\_arr, // array of x-es on the range

y\_arr; // array of f(x)-es on the range

for (unsigned long i = 0; i < n; ++i)

{

x\_arr.push\_back(a + (i + 1) \* (b - a) / (n + 1));

y\_arr.push\_back(f(s, x\_arr[i]));

}

x\_min\_index = FindMinValue(y\_arr, n);

x\_max\_index = FindMaxValue(y\_arr, n);

x\_min = x\_arr[x\_min\_index];

x\_max = x\_arr[x\_max\_index];

work\_time = double(clock() - start\_time) / CLK\_TCK;

// output result

int k = 0; // number of digits after comma

std::string result = ""; // result of method's work

while (acc < 1)

{

acc \*= 10;

k++;

}

if (x\_min == x\_max)

result = "Function is independent of x. So f(min) = f(max) = " + ToString(RoundTo(f(s, x\_max), k), k);

else

{

result = "y(min) = " + ToString(RoundTo(f(s, x\_min), k), k) + " at x(min) = " + ToString(RoundTo(x\_min, k), k);

result += "\ny(max) = " + ToString(RoundTo(f(s, x\_max), k), k) + " at x(min) = " + ToString(RoundTo(x\_max, k), k);

}

result += "\nNumber of iterations: " + std::to\_string(n);

result += "\nRunning time of the method: " + std::to\_string(work\_time) + " s";

return result;

}

1. Dichotomy.h

/\*\*

\* Used to describe the prototype of the DichotomyMethod() function.

\*/

#pragma once

#include "stdafx.h"

#include "functions.h"

namespace Dichotomy

{

/\*\*

\* Dichotomy method, which finds min and max

\* value of the function on a range.

\*

\* @param s Given function to calculate.

\* @param a Lower bound of the function.

\* @param b Higher bound of the function.

\* @param acc Accuracy of the calculations.

\* @returns String, which contents result and information

\* about calculations.

\*/

std::string DichotomyMethod(std::string, double, double, double);

}

1. Dichotomy.cpp

#include "Dichotomy.h"

/\*\*

\* Dichotomy method, which finds min and max

\* value of the function on a range.

\*

\* @param s Given function to calculate.

\* @param a Lower bound of the function.

\* @param b Higher bound of the function.

\* @param acc Accuracy of the calculations.

\* @returns String, which contents result and information

\* about calculations.

\*/

std::string Dichotomy::DichotomyMethod(std::string s, double a, double b, double acc)

{

unsigned int iterations = 0, // number of iterations

start\_time = clock(); // start time of working method

double x1, // first x-value for golden section search

x2, // second x-value for golden section search

y1, // first f(x)-value for golden section search

y2, // second f(x)-value for golden section search

x\_min, // min x value

x\_max, // max x value

a\_original, // initial start of the range

b\_original, // initial end of the range

work\_time = 0, // method's work time

epsilon = 0.00001; // some small number

a\_original = a;

b\_original = b;

// find minimum of the function

do

{

x1 = (a + b - epsilon) / 2;

y1 = f(s, x1);

x2 = (a + b + epsilon) / 2;

y2 = f(s, x2);

if (y1 <= y2)

b = x2;

else

a = x1;

iterations++;

} while (abs(b - a) > acc);

x\_min = (a + b) / 2;

a = a\_original;

b = b\_original;

// find maximum of the function

do

{

x1 = (a + b - epsilon) / 2;

y1 = f(s, x1);

x2 = (a + b + epsilon) / 2;

y2 = f(s, x2);

if (y1 >= y2)

b = x2;

else

a = x1;

iterations++;

} while (abs(b - a) > acc);

x\_max = (a + b) / 2;

work\_time = double(clock() - start\_time) / CLK\_TCK;

// output result

int n = 0; // number of digits after comma

std::string result = ""; // result of method's work

while (acc < 1)

{

acc \*= 10;

n++;

}

if (x\_min == x\_max)

result = "Function is independent of x. So f(min) = f(max) = " + ToString(RoundTo(f(s, x\_max), n), n);

else

{

result = "y(min) = " + ToString(RoundTo(f(s, x\_min), n), n) + " at x(min) = " + ToString(RoundTo(x\_min, n), n);

result += "\ny(max) = " + ToString(RoundTo(f(s, x\_max), n), n) + " at x(min) = " + ToString(RoundTo(x\_max, n), n);

}

result += "\nNumber of iterations: " + std::to\_string(iterations);

result += "\nRunning time of the method: " + std::to\_string(work\_time) + " s";

return result;

}

1. functions.h

/\*\*

\* Used to describe all prototypes of functions in the file functions.cpp.

\*/

#pragma once

#include "stdafx.h"

/\*\*

\* Replacing all powers by multiplying and

\* remove all spaces in the input string.

\*

\* @param s Given string.

\*/

void PowerReplace(std::string &);

/\*\*

\* Saves given string to the file "extremums.txt".

\*

\* @param s Given string.

\*/

void SaveToFile(std::string);

/\*\*

\* Replaces in string one substring by another.

\*

\* @param str Given string.

\* @param from Given substring that you want to replace.

\* @param to Given substring to which you want to replace.

\* @returns Changed string.

\*/

std::string ReplaceAll(std::string, const std::string&, const std::string&);

/\*\*

\* Converts double number to string

\*

\* @param a Given number.

\* @param n Number of digits after comma.

\* @returns Created of the number string.

\*/

std::string ToString(double, int);

/\*\*

\* Finds max value in an array.

\*

\* @param arr Giver array.

\* @param n Length of the array.

\* @returns Max array value.

\*/

double FindMaxValue(std::vector<double>, unsigned long);

/\*\*

\* Finds min value in an array.

\*

\* @param arr Giver array.

\* @param n Length of the array.

\* @returns Max array value.

\*/

double FindMinValue(std::vector<double>, unsigned long);

/\*\*

\* Reverse polish algorithm, which calculates list with numbers

\* and operations and gives calculated value.

\*

\* @param queue List with numbers and operations, which received from function.

\* @returns Value of the calculated list.

\*/

double ReversePolish(std::list<std::string>);

/\*\*

\* Calculates input as string function in point x.

\*

\* @param s Given function to calculate.

\* @param x The point in which you want to calculate function.

\* @returns Value of the calculated function in point.

\*/

double f(std::string, double);

/\*\*

\* Rounds given number up to a given number of digits.

\*

\* @param a Given number.

\* @param n Number of digits to round.

\* @returns Rounded number.

\*/

double RoundTo(double, int);

/\*\*

\* Checks is given function has correct format.

\*

\* @param s Given function.

\* @returns Has correct format or not.

\*/

bool CheckFunction(std::string);

/\*\*

\* Checks is string a number.

\*

\* @param s Given string.

\* @returns Is number or not.

\*/

bool IsNumber(std::string);

/\*\*

\* Shunting algorithm function helps to calculate input as string function

\* in point x.

\*

\* @param s Given function to calculate.

\* @param x The point in which you want to calculate function.

\* @returns List with numbers and operations, which received from function.

\*/

std::list<std::string> ShuntingAlgorithm(std::string, double);

1. functions.cpp

#include "functions.h"

#include "stdafx.h"

/\*\*

\* Checks is given function has correct format.

\*

\* @param s Given function.

\* @returns Has correct format or not.

\*/

bool CheckFunction(std::string s)

{

int n = s.length(), // length of the string

counter = 0; // number of symbols, which can be in the expression with double numbers

std::vector<char> can\_content; // symbols, which can be in the expression with double numbers

can\_content = { '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', 'x', ' ', '+', '-', '\*', '/', '^', '(', ')' };

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

if (std::find(can\_content.begin(), can\_content.end(), s[i]) != can\_content.end())

counter++;

return !!(counter == n);

}

/\*\*

\* Checks is string a number.

\*

\* @param s Given string.

\* @returns Is number or not.

\*/

bool IsNumber(std::string s)

{

int n = s.length(), // length of the string

counter = 0; // counter

if (n == 0)

return false;

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

if (isdigit(s[i]) || s[i] == ',' || s[i] == '-')

counter++;

return !!(counter == n);

}

/\*\*

\* Saves given string to the file "extremums.txt".

\*

\* @param s Given string.

\*/

void SaveToFile(std::string s)

{

std::ofstream f("extremums.txt"); // output file

f << s;

f.close();

}

/\*\*

\* Rounds given number up to a given number of digits.

\*

\* @param a Given number.

\* @param n Number of digits to round.

\* @returns Rounded number.

\*/

double RoundTo(double a, int n)

{

int mult = std::pow(10, n); // stores 10 in some power

a \*= mult;

a = std::round(a);

return a / mult;

}

/\*\*

\* Converts double number to string

\*

\* @param a Given number.

\* @param n Number of digits after comma.

\* @returns Created of the number string.

\*/

std::string ToString(double a, int n)

{

std::string s = std::to\_string(a); // string from double number

int p = s.find("."); // position of dot symbol in the string

s = s.substr(0, p + n + 1);

return s;

}

/\*\*

\* Replaces in string one substring by another.

\*

\* @param str Given string.

\* @param from Given substring that you want to replace.

\* @param to Given substring to which you want to replace.

\* @returns Changed string.

\*/

std::string ReplaceAll(std::string str, const std::string& from, const std::string& to) {

size\_t start\_pos = 0; // start position in the string

while ((start\_pos = str.find(from, start\_pos)) != std::string::npos) {

str.replace(start\_pos, from.length(), to);

start\_pos += to.length();

}

return str;

}

/\*\*

\* Finds max value in an array.

\*

\* @param arr Giver array.

\* @param n Length of the array.

\* @returns Max array value.

\*/

double FindMaxValue(std::vector<double> arr, unsigned long n)

{

unsigned long max\_index = 0; // index of max element in the array

double max = arr[0]; // initial "max" value

for (unsigned long i = 0; i < n; ++i)

if (arr[i] > max)

{

max = arr[i];

max\_index = i;

}

return max\_index;

}

/\*\*

\* Finds min value in an array.

\*

\* @param arr Giver array.

\* @param n Length of the array.

\* @returns Max array value.

\*/

double FindMinValue(std::vector<double> arr, unsigned long n)

{

unsigned long min\_index = 0; // index of min element in the array

double min = arr[0]; // initial "min" value

for (unsigned long i = 0; i < n; ++i)

if (arr[i] < min)

{

min = arr[i];

min\_index = i;

}

return min\_index;

}

/\*\*

\* Replacing all powers by multiplying and

\* remove all spaces in the input string.

\*

\* @param s Given string.

\*/

void PowerReplace(std::string &s)

{

int pos, // position of "x^" substring

n, // end of substring with "x^"

i; // counter

s = ReplaceAll(s, " ", "");

while (s.find("x^") != -1)

{

pos = s.find("x^");

n = s[pos + 2] - '0';

s.erase(pos, 3);

std::string power = "";

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

power += "x\*";

power.erase(power.end() - 1);

s.insert(pos, power);

}

}

/\*\*

\* Shunting algorithm function helps to calculate input as string function

\* in point x.

\*

\* @param s Given function to calculate.

\* @param x The point in which you want to calculate function.

\* @returns List with numbers and operations, which received from function.

\*/

std::list<std::string> ShuntingAlgorithm(std::string s, double x)

{

bool numbers\_before; // whether there are numbers before

int i; // counter

std::list<char> stack; // stack for string's elements

std::list<std::string> queue; // result queue

std::string prev = "o"; // previous char in the string

std::string number; // number in the string

std::map<char, int> precedence = { // dictionary with prioriry of operations

{ '-', 1 },{ '+', 1 },

{ '\*', 2 },{ '/', 2 },

{ '^', 3 }

};

// replace 'x' by x-value

s = ReplaceAll(s, "x", std::to\_string(x));

// replace all '--' by '+'

s = ReplaceAll(s, "--", "+");

if (s[0] == '+')

s.erase(s.begin(), s.begin() + 1);

// work with '-'

numbers\_before = false;

i = 0;

while (i < s.length())

{

if (isdigit(s[i]))

numbers\_before = true;

i++;

if (s[i] == '-' && numbers\_before && isdigit(s[i - 1]))

{

s.insert(i, "+");

i++;

}

}

for (int i = 0; i < s.length(); ++i)

{

if (isdigit(s[i]) || s[i] == '.' || s[i] == '-')

{

// It's a number

if (prev.length() == 1)

{

if (isdigit(prev[0]) || prev[0] == '.' || prev[0] == '-')

{

number = prev + std::string(1, s[i]);

queue.pop\_back();

queue.push\_back(number);

prev = number;

}

else

{

queue.push\_back(std::string(1, s[i]));

prev = s[i];

}

}

else

{

number = prev + std::string(1, s[i]);

queue.pop\_back();

queue.push\_back(number);

prev = number;

}

}

else

{

if (s[i] == '+' || s[i] == '-' || s[i] == '\*' || s[i] == '/' || s[i] == '^')

{

// It's an operator

while (!stack.empty())

{

if (precedence[stack.back()] > precedence[s[i]])

{

queue.push\_back(std::string(1, stack.back()));

stack.pop\_back();

}

else

break;

}

stack.push\_back(s[i]);

}

else

{

if (s[i] == '(')

stack.push\_back('(');

if (s[i] == ')')

{

while (!stack.empty())

{

if (stack.back() != '(')

{

queue.push\_back(std::string(1, stack.back()));

stack.pop\_back();

}

else

{

stack.pop\_back();

break;

}

}

}

}

prev = s[i];

}

}

// While there's operators on the stack, pop them to the queue

while (!stack.empty())

{

queue.push\_back(std::string(1, stack.back()));

stack.pop\_back();

}

return queue;

}

/\*\*

\* Reverse polish algorithm, which calculates list with numbers

\* and operations and gives calculated value.

\*

\* @param queue List with numbers and operations, which received from function.

\* @returns Value of the calculated list.

\*/

double ReversePolish(std::list<std::string> queue)

{

double a, // first number for math operation

b, // second number for math operation

result = 0; // result of math operation

std::list<double>fstack; // stack with numbers and math operators

std::string str; // string, which contains current symbol from queue

for (std::list<std::string>::iterator p = queue.begin(); p != queue.end(); ++p)

{

str = \*p;

if (str[0] == '-')

{

fstack.push\_back(stod(str));

continue;

}

if (isdigit(str[0]))

fstack.push\_back(stod(str));

else

{

a = fstack.back();

fstack.pop\_back();

b = fstack.back();

fstack.pop\_back();

switch (str[0])

{

case '+':

result = b + a;

break;

case '-':

result = b - a;

break;

case '\*':

result = b \* a;

break;

case '/':

result = (double)b / a;

break;

case '^':

result = (double)pow((double)b, a);

break;

}

fstack.push\_back(result);

}

}

return fstack.back();

}

/\*\*

\* Calculates input as string function in point x.

\*

\* @param s Given function to calculate.

\* @param x The point in which you want to calculate function.

\* @returns Value of the calculated function in point.

\*/

double f(std::string s, double x)

{

std::list<std::string>queue = ShuntingAlgorithm(s, x); // queue with numbers and math operators

return ReversePolish(queue);

}