**Форма № Н-6.01**

ЗАТВЕРДЖЕНО

Наказ Міністерства освіти і науки

29 березня 2012 року № 384

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра програмного забезпечення комп’ютерних систем

КУРСОВИЙ ПРОЕКТ

на тему:

**ДОСЛІДЖЕННЯ ФУНКЦІЙ   
НА МАКСИМУМ І МІНІМУМ**

Студентки 2 курсу 243 групи

напряму підготовки

«Програмна інженерія»

Вітталь-Комолової К.Є.

Керівник

Доцент кафедри МПУіК,

Лазорик В.В.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

**Члени комісії** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

Чернівці – 2015

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра програмного забезпечення комп’ютерних систем

**ЗАТВЕРДЖУЮ**

Завідувач кафедри

доктор фіз.-мат.наук, проф.

\_\_\_\_\_\_\_\_С.Е. Остапов

”\_\_\_”\_\_\_\_\_\_\_\_\_\_2015 р.

|  |  |
| --- | --- |
| *Підпис і дата* |  |
| *Інв. № дубл.* |  |
| *Зам. інв. №* |  |
| *Підпис і дата* |  |
| *Інв. № ориг.* |  |

**ДОСЛІДЖЕННЯ ФУНКЦІЙ   
НА МАКСИМУМ І МІНІМУМ**

ЛИСТ ЗАТВЕРДЖЕННЯ

**«УЗГОДЖЕНО»**

Керівник проекту

\_\_\_\_\_\_\_\_\_\_\_В.В. Лазорик

”\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2015 р.

Виконавець

СтудентКА 243 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_К.Є.Вітталь-Комолова

”\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_ 2015 р.

Нормоконтролер

\_\_\_\_\_\_\_\_\_\_\_\_Р.І. Овчар

”\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_ 2015 р.

2015 ЛЗ

Чернівецький національний університет імені Юрія Федьковича

(назва вузу)

Інститут  *фізико-технічних та комп’ютерних наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Кафедра *\_ програмного забезпечення комп’ютерних систем\_\_\_\_\_\_\_\_*

Напрям підготовки \_ *Програмна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**ЗАТВЕРДЖУЮ**

зав. кафедрою\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2015 р.

**ЗАВДАННЯ НА КУРСОВИЙ ПРОЕКТ**

Вітталь-Комолова Каріна Євгенівна

(прізвище, ім’я, по батькові)

1. Тема роботи *Програма «Дослідження функцій на максимум і мінімум»*

Затверджено наказом по університету від «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2015 р. №\_\_\_\_\_\_

1. Термін подачі студентом закінченого проекту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Вихідні дані до проекту (роботи) *«Середовище розробки Visual Studio 2013, Гради Буч: Объектно-ориентированный анализ и проектирование»*
3. Зміст документації до кваліфікаційної роботи
4. (перелік питань, що їх належить розробити) *Розробити програму «Дослідження функцій на максимум і мінімум»*

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

1. Перелік графічного матеріалу

(з точним зазначенням обов’язкових креслень) блок-схема (рис. 1),\_\_\_\_\_\_ Головне вікно програми (рис. 2)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

1. Консультанти по проекту (роботі), із зазначенням розділів проекту, що стосуються їх

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Консультант | Підпис, дата | |
| Завдання видав | Завдання прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівники \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Завдання прийняв до виконання\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва етапів кваліфікаційної роботи | Термін виконання роботи |
| 1 | Одержання технічного завдання | 01.12.2014р. |
| 2 | Аналіз літератури | 15.01.2015р. |
| 3 | Розробка інтерфейсу | 20.02.2015р. |
| 4 | Реалізація програми | 02.03.2015р. |
| 5 | Тестування та налагодження програми | 05.04.2015р. |
| 6 | Оформлення програмної документації | 14.04.2015р. |
| 7 | Представлення готової роботи | 17.04.2015р. |
| 8 | Захист роботи | згідно розкладу |

Студентка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник проекту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**ЗМІСТ**

[Теоретична частина 7](#_Toc417467813)

[Вступ 8](#_Toc417467814)

[1. ТЕХНІЧНЕ ЗАВДАННЯ 9](#_Toc417467815)

[1.1. Підстави для розробки 9](#_Toc417467816)

[1.2. Призначення розробки 9](#_Toc417467817)

[1.3. Аналіз вимог до програмного забезпечення 9](#_Toc417467818)

[1.3.1. Функціональні вимоги 9](#_Toc417467819)

[1.3.2. Вимоги до складу та параметрів технічних засобів 9](#_Toc417467820)

[1.3.3. Вимоги до інтерфейсу 10](#_Toc417467821)

[1.3.4. Вимоги до інформаційної та програмної сумісності 10](#_Toc417467822)

[1.3.5. Вимоги до тестування програмного забезпечення 10](#_Toc417467823)

[1.4. Вимоги до програмної документації 11](#_Toc417467824)

[1.4.1. Склад супроводжувальної документації 11](#_Toc417467825)

[1.4.2. Вимоги до супроводжувальної документації 11](#_Toc417467826)

[1.5. Стадії та етапи розробки 11](#_Toc417467827)

[1.6. Порядок контролю і приймання 12](#_Toc417467828)

[Практична частина 13](#_Toc417467829)

[2. АРХІТЕКТУРА, ФУНКЦІОНАЛЬНІ ТА ТЕХНІЧНІ ПОКАЗНИКИ 14](#_Toc417467830)

[2.1. Призначення та область застосування 14](#_Toc417467831)

[2.2. Опис та обґрунтування обраної архітектури 14](#_Toc417467832)

[2.3. Функціональна специфікація 15](#_Toc417467833)

[2.3.1. Опис функціональних можливостей 15](#_Toc417467834)

[2.3.2. Опис інтерфейсу користувача 15](#_Toc417467835)

[2.4. Технічна специфікація 16](#_Toc417467836)

[2.4.1. Опис і обґрунтування вхідних і вихідних даних 16](#_Toc417467837)

[2.4.2. Опис модулів 16](#_Toc417467838)

[3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 19](#_Toc417467839)

[3.1. Опис і обґрунтування обраних програмних засобів 19](#_Toc417467840)

[3.2. Опис програми 19](#_Toc417467841)

[3.2.1. Функціональні можливості 19](#_Toc417467842)

[3.2.2. Опис логічної структури 19](#_Toc417467843)

[3.2.3. Використані технічні засоби 20](#_Toc417467844)

[4. ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ 21](#_Toc417467845)

[4.1. Об’єкт випробувань 21](#_Toc417467846)

[4.2. Використані технічні засоби 21](#_Toc417467847)

[4.3. Порядок та методика випробувань 21](#_Toc417467848)

[4.4. Результати випробувань 22](#_Toc417467849)

[Висновок 25](#_Toc417467850)

[Список використаних джерел 26](#_Toc417467851)

[Додатки 27](#_Toc417467852)

[Class MyForm 27](#_Toc417467853)

[Class MyConvert 46](#_Toc417467854)

[Class MyExceptions 48](#_Toc417467855)

[Class ParserResult 50](#_Toc417467856)

[Class MathParser 51](#_Toc417467857)

[Class MaxMinResult 57](#_Toc417467858)

[Class Extreme 59](#_Toc417467859)

[Class GoldenSection 63](#_Toc417467860)

Теоретична частина

Вступ

Призначенням даного проекту являється дослідження різних методів для пошуку екстремумів, максимуму та мінімуму функцій. Зараз все більше розвиваються комп′ютерні технології і стають важливими проблеми оптимізації та мінімізації математичних методів. Проблему оптимізації задач нелінійного програмування я вважаю актуальною на даний момент, тому обрала тему, яка розглядає одну з цих задач. Нелінійне програмування використовується для задач, які мають певні обмеження або цільова функція не є лінійною.

Методи розв′язування задач нелінійного програмування поділяються на декілька видів. Ці види представлені на рис. 1.



*Рис. 1. Види задач нелінійного програмування*

Так як задача проекту полягає в пошуку екстремумів, а також максимуму та мінімуму функції, то було обрано два методи, якими буде розв′язана поставлена задача. Перший метод – метод золотого січення реалізовує лише пошук максимуму та мінімуму, а другий метод – метод пошуку похідних та екстремумів, а потім визначення максимуму та мінімуму.

1. ТЕХНІЧНЕ ЗАВДАННЯ

1.1. Підстави для розробки

Розробка програмного забезпечення виконується на підставі рішення засідання кафедри ПЗКС про затвердження тем дипломних та курсових робіт.

1.2. Призначення розробки

Призначенням даної розробки є створення програми для пошуку екстремумів, максимуму та мінімуму функції.

1.3. Аналіз вимог до програмного забезпечення

1.3.1. Функціональні вимоги

До програмного забезпечення висуваються наступні вимоги:

* Аналіз рядку, введеного користувачем
* Пошук максимуму функції
* Пошук мінімуму функції

1.3.2. Вимоги до складу та параметрів технічних засобів

Для нормальної роботи розробленого програмного продукту необхідний персональний комп’ютер з наступними мінімальними характеристиками:

|  |  |
| --- | --- |
| **Параметр** | **Значення** |
| Процесор | Intel Core 2 Duo 1.8 Ghz |
| Пам’ять | 1 GB RAM |
| Жорсткий диск | 40 GB |

1.3.3. Вимоги до інтерфейсу

Програма повинна мати інтуїтивно зрозумілий, зручний та візуально приємний інтерфейс.

Програмне забезпечення повинно бути оформлене у вигляді стандартного вікна операційної системи Windows.

Програма повинна складатися з:

* Системи введення символьного рядку
* Системи пошуку максимуму заданої функції
* Системи пошуку мінімуму функції

1.3.4. Вимоги до інформаційної та програмної сумісності

Необхідною вимогою до програмного забезпечення є встановлена операційна система класу Windows XP/7. Окремих вимог не висувається.

1.3.5. Вимоги до тестування програмного забезпечення

Для тестування програмного забезпечення необхідно виконати наступні дії:

* запустити програму;
* перейти до введення функції однієї змінної у символьний рядок;
* ввести межі та точність пошуку;
* вибрати пошук максимуму функції та натиснути кнопку результат;
* вибрати пошук мінімуму функції та натиснути кнопку результат;
* переглянути результати пошуку;
* повторити попередні дії з іншою функцією;
* закрити програму;

Після успішного виконання цих дій тестування програми може вважатися завершеним.

1.4. Вимоги до програмної документації

1.4.1. Склад супроводжувальної документації

Програмне забезпечення постачається разом із супроводжувальною документацією, в склад якої входить:

* технічне завдання;
* опис та обґрунтування вибраної архітектури;
* функціональна специфіка;
* технічна специфікація;
* опис програми;
* програма та методика випробувань;

1.4.2. Вимоги до супроводжувальної документації

Текст супроводжувальної документації набирається на комп’ютері шрифтом Times New Roman 14 пт. через 1,5 інтервал. Розміщення – книжкове.

Поля зліва – не менше 2,5 см., решта – не менше 1,5 см. Вирівнювання – по ширині. Відступ першого рядка абзацу – 1,27 см. Нумерація сторінок – наскрізна в усій роботі. Сторінки нумеруються по центру верхнього контитулу. На титульному аркуші та листі затвердження нумерація сторінок не ставиться, але вони враховуються в загальній кількості сторінок.

1.5. Стадії та етапи розробки

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів кваліфікаційної розробки | Термін виконання етапів розробки | Примітка |
| 1 | Одержання технічного завдання | 01.12.2014p. |  |
| 2 | Аналіз літератури | 15.01.2015р. |  |
| 3 | Розробка інтерфейсу | 20.02.2015р. |  |
| 4 | Реалізація програми | 02.03.2015р. |  |
| 5 | Тестування та налагодження | 05.04.2015р. |  |
| 6 | Оформлення програмної документації | 14.04.2015р. |  |
| 7 | Представлення готової роботи | 17.04.2015р. |  |
| 8 | Захист роботи | Згідно розкладу |  |

1.6. Порядок контролю і приймання

Програмне забезпечення повинне бути випробуване та протестоване на наявність помилок.

Порядок контролю і прийому проекту полягає в наступному:

* перевірка правильного функціонування програми керівником проекту;
* перевірка відповідності стандартам оформлення документації;
* перевірка правильного оформлення програмної документації керівником проекту;
* попередній захист курсової роботи;

Практична частина

2. АРХІТЕКТУРА, ФУНКЦІОНАЛЬНІ ТА ТЕХНІЧНІ ПОКАЗНИКИ

2.1. Призначення та область застосування

Призначення даної програми є дослідження різноманітних функції на екстремуми, максимум та мінімум.

2.2. Опис та обґрунтування обраної архітектури

Для даного проекту була вибрана об’єктно-зорієнтована архітектура, оскільки вона є найпоширенішою та найкраще підходить для розробки програм складного рівня. Були проаналізовані декілька методів пошуку максимуму та мінімуму функції і обрано ті, якій найбільш підходять даній задачі.

Роботу розробленої програми можна показати наступною блок-схемою, що показана на рис. 2.



*Рис. 2. Блок-схема програми*

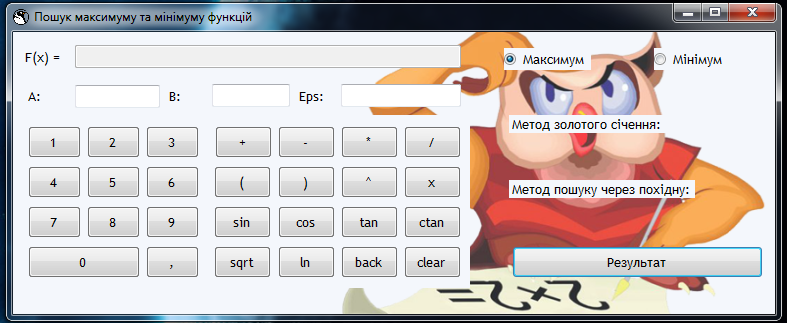
**2.3.** Функціональна специфікація

2.3.1. Опис функціональних можливостей

Розроблена програма надає можливість реалізовувати пошук максимуму та мінімуму заданої в рядок функції. Її функціонал дозволяє користувачеві вводити символьний рядок з форми через кнопки, задавати межі знаходження функції та точність обчислень. Також програма надає можливість аналізувати результат пошуку двома методами: методом золотого січення і методом пошуку похідних.

2.**3.2.** Опис інтерфейсу користувача

На рисунку 3. представлено інтерфейс програми. Даний інтерфейс складається з декількох блоків. Зліва розташований блок для введення початкових даних. Та кнопки з цифрами, операціями та функціями, які дозволяють здійснювати ввід початкових даних в символьний рядок. Справа знаходиться блок результатів. Користувач має змогу обрати пошук максимуму чи мінімуму. Після цього натиснувши кнопку результат він може побачити результати пошуку двома методами.



*Рис.4. Вигляд інтерфейсу*

**2.4.** Технічна специфікація

**2.4.1.** Опис і обґрунтування вхідних і вихідних даних

Вхідними даними для розробленого продукту є символьний рядок заданий через кнопки на формі, а також введені з клавіатури дані в поля про межі заданої функції і точність розбиття. Крім цього передбачений вивід про помилки введення. А також при натисненні кнопки результату буде виведена інформація про результат пошуку двома способами.

**2.4.2.** Опис модулів

Робота даного програмного забезпечення реалізується в наступних модулях:

* MyForm – реалізовано головну форму програми, за допомогою якої користувач взаємодіє з програмою;
* MathParser – клас, в якому реалізовано аналіз символьного рядка для обрахунку результату методом рекурсивного спуску;
* ParserResult – клас, де зберігається результат дій з парсером, а саме частина рядка, яку вже проаналізували і решту рядку;
* Extreme – клас, в якому реалізовано алгоритм пошуку похідної функції в точці, а також аналіз значень функції, де похідна дорівнює нулю і пошук максимуму та мінімуму;
* GoldenSection – клас, в якому реалізовано алгоритм пошуку максимуму та мінімуму функції методом золотого січення;
* MaxMinResult – клас, в якому зберігаються результати пошуку максимального та мінімального значення функцій;
* MyConvert – клас, в якому реалізовано переведення типів, таких як string, System:: String^ та double;
* MyExceptions – клас, в якому реалізовано виключні ситуації;

**Клас MathParser** реалізований за допомогою методу рекурсивного спуску. Суть даного методу полягає в тому, що він розділяється на свої підзадачі. У свою чергу підзадача повинна працювати тільки з тим, з чим вміє працювати, якщо умови не задовольняються передавати управління далі. Якщо ж умови задоволені, робимо обчислення і передаємо частину необробленого тексту, що залишилася. Виконання відбувається до тих пір, поки текст все ще є або ж якщо жодна підзадача не може обробити поточний стан. Також має значення пріоритет підзадачі. Міняючи пріоритет підзадачі поміняється і поведінка парсеру. Оскільки ми будемо реалізовувати парсер математичних формул, то для розставляння пріоритетів ми будемо керуватися пріоритетами математичних операцій:

* функція і змінна
* дужки
* множення і ділення
* додавання і віднімання

**Клас GoldenSection** визначає максимум та мінімум функції за допомогою алгоритму золотого січення.

Алгоритм:

* На першій ітерації заданий відрізок ділиться двома симетричними відносно його центру точками і визначається значення в цих точках.
* Після чого той з кінців відрізка, до якого серед двох знову поставлених точок ближче виявилася та, значення в якій максимальне (для випадку пошуку мінімуму і навпаки), відкидають.
* На наступній ітерації в силу показаної вище властивості золотого перетину вже треба шукати всього одну нову точку.
* Процедура продовжується до тих пір, поки не буде досягнута задана точність.

Клас Extreme визначає екстремуми функції, а далі визначаються максимум та мінімум.

Алгоритм:

* Знаходження похідної функції в точці
* Пошук екстремумів функції в кожному значенні проміжку
* Запис знайдених екстремумів у вектор дійсних чисел
* Аналіз екстремумів і визначення максимуму та мінімуму фунцій

3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Опис і обґрунтування обраних програмних засобів

Розробка програмного забезпечення здійснена в середовищі Visual Studio 2013 C++. Дане інтегроване середовище розробки – зручний засіб розробки віконних програм для операційної системи Windows XP/7. В ньому поєднані потужні можливості мови програмування С++ та величезний набір візуальних компонентів, які, в свою чергу, дозволяють розробнику легко швидко та якісно створювати програмні продукти з візуальним інтерфейсом. Також, візуальна бібліотека містить елементи, що спрощують діалогову взаємодію користувача та додатку (діалогові вікна, відкриття та збереження файлів, налаштування параметрів кольорів, шрифту та іншого…)

3.2. Опис програми

3.2.1. Функціональні можливості

В створеному програмному забезпеченні реалізовано наступні функції:

* Аналіз символьного рядку та його обрахунок;
* Пошук максимуму/мінімуму функції шляхом методу золотого січення;
* Пошук максимуму/мінімуму функції шляхом пошуку екстремумів функції;

3.2.2. Опис логічної структури

Логічну структуру даного програмного забезпечення зручно подати у вигляді наступної блок-схеми рис. 3.



*Рис. 3. Логічна структура програми*

3.2.3. Використані технічні засоби

Розробка програмного продукту здійснювалася на персональному комп’ютері з наступною конфігурацією:

* Процесор – Intel Pentium CPU B960 @ 2,2 GHz 2,2 GHz;
* ОЗП – 6 GB;
* Відеоадаптер – Nvidia Geforce GT 630M 1 GB;
* Жорсткий диск – 500 GB;
* DVD-RW – PLDS DS8A8SH ATA Device

Даний комп’ютер працює під управлінням ОС Windows 7.

4. ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

4.1. Об’єкт випробувань

Об’єктом випробувань є створене в процесі виконання курсової роботи програмне забезпечення. Для перевірки коректності роботи програми було проведено ряд тестів.

4.2. Використані технічні засоби

При проведенні тестування використовувалися технічні засоби, описані в підрозділі 3.3.1 «Використані технічні засоби». Під час проведення тестів використовувалася система ручного тестування.

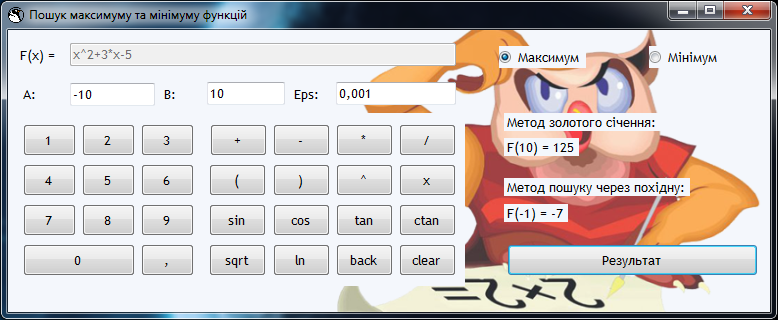
4.3. Порядок та методика випробувань

Порядок проведення випробувань програмного забезпечення:

* запустити програму;
* перейти до введення функції однієї змінної у символьний рядок;
* ввести межі та точність пошуку;
* вибрати пошук максимуму функції та натиснути кнопку результат;
* вибрати пошук мінімуму функції та натиснути кнопку результат;
* переглянути результати пошуку;
* повторити попередні дії з іншою функцією;
* закрити програму;

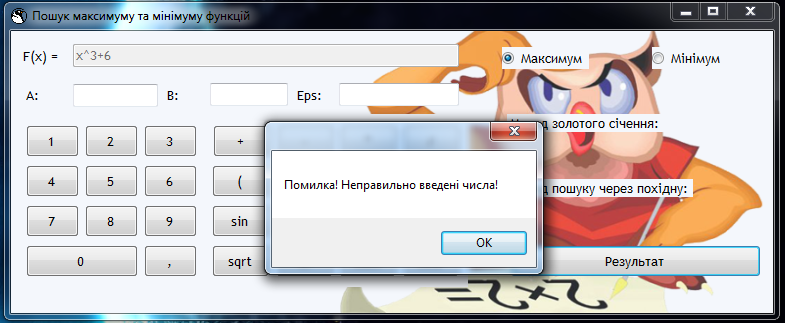
4.4. Результати випробувань

На рисунку 4 представлено роботу програми. У символьний рядок введено функцію, задані межі та точність пошуку. Обрано пошук максимуму функції та обчислено результат. На екрані виведено результат обчислень двома методами.



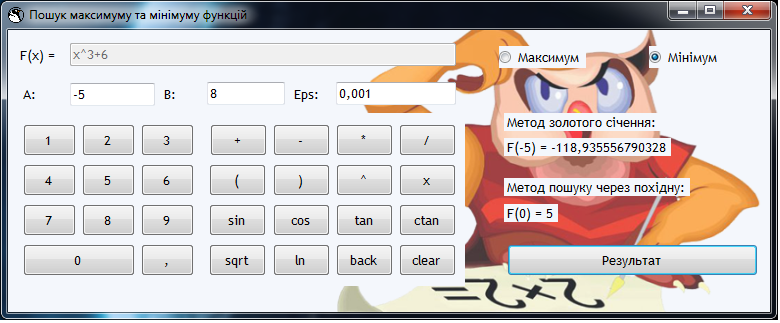
*Рис.4. Результат роботи програми*

На рисунку 5 було введено текстовий рядок, обрано пошук максимуму, але не вказано межі пошуку та точність обчислень. При натисненні кнопки результату програма покаже повідомлення про помилку.



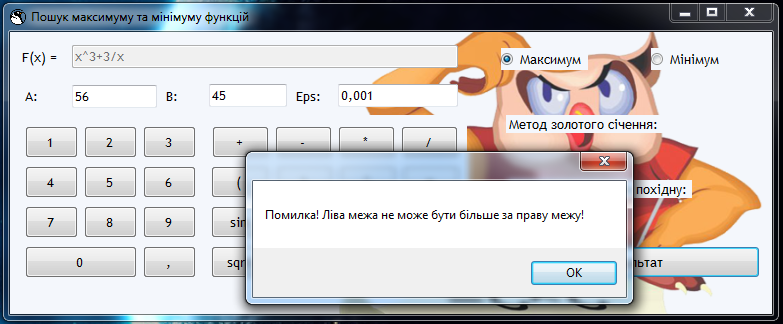
*Рис. 5. Результат роботи програми при порожніх рядках*

На рисунку 6 представлено роботи програми, коли обрано пошук мінімуму функції. У символьному рядку задано функцію, також задано межі функції та точність пошуку. Обрано пошук мінімуму функції та натиснуто кнопку результат. На екрані виведено результати пошуку мінімуму заданої функції двома методами.



*Рис.6. Результат роботи програми*

На рисунку 7 показано роботу програми, якщо ми введемо ліву межу більшою за праву. Було введено функцію в символьний рядок, межі функції та точність обчислень, обрано пошук максимуму функції та натиснену кнопку результату, але так як виникла виключна ситуація, то програма викликала повідомлення про помилку.



*Рис. 7. Результат роботи програми коли ліва межа більша за праву*

Випробування програмного продукту проводилось шляхом тестування різних введених функцій. При тестуванні помилок не виявлено. Програма працює правильно.

Висновок

В результаті виконання курсової роботи було розроблено програму, яка шукає максимум та мінімум заданої функції. Програма дозволяє полегшити користувачу пошук точок максимуму та мінімуму великих та об′ємних функцій.

Тестування показало працездатність програмного забезпечення та відповідність вимогам. Повідомлення про всі помилки виводяться на екран.

Дане програмне забезпечення має наступні можливості:

* введення функції у символьний рядок за допомогою кнопок
* введення меж заданої функції з клавіатури
* введення з клавіатури точності пошуку
* пошук максимуму функції у точці двома методами
* пошук мінімуму функції у точці двома методами
* перегляд результатів

Дана розробка може допомогти як метод оптимізації пошуку функції. А також, вона забезпечує швидку роботу програми для визначення максимуму та мінімуму.

Список використаних джерел

Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++ / Гради Буч. — М.: «Издательство Бином», СПб: «Невский диалект», 1998. — 560 с.

Волков О.А. Чисельні методи : Навчальний посібник / О.А.Волков. — М.: Наука, 1987. — 248 с.

Дьяконов В.П. MATLAB. Аналіз, ідентифікація та моделювання систем / В.П.Дьяконов, В.В.Круглов. — СПб.: Пітер, 2002. — 448 с.

Метод рекурсивного спуску. [Електронний ресурс]: Режим доступу до документу:<https://ru.wikipedia.org/wiki/Метод_рекурсивного_спуска>

Додатки

Class MyForm

//MyForm.h

#pragma once

#include "GoldenSection.h"

#include "MyExceptions.h"

#include <math.h>

#include "Extreme.h"

using namespace std;

namespace MyCourseWork {

MyExceptions excep;

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>

/// Сводка для MyForm

/// </summary>

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

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

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

protected:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

private: System::Windows::Forms::Panel^ panel1;

private: System::Windows::Forms::Panel^ panel2;

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

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

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

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

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

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

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

protected:

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

this->panel1 = (gcnew System::Windows::Forms::Panel());

this->panel2 = (gcnew System::Windows::Forms::Panel());

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

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

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

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

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

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

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

this->panel1->SuspendLayout();

this->panel2->SuspendLayout();

this->SuspendLayout();

//

// tFunction

//

this->tFunction->Enabled = false;

this->tFunction->Location = System::Drawing::Point(62, 13);

this->tFunction->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->tFunction->Size = System::Drawing::Size(386, 23);

this->tFunction->TabIndex = 1;

//

// lFunction

//

this->lFunction->AutoSize = true;

this->lFunction->Location = System::Drawing::Point(9, 16);

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

this->lFunction->Size = System::Drawing::Size(47, 18);

this->lFunction->TabIndex = 2;

this->lFunction->Text = L"F(x) = ";

//

// tEps

//

this->tEps->Location = System::Drawing::Point(328, 52);

this->tEps->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->tEps->Size = System::Drawing::Size(120, 23);

this->tEps->TabIndex = 4;

//

// lEps

//

this->lEps->AutoSize = true;

this->lEps->Location = System::Drawing::Point(283, 55);

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

this->lEps->Size = System::Drawing::Size(33, 18);

this->lEps->TabIndex = 5;

this->lEps->Text = L"Eps:";

//

// tLeftBorder

//

this->tLeftBorder->Location = System::Drawing::Point(62, 53);

this->tLeftBorder->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->tLeftBorder->Size = System::Drawing::Size(85, 23);

this->tLeftBorder->TabIndex = 7;

//

// tRightBorder

//

this->tRightBorder->Location = System::Drawing::Point(199, 52);

this->tRightBorder->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->tRightBorder->Size = System::Drawing::Size(78, 23);

this->tRightBorder->TabIndex = 8;

//

// lLeftBorder

//

this->lLeftBorder->AutoSize = true;

this->lLeftBorder->Location = System::Drawing::Point(12, 55);

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

this->lLeftBorder->Size = System::Drawing::Size(21, 18);

this->lLeftBorder->TabIndex = 9;

this->lLeftBorder->Text = L"A:";

//

// lRightBorder

//

this->lRightBorder->AutoSize = true;

this->lRightBorder->Location = System::Drawing::Point(153, 55);

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

this->lRightBorder->Size = System::Drawing::Size(20, 18);

this->lRightBorder->TabIndex = 10;

this->lRightBorder->Text = L"B:";

//

// bMul

//

this->bMul->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bMul->Location = System::Drawing::Point(129, 11);

this->bMul->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bMul->Size = System::Drawing::Size(57, 32);

this->bMul->TabIndex = 11;

this->bMul->Text = L"\*";

this->bMul->UseVisualStyleBackColor = true;

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

//

// bMinus

//

this->bMinus->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bMinus->Location = System::Drawing::Point(66, 11);

this->bMinus->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bMinus->Size = System::Drawing::Size(57, 32);

this->bMinus->TabIndex = 12;

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

this->bMinus->UseVisualStyleBackColor = true;

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

//

// bPlus

//

this->bPlus->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bPlus->Location = System::Drawing::Point(3, 11);

this->bPlus->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bPlus->Size = System::Drawing::Size(57, 32);

this->bPlus->TabIndex = 13;

this->bPlus->Text = L"+";

this->bPlus->UseVisualStyleBackColor = true;

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

//

// bDiv

//

this->bDiv->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bDiv->Location = System::Drawing::Point(192, 11);

this->bDiv->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bDiv->Size = System::Drawing::Size(57, 32);

this->bDiv->TabIndex = 15;

this->bDiv->Text = L"/";

this->bDiv->UseVisualStyleBackColor = true;

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

//

// bTan

//

this->bTan->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bTan->Location = System::Drawing::Point(129, 91);

this->bTan->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bTan->Size = System::Drawing::Size(57, 32);

this->bTan->TabIndex = 16;

this->bTan->Text = L"tan";

this->bTan->UseVisualStyleBackColor = true;

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

//

// bCtan

//

this->bCtan->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bCtan->Location = System::Drawing::Point(192, 91);

this->bCtan->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bCtan->Size = System::Drawing::Size(57, 32);

this->bCtan->TabIndex = 17;

this->bCtan->Text = L"ctan";

this->bCtan->UseVisualStyleBackColor = true;

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

//

// bSin

//

this->bSin->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bSin->Location = System::Drawing::Point(2, 91);

this->bSin->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bSin->Size = System::Drawing::Size(57, 32);

this->bSin->TabIndex = 18;

this->bSin->Text = L"sin";

this->bSin->UseVisualStyleBackColor = true;

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

//

// bCos

//

this->bCos->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bCos->Location = System::Drawing::Point(66, 91);

this->bCos->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bCos->Size = System::Drawing::Size(57, 32);

this->bCos->TabIndex = 19;

this->bCos->Text = L"cos";

this->bCos->UseVisualStyleBackColor = true;

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

//

// bLeftBracket

//

this->bLeftBracket->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bLeftBracket->Location = System::Drawing::Point(2, 51);

this->bLeftBracket->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bLeftBracket->Size = System::Drawing::Size(57, 32);

this->bLeftBracket->TabIndex = 20;

this->bLeftBracket->Text = L"(";

this->bLeftBracket->UseVisualStyleBackColor = true;

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

//

// bRightBracket

//

this->bRightBracket->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bRightBracket->Location = System::Drawing::Point(66, 51);

this->bRightBracket->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bRightBracket->Size = System::Drawing::Size(57, 32);

this->bRightBracket->TabIndex = 21;

this->bRightBracket->Text = L")";

this->bRightBracket->UseVisualStyleBackColor = true;

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

//

// bPow

//

this->bPow->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bPow->Location = System::Drawing::Point(129, 51);

this->bPow->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bPow->Size = System::Drawing::Size(57, 32);

this->bPow->TabIndex = 22;

this->bPow->Text = L"^";

this->bPow->UseVisualStyleBackColor = true;

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

//

// bLn

//

this->bLn->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bLn->Location = System::Drawing::Point(66, 131);

this->bLn->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bLn->Size = System::Drawing::Size(57, 32);

this->bLn->TabIndex = 23;

this->bLn->Text = L"ln";

this->bLn->UseVisualStyleBackColor = true;

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

//

// bNull

//

this->bNull->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bNull->Location = System::Drawing::Point(3, 131);

this->bNull->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bNull->Size = System::Drawing::Size(112, 32);

this->bNull->TabIndex = 24;

this->bNull->Text = L"0";

this->bNull->UseVisualStyleBackColor = true;

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

//

// bOne

//

this->bOne->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bOne->Location = System::Drawing::Point(3, 11);

this->bOne->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bOne->Size = System::Drawing::Size(53, 32);

this->bOne->TabIndex = 25;

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

this->bOne->UseVisualStyleBackColor = true;

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

//

// bTwo

//

this->bTwo->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bTwo->Location = System::Drawing::Point(62, 11);

this->bTwo->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bTwo->Size = System::Drawing::Size(53, 32);

this->bTwo->TabIndex = 26;

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

this->bTwo->UseVisualStyleBackColor = true;

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

//

// bThree

//

this->bThree->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bThree->Location = System::Drawing::Point(121, 11);

this->bThree->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bThree->Size = System::Drawing::Size(53, 32);

this->bThree->TabIndex = 27;

this->bThree->Text = L"3";

this->bThree->UseVisualStyleBackColor = true;

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

//

// bFour

//

this->bFour->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bFour->Location = System::Drawing::Point(3, 51);

this->bFour->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bFour->Size = System::Drawing::Size(53, 32);

this->bFour->TabIndex = 28;

this->bFour->Text = L"4";

this->bFour->UseVisualStyleBackColor = true;

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

//

// bFive

//

this->bFive->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bFive->Location = System::Drawing::Point(62, 51);

this->bFive->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bFive->Size = System::Drawing::Size(53, 32);

this->bFive->TabIndex = 29;

this->bFive->Text = L"5";

this->bFive->UseVisualStyleBackColor = true;

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

//

// bSix

//

this->bSix->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bSix->Location = System::Drawing::Point(121, 51);

this->bSix->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bSix->Size = System::Drawing::Size(53, 32);

this->bSix->TabIndex = 30;

this->bSix->Text = L"6";

this->bSix->UseVisualStyleBackColor = true;

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

//

// bSeven

//

this->bSeven->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bSeven->Location = System::Drawing::Point(3, 91);

this->bSeven->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bSeven->Size = System::Drawing::Size(53, 32);

this->bSeven->TabIndex = 31;

this->bSeven->Text = L"7";

this->bSeven->UseVisualStyleBackColor = true;

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

//

// bEight

//

this->bEight->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bEight->Location = System::Drawing::Point(62, 91);

this->bEight->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bEight->Size = System::Drawing::Size(53, 32);

this->bEight->TabIndex = 32;

this->bEight->Text = L"8";

this->bEight->UseVisualStyleBackColor = true;

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

//

// bNine

//

this->bNine->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bNine->Location = System::Drawing::Point(121, 91);

this->bNine->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bNine->Size = System::Drawing::Size(53, 32);

this->bNine->TabIndex = 33;

this->bNine->Text = L"9";

this->bNine->UseVisualStyleBackColor = true;

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

//

// bComa

//

this->bComa->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bComa->Location = System::Drawing::Point(121, 131);

this->bComa->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bComa->Size = System::Drawing::Size(53, 32);

this->bComa->TabIndex = 34;

this->bComa->Text = L",";

this->bComa->UseVisualStyleBackColor = true;

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

//

// bSqrt

//

this->bSqrt->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bSqrt->Location = System::Drawing::Point(2, 131);

this->bSqrt->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bSqrt->Size = System::Drawing::Size(57, 32);

this->bSqrt->TabIndex = 35;

this->bSqrt->Text = L"sqrt";

this->bSqrt->UseVisualStyleBackColor = true;

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

//

// bBack

//

this->bBack->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bBack->Location = System::Drawing::Point(129, 131);

this->bBack->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bBack->Size = System::Drawing::Size(57, 32);

this->bBack->TabIndex = 36;

this->bBack->Text = L"back";

this->bBack->UseVisualStyleBackColor = true;

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

//

// bClear

//

this->bClear->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bClear->Location = System::Drawing::Point(192, 131);

this->bClear->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bClear->Size = System::Drawing::Size(57, 32);

this->bClear->TabIndex = 37;

this->bClear->Text = L"clear";

this->bClear->UseVisualStyleBackColor = true;

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

//

// bXVariable

//

this->bXVariable->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->bXVariable->Location = System::Drawing::Point(192, 51);

this->bXVariable->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bXVariable->Size = System::Drawing::Size(57, 32);

this->bXVariable->TabIndex = 38;

this->bXVariable->Text = L"x";

this->bXVariable->UseVisualStyleBackColor = true;

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

//

// panel1

//

this->panel1->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"panel1.BackgroundImage")));

this->panel1->Controls->Add(this->bDiv);

this->panel1->Controls->Add(this->bClear);

this->panel1->Controls->Add(this->bXVariable);

this->panel1->Controls->Add(this->bBack);

this->panel1->Controls->Add(this->bPlus);

this->panel1->Controls->Add(this->bSqrt);

this->panel1->Controls->Add(this->bMinus);

this->panel1->Controls->Add(this->bMul);

this->panel1->Controls->Add(this->bLeftBracket);

this->panel1->Controls->Add(this->bRightBracket);

this->panel1->Controls->Add(this->bPow);

this->panel1->Controls->Add(this->bSin);

this->panel1->Controls->Add(this->bCos);

this->panel1->Controls->Add(this->bTan);

this->panel1->Controls->Add(this->bCtan);

this->panel1->Controls->Add(this->bLn);

this->panel1->Location = System::Drawing::Point(199, 83);

this->panel1->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

this->panel1->Name = L"panel1";

this->panel1->Size = System::Drawing::Size(258, 173);

this->panel1->TabIndex = 39;

//

// panel2

//

this->panel2->Controls->Add(this->bNull);

this->panel2->Controls->Add(this->bOne);

this->panel2->Controls->Add(this->bComa);

this->panel2->Controls->Add(this->bTwo);

this->panel2->Controls->Add(this->bNine);

this->panel2->Controls->Add(this->bThree);

this->panel2->Controls->Add(this->bEight);

this->panel2->Controls->Add(this->bFour);

this->panel2->Controls->Add(this->bSeven);

this->panel2->Controls->Add(this->bFive);

this->panel2->Controls->Add(this->bSix);

this->panel2->Location = System::Drawing::Point(12, 83);

this->panel2->Name = L"panel2";

this->panel2->Size = System::Drawing::Size(181, 173);

this->panel2->TabIndex = 40;

//

// rbMin

//

this->rbMin->AutoSize = true;

this->rbMin->Location = System::Drawing::Point(641, 16);

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

this->rbMin->Size = System::Drawing::Size(75, 22);

this->rbMin->TabIndex = 43;

this->rbMin->Text = L"Мінімум";

this->rbMin->UseVisualStyleBackColor = true;

this->rbMin->CheckedChanged += gcnew System::EventHandler(this, &MyForm::rbMin\_CheckedChanged);

//

// lDerivative

//

this->lDerivative->AutoSize = true;

this->lDerivative->Location = System::Drawing::Point(496, 148);

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

this->lDerivative->Size = System::Drawing::Size(186, 18);

this->lDerivative->TabIndex = 14;

this->lDerivative->Text = L"Метод пошуку через похідну:";

//

// bResult

//

this->bResult->Location = System::Drawing::Point(499, 214);

this->bResult->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

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

this->bResult->Size = System::Drawing::Size(251, 32);

this->bResult->TabIndex = 0;

this->bResult->Text = L"Результат";

this->bResult->UseVisualStyleBackColor = true;

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

//

// rbMax

//

this->rbMax->AutoSize = true;

this->rbMax->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"rbMax.BackgroundImage")));

this->rbMax->Checked = true;

this->rbMax->Location = System::Drawing::Point(491, 16);

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

this->rbMax->Size = System::Drawing::Size(87, 22);

this->rbMax->TabIndex = 42;

this->rbMax->TabStop = true;

this->rbMax->Text = L"Максимум";

this->rbMax->UseVisualStyleBackColor = true;

this->rbMax->CheckedChanged += gcnew System::EventHandler(this, &MyForm::rbMax\_CheckedChanged);

//

// lGoldenSection

//

this->lGoldenSection->AutoSize = true;

this->lGoldenSection->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"lGoldenSection.Image")));

this->lGoldenSection->Location = System::Drawing::Point(496, 83);

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

this->lGoldenSection->Size = System::Drawing::Size(157, 18);

this->lGoldenSection->TabIndex = 6;

this->lGoldenSection->Text = L"Метод золотого січення:";

//

// lDerivativeResult

//

this->lDerivativeResult->AutoSize = true;

this->lDerivativeResult->Location = System::Drawing::Point(496, 174);

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

this->lDerivativeResult->Size = System::Drawing::Size(0, 18);

this->lDerivativeResult->TabIndex = 44;

//

// lGoldenResult

//

this->lGoldenResult->AutoSize = true;

this->lGoldenResult->Location = System::Drawing::Point(496, 108);

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

this->lGoldenResult->Size = System::Drawing::Size(0, 18);

this->lGoldenResult->TabIndex = 45;

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(7, 18);

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

this->BackColor = System::Drawing::SystemColors::InactiveBorder;

this->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"$this.BackgroundImage")));

this->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->ClientSize = System::Drawing::Size(762, 282);

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

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

this->Controls->Add(this->panel2);

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

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

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

this->Controls->Add(this->panel1);

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

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

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

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

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

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

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

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

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

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

this->DoubleBuffered = true;

this->Font = (gcnew System::Drawing::Font(L"Trebuchet MS", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->Margin = System::Windows::Forms::Padding(3, 4, 3, 4);

this->Name = L"MyForm";

this->Text = L"Пошук максимуму та мінімуму функцій";

this->panel1->ResumeLayout(false);

this->panel2->ResumeLayout(false);

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

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

{

//створення екземпляру класу золотого січення та аналізу екстремумів

GoldenSection goldenSection;

Extreme extreme;

//змінна, яка відповідає за перевірку правильного вводу в поля

bool convertError = false;

//перевірка чи в полях меж та точності лише числа

try

{

double a = double::Parse(tLeftBorder->Text);

a = double::Parse(tRightBorder->Text);

a = double::Parse(tEps->Text);

}

catch (System::FormatException^ e)

{

MessageBox::Show("Помилка! Неправильно введені числа!");

convertError = true;

}

if (convertError){}

else

//Перевірка полів на заповненість

if (tLeftBorder->Text == "" || tRightBorder->Text == "" || tEps->Text == "" || tFunction->Text == "")

{

MessageBox::Show("Помилка! Поля не можуть бути порожніми! ");

}

else

//Перевірка того, що кількість дужок співпадає

if (!excep.isValidBrackets(MyConvert::toString(tFunction->Text)))

{

MessageBox::Show("Помилка! Кількість відкриваючих і закриваючих дужок не співдає! ");

}

else

//Перевірка того, що ліва межа менше за праву межу

if (!excep.isCorrectBorders(MyConvert::toDouble(tLeftBorder->Text), MyConvert::toDouble(tRightBorder->Text)))

{

MessageBox::Show("Помилка! Ліва межа не може бути більше за праву межу! ");

}

//Виконання основного коду

else

{

//Отримання значень з полів

goldenSection.setFunction(MyConvert::toString(tFunction->Text));

extreme.setFunction(MyConvert::toString(tFunction->Text));

goldenSection.setEps(MyConvert::toDouble(tEps->Text));

extreme.setEps(MyConvert::toDouble(tEps->Text));

goldenSection.setLeftBorder(MyConvert::toDouble(tLeftBorder->Text));

goldenSection.setRightBorder(MyConvert::toDouble(tRightBorder->Text));

extreme.setLeftBorder(MyConvert::toDouble(tLeftBorder->Text));

extreme.setRightBorder(MyConvert::toDouble(tRightBorder->Text));

//Пошук максимуму

if (rbMax->Checked)

{

try

{

goldenSection.findMax();

double tmp = goldenSection.result.getXMax();

lGoldenResult->Text = "";

lGoldenResult->Text += "F(" + round(tmp \* 100000 / 100000.00).ToString() + ") = ";

tmp = goldenSection.result.getMax();

lGoldenResult->Text += round(tmp \* 100000 / 100000.00).ToString() + "\n";

bool foundExcep = false;

extreme.findExtremes();

try

{

extreme.findMax();

}

catch (...)

{

lDerivativeResult->Text = "Не знайдено екстремумів функції";

foundExcep = true;

}

if (!foundExcep)

{

tmp = extreme.result.getXMax();

lDerivativeResult->Text = "";

lDerivativeResult->Text += "F(" + round (tmp \* 1000000 / 1000000.00).ToString() + ") = ";

tmp = extreme.result.getMax();

lDerivativeResult->Text += round (tmp \* 1000000 / 1000000.00).ToString() + "\n";

}

}

catch (...)

{

MessageBox::Show("Помилка! Знайдено неочікувану помилку!");

}

}

//пошук мінімуму

else

{

try

{

goldenSection.findMin();

double tmp = goldenSection.result.getXMin();

lGoldenResult->Text = "";

lGoldenResult->Text = "F(" + (tmp \* 1000000 / 1000000.0).ToString() + ") = ";

tmp = goldenSection.result.getMin();

lGoldenResult->Text += (tmp \* 1000000 / 1000000.0).ToString();

bool foundExcep = false;

extreme.findExtremes();

try

{

extreme.findMin();

}

catch (...)

{

lDerivativeResult->Text = "Не знайдено екстремумів функції";

foundExcep = true;

}

if (!foundExcep)

{

tmp = extreme.result.getXMin();

lDerivativeResult->Text = "";

lDerivativeResult->Text += "F(" + (tmp \* 1000000 / 1000000.0).ToString() + ") = ";

tmp = extreme.result.getMin();

lDerivativeResult->Text += (tmp \* 1000000 / 1000000.0).ToString();

}

}

catch (...)

{

MessageBox::Show("Помилка! Знайдено неочікувану помилку!");

}

}

}

}

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

{

tFunction->Text += "1";

}

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

{

tFunction->Text += "2";

}

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

{

tFunction->Text += "3";

}

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

{

tFunction->Text += "4";

}

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

{

tFunction->Text += "5";

}

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

{

tFunction->Text += "6";

}

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

{

tFunction->Text += "7";

}

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

{

tFunction->Text += "8";

}

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

{

tFunction->Text += "9";

}

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

{

tFunction->Text += "0";

}

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

{

tFunction->Text += ",";

}

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

{

tFunction->Text += "+";

}

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

{

tFunction->Text += "-";

}

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

{

tFunction->Text += "\*";

}

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

{

tFunction->Text += "/";

}

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

{

tFunction->Text += "(";

}

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

{

tFunction->Text += ")";

}

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

{

tFunction->Text += "^";

}

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

{

tFunction->Text += "x";

}

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

{

tFunction->Text += "sin(";

}

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

{

tFunction->Text += "cos(";

}

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

{

tFunction->Text += "tan(";

}

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

{

tFunction->Text += "ctan(";

}

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

{

tFunction->Text += "sqrt(";

}

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

{

tFunction->Text += "ln(";

}

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

{

string tmp;

tmp = MyConvert::toString(tFunction->Text);

if (tmp.length() != 0)

{

tmp = tmp.erase(tmp.length() - 1, 1);

}

tFunction->Text = MyConvert::toSysString(tmp);

}

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

{

tFunction->Text = "";

lDerivativeResult->Text = "";

lGoldenResult->Text = "";

}

private: System::Void rbMax\_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

{

rbMin->Checked = false;

}

private: System::Void rbMin\_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

{

rbMax->Checked = false;

}

};

};

//MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

MyCourseWork::MyForm form;

Application::Run(%form);

}

Class MyConvert

//MyConvert.h

#pragma once

#include <msclr\marshal\_cppstd.h>

#include <string>

#include "MyExceptions.h"

using namespace System::Runtime::InteropServices;

using namespace std;

class MyConvert //Клас, який включає в себе користувацькі переведення типів

{

public:

MyExceptions excep;

//Конструктор без параметрів

MyConvert();

//Перевід з System::String в double

static double toDouble(System::String^ stringValue);

//Перевід з Std::String в double

static double toDouble(string stringValue);

//Перевід з Std::String в System::String

static System::String^ toSysString(string \_value);

//Перевід з System::String в Std::String

static string toString(System::String^ value);

};

//MyConvert.cpp

#include "MyConvert.h"

//Конструктор без параметрів

MyConvert::MyConvert()

{

}

//Перевід з System::String в double

double MyConvert::toDouble(System::String^ stringValue)

{

double value;

try

{

value = double::Parse(stringValue);

}

catch (System::FormatException ^e)

{

//throw new MyExceptions();

return 0;

}

catch (...)

{

return 0;

}

return value;

}

//Перевід з Std::String в double

double MyConvert::toDouble(string stringValue)

{

double value;

try

{

value = atof(stringValue.c\_str());

}

catch (System::FormatException ^e)

{

//throw new MyExceptions();

return 0;

}

catch (...)

{

return 0;

}

return value;

}

//Перевід з Std::String в System::String

System::String^ MyConvert::toSysString(string \_value)

{

return gcnew System::String(\_value.c\_str());

}

//Перевід з System::String в Std::String

string MyConvert::toString(System::String^ value)

{

return msclr::interop::marshal\_as<std::string>(value);

}

Class MyExceptions

//MyExceptions.h

#pragma once

#include <exception>

#include <string>

using namespace std;

class MyExceptions //Клас, який включає в себе всі користувацькі виключні ситуації

{

public:

//конструктор без параметрів

MyExceptions();

//Перевірка того, що кількість дужок які закриваються і відкриваються однакова

bool isValidBrackets(string s);

//Перевірка правильності вводу меж визначення функції

bool isCorrectBorders(double leftBorder, double rightBorder);

//Перевірка чи в числі лише одна кома

const char\* notValidNum();

//Перевірка чи є в рядку число

const char\* canNotGetValidNum();

//Перевірка правильного значення аргументу функції

const char\* notValidArgument();

//Ділення на нуль

const char\* divisionByZero();

//Перевірка того, що знайшли екстремуми

const char\* noZeroDerivative();

};

//MyExceptions.cpp

#include "MyExceptions.h"

//конструктор без параметрів

MyExceptions::MyExceptions()

{

}

//Перевірка чи в числі лише одна кома

const char\* MyExceptions::notValidNum() { return "Not valid number"; }

//Перевірка чи є в рядку число

const char\* MyExceptions::canNotGetValidNum() { return "Can't get valid number"; }

//Перевірка правильного значення аргументу функції

const char\* MyExceptions::notValidArgument() { return "Not valid argument"; }

//Перевірка того, що знайшли екстремуми

const char\* MyExceptions::noZeroDerivative() { return "Not found derivatives equal to zero"; }

//Ділення на нуль

const char\* MyExceptions::divisionByZero() { return "Division by zero"; }

//Перевірка того, що кількість дужок які закриваються і відкриваються однакова

bool MyExceptions::isValidBrackets(string s)

{

int openBrackets = 0;

int closeBrackets = 0;

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

{

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

openBrackets++;

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

closeBrackets++;

}

if (openBrackets != closeBrackets)

{

return false;

}

return true;

}

//Перевірка правильності вводу меж визначення функції

bool MyExceptions::isCorrectBorders(double leftBorder, double rightBorder)

{

return leftBorder < rightBorder;

}

Class ParserResult

//ParserResult.h

#pragma once

#include <string>

using namespace std;

class Result

{

public:

double acc; //частина рядка, який вже обробили

string rest; //частина рядка, яка залишилась

//збереження результату обрахунку рядку

Result(double v, string r);

};

//ParserResult.cpp

#include "ParserResult.h"

//збереження нашого результату обрахунку рядку

Result::Result(double v, string r)

{

acc = v;

rest = r;

}

Class MathParser

//MathParser.h

#pragma once

#include <map>

#include <string>

#include <iostream>

#include <conio.h>

#include "ParserResult.h"

#include <ctype.h>

#include <stdlib.h>

#include "MyExceptions.h"

#include "MyConvert.h"

using namespace std;

class MathParser //Клас, який аналізує рядок і рахує його значення

{

public:

MyExceptions e;

private:

map<string, double> variables;

bool negativeVariable = false;

//метод, який переводить у радіани

double toRadians(double angle);

//конструктор без параметрів

public: MathParser();

//встановлюємо значення змінної

public: void setVariable(string variableName, double variableValue);

//отримуємо значення змінної

public: double getVariable(string variableName);

//метод в який передаємо рядок для визначення значення

public: double Parse(string s);

//пошуку плюсу мінусу та обчислення

private: Result PlusMinus(string s);

//пошук дужок

private: Result Bracket(string s);

// пошук функції або змінної

private: Result FunctionVariable(string s);

//множення, ділення або пошук степеня та обчислення результату

private: Result MulDiv(string s);

//пошук числа

private: Result Num(string s);

// Визначення всіх функцій які ми можемо використовувати

private: Result processFunction(string func, Result r);

};

//MathParser.cpp

#include "MathParser.h"

//метод, який переводить у радіани

double MathParser::toRadians(double angle)

{

return angle \* 3.14 / 180.0;

}

//конструктор без параметрів

MathParser::MathParser()

{

variables = map<string, double>();

}

//встановлюємо значення змінної

void MathParser::setVariable(string variableName, double variableValue)

{

variables[variableName] = variableValue;

}

//отримуємо значення змінної

double MathParser::getVariable(string variableName)

{

if (variables.empty())

{

return 0.0;

}

map<string, double>::iterator it;

it = variables.find(variableName);

if (negativeVariable)

{

setVariable(variableName, (-1)\*(\*it).second);

}

return (\*it).second;

}

//метод в який передаємо рядок для визначення значення

double MathParser::Parse(string s)

{

Result result = PlusMinus(s);

if (result.rest != "")

{

throw e.notValidNum();

}

return result.acc;

}

//пошуку плюсу мінусу та обчислення

Result MathParser::PlusMinus(string s)

{

Result current = MulDiv(s);

double acc = current.acc;

while (current.rest.length() > 0)

{

if (!(current.rest[0] == '+' || current.rest[0] == '-' )) break;

char sign = current.rest[0];

string next = current.rest.substr(1, s.length());

current = MulDiv(next);

if (sign == '+')

{

acc += current.acc;

}

/\*else if (sign == '^')

{

}\*/

else

{

acc -= current.acc;

}

}

return Result(acc, current.rest);

}

//пошук дужок

Result MathParser::Bracket(string s)

{

char zeroChar = s[0];

if (zeroChar == '(')

{

Result r = PlusMinus(s.substr(1, s.length()));

if (r.rest != "" && r.rest[0] == ')')

{

r.rest = r.rest.substr(1, s.length());

}

else

{

cout << "Error: not close bracket";

}

return r;

}

return FunctionVariable(s);

}

// пошук функції або змінної

Result MathParser::FunctionVariable(string s)

{

string f = "";

int i = 0;

if (s[0] == '-' && isalpha(s[1]))

{

negativeVariable = true;

s = s.substr(1, s.length());

}

// пошук функції або змінної(ім\*я обов\*язково повинно починатись з букви

while ((unsigned)i <= s.length() && (isalpha(s[i]) /\* || (isdigit(s[i])&& i >= 0)\*/))

{

f += s[i];

i++;

}

if (f != "")

{ // якщо щось знайшли

if (s.length() >= (unsigned)i && s[i] == '(')

{ // і наступний символ дужка - це функція

Result r = Bracket(s.substr(i, s.length()));//was f.length

return processFunction(f, r);

}

else

{ // інакше - змінна

return Result(getVariable(f), s.substr(i, s.length()));//erase=substr;was f.length

}

}

return Num(s);

}

//множення, ділення або пошук степеня та обчислення результату

Result MathParser::MulDiv(string s)

{

Result current = Bracket(s);

double acc = current.acc;

while (true)

{

if (current.rest.length() == 0)

{

return current;

}

char sign = current.rest[0];

if ((sign != '\*' && sign != '/' &&sign != '^')) return current;

string next = current.rest.substr(1, s.length());

Result right = Bracket(next);

if (sign == '\*')

{

acc \*= right.acc;

}

else

if (sign == '/')

{

if (right.acc == 0)

{

throw e.divisionByZero();

}

acc /= right.acc;

}

else

if (sign == '^')

{

acc = pow(acc, right.acc);

}

current = Result(acc, right.rest);

}

}

//пошук числа

Result MathParser::Num(string s)

{

int i = 0;

int dot\_cnt = 0;

bool negative = false;

// число також може починатись з мінуса

if (s[0] == '-' && isdigit(s[1]))

{

negative = true;

s = s.substr(1, s.length());

}

// дозволяємо лише цифри та точку

while ((unsigned)i < s.length() && (isdigit(s[i]) || s[i] == '.'))

{

// але також робимо перевірку чи лише одна точка

if (s[i] == '.' && ++dot\_cnt > 1)

{

throw e.notValidNum();

}

i++;

}

if (i == 0)

{ // якщо щось схоже на число ми не знайшли

throw e.canNotGetValidNum();

//negativeVariable = true;

//return FunctionVariable(s);

}

else

{

double dPart = atof(s.substr(0, i).c\_str());

if (negative) dPart = -dPart;

string restPart = s.substr(i, s.length());

return Result(dPart, restPart);

}

}

// Визначення всіх функцій які ми можемо використовувати

Result MathParser::processFunction(string func, Result r)

{

try

{

if (func == "sin")

{

return Result(sin(toRadians(r.acc)), r.rest);

}

else if (func == "cos")

{

return Result(cos(toRadians(r.acc)), r.rest);

}

else if (func == "tan")

{

return Result(tan(toRadians(r.acc)), r.rest);

}

else if (func == "ctan")

{

if (tan(toRadians(r.acc) == 0))

{

throw e.divisionByZero();

}

return Result(1 / tan(toRadians(r.acc)), r.rest);

}

else if (func == "exp")

{

return Result(exp(r.acc), r.rest);

}

else if (func == "ln")

{

if (MyConvert::toDouble(r.rest) < 1)

{

throw e.notValidArgument();

}

return Result(log(r.acc), r.rest);

}

else if (func == "sqrt")

{

if (MyConvert::toDouble(r.rest) < 0)

{

throw e.notValidArgument();

}

return Result(sqrt(r.acc), r.rest);

}

else

{

cout << "function '" + func + "' is not defined";

}

}

catch (...)

{

}

return r;

}

Class MaxMinResult

//MaxMinResult.h

#pragma once

class MaxMinResult

{

private:

//значення х, коли функція досягає мінімального значення

double \_xMin;

//значення х, коли функція досягає максимального значення

double \_xMax;

//мінімальне значення функції

double \_min;

//максимальне значення функції

double \_max;

public:

MaxMinResult();

//Конструктор з параметрами

MaxMinResult(double min, double xMin, double max, double xMax);

void setXMin(double xMin);

double getXMin();

void setXMax(double xMax);

double getXMax();

void setMin(double min);

double getMin();

void setMax(double max);

double getMax();

};

//MaxMinResult.cpp

#include "MaxMinResult.h"

MaxMinResult::MaxMinResult()

{

\_min = 0;

\_xMin = 0;

\_max = 0;

\_xMax = 0;

}

//Конструктор з параметрами

MaxMinResult::MaxMinResult(double min, double xMin, double max, double xMax)

{

\_min = min;

\_xMin = xMin;

\_max = max;

\_xMax = xMax;

}

void MaxMinResult::setXMin(double xMin)

{

\_xMin = xMin;

}

double MaxMinResult::getXMin()

{

return \_xMin;

}

void MaxMinResult::setXMax(double xMax)

{

\_xMax = xMax;

}

double MaxMinResult::getXMax()

{

return \_xMax;

}

void MaxMinResult::setMin(double min)

{

\_min = min;

}

double MaxMinResult::getMin()

{

return \_min;

}

void MaxMinResult::setMax(double max)

{

\_max = max;

}

double MaxMinResult::getMax()

{

return \_max;

}

Class Extreme

//Extreme.h

#pragma once

#include <string>

#include <vector>

#include "MathParser.h"

#include "MaxMinResult.h"

using namespace std;

class Extreme

{

public:

MaxMinResult result;

MyExceptions excep;

private:

string \_function;

double \_eps;

double \_leftBorder;

double \_rightBorder;

vector <double> \_derivative;

vector <double> \_points;

MathParser mathParser;

public:

//конструктор без параметрів

Extreme();

//метод, який змінює коми на крапки

void changeComasWithDots(string& str);

//встановлює значення функції

void setFunction(string function);

//отримує значення функції

string getFunction();

//встановлює значення точності

void setEps(double eps);

//отримує значення точності

double getEps();

//отримує значення правої межі

double getRightBorder();

//отримує значення лівої межі

double getLeftBorder();

//встановлює значення лівої межі

void setLeftBorder(double leftBorder);

//встановлює значення правої межі

void setRightBorder(double rightBorder);

//метод, який шукає значення похідної в точці

double findDerivative(double point);

//метод, який шукає екстремуми функції

void findExtremes();

// метод, який визначає значення елементу, де функція набуває максимального значення

void findMax();

//метод, який визначає значення елементу, де функція набуває мінімального значення

void findMin();

};

//Extreme.cpp

#include "Extreme.h"

//конструктор без параметрів

Extreme::Extreme()

{

\_derivative = {};

\_points = {};

\_eps = 0.01;

\_function = "";

}

//метод, який змінює коми на крапки

void Extreme::changeComasWithDots(string& str)

{

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

{

if (str[i] == ',')

{

str[i] = '.';

}

}

}

//встановлює значення функції

void Extreme::setFunction(string function)

{

changeComasWithDots(function);

\_function = function;

}

//отримує значення функції

string Extreme::getFunction()

{

return \_function;

}

//встановлює значення точності

void Extreme::setEps(double eps)

{

\_eps = eps;

}

//отримує значення точності

double Extreme::getEps()

{

return \_eps;

}

//отримує значення правої межі

double Extreme::getRightBorder()

{

return \_rightBorder;

}

//отримує значення лівої межі

double Extreme::getLeftBorder()

{

return \_leftBorder;

}

//встановлює значення лівої межі

void Extreme::setLeftBorder(double leftBorder)

{

\_leftBorder = leftBorder;

}

//встановлює значення правої межі

void Extreme::setRightBorder(double rightBorder)

{

\_rightBorder = rightBorder;

}

//метод, який шукає значення похідної в точці

double Extreme::findDerivative(double point)

{

mathParser.setVariable("x", point + \_eps);

double tmp = mathParser.Parse(\_function);

mathParser.setVariable("x", point);

tmp -= mathParser.Parse(\_function);

return tmp / \_eps;

}

//метод, який шукає екстремуми функції

void Extreme::findExtremes()

{

double tmp;

for (double i = \_leftBorder; i < \_rightBorder; i += \_eps)

{

tmp = findDerivative(i);

if (fabs(tmp) <= \_eps\*100 )

{

mathParser.setVariable("x", i);

\_derivative.push\_back(mathParser.Parse(\_function));

\_points.push\_back(i);

}

}

}

// метод, який визначає значення елементу, де функція набуває максимального значення

void Extreme::findMax()

{

if (\_derivative.empty())

{

throw excep.noZeroDerivative();

}

int max = \_derivative[0];

int index = \_points[0];

for (int i = 0; i < \_derivative.size(); i++)

{

if (max < \_derivative[i])

{

index = \_points[i];

max = \_derivative[i];

}

}

result.setMax(max);

result.setXMax(index);

}

//метод, який визначає значення елементу, де функція набуває мінімального значення

void Extreme::findMin()

{

if (\_derivative.empty())

{

throw excep.noZeroDerivative();

}

int min = \_derivative[0];

int index = \_points[0];

for (int i = 0; i < \_derivative.size(); i++)

{

if (min>\_derivative[i])

{

index = \_points[i];

min = \_derivative[i];

}

}

result.setMin(min);

result.setXMin(index);

}

Class GoldenSection

//GoldenSection.h

#pragma once

#include <string>

#include "MaxMinResult.h"

#include "MathParser.h"

using namespace std;

class GoldenSection

{

public:

MaxMinResult result;

private:

string \_function;

double \_eps; //точність

double \_leftBorder;

double \_rightBorder;

MathParser mathParser;

const double t = (1 + sqrt(5)) / 2;//константа золотого січення

public:

//конструктор без параметрів

GoldenSection();

//конструктор з параметром

GoldenSection(string function);

//метод, який змінює коми на крапки

void changeComasWithDots(string& str);

//встановлюємо значення точності

void setEps(double eps);

// встановлюємо значення лівої межі

void setLeftBorder(double leftBorder);

//встановлюємо значення правої межі

void setRightBorder(double rightBorder);

//отримує значення точності

double getEps();

//отримуємо значення лівої межі

double getRightBorder();

//отримуємо значення правої межі

double getLeftBorder();

//встановлюємо значення функції

void setFunction(string function);

//отримуємо значення функції

string getFunction();

//пошук максимуму та мінімуму методом золотого січення

void find(bool isLess);

//визначає максимум

void findMax();

//визначає мінімум

void findMin();

};

//GoldenSection.cpp

#include "GoldenSection.h"

//конструктор без параметрів

GoldenSection::GoldenSection()

{

\_function = "";

\_eps = 0.0001;

\_leftBorder = -1;

\_rightBorder = 1;

}

//конструктор з параметром

GoldenSection::GoldenSection(string function)

{

\_function = function;

}

//метод, який змінює коми на крапки

void GoldenSection::changeComasWithDots(string& str)

{

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

{

if (str[i] == ',')

{

str[i] = '.';

}

}

}

//встановлюємо значення точності

void GoldenSection::setEps(double eps)

{

\_eps = eps;

}

// встановлюємо значення лівої межі

void GoldenSection::setLeftBorder(double leftBorder)

{

\_leftBorder = leftBorder;

}

//встановлюємо значення правої межі

void GoldenSection::setRightBorder(double rightBorder)

{

\_rightBorder = rightBorder;

}

//отримує значення точності

double GoldenSection::getEps()

{

return \_eps;

}

//отримуємо значення лівої межі

double GoldenSection::getRightBorder()

{

return \_rightBorder;

}

//отримуємо значення правої межі

double GoldenSection::getLeftBorder()

{

return \_leftBorder;

}

//встановлюємо значення функції

void GoldenSection::setFunction(string function)

{

changeComasWithDots(function);

\_function = function;

}

//отримуємо значення функції

string GoldenSection::getFunction()

{

return \_function;

}

//пошук максимуму та мінімуму методом золотого січення

void GoldenSection::find(bool isLess)

{

mathParser.setVariable("x", \_leftBorder);

double tmp = mathParser.Parse(\_function);

result.setMax(tmp);

tmp = mathParser.Parse(\_function);

result.setMin(tmp);

result.setXMax(\_leftBorder);

result.setXMin(\_leftBorder);

do

{

double x1 = \_rightBorder - ((\_rightBorder - \_leftBorder) / t);

double x2 = \_leftBorder + ((\_rightBorder - \_leftBorder) / t);

mathParser.setVariable("x", x1);

double y1 = mathParser.Parse(\_function);

mathParser.setVariable("x", x2);

double y2 = mathParser.Parse(\_function);

if (isLess){

if (y1 >= y2) //min

{

\_leftBorder = x1;

}

else

{

\_rightBorder = x2;

}

}

else

{

if (y1 <= y2) //max

{

\_leftBorder = x1;

}

else

{

\_rightBorder = x2;

}

}

} while (fabs(\_leftBorder - \_rightBorder) > \_eps);

if (isLess){

result.setMin(mathParser.Parse(\_function));

result.setXMin(\_leftBorder);

//return goldenResult.getMin();

}

else

{

result.setMax(mathParser.Parse(\_function));

result.setXMax(\_leftBorder);

//return goldenResult.getMin();

}

//return 0.0;

}

//визначає максимум

void GoldenSection::findMax()

{

find(false);

}

//визначає мінімум

void GoldenSection::findMin()

{

find(true);

}