Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп’ютерних технологій

Кафедра радіофізики та комп’ютерних технологій

КУРСОВА РОБОТА

на тему: «Розробка інтерпретатора для побудови графіків функцій»

Виконав ст. 2 курсу

гр. ФЕІ-25с

\_\_\_\_\_\_\_\_\_\_Тєльний Б.

Науковий керівник

\_\_\_\_\_\_\_\_\_\_доц. Іван ХВИЩУН

Львів 2024

ЗМІСТ

ВСТУП

Аналіз математичних функцій є важливою складовою багатьох технічних і наукових завдань. Графік дозволяє краще зрозуміти поведінку функції, проаналізувати її властивості, а також знаходити закономірності у складних даних.

Ця курсова робота присвячена розробці інтерпретатора для побудови графіків математичних функцій. Актуальність теми зумовлена тим, що в багатьох існуючих рішеннях відсутня простота використання, або вони вимагають великих обчислювальних ресурсів. Створення програмного продукту, за допомогою сучасних мов програмування, дозволяє отримати функціональний інструмент із широкими можливостями.

Об’єктом дослідження є програмні засоби для побудови графіків функцій. Предметом дослідження є методи реалізації інтерпретатора, що забезпечує обробку математичних виразів, їх візуалізацію та взаємодію з користувачем за допомогою графічного інтерфейсу.

Мета роботи полягає у створенні програми, яка дозволяє легко задавати математичні функції, будувати їх графіки та зберігати результати табуляції.

Розробка програми складалася з кількох етапів: вибір технологій, реалізація алгоритмів обробки функцій, створення графічного інтерфейсу та тестування готового продукту. Було детально розглянуто реалізацію алгоритму переведення математичних виразів у зворотний польський запис, метод обчислення значення функції і принципи створення зручного інтерфейсу для взаємодії з користувачем.

Сфери застосування створеного програмного забезпечення дуже широкі. Воно може бути корисним у навчальному процесі для викладачів і студентів, у наукових дослідженнях для аналізу складних функцій, а також для фахівців, які працюють з математичними моделями. Програма дозволяє значно полегшити процес побудови графіків, заощаджуючи час і ресурси.

Таким чином, виконання роботи спрямоване на створення універсального інструменту, який дозволить легко виконувати побудову графіків і сприятиме розвитку навичок програмування, аналізу даних і створення програмного забезпечення.

РОЗДІЛ 1. ТЕХНОЛОГІЧНІ ЗАСОБИ

* 1. Інтерпретована об’єктно-орієнтована мова програмування Python

Python є мовою програмування високого рівня, загального призначення, яка була створена Гвідо ван Россумом у 1991 році. Вона вирізняється простим і зрозумілим синтаксисом, який забезпечує високу читабельність коду, та широким спектром застосувань, включаючи веб-розробку, аналіз даних, автоматизацію, машинне навчання тощо. [1]

Python має ряд ключових характеристик, які виділяють її серед інших мов програмування:

* Код Python виконується рядок за рядком за допомогою інтерпретатора, що спрощує налагодження та дозволяє уникати компіляції;
* Python підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє структурувати код у вигляді класів і об'єктів;
* Програми на Python можуть виконуватися на різних операційних системах, таких як Windows, macos, Linux;
* Python постачається із множиною вбудованих модулів для роботи;
* Синтаксис Python інтуїтивно зрозумілий, що полегшує сприйняття коду.

Мова Python використовується у багатьох галузях, зокрема і для візуалізації та аналізу даних, завдяки таким бібліотекам, як NumPy [1, c.33] та Matplotlib. [1, c.217]

* + 1. Переваги Python:
* Легкий та інтуїтивно зрозумілий код.
* Велика кількість навчальних матеріалів і активна спільнота розробників.
* Універсальність і багато вбудованих інструментів.
* Динамічна типізація та автоматичне керування пам’яттю. [2]
  + 1. Недоліки Python:
* Порівняно низька швидкодія, у порівнянні з компільованими мовами програмування [2].
* Відсутність суворої типізації
  1. Бібліотеки
     1. Tkinter

Tkinter – це інтерфейс до інструментарію GUI Tk, ця бібліотека є стандартною Python, тому не потребує встановлення. Tk не є окремою бібліотекою, а складається з кількох окремих модулів, кожен з яких має свої функції та власну офіційну документацію. [3, c.12]

Tkinter має кілька ключових особливостей, які роблять його зручним для створення простих та ефективних графічних інтерфейсів:

* Tkinter має інтуїтивно зрозумілий API.
* Доступний на більшості платформ Unix, включаючи macos, а також у системах Windows.
* Бібліотека надає інструменти для створення різних елементів інтерфейсу, таких як кнопки, текстові поля, списки, тощо.
* Tkinter постачається разом із Python і не потребує встановлення додаткових модулів для базової роботи.

Основним елементом є головне вікно програми, яке служить базою для розміщення всіх інших компонентів.

Tkinter надає широкий набір віджетів, серед яких:

* Button – кнопки.
* Label – текстові або графічні підписи.
* Entry – однорядкові текстові поля для вводу.

Для розташування віджетів у вікні tkinter використовує три менеджери компоновки:

1. pack() – розташування елементів у вертикальному або горизонтальному напрямку.
2. grid() – розташування елементів у вигляді таблиці.
3. place() – точне позиціонування елементів за координатами.
   * 1. Бібліотека Ttkbootstrap

Ttkbootstrap – це бібліотека для створення сучасних і стильних графічних інтерфейсів користувача (GUI) на Python, що є надбудовою над стандартною бібліотекою Tkinter і використовує віджети з модулю ttk (themed Tkinter). Ttkbootstrap надає велику кількість попередньо налаштованих стилів для віджетів, що дозволяє значно покращити зовнішній вигляд додатків, створених за допомогою Tkinter. Завдяки цьому можна створювати привабливий інтерфейс та вручну налаштовувати стилі. [4]

Ttkbootstrap розширює можливості стандартних віджетів Tkinter та додає наступні переваги:

* Стильний, сучасний вигляд віджетів, що нагадує популярні дизайнерські бібліотеки, такі як Bootstrap.
* Щоб почати використовувати Ttkbootstrap, достатньо мінімальних змін у коді Tkinter. Це робить бібліотеку дуже зручною для розробників, знайомих з Tkinter. [4]
* Підтримує багато стилів, кольорових схем та компонентів, що дозволяє легко налаштовувати інтерфейс.
* Є попередньо налаштовані теми, що дозволяє змінювати вигляд додатку всього за кілька рядків коду.
  + 1. Matplotlib

Matplotlib є однією із найпопулярніших бібліотек Python для візуалізації даних. Вона надає великий набір інструментів для створення різноманітних графіків і діаграм, таких як лінійні графіки, гістограми, діаграми розсіяння, графіки функцій, 3D-графіки та багато іншого. Бібліотека є потужним інструментом для аналізу даних і використовується в наукових, інженерних та статистичних розробках.

Бібліотека надає багато параметрів для налаштування вигляду графіків. Можна змінювати шрифти для всіх текстових елементів. Задавати колір графіка, фону, осей та інших елементів. Кожну лінію можна стилізувати відповідно до потреб. [2, c.217]

Основні компоненти Matplotlib**:**

* Figure: Це основний контейнер, що містить один або більше графіків.
* Axes: Це область в межах Figure, на якій безпосередньо будується графік. Один Figure може містити кілька осей для різних графіків.
* Plot: Це безпосередньо графік або лінія, яка відображається на осі (Axes). Наприклад, лінійний графік або точковий графік.

Переваги Matplotlib:

* можна детально налаштовувати зовнішній вигляд графіків, додаючи різноманітні елементи та змінюючи кольори, шрифти, стилі тощо.
* Matplotlib може працювати з іншими бібліотеками, такими як Ttkbootstrap для відображення графіка безпосередньо на головному вікні програми.

Недоліки Matplotlib:

* може здаватися складною через безліч параметрів налаштувань.
* досить повільно працює на великих обсягах даних.



РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМИ

* 1. Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) – це безкоштовне, потужне і легке середовище розробки, яке підтримує різні мови програмування та інструменти для розробки. Воно користується великою популярністю серед розробників завдяки своїм можливостям, гнучкості та великій кількості плагінів, що дозволяють налаштувати робоче середовище під будь-які потреби.

Було обрано саме це середовище розробки, через наступні переваги:

* VS Code є набагато легшим за багато інших, таких як PyCharm чи Eclipse. Це забезпечує швидший запуск і менше навантаження на систему.
* середовище підтримує розширення, що значно полегшує процес написання коду. Наприклад, для Python є розширення Python, яке забезпечує автодоповнення, дебаггер, інтеграцію з Jupyter Notebook, а також підтримку віртуальних середовищ.
* має вбудовану підтримку для роботи з системами контролю версій. Можна виконувати основні операції безпосередньо в редакторі.
* вбудований термінал дозволяє запускати командний рядок без виходу з редактора. Це зручний інструмент для виконання команд без необхідності відкривати окреме вікно терміналу.
* має вбудований дебаггер, який дозволяє зручно відлагоджувати код за допомогою точок зупинки, перегляду змінних, стека викликів тощо.
* VS Code може допомогти в написанні коду, підказуючи синтаксис, методи та функції для певних об'єктів, класів або бібліотек.
  1. Stack

Стек – це структура даних, що працює за принципом "останній прийшов – перший вийшов" (LIFO, Last In First Out). Це означає, що елементи додаються в структуру даних тільки в один кінець (верх стека) і видаляються також з цього ж кінця. [5, с.247]

Основні операції, які підтримуються стеком:

* Push() – додає елемент на верхівку стека
* Pop() – видаляє елемент з верхівки стека
* Empty() – повертає True, якщо стек порожній
  1. Зворотний польський запис

Зворотний польський запис – це форма математичного запису, у якій оператори записуються після операндів. Цей підхід дозволяє обчислювати математичні вирази без використання дужок і пріоритетів операцій, що робить його зручним для комп'ютерної реалізації.

Зворотний польський запис був розроблений у середині 1950-х років австралійським філософом і спеціалістом з теорії обчислювальних машин Чарлзом Гембліном. Він базувався на польській нотації, яку ще у 1920 році запропонував польський математик Ян Лукашевич. Результати роботи Гембліна були представлені на конференції в червні 1957 року, а пізніше опубліковані в 1957 і 1962 роках. [6]

* + 1. Структура зворотного польського запису

1. Запис операцій складається з послідовності операндів та знаків операцій. При письмовому оформленні виразу операнди розділяються пробілами.
2. Обчислення виразу відбувається зліва направо.
3. Коли у виразі зустрічається знак операції, виконується відповідна дія над двома останніми операндами, які йдуть перед цим знаком, у порядку їх запису.
4. Результат операції замінює у виразі її операнди та знак, після чого обчислення продовжується за тим самим алгоритмом.
5. Підсумковим результатом є значення останньої виконаної операції.

Так як людині простіше сприймати інфіксну форму запису (оператор ставиться між операндами), потрібно вміти переводити запис у такій формі до вигляду зворотного польського запису. Алгоритм був вигаданий Едсгером Дейкстрою та вперше опублікований в 1961 р. Алгоритм базується на структурі даних stack. [7]

* + 1. Алгоритм Дейкстри

1. Проходимо по кожному елементу рядка.
2. При знаходженні числа додаємо його в рядок відповіді.
3. При знаходженні оператора додаємо його в стек.
4. Додаємо до рядка відповіді всі оператори, пріоритет яких вище, за оператор, що розглядається зараз.
5. Якщо зустрічаємо ‘(‘ понижуємо пріоритет всіх наступних операторів
6. Якщо зустрічаємо ‘)’ виконуємо пункт 4.

Для обчислення значення виразу, записано в зворотній польській формі, теж зручно використовувати stack.

* + 1. Обчислення значення виразу

1. Проходимо по кожному елементу рядка.
2. Якщо це число або змінна, то додаємо його в стек.
3. Якщо це оператор, то витягаємо з стека два останніх елементи, здійснюємо операцію та додаємо результат в стек.
4. Якщо це функція, то витягаємо з стека останній елемент, визначаємо значення функції від цього елемента та додаємо результат в стек.

В кінці стек міститиме один елемент – результат.

* 1. Опис GUI

Графічний інтерфейс (Graphical User Interface, GUI) – це форма взаємодії користувача із програмним забезпеченням за допомогою візуальних елементів, які відображаються на екрані. Основною метою графічного інтерфейсу є забезпечення зручного, інтуїтивно зрозумілого способу керування програмами або пристроями. Для зручної роботи користувача створено графічний інтерфейс за допомогою бібліотек Tkinter та ttkbootstrap. Інтерфейс програми організований таким чином, щоб забезпечити швидкий доступ до основного функціоналу.

* + 1. Основні компоненти GUI

1. Введення функції.

Вікно містить текстове поле, де користувач вводить функції

1. Налаштування меж і кроку.

Для побудови графіка користувач має вказати:

* Ліву межу
* Праву межу
* Крок табуляції

1. Відображення графіка функції.

Графік відображається на полотні, створеного за допомогою matplotlib, поділки маштабуються, щоб повністю побачити всі значення в межах заданого діапазону.

* + 1. Взаємодія з програмою

1. Натиснувши кнопку «Побудувати графік», користувач запускає алгоритм обчислення і відображення графіка введеної функції. У випадку помилки (наприклад, введення некоректних меж або функції) виводиться повідомлення про помилку.
2. Графік функції відображається в центральній частині вікна. Він включає осі координат, підписи осей, назву графіка і сітку для зручності аналізу.
3. Користувач може зберегти, у файл формату .dat, табуляцію значень x та y, натиснувши кнопку «Зберегти табуляцію». У цьому файлі міститься індекс точки, координата x та відповідне їй значення y.

Інтерфейс підтримує динамічну взаємодію: введення даних у текстові поля, обробка помилок в реальному часі, збереження результатів у файл для подальшого аналізу.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМИ

1. 1. Стандартизація вигляду функції

Стандартизація зовнішнього вигляду математичної функції – це процес приведення введених користувачем виразів до уніфікованої форми, яка є однорідною та легко оброблюваною програмою. Це перетворення має вирішальне значення для обробки введених даних користувача, які можуть містити варіації або невідповідності, наприклад відсутні оператори або нетрадиційний синтаксис. Застосовуючи системний підхід, програма інтерпретує функцію таким чином, що узгоджується з її внутрішньою логікою для подальших обчислень.

Клас Function використовує метод під назвою \_\_make\_beautiful ([див. додаток А](#Додаток_А).) для вирішення цього завдання. Цей метод готує функцію для синтаксичного аналізу шляхом корегування введених користувачами даних. Одним із ключових аспектів цього методу є його здатність обробляти неявні операції. Наприклад, користувач може ввести 2x замість явного написання 2\*x. Процес стандартизації виявляє це та вставляє необхідний оператор множення.

Іншою особливістю є заміна математичних констант, таких як e та pi, їхніми числовими еквівалентами. Цей крок гарантує, що функція готова до чисельного обчислення без додаткових пошуків. Наприклад, такий вхід, як 2pi, перетворюється на 2\*3,141592653589793, що робить його миттєво обчислюваним.

Програма також обробляє унарні оператори, такі як унарний мінус, гарантуючи, що вони правильно інтерпретуються в контексті дужок. Якщо користувач вводить -x + 2, програма перетворює це на 0 - x + 2, забезпечуючи належне розуміння віднімання.

Запобігання помилкам є значною перевагою стандартизації. Цей процес не тільки підвищує точність обчислень, але й робить програму зручнішою для користувача завдяки прийняттю ширшого діапазону стилів введення. Процес стандартизації готує функцію до наступних кроків. Це гарантує, що функція відповідає правилам математичного синтаксису, залишаючись зрозумілою для програми. Цей крок важливий для підтримки цілісності обчислень і отримання значущих результатів.

* 1. Обробка математичних операцій

Обробка математичних операцій є основою функціональності програми, що дозволяє оцінювати визначені користувачем функції в заданих діапазонах вхідних даних. Реалізація використовує алгоритм, який перетворює функцію у формат, придатний для обчислення, а потім оцінює її для різних значень.

Для цього програма використовує зворотній польський запис (ЗПЗ). Цей формат спрощує обчислення, забезпечуючи природне керування пріоритетом операторів і асоціативністю.

Перетворення в ЗПЗ обробляється методом \_\_parce у класі Function ([див. додаток Б](#Додаток_Б).). Цей метод використовує підхід на основі стека для аналізу функції. Під час обробки вхідних даних він розділяє оператори, функції та операнди, гарантуючи, що структура функції відповідає правилам ЗПЗ.

Обчислення значення функції в точці виконується методом \_\_get\_value у класі Function ([див. додаток В](#Додаток_В).). Кожен операнд поміщається в стек. Коли зустрічається оператор, операнди витягуються зі стеку, виконується операція, а результат повертається в стек. Цей процес триває, доки не буде обчислено весь вираз.

Програма підтримує широкий спектр математичних операцій, включаючи арифметичні, тригонометричні та логарифмічні функції. Кожна операція реалізована за модульним принципом, що дозволяє легко розширювати, якщо в майбутньому потрібно буде додати нові функції. Наприклад, функція sin обчислюється за допомогою вбудованої функції sin Python, а логарифмічні функції обчислюються за допомогою log.

Обробка помилок є невід'ємною частиною обробки. Програма призначена для виявлення та звітування про такі проблеми, як виклик математичної функції без достатньої кількості аргументів чи спроба виконання операції, без достатньої кількості операндів. Надаючи змістовні повідомлення про помилки, програма гарантує, що користувачі можуть виправити свої введення та повторити спробу.

Цей надійний підхід до обробки математичних операцій дозволяє програмі ефективно обробляти складні вирази. Він перетворює дані користувача в точні результати, зберігаючи при цьому продуктивність.

* 1. Створення графічного інтерфейсу програми

Створений з використанням бібліотеки tkinter та її розширення ttkbootstrap, інтерфейс поєднує функціональність з інтуїтивно зрозумілим дизайном, щоб зробити математичні обчислення доступними для користувачів.

Інтерфейс складається з секції введення, секції керування та секції відображення ([див. додаток Г](#Додаток_Г).). У розділі введення користувачі можуть визначити функцію, яку потрібно побудувати, вказати діапазон та розмір кроку для обчислення. Значення за замовчуванням надаються для цих полів, щоб допомогти користувачам під час початкової взаємодії з програмою. Наприклад, стандартною функцією є sin(x) із діапазоном від -10 до 10 і розміром кроку 0,1.

Розділ керування містить кнопки для виконання таких дій, як побудова графіка та збереження результатів. Кнопка побудови графіка запускає обчислення та візуалізацію графіка, а кнопка збереження дозволяє користувачам експортувати обчислені значення у файл ([див. додаток Д](#Додаток_Д).). Кнопка збереження спочатку вимкнена та стає активною лише після успішного обчислення, гарантуючи, що користувачі не зможуть зберегти неповні або помилкові результати.

У розділі відображення будується графік ([див. додаток Е](#Додаток_Е).). Використовуючи бібліотеку matplotlib, програма створює візуальне представлення функції в заданому діапазоні. Для наочності графік містить осі, лінії сітки та позначки. Користувачі можуть легко інтерпретувати графік і визначати ключові характеристики функції, такі як максимуми, мінімуми чи точки перегину.

Інтерфейс розроблено для зручної обробки помилок. Якщо користувач вводить недійсну функцію або діапазон, програма відображає повідомлення про помилку з поясненням проблеми. Цей цикл зворотного зв’язку гарантує, що користувачі можуть виправити свої введення та продовжити виконувати обчислення.

Дизайн інтерфейсу зосереджений на зручності та естетичності. Використовуючи можливості створення тем у ttkbootstrap, програма досягає сучасного вигляду, що приваблює користувачів. Макет організований таким чином, щоб мінімізувати безлад і зробити навігацію інтуїтивно зрозумілою.

РОЗДІЛ 4. ТЕСТУВАННЯ ПРОГРАМИ

A screen shot of a graph

Description automatically generated

Ця функція перевіряє базові операції додавання та множення, а також демонструє, що операції виконуються в правильному порядку.

A screenshot of a graph

Description automatically generated

Ця функція перевіряє роботу з базовими тригонометричними функціями та їх комбінаціями.

A screen shot of a graph

Description automatically generated

Ця функція тестує піднесення до степеня та логарифмування. Також демонструє, що на тій частині координатної площини, де логарифм не визначено, нічого не будується.

* + 1. Побудова графіка

A screenshot of a graph

Description automatically generated

Складна комбінована функція

* + 1. Збереження табуляції

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

Табуляція зберігається в форматі \*.dat, її можна використовувати для глибшого аналізу функції. Якщо значення функції в точці не є визначеним, записане значення відображається, як nan.

ВИСНОВОК

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вандерплас Д. Python Data Science Handbook / Джейс Вандерплас. – Sebastopol, 2016. – 530 с. – (O`Reilly Media).
2. Мови програмування для веброзробки та супутні технології [Електронний ресурс] // Друкарня. – 2024. – Режим доступу до ресурсу: <https://drukarnia.com.ua/articles/movi-programuvannya-dlya-vebrozrobki-ta-suputni-tekhnologiyi-wzeAr>.
3. Грейсон Д. Е. Python and Tkinter Programming / Джон Е. Грейсон. – Greenwich, 2000. – 659 с. – (Manning Publications Co).
4. ttkbootstrap [Електронний ресурс] // github. – 22. – Режим доступу до ресурсу: <https://github.com/israel-dryer/ttkbootstrap?tab=readme-ov-file>.
5. Вступ до алгоритмів (переклад з англійської) / Т.Корвен, Ч. Лейзерсон, Р. Рівест, К. Стайн., 2019. – 1288 с. – (К. І. С.).
6. Складнюк М. Арифметичний калькулятор (Курсова робота) / Складнюк Максим – Бердичів, 2020.
7. Дейкстра Е. В. Stichting Mathematisch Centrum : дис. докт. / Дейкстра Едсгер Вібе – Амстердам, 1961. – 34 с.

ДОДАТКИ

**Додаток А**. Реалізація функції \_\_make\_beautiful()

def \_\_make\_beautiful(self, s):

        s = "(" + s + ")"

        s = s.replace(" ", "")

        temp = ""

        for i in range(len(s)):

            if s[i] in '-+' and s[i - 1] in '(':

                temp += "0"

            if i > 0 and s[i] in '(xcltsep' and s[i - 1] in '0123456789)x':

                temp += "\*"

            temp += s[i]

        s = temp

        s = s.replace("e", str(e))

        s = s.replace("pi", str(pi))

        return s

**Додаток Б**. Реалізація функції \_\_parce()

def \_\_parce(self, s):

        try:

            self.\_\_f.clear()

            operators = deque()

            func = deque()

            temp = ""

            for char in s:

                if char == '(':

                    operators.append('(')

                    func.append(self.\_\_get\_func(temp))

                    temp = ""

                elif char == ')':

                    if temp:

                        if temp == "x":

                            self.\_\_f.append(temp)

                        else:

                            self.\_\_f.append(float(temp))

                        temp = ""

                    while operators and operators[-1] != '(':

                        self.\_\_f.append(operators.pop())

                    if func[-1] != self.\_Operation.RETURNER:

                        self.\_\_f.append(func.pop())

                    else:

                        func.pop()

                    operators.pop()

                elif self.\_\_is\_operator(char):

                    if temp:

                        if temp == "x":

                            self.\_\_f.append(temp)

                        else:

                            self.\_\_f.append(float(temp))

                        temp = ""

                    cur\_operator = self.\_\_get\_func(char)

                    while operators and operators[-1] != '(' and self.\_\_less\_or\_same\_priority(operators[-1], cur\_operator):

                        self.\_\_f.append(operators.pop())

                    operators.append(cur\_operator)

                else:

                    temp += char

        except:

            raise self.\_Error(self.\_Error.BADFUNC)

**Додаток В**. Реалізація функції \_\_get\_value()

def \_\_get\_value(self, value):

        try:

            val\_stack = deque()

            for cur in self.\_\_f:

                if cur == "x":

                    val\_stack.append(value)

                elif type(cur) is float:

                    val\_stack.append(cur)

                elif cur < 15:

                    if not val\_stack:

                        raise self.\_Error(self.\_Error.FUNC)

                    val\_stack.append(self.\_\_make\_unar\_operation(val\_stack.pop(), cur))

                else:

                    if not len(val\_stack) > 1:

                        raise self.\_Error(self.\_Error.BINAROPERATION)

                    b, a = val\_stack.pop(), val\_stack.pop()

                    val\_stack.append(self.\_\_make\_binar\_operation(a, cur, b))

            if val\_stack:

                return val\_stack.pop()

            raise self.\_Error(self.\_Error.BADFUNC)

        except self.\_Error as e:

            self.function = None

            raise e

        except Exception as e:

            return float('nan')

**Додаток Г**. Створення графічного інтерфейсу

main\_window = ttk.Window(title="Побудова графіків функцій", resizable=(False, False), position=(100, 100))

controls = ttk.Frame(main\_window)

controls.pack(padx=5, pady=5)

ttk.Label(controls, text="Функція y = f(x):").grid(row=0, column=0, padx=5, pady=5)

entry\_f = ttk.Entry(controls, width=70)

entry\_f.insert(0, "sin(x)")

entry\_f.grid(row=0, column=1, padx=5, pady=5)

ttk.Label(controls, text="Ліва межа:").grid(row=1, column=0, padx=5, pady=5)

entry\_a = ttk.Entry(controls, width=70)

entry\_a.insert(0, "-10")

entry\_a.grid(row=1, column=1, padx=5, pady=5)

ttk.Label(controls, text="Права межа:").grid(row=2, column=0, padx=5, pady=5)

entry\_b = ttk.Entry(controls, width=70)

entry\_b.insert(0, "10")

entry\_b.grid(row=2, column=1, padx=5, pady=5)

ttk.Label(controls, text="Крок (h):").grid(row=3, column=0, padx=5, pady=5)

entry\_h = ttk.Entry(controls, width=70)

entry\_h.insert(0, "0.1")

entry\_h.grid(row=3, column=1, padx=5, pady=5)

ttk.Button(controls, text="Побудувати графік", command=plot\_graph).grid(row=4, column=0, pady=10)

button\_save = ttk.Button(controls, text="Зберегти табуляцію", command=save, state=tk.DISABLED)

button\_save.grid(row=4, column=1, pady=10)

plot = ttk.Frame(main\_window)

plot.pack()

fig = Figure()

cur\_func = fig.add\_subplot()

canvas = FigureCanvasTkAgg(fig, master=plot)

canvas\_widget = canvas.get\_tk\_widget()

canvas\_widget.pack()

main\_window.mainloop()

**Додаток Д**. Збереження результатів табуляції

def save():

    path = tk.filedialog.asksaveasfilename(defaultextension=".dat", filetypes=[("DAT Files", "\*.dat")])

    if path:

        with open(path, 'w', encoding='utf-8') as file:

            file.write('№\tx\ty\n')

            for i in range(len(x)):

                file.write(f'{i + 1}\t{x[i]}\t{y[i]}\n')

            tk.messagebox.showinfo("Save Successful", f"Results saved to {path}")

**Додаток Е**. Табуляція та побудова графіка

def plot\_graph():

    try:

        f.set\_function(entry\_f.get())

        # print(f.get\_rpn())

        a = float(entry\_a.get())

        b = float(entry\_b.get())

        h = float(entry\_h.get())

        if a >= b:

            raise ValueError("Ліва межа повинна бути меншою за праву.")

        if h <= 0:

            raise ValueError("Крок має бути більшим за 0.")

        global x, y

        x, y = [], []

        i = a

        while i <= b:

            x.append(i)

            y.append(f(i))

            i += h

        cur\_func.clear()

        cur\_func.set\_xlim(a, b)

        cur\_func.axhline(0, color="black")

        cur\_func.axvline(0, color="black")

        cur\_func.grid(linestyle="--")

        cur\_func.set\_title(f"Графік функції {f.function[1:-1]}")

        cur\_func.set\_xlabel("x")

        cur\_func.set\_ylabel("y")

        cur\_func.plot(x, y)

        canvas.draw()

        button\_save.config(state=tk.ACTIVE)

x, y = [], []

f = Function()