Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

кафедра програмних засобів

Самостійна робота

з дисципліни «Основи програмної інженерії» на тему:

«Калькулятор»

Виконав:

ст.групи КНТ-113сп Іван ЩЕДРОВСЬКИЙ

Прийняв:

ст.викладач Олександр КАЧАН

2023

РЕФЕРАТ

КАЛЬКУЛЯТОР, WINDOWS, VISUAL STUDIO, МОВА C#, ФОРМА, ЗВІТ, КЕРІВНИЦТВО ПРОГРАМІСТА

Пояснювальна записка складається з чотирьох розділів.

Розділ «Опис предметної області» містить основні поняття, алгоритм роботи програми та інформаційних потоків підприємства.

Розділ «Постанова завдання» включає мету створення та функції програми, вимоги до проектованої системи, вимоги до надійності програмного продукту, умови роботи програми.

Розділ «Програмування» містить обґрунтування вибору середовища розробки та функціонування системи, основні рішення щодо реалізації компонентів системи.

Розділ «Методика роботи користувача з програмою» включає в себе інструкцію користування програмою.

ЗМІСТ

1 ВСТУП

В сучасному світі спостерігається нестримний розвиток технологій, який приводить до заглиблення взаємозв'язків із сферою комп'ютерних технологій та загальної автоматизації. Це перетворює традиційний підхід до виконання завдань, оскільки тепер багато операцій, які раніше виконували люди, автоматизовані різноманітними програмами. Такий тренд робить роботу, навчання та самовдосконалення більш ефективними.

Широке використання комп'ютерів обґрунтоване їхніми унікальними можливостями, серед яких зберігання великого обсягу інформації, швидка та зручна обробка даних, підвищення точності розрахунків, спрощення введення інформації, зручність структурованого зберігання даних, швидка передача інформації на великі відстані та мінімізація помилок людини. Використання обчислювальної потужності комп'ютерів виявляється розумним рішенням порівняно з традиційними методами розрахунків та обробки інформації.

Ця технологічна трансформація обумовлює появу інноваційних інструментів, таких як калькулятор, який стає невід'ємною частиною нашого повсякденного життя. Використання таких інструментів полегшує не лише процеси вивчення та розвитку, але і забезпечує більш ефективні та точні результати у сферах, де використання технологій стає невід'ємним аспектом. Такий підхід визначає новий етап в еволюції суспільства, сприяючи стрімкому росту інформаційних технологій та покращенню якості життя у всіх сферах людської діяльності.

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Основні поняття

Калькулятор - це електронний пристрій чи програмне забезпечення, яке використовується для виконання різноманітних операцій над числами або формулами алгебри. Він може бути вбудований в різні пристрої, такі як мобільні телефони, комп'ютери, планшети чи смарт-годинники. Калькулятори замінили механічні обчислювальні пристрої, такі як абаки, рахунки та логарифмічні лінійки, розширюючи можливості обчислень.

Залежно від функціональності та призначення виділяють різні типи калькуляторів, такі як найпростіші, бухгалтерські, інженерні, фінансові, програмовані та графічні. Спеціалізовані калькулятори призначені для конкретних областей, таких як фінанси чи будівництво.

Існують настільні та компактні (кишенькові) калькулятори, а деякі моделі мають інтерфейси для підключення до комп'ютерів, друкувальних пристроїв чи зовнішніх модулів пам'яті. Сучасні технології дозволяють вбудовувати функції калькулятора в різноманітні пристрої, від персональних комп'ютерів та стільникових телефонів до наручних годинників.

Термін "калькулятор" також застосовується до спеціалізованих програм, вбудованих у веб-сайти або побутову техніку, які виконують різні розрахунки, наприклад, калькулятор калорій або розмірів одягу. Такий інструментарій став необхідною частиною нашого повсякденного життя, полегшуючи обчислення та спрощуючи багато рутинних завдань.

2.2 Основний алгоритм

Для побудови синтаксичного аналізатора я використовував Метод рекурсивного спуску з LL(1). Цей метод є ефективним і забезпечує чітку та зрозумілу структуру аналізатора, спрощуючи розробку та розуміння коду.

Метод рекурсивного спуску з LL(1) базується на рекурсивних функціях, які відповідають правилам граматики. Для забезпечення LL(1) властивості, я використовував передбачуваність та унікальність правил для кожного нетерміналу. Це дозволяє визначити, яке правило використовувати на основі першого терміналу, що слідує за кожним нетерміналом.

Однією з ключових переваг методу є його легкість в реалізації та зрозумілість коду. Кожна рекурсивна функція відповідає конкретному нетерміналу граматики, що спрощує відладку та модифікацію коду. Крім того, можливість працювати з LL(1) граматикою робить алгоритм високопродуктивним та швидким у виконанні.

Використання Методу рекурсивного спуску з LL(1) дозволило ефективно реалізувати синтаксичний аналізатор, що забезпечує надійну обробку вхідних даних відповідно до граматичних правил мови програмування.

3 ПОСТАНОВА ЗАВДАННЯ

3.1 Мета створення програми

Створення ефективного та надійного синтаксичного аналізатора для математичних виразів, використовуючи метод рекурсивного спуску з LL(1). Основні цілі включають забезпечення точності обчислень, ефективної обробки виразів з врахуванням пріоритету операцій, а також створення легко розширюваної та зрозумілої структури для майбутніх модифікацій та додаткових функцій.

3.2 Функції програми

Програма повинна реалізувати такі функції:

- віднімання операндів;
- додавання операндів;
- множення операндів;
- ділення операндів;
- знаходження сінусу від першого операнду;
- знаходження косінусу від першого операнду;
- знаходження тангенсу від першого операнду;
- знаходження котангенсу від першого операнду;
- знаходження sec від першого операнду;
- знаходження csc від першого операнду;
- обчислення ступеней;
- обчислення логарифму;
- обчислення натурального логарифму;
- програма повинна мати зручний для роботи інтерфейс;

– програма повинна мати зручні для редагування операндів інструменти;

– можливість працювати в радіанах та градусах;

3.3 Вимоги до проєктованої системи

Вимоги до проєктованої системи:

– програма повинна мати зручний, максимально орієнтований на будь-якого користувача інтерфейс;

– програма повинна мати зручні для редагування операндів інструменти;

– програма повинна працювати без інсталяції будь-якого програмного забезпечення;

– обов’язкове виведення на екран результатів обчислення;

3.4 Умови роботи програми

Для функціонування розробленого програмного продукту, а також в разі його доопрацювання, необхідно мати таку мінімальну апаратну платформу: ПК з тактовою частотою мікропроцесора не менше 2 ГГц, з оперативною пам’яттю з мінімальним об’ємом 1Гб для Windows XP/1 Гб для Windows 11/10, з дисковим простором 5 Гб, необхідним для інсталяції додатка. На персональному комп’ютері повинно бути встановлено наступне програмне забезпечення: операційна система Windows XP/7/8/10/11, Microsoft .NET Framework 4.0.

Програма повинна нормально функціонувати під керуванням операційної системи Microsoft Windows 11.

4 ПРОГРАМУВАННЯ

4.1 Обґрунтування вибору середовища розробки системи

Для написання даного програмного продукту обрано таку мову програмування, як C#, а також середовище швидкої та зручної розробки додатків Visual Studio 2022.

Мова C# є об'єктно-орієнтованою та подійно-орієнтованою мовою програмування. Це означає, що весь програмний код повинен бути розміщений в класах. Ядро мови C# досить компактне і практично співпадає з C++. Вся потужність мови реалізована за допомогою класів. До основних понять мови можна віднести ідентифікатори, ключові слова, знаки операцій і роздільники, літерали. Важливо пам'ятати, що компілятор C# чутливий до регістру, тобто, Value і value – це різні ідентифікатори.

C# є суворо типізованою мовою, тобто, кожна змінна повинна бути визначена в програмі до першого використання. Система типів C# складніша за систему C++ за своєю реалізацією. На відміну від C++ визначати змінні можна у будь якому місці програми, а не на її початку. Область дії змінної – блок коду (частина коду, обмежена фігурними дужками {}).

При написанні програми важливо дотримуватися певного стилю іменування змінних. Для цього пропонуються дві нотації: camel - для імен змінних і Pascal – для імен методів і інших ідентифікаторів.

Програми C# виконуються на платформі .NET Framework, яка інтегрована в Windows і містить віртуальну загальномовне середовище виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR Microsoft представляє собою комерційну реалізацію міжнародного стандарту Common Language Infrastructure (CLI), який служить основою для створення середовищ і розробки, що дозволяють спільно використовувати різні мови і бібліотеки.

Вихідний код, написаний на мові C# складений проміжною мовою (IL), яка відповідає специфікаціям CLI. Код на мові IL і ресурси, в тому числі точкові малюнки та рядки, зберігаються на диск у вигляді виконуваного файлу (звичайно з розширенням .exe або .dll). Такий файл називається складанням. Збірка містить маніфест з інформацією про типи, версії, вимог безпеки, мовою і регіональних параметрах для цієї збірки.

При виконанні програми C# середовище CLR завантажує складання і виконує різні дії в залежності від відомостей, які зберігаються в маніфесті. Якщо виконуються всі вимоги безпеки, середовище CLR виконує JIT-компіляції коду на мові IL в інструкції машинного мови. Також середовище CLR виконує інші операції, наприклад автоматичне прибирання сміття, обробку винятків і управління ресурсами. Код, виконуваний середовищем CLR, іноді називають "керованим кодом", щоб підкреслити відмінності цього підходу від "некерованого коду", який відразу компілюється в машинну мову для певної системи. На схемі показані зв'язки між файлами вихідного коду C#, бібліотеками класів .NET Framework, збірками і середовищем CLR, існуючі під час компіляції і під час виконання.

Взаємодія між мовами є найважливішою можливістю .NET Framework. Створюваний компілятором C# код IL відповідає специфікації загальних типів (CTS). Це означає, що цей код IL може успішно взаємодіяти з кодом, створеним з Visual Basic та Visual C++ для платформи .NET або будь-якого іншого CTS-сумісного мови, яких існує вже більше 20. Одна збірка може містити кілька модулів, написаних на різних мовах .NET, і всі типи можуть посилатися один на одного, як якщо б вони були написані на одній мові. Крім служб часу виконання, платформа .NET Framework містить велику бібліотеку, в яку входить понад 4000 класів. Ці класи розподілені по просторах імен, відповідним різним корисних функцій: від операцій файлового вводу і виводу до управління рядками, від синтаксичного аналізу XML до елементів управління Windows Forms. Зазвичай додатка C# активно використовують бібліотеку класів .NET Framework для вирішення типових завдань взаємодії.

Середовище розробки Microsoft Visual Studio 2010 – продукт фірми Майкрософт, який включає в себе інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Цей продукт дозволяє розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Visual Studio 2022 продовжує підтримувати розробку класичних програм та Магазину додатків Windows. Visual Studio, по мірі розвитку ОС Windows, також розвивається разом з нею. У Visual Studio 2010 бібліотеки і мови платформи .NET, а також мову C++, зазнали значних удосконалень, які застосовні до всіх версій Windows. .NET Framework .NET Framework 4.6 Майкрософт пропонує близько 150 нових API і 50 оновлених API для використання додаткових сценаріїв. Наприклад, все більше колекцій тепер реалізують IReadOnlyCollection <T>, що спрощує їх використання. Крім того, згадана раніше платформа ASP.NET 5 пропонує компактну платформу .NET для створення сучасних хмарних додатків. Додатки з Магазину Windows, написані на C# для платформи .NET Framework, тепер можуть отримати перевагу власної платформи .NET, яка компілює програми з власним кодом, а не з IL, а .NET Framework 4.6 також додає RyuJIT, 64-розрядний JIT-компілятор. Нові компілятори C# і VB ("Roslyn") значно зменшують час компонування і надають зрозумілі API аналізу коду. Visual Studio 2010 використовує перевагу Roslyn для надання додаткової рефакторінгу, включаючи вбудоване перейменування, аналізатори і швидкі виправлення. Мови C# і Visual Basic містять безліч невеликих удосконалень, які стосуються базової мови і підтримки IDE. Ці удосконалення роблять процес написання коду .NET ще більш інтуїтивним, зручним і продуктивним.

Новий компілятор Roslyn для C# і Visual Basic не тільки швидше компілює, але також дозволяє використовувати повністю нові сценарії, такі як динамічний аналіз коду, який надає докладний і настроюється відгук і пропозиції безпосередньо в редакторі коду по мірі введення даних. У Visual Studio 2022 лампочки відображаються в лівій частині (при використанні клавіатури) або в підказці (при наведенні покажчика миші на помилку). Лампочка повідомляє в режимі реального часу, що компілятор (можливо, використовує набором правил) виявив проблему в коді і пропонує варіант її вирішення. Якщо ви бачите лампочку, клацніть її для отримання пропозицій, що вимагають дій.

До можливостей C# відносяться наступні:

– реалізація розроблювачем максимально гнучкого та зручного інтерфейсу для свого додатку, який зможе задовольняти потреби навіть споживача з високими вимогами (Для цього використовуються вбудовані елементи керування – візуально створені об'єкти із заданим набором властивостей та методів, які програміст має можливість змінювати);

– створення різноманітних багаторівневих меню, а також панелей інструментів;

– обробка подій миші та клавіатури, виведення на екран різноманітних графічних зображень, а також різноманітних даних;

– керування кольором, налаштовування принтеру, використання стандартних діалогів;

Усі вище перераховані можливості дозволили зупинити свій вибір для рішення поставленої задачі на мові програмування високого рівня C#, та середовищі розробки Visual Studio 2022.

4.2 Обґрунтування вибору середовища функціонування системи

Даний програмний продукт може функціонувати в операційних системах сімейства Windows. Але перевага була надана операційній системі Windows 11. Корпорація Microsoft вперше продемонструвала попередню робочу версію Windows 11, розповівши про її основні переваги і нововведення 28 жовтня 2008 року на конференції розробників в Лос-Анджелесі.[6]

Найвідчутнішим нововведенням розробник називає широке застосування інтерфейсу вводу, що управляється дотиком (touchscreen), в варіанті реагування на кілька дотиків водночас (multi-touch).

Істотно перероблений інтерфейс системи, панель завдань і меню «Пуск». Тепер назви програм і заголовки відкритих документів в панелі завдань не відображаються – тут немає ніякого тексту, а тільки іконки. Дістати доступ до основних функцій можна, не розгортаючи додатків, – досить натиснути на відповідну іконку в панелі завдань правою кнопкою миші. У меню «Пуск» з'явився доступ до тек, що найчастіше відкривалися.

Пошук став більш зручним. Необхідно ввести потрібний елемент у поле пошуку, розташованому в меню "Пуск", – і результати відобразяться миттєво та будуть згруповані за категоріями (документи, зображення, музика, електронна пошта та програми). Здійснюючи пошук у папці або бібліотеці, можна настроїти його, наприклад, застосувавши фільтр дати чи типу файлу. Коли пошук буде завершено, за допомогою панелі попереднього перегляду здійснюється швидке ознайомлення з результатами.

При натисканні мишею на вільне місце на робочому столі, всі відкриті вікна стають прозорими – це допомагає дістатися до віджетів, які в Windows 11 можуть розташовуватися довільно на робочому столі, а не в його певній частині, як в Vista. При перетягуванні вікна воно збільшується, коли користувач утримує його мишкою, і зменшується, коли відпускає в потрібному місці (як в Mac OS). Якщо відкрите вікно підвести до краю робочого столу, воно зменшиться до 50% від номінального розміру – це зручно для організації вікон. Покращено виведення зображення на мульти- екранні системи.

Операційній системі Windows 11 потрібно менше часу на перехід в сплячий режим і вихід з нього, а також на відновлення зв'язку з бездротовими мережами Wi-Fi. Установка і підготовка до роботи запам'ятовуючих та інших пристроїв, що підключаються через роз’єм USB, «займає лічені секунди», пошук інформації на комп'ютері і сортування його результатів відбуваються також значно швидше.

На робочому столі Windows 11 можна розмістити віджети – невеликі додатки, що показують час, картинки, які транслюють текстові новини, що програють музичні або відеофайли і так далі.

В Windows 11 впроваджено нову вдосконалену технологію Natural ClearType, яка має зробити шрифти ще гладкішими.

До операційної системи також вбудовано близько 120 фонових малюнків, унікальних для кожної країни і мовної версії. Всі версії включають 50 нових шрифтів. У існуючих шрифтах пророблена робота над коректним відображенням всіх символів. Windows 7 – перша версія Windows, яка включає більше шрифтів для відображення нелатинських символів, ніж для відображення латинських. Для зручності управління панель управління шрифтами також піддалася поліпшенню. За умовчанням, в ній відображуватимуться лише ті шрифти, розкладка для яких встановлена в системі. Реалізована підтримка Unicode 5.1. Панель пошуку Instant Search тепер розпізнає більше мов. [5]

Нові функції інтерфейсу:

– *shake – y* інтерфейс Windows Aero додана нова функція Aero Shake, що дозволяє скрутити всі неактивні застосування рухом миші; для її активації досить захопити заголовок вікна і трохи «потрясти» вліво-управо;

– *peek – ф*ункція Aero Peek дозволяє відображувати зменшені копії вікон при наведенні миші на значок панелі завдань, перемикатися між вікнами додатку простим кликом по значку, перетягувати і фіксувати на панелі завдань різні вікна і додатки, переглядати робочий стіл одним наведенням в спеціальну область екрану і багато що інше;

– *snap* – аналогічно функції Shake функція Aero Snap дозволяє рухом миші розгортати вікно підлоги-екрану, весь екран або лише по вертикальній осі.

Значною перевагою є просте надання спільного доступу членам домашньої групи за допомогою групи Домашня група. Необхідно лише два або більше ПК під керуванням Windows 11 і можна обмінюватися музикою, зображеннями, відео та документами з іншими користувачами ПК вашої оселі.

Швидкість у Windows 11 підвищено за основними показниками. Тепер операційна система використовує менше пам’яті та запускає фонові служби лише за потреби. Розробники цієї операційної системи працювали над тим, щоб прискорити такі операції, як запуск програм, перехід комп’ютера до режиму сну та відновлення його роботи, а також повторне встановлення підключення до бездротових мереж. Завдяки підтримці 64-розрядних версій можна скористатися всіма перевагами сучасних потужних комп’ютерів, оснащених 64-розрядним процесором.

Покращена робота в бездротовій мережі. Підключення до бездротових мереж на ноутбуці виконується кількома простими рухами. Для цього потрібно лише вибрати потрібну мережу зі списку доступних на панелі завдань та підключитися. Windows запам’ятає мережу, щоб наступного разу встановити до неї підключення автоматично.

Ефективне відтворення із пристроїв завдяки Device Stage – це нова функція у Windows 11, що працює як домашня сторінка для портативних музичних програвачів, смартфонів і принтерів. Після підключення будь-якого сумісного пристрою до ПК на екрані відобразиться меню з інформацією та популярними завданнями, зокрема заряд акумулятора, кількість готових до завантаження фотографій, а також параметри друку.

Центр підтримки – це нова функція у Windows 11, яка дає змогу стежити за повідомленнями стосовно налаштувань безпеки й обслуговування. Повідомлення можна вмикати та вимикати для таких об’єктів, як технологія Windows Defender або Служба захисту користувачів. Якщо Windows потребує уваги, праворуч на панелі завдань відобразиться відповідне сповіщення. При

натисненні на нього можна переглянути рекомендовані способи вирішення будь-яких проблем.

4.3 Основні рішення щодо реалізації компонентів системи

4.3.1 Використані компоненти

Для розробки даного програмного продукту було використано такі елементи(таблиця 4.1).

Таблиця 4.1 — Використані компоненти

Form	Використано для розроблення інтерфейсу програми
Button	Використано для вводу інформації в програму
TextBox	Використано для виводу інформації на інтерфейс програми
TabControl	Використано для табів
TableLayoutPanel	Використано для сітки

4.3.2 Введення інформації у програму

Запис операндів відбувається за допомогою окремих методів на кожній кнопці, які редагують вміст textbox.

Після запису першого числа слід обрати потрібну операцію. Натискаючи на операнд він також як текст додається до textbox. Слід зауважити, що не всі операції потребують двох операндів, деякі працюють лише з першим. Такі операції як sin, cos, tan, arc, csc, sec, log, ln працюють з першим операндом.

Після запису першого операнду та операції відбувається перевірка на наявність знаку операції. Якщо він є – то записується другий операнд. Процес такий же як і запис першого операнду.

4.3.3 Виведення інформації з програми

Коли користувач ввів всі потрібні данні він може запустити алгоритм знаходження відповіді. Якщо все добре – йому покажеться відповідь, якщо погано – повідомлення про помилку

4.3.4 Обчислення

Для обчислення операції було використано методи Math та метод рекурсивного спуску, для якого потрібен парсер типу LL(k). В цій роботі використовується стандартний парсер LL(1) який розуміє, яку саме граматику використовувати базуючись на одному наступному символі.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ

Для старту роботи ти треба запустити файл Calculator.exe. Відкриється ось таке вікно, показане на рисунку рисунку 5.1

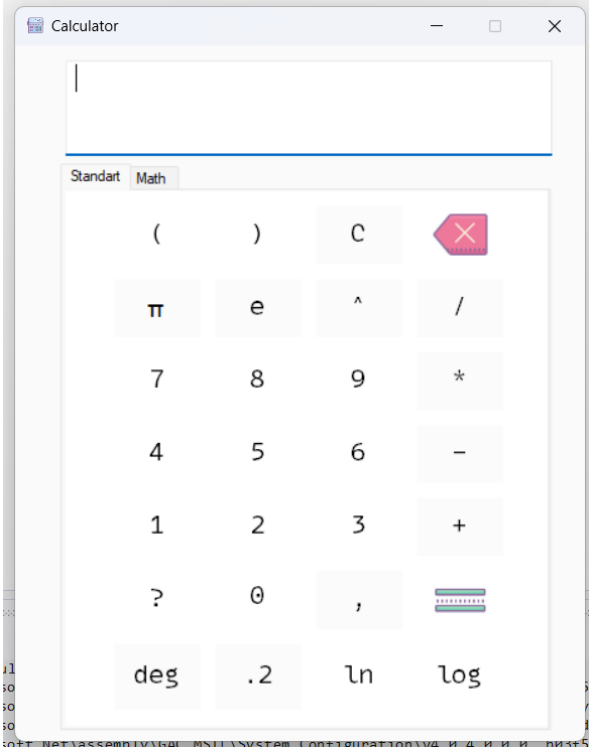

Рисунок 5.1 – Інтерфейс програми

Слід ввести необхідне для розрахунків перше число, просто натискаючи по клавішам 0-9. Показано на рисунку 5.2.

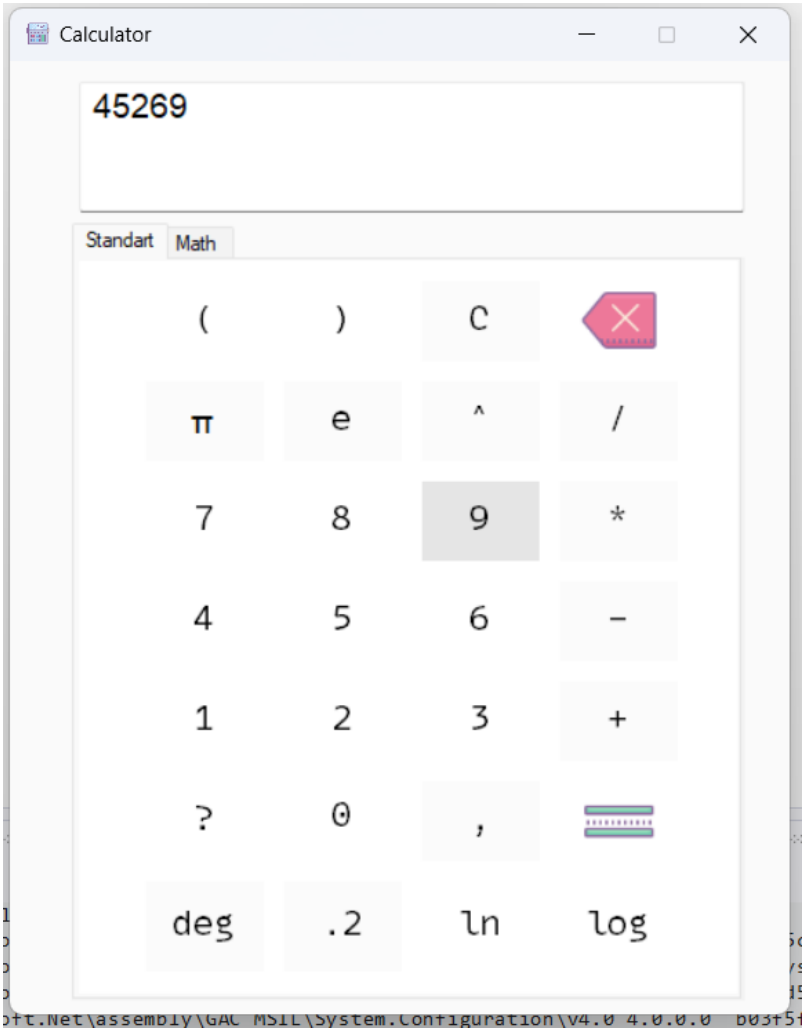

Рисунок 5.2 – Інтерфейс програми з одним операндом

Потім слід обрати потрібну операцію та ввести друге число. Показано на рисунку 5.3.

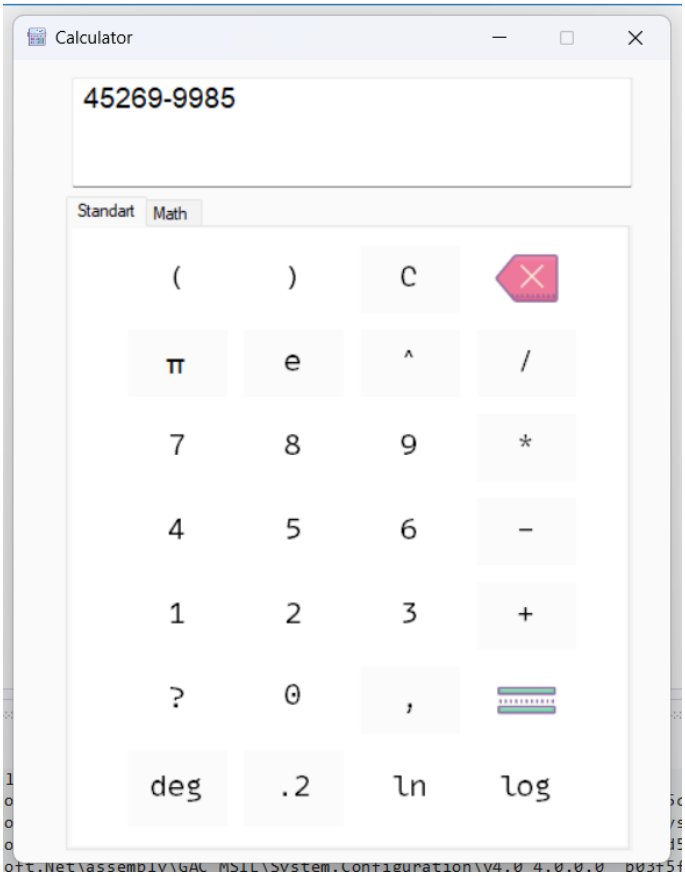

Рисунок 5.3 – Інтерфейс програми з двома операндами

Далі слід натиснути кнопку дорівнює і програма виведе результат на екран. Показана на рисунку 5.4.

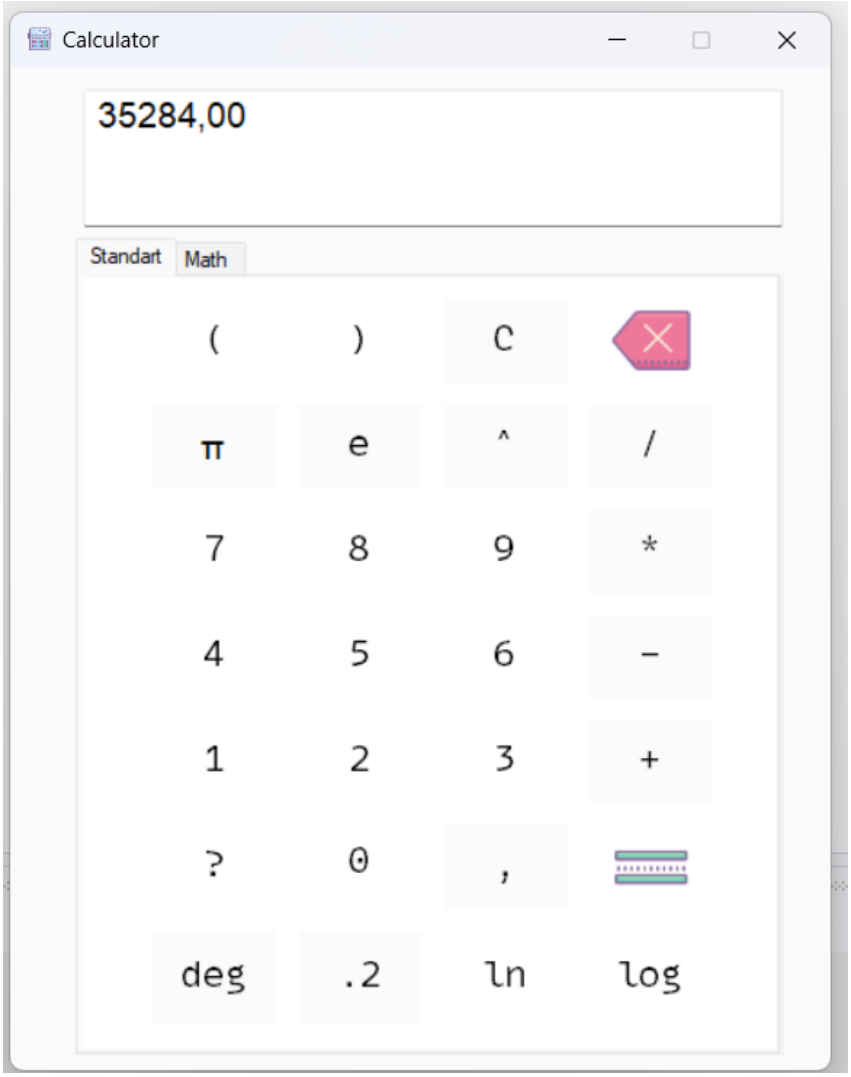

Рисунок 5.4 – Результат обчислення

Також якщо потрібно використовувати додаткові функції, то потрібно перейти на вкладку Math та натиснути на потрібні кнопки для розрахування більш розширених прикладів. Рисунок 5.5.

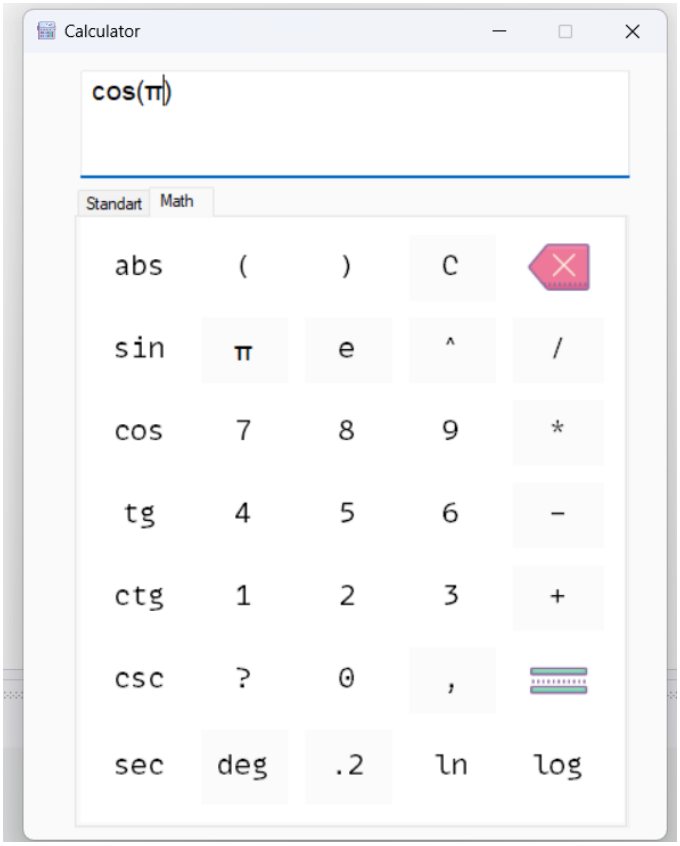

Рисунок 5.5 – Використовування додаткових функцій

6 ВИСНОВКИ

У результаті виконання самостійної роботи було розроблено калькулятор, який може виконувати базові та складні математичні операції. Додаток було створено за допомогою середовища розробки Visual Studio та мови програмування C#. Він працює в операційних системах сімейства Windows.

Розробка калькулятора була завершена успішно. Він є функціональним та зручним у використанні. Календар може бути корисним для широкого кола користувачів, у тому числі студентів, школярів, а також фахівців різних галузей.

Для підвищення функціональності калькулятора можна додати такі можливості:

- підтримку більшої кількості операцій;
- можливість обчислення чисел з плаваючою комою;
- інтерфейс користувача, адаптований для різних типів пристроїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hatch, S.V. Computerized Engine Controls / S.V. Hatch. – Boston: Cengage Learning, 2016. – 688 p.

2. Czichos, H. Measurement, Testing and Sensor Technology. Fundamentals and Application to Materials and Technical Systems / H. Czichos. – Berlin: Springer, 2018. – 213 p.

3. Kaźmierczak, J. Data Processing and Reasoning in Technical Diagnostics / J. Kaźmierczak, W. Cholewa. – Warszawa: Wydawnictwa Naukowo-Techniczne, 1995. – 186 p.

4. Diagnostics as a Reasoning Process: From Logic Structure to Software Design / [M. Cristani, F. Olivieri, C. Tomazzoli, L. Vigano, M. Zorzi] // Journal of Computing and Information Technology. – 2018. – Vol. 27 (1). – P. 43-57.

5. Wieczorek, A.N. Analysis of the Possibility of Integrating a Mining Right-Angle Planetary Gearbox with Technical Diagnostics Systems / A.N. Wieczorek // Scientific Journal of Silesian University of Technology. Series Transport. – 2016. – Vol. 93. – P. 149-163.

6. Tso, B. Classification Methods for Remotely Sensed Data / B. Tso, P.M. Mather. – Boca Raton : CRC Press, 2016. – 352 p.

7. Oppermann, A. Regularization in Deep Learning – L1, L2, and Dropout [Electronic resource]. – Access mode: https://www.deeplearning-academy.com/p/ai-wiki-regularization.

8. Classic Regularization Techniques in Neural Networks [Electronic resource]. – Access mode: https://medium.com/@ODSC/classic-regularization-techniques-in-neural-networks-68bccee03764.

ДОДАТОК А - КОД ПРОГРАМИ

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Calculator_OPI
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        static bool errorActive = false;

        /* Logic */
        static char token;

        static string inputString = "";
        static int inputStringIndex = 0;

        static double t(string input)
        {
            inputString = input;
            inputStringIndex = 0;

            token = DequeueToken();

            double result = Expr();

            return result;
        }

        static void error(string msg)
        {
            Console.Error.WriteLine(msg);
            Environment.Exit(1);
        }

        static char DequeueToken()
        {
            if (inputString.Length > inputStringIndex)
            {
                return inputString[inputStringIndex++];
            }

            return '\0';
        }

        // sin -> s
        // cos -> c
        // tg - t
        // ctg -> b

        static double Factor()
        {
            double value = double.MaxValue;
```

```
                    Console.WriteLine("Input token = " + token);
                    if (token == '(')
                    {
                        match('(');
                        value = Expr();
                        match(')');
                    }
                    else if (token == ')')
                    {
                        value = 0;
                    }
                    else if (char.IsDigit(token) || token == '.' || token == '+'
|| token == '-' || token == 'π' || token == 'e')
                    {
                         char previousToken = token;

                        if (char.IsDigit(token))
                        {
                            value = double.Parse(token.ToString());
                        }
                        else if (token == 'π')
                        {
                            value = System.Math.PI;
                        }
                        else if (token == 'e')
                        {
                            value = System.Math.E;
                        }

                        token = DequeueToken();

                        if (token == '.')
                        {
                            previousToken = '.';
                            token = DequeueToken();
                        }

                        if (char.IsDigit(token))
                        {
                            double newFactor = Factor();

                            if (previousToken == '.')
                            {
                                value = Convert.ToDouble(value.ToString() + ","
+ newFactor.ToString());
                            }
                            else if (value == double.MaxValue)
                            {
                                int negative = (previousToken == '-') ? -1 : 1;

                                value = negative * newFactor;
                            }
                            else
                            {
                                value    =    double.Parse(value.ToString()    +
newFactor.ToString());
                            }
                        }
                    }
                    else if (char.IsLetter(token))
                    {
                        string functionName = "";
                        while (char.IsLetter(token))
```

```
                    {
                        functionName += token;
                        token = DequeueToken();
                    }

                    if (token == '(')
                    {
                        match('(');
                        double expresion = Expr();
                        match(')');

                        switch (functionName.ToLower())
                        {
                            case "sin":
                                value                                       =
System.Math.Sin(transformToDegMode(expresion));
                                break;
                            case "cos":
                                value                                       =
System.Math.Cos(transformToDegMode(expresion));
                                break;
                            case "tan":
                                value                                       =
System.Math.Tan(transformToDegMode(expresion));
                                break;
                            case "cot":
                                value            =            1            /
System.Math.Tan(transformToDegMode(expresion));
                                break;
                            case "sec":
                                value            =            1            /
System.Math.Cos(transformToDegMode(expresion));
                                break;
                            case "csc":
                                value            =            1            /
System.Math.Sin(transformToDegMode(expresion));
                                break;
                            case "ln":
                                value = System.Math.Log(expresion);
                                break;
                            case "log":
                                value = System.Math.Log10(expresion);
                                break;
                            case "abs":
                                value = System.Math.Abs(expresion);
                                break;
                            default:
                                errorActive = true;
                                break;
                        }
                    }
                    else
                    {
                        errorActive = true;
                    }
                }
                else
                {
                    errorActive = true;
                }

                Console.WriteLine("value = " + value.ToString());

                return value;
```

```
}

static double SomeLevel()
{
    double value = Factor();

    while (token == '^')
    {
        switch (token)
        {
            case '^':
                match('^');
                value = System.Math.Pow(value, Factor());
                break;
            default:
                errorActive = true;
                break;
        }
    }

    return value;
}


static double Term()
{
    double value = SomeLevel();

    while (token == '*' || token == '/')
    {
        switch (token)
        {
            case '*':
                match('*');
                value *= SomeLevel();
                break;
            case '/':
                match('/');
                value /= SomeLevel();
                break;
            default:
                errorActive = true;
                break;
        }
    }

    return value;
}

static double Expr()
{
    double value = Term();

    while (token == '+' || token == '-')
    {
        switch (token)
        {
            case '+':
                match('+');
                value += Term();
                break;
            case '-':
                match('-');
                value -= Term();
```

```
                    break;
                default:
                    errorActive = true;
                    break;
            }
        }

        return value;
    }

    static void match(char expected)
    {
        if (token == expected)
        {
            token = DequeueToken();
            return;
        }

        errorActive = true;
    }

    static double transformToDegMode(double value)
    {
        if (degreeMode == "rad")
        {
            return value;
        }

        return value * System.Math.PI / 180;
    }

    /* Logic end */

    static string degreeMode = "deg";
    static int afterDotNumbers = 2;
    static int AFTER_DOT_MAX = 5;

    private void Form1_Load(object sender, EventArgs e)
    {

    }

    private void button_0_Click(object sender, EventArgs e)
    {
        textBox1.Text += '0';
    }

    private void button_result_Click(object sender, EventArgs e)
    {
        double answer = t(textBox1.Text.Replace(',', '.'));

        if (errorActive == true)
        {
            MessageBox.Show("Error!");
            answer = 0;
        }

        textBox1.Text  =  answer.ToString("0."  +  new  string('0',
afterDotNumbers));
    }

    private void button_3_Click(object sender, EventArgs e)
    {
        textBox1.Text += '3';
```

```
}

private void button_2_Click(object sender, EventArgs e)
{
    textBox1.Text += '2';
}

private void button_1_Click(object sender, EventArgs e)
{
    textBox1.Text += '1';
}

private void button_4_Click(object sender, EventArgs e)
{
    textBox1.Text += '4';
}

private void button_5_Click(object sender, EventArgs e)
{
    textBox1.Text += '5';
}

private void button_6_Click(object sender, EventArgs e)
{
    textBox1.Text += '6';
}

private void button_7_Click(object sender, EventArgs e)
{
    textBox1.Text += '7';
}

private void button_8_Click(object sender, EventArgs e)
{
    textBox1.Text += '8';
}

private void button_9_Click(object sender, EventArgs e)
{
    textBox1.Text += '9';
}

private void button_plus_Click(object sender, EventArgs e)
{
    textBox1.Text += '+';
}

private void button_minus_Click(object sender, EventArgs e)
{
    textBox1.Text += '-';
}

private void button_multiply_Click(object sender, EventArgs e)
{
    textBox1.Text += '*';
}

private void button_division_Click(object sender, EventArgs e)
{
    textBox1.Text += '/';
}

private void button_del_Click(object sender, EventArgs e)
{
```

```
                if (textBox1.Text.Length > 0)
                {
                    textBox1.Text                                                    =
textBox1.Text.Remove(textBox1.Text.Length - 1);
                }
            }

            private void button_c_Click(object sender, EventArgs e)
            {
                textBox1.Text = "";
            }

            private void button_pow_Click(object sender, EventArgs e)
            {
                textBox1.Text += '^';
            }

            private void button_dot_Click(object sender, EventArgs e)
            {
                textBox1.Text += ',';
            }

            private void button_pi_Click(object sender, EventArgs e)
            {
                textBox1.Text += 'π';
            }

            private void button_exp_Click(object sender, EventArgs e)
            {
                textBox1.Text += 'e';
            }

            private void button_hint_Click(object sender, EventArgs e)
            {
                MessageBox.Show("Made     by     @ltlaitoff     |     Ivan
Shchedrovskyi<ltlaitoff@gmail.com>");
            }

            private void button_log_Click(object sender, EventArgs e)
            {
                textBox1.Text += "log()";
            }

            private void button_ln_Click(object sender, EventArgs e)
            {
                textBox1.Text += "ln()";
            }

            private   void   button_change_after_dot_Click(object   sender,
EventArgs e)
            {

                afterDotNumbers++;

                if (afterDotNumbers > AFTER_DOT_MAX)
                {
                    afterDotNumbers = 1;
                }

                button_after_dot_1.Text = "." + afterDotNumbers.ToString();
                button_after_dot_2.Text = "." + afterDotNumbers.ToString();
            }

            private void button_deg_mode_Click(object sender, EventArgs e)
```

```
        {
            if (degreeMode == "rad")
            {
                degreeMode = "deg";
            } else
            {
                degreeMode = "rad";
            }

            deg_mode_1.Text = degreeMode;
            deg_mode_2.Text = degreeMode;
        }

        private void button_abs_Click(object sender, EventArgs e)
        {
            textBox1.Text += "abs()";
        }

        private void button_sin_Click(object sender, EventArgs e)
        {
            textBox1.Text += "sin()";
        }

        private void button_cos_Click(object sender, EventArgs e)
        {
            textBox1.Text += "cos()";
        }

        private void button_tg_Click(object sender, EventArgs e)
        {
            textBox1.Text += "tg()";
        }

        private void button_ctg_Click(object sender, EventArgs e)
        {
            textBox1.Text += "ctg()";
        }

        private void button_scs_Click(object sender, EventArgs e)
        {
            textBox1.Text += "csc()";
        }

        private void button_sec_Click(object sender, EventArgs e)
        {
            textBox1.Text += "sec()";
        }

        private void button_left_quot_Click(object sender, EventArgs e)
        {
            textBox1.Text += "(";
        }

        private void button_right_quot_Click(object sender, EventArgs e)
        {
            textBox1.Text += ")";
        }
    }
}
```