МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ЗАПОРІЗЬКИЙ ЕЛЕКТРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Циклова комісія спеціальності

121 Інженерія програмного забезпечення

спеціалізація «Розробка програмного забезпечення»

### РОЗРОБКА ПРОГРАМИ ДЛЯ ЧИСЛОВОГО РІШЕННЯ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ

### Пояснювальна записка до курсової роботи

121.44.07.01 ПЗ

#### 

Викладач Алла СМІРНОВА

Члени комісії Ян БАСОК

Данило БОРОВИК

Студент гр. РПЗ 20 1/9 Олексій ДОЛИННИЙ

2023

**РЕФЕРАТ**

Пояснювальна записка до курсової роботи містить 58 сторінок,   
11 рисунків, 2 таблиці, 2 додатки, 13 джерел.

Метою курсової роботи є розробка програмного продукту для спрощення роботи з диференціальними рівняннями.

Програмне забезпечення розроблене у середовищі Microsoft Visual   
Studio 2019 за допомогою візуальної мови програмування C#.

Програма працює на ПК під керуванням операційної системи Microsoft Windows 10. Застосунок має зручний, максимально орієнтований на кінцевого користувача інтерфейс.

В загальному розділі розглянуті такі питання: опис предметної області, постанова завдання, обґрунтування вибору середовища розробки системи та середовища функціонування системи.

В спеціальному розділі розглянуті основні рішення щодо реалізації компонентів системи, методика роботи користувача з системою, яка включає інструкцію програмісту та інструкцію оператору.

ДИФЕРЕНЦІАЛЬНІ РІВНЯННЯ, ПОХІДНА, ЗАДАЧА КОШІ, ПОРЯДОК РІВНЯННЯ, C#, ОПЕРАЦІЙНА СИСТЕМА WINDOWS, ІНТЕРФЕЙС

**ЗМІСТ**

Вступ 4

1. Загальний розділ 6
   1. Опис предметної області 6
   2. Постанова завдання 9
   3. Обґрунтування вибору середовища розробки системи 10
   4. Обґрунтування вибору середовища функціонування системи 16
2. Спеціальний розділ 18
   1. Основні рішення щодо реалізації компонентів системи 18
   2. Алгоритми вирішення диференціальних рівнянь 20
   3. Методика роботи користувача з системою 24
      1. Керівництво програміста 24
      2. Керівництво оператора 25

Висновки 31

Перелік джерел посилання 32

Додаток А (обов’язковий) Текст програми 33

Додаток Б (обов’язковий) Приклади звітів 57

**ВСТУП**

Метою створення додатку була необхідність чисельного рішення диференціальних рівнянь.

При вирішенні багатьох практичних завдань, необхідно знаходити значення невідомої функції з рівняння, яке містить, поряд з цією невідомою функцією, її похідні. Такі рівняння звуться диференціальними, та відокремлюються в окрему галузь математики.

Диференціальні рівняння відіграють значну роль у математичному моделюванні багатьох завдань у різних галузях науки та техніки. Вони широко використовуються на практиці, зокрема для опису перехідних процесів, коливань, теплопровідності, деформації балок і пластин, поширення електричного струму у провіднику тощо. Тому їх вирішення завжди було одним із найважливіших математичних завдань.

Перші найпростіші диференціальні рівняння з’явилися ще дуже давно, у часи Ісаака Ньютона. З того часу ця область математики неухильно розширювалась. Диференціальні рівняння набули величезного прикладного значення, оскільки вони є інструментом дослідження багатьох завдань у природознавстві і техніці. Вони широко використовуються в механіці, астрономії, фізиці, а також у багатьох завданнях хімії та біології. Причина цього криється в тому, що вони дозволяють кількісно висловити основні закони, яким підпорядковуються певні явища та процеси.

Багато фізичних законів можуть бути записані у вигляді диференціальних рівнянь. Однак їх інтегрування є складним завданням. Деякі диференціальні рівняння можна вирішити явно, тобто знайти потрібну функцію за допомогою формули. Для інших рівнянь такі зручні формули й досі не знайдені, і в таких випадках рішення знаходять із використанням цифрових електронно-обчислювальних пристроїв.

З розвитком інформаційних технологій досягли величезного прогресу й комп’ютери. З використанням методів чисельного наближеного інтегрування диференціальних рівнянь, які забезпечують зручні обчислення з досить високою точністю, сучасна обчислювальна техніка дозволяє отримувати числові результати економічно та швидко. [1]

Мета створення даної програми – спрощення рішення диференціальних рівнянь за допомогою чисельних методів.

Програма розроблена за допомогою середовища програмування   
Visual Studio 2019 на мові програмування C#.

1. **ЗАГАЛЬНИЙ РОЗДІЛ**

**1.1 Опис предметної області**

Предметну область можна визначити як сферу людської діяльності, виділену і описану згідно зі встановленими критеріями.

Під терміном «модель» будемо розуміти будь-який математичний об’єкт, який описує властивості системи, що досліджується.

При цьому під моделлю предметної області розуміється деяка система, що імітує структуру або функціонування досліджуваної предметної області і відповідає основній вимозі - бути адекватної цій області.

Предметною областю являються чисельні методи рішення диференціальних рівнянь.

Теорія диференціальних рівнянь — розділ математики, який розглядає теорію та способи розв'язування диференціальних рівнянь.

Диференціальні рівняння — рівняння, що встановлюють залежність між незалежними змінними, числами (параметрами), невідомими функціями та їхніми похідними.

У застосуваннях математики часто виникають задачі, в яких залежність одного параметра від іншого є невідомою, але можливо записати вираз для швидкості зміни одного параметра відносно іншого (похідної). У цьому випадку задача зводиться до знаходження функції за її похідною відносно з деяких інших виразів.

У випадку одного аргументу диференціальне рівняння називається звичайним; у випадку декількох аргументів — диференціальним рівнянням з частинними похідними.

Порядком диференціального рівняння називається найвищий порядок похідної, що входить до рівняння. [1]

Розв'язком диференціального рівняння порядку n називається функція, що має похідні, до n-ного порядку включно на деякому інтервалі, підставлення якої у рівняння перетворює його у тотожність. Якщо рівняння має розв'язок, то не один, а нескінченну множину; розв'язок може залежати не лише від аргументу, але також від однієї або декількох довільних сталих чи функцій.

Задача Коші — одна з основних задач теорії диференціальних рівнянь — полягає в пошуку розв'язку (інтеграла) диференціального рівняння, що задовольняє початковим умовам (початковим даним).

Початковими умовами або граничними умовами називаються додаткові умови, що накладаються на функцію при розв'язку конкретної задачі, що приводить до диференціального рівняння.

Чисельні методи — методи наближеного або точного розв'язування задач чистої або прикладної математики, які ґрунтуються на побудові послідовності дій над скінченною множиною чисел.

Чисельні методи можна поділити за способом дискретизації на проєкційні, скінченно-різницеві та проєкційно-різницеві, а за способом розв'язання лінійної системи — на прямі, ітераційні та комбіновані методи.

При ітераційних методах можна віднайти розв'язок задачі, використовуючи низку формул, де кожний новий уточнений наближений розв'язок xk обчислюється через попередній xk-1, тобто xk = f(xk-1). Ітераційний процес пошуку наближеного розв'язку завершується тоді, коли виконається умова |xk – xk-1| ≤ ε.

Метод Ейлера - чисельний спосіб розв'язання систем звичайних диференціальних рівнянь. Метод Ейлера є очевидним, однокроковим методом першого порядку точності. Це найбільш базовий вид чисельних методів інтегрування звичайних диференціальних рівнянь. [2]

Чисельне рішення за методом Ейлера для точки xn+1 задається формулою:

де h – крок інтегрування, n – номер ітерації.

Метод Ейлера був історично першим методом чисельного вирішення завдання Коші. О. Коші використав цей метод для доказу існування розв'язання задачі Коші. Через невисоку точність і обчислювальну нестійкість для практичного знаходження рішень завдання Коші метод Ейлера застосовується рідко. Однак через свою простоту метод Ейлера знаходить своє застосування в теоретичних дослідженнях диференціальних рівнянь, завдань варіаційного обчислення та ряду інших математичних проблем.

З метою підвищити точність та стійкість обчислення рішення використовується модифікація методу Ейлера:

Методи Рунге - Кутта — важлива група чисельних методів розв’язування звичайних диференціальних рівнянь. [3]

До класу методів Рунге - Кутта відносяться явний метод Ейлера і модифікований метод Ейлера з перерахунком. Найчастіше використовується та реалізований у різних математичних пакетах класичний метод Рунге — Кутта, що має четвертий порядок точності.

Метод Рунге - Кутта четвертого порядку під час обчислень з постійним кроком інтеграції настільки широко розповсюджений, що його часто називають просто методом Рунге - Кутта. [4]

Тоді значення невідомої функції в точці xn+1 обчислюється відносно значення в попередній точці xn за формулою:

де h — крок інтегрування, а коефіцієнти kn розраховуються наступним чином:

Метод Адамса — група методів чисельного інтегрування звичайних диференційних рівнянь, які дозволяють обчислювати таблицю наближених значень розв'язку за даними в початкових точках. Метод Адамса є багатокроковим методом, тому має значно вищу точність обчислень ніж однокрокові.

Найпростішим багатокроковим методом є двокроковий метод Адамса-Бешфорта.

Цей метод для отримання наступного значення, yn+2, потребує два значення, yn+1 і yn. Один з підходів полягає у використанні y1, обчисленого будь-яким іншим методом. Для двокрокового методу Адамса-Бешфорта наступне значення функції обчислюється за формулою: [5]

У програмі реалізовано наступні методи: модифікований метод Ейлера; класичний метод Рунге-Кутта; метод Адамса-Бешфорта, початкова послідовність значень для якого розраховується за методом Рунге-Кутта.

**1.2 Постанова завдання**

Мета створення даної програми – спрощення рішення диференціальних рівнянь за допомогою чисельних методів. Створюване програмне забезпечення повинно мати наступні функції:

* зручне введення досліджуваного рівняння;
* вибір методу розв’язання;
* швидке обчислення за обраним методом;
* зміна початкових умов для рівняння;
* перегляд результату обчислень;
* графічне відображення результату у вигляді графіків;
* експорт результатів обчислень в окремий файл.

До основних вимог проектованої системи відносяться:

* виведення на екран вихідних форм;
* інтерфейс повинен бути виконаний у вигляді стандартних вікон Windows;
* програма повинна надавати можливість переходу з однієї форми на іншу.

До основних вимог до надійності програми відносяться наступні:

* коректна інтерпретація і зберігання інформації;
* перевірка на коректність введеної користувачем інформації.

Для нормальної роботи програми необхідно мати персональний комп’ютер з такими характеристиками:

* процесор з тактовою частотою не менш 1 ГГц;
* оперативна пам'ять 1 Гб та більше;
* вільний дисковий простір від 100 Мб.

Окрім цього повинні бути у наявності монітор з мінімальним розширенням 854х480 пікселів, клавіатура, та маніпулятор «миша».

На всі пристрої, що використовуються в системі, потрібні драйвери цих пристроїв.

Програма повинна нормально функціонувати під керуванням операційної системи Microsoft Windows 10.

Для перенесення програми на інший комп’ютер достатньо скопіювати папку, яка містить завантажувальний файл програми. Додаткових дій з інсталяції програма не потребує.

**1.3 Обґрунтування вибору середовища розробки системи**

Розробка програмного забезпечення виконана в візуальному середовищі програмування Visual Studio 2019.

Інтегроване середовище розробки Microsoft Visual Studio – це дуже потужний засіб, створений корпорацією Microsoft, який можна використовувати для редагування, налагодження та складання коду, а також для публікації програми. Крім стандартного редактора та відладчика, Visual Studio включає компілятори, засоби завершення коду, графічні конструктори та інструменти для вдосконалення процесу розробки програмного забезпечення, що дозволяє взагалі не залишати цього середовища піж час розробки. Visual Studio не тільки надає всі необхідні інструменти для роботи з програмами, але також допомагає правильно організувати їх. Вона виявляється найбільш ефективною для роботи над великими проєктами, хоча з тим самим успіхом може бути використана і для розробки невеликих програм.

Visual Studio 2019 є найбільш повним інтегрованим середовищем розробки для розробників .NET і C++ у Windows. Повноцінний набір інструментів та функцій дозволяє покращити та вдосконалити кожен етап розробки програмного забезпечення. [6]

.NET — це безкоштовна платформа з відкритим кодом для створення сучасних програм і потужних хмарних служб. .NET є безкоштовним проєктом із відкритим вихідним кодом, розроблений і підтримуваний на GitHub. Він дає змогу писати, запускати та створювати на кількох платформах, включаючи Windows, Linux і macOS. Завдяки великій екосистемі підтримки та потужним інструментам .NET є найпродуктивнішою платформою для розробників. [7]

Новий .NET Framework 4.7.2 - це наступне велике оновлення після випуску v4.7.1 у жовтні 2017 року, в якому додано підтримку .NET Standard 2.0, що визначає API, які повинні надавати всі відповідні реалізації .NET для забезпечення узгодженого використання API в проєкти розробки, замінюючи попередні переносні бібліотеки класів (PCL) як засіб створення бібліотек для всіх сценаріїв.

.NET Framework 4.7.2 містить безліч переваг, включаючи підтримку Windows Forms і WPF, додавання нових API-інтерфейсів JSON, підтримку ARM64 і високою продуктивністю по всіх напрямках. У цю версію також входить C# 9, який включає потоки з нульовим значенням, асинхронні потоки та інші шаблони. Включений F# 5, орієнтований на спрощення синтаксису.

Він активно використовується командами Microsoft та інших компаній у виробництві та для тестування продуктивності. Ці команди демонструють відмінні результати, що демонструють приріст продуктивності та/або можливості зниження витрат на хостинг для своїх веб-додатків.

.NET Framework, як і раніше, залишається оптимальним вибором для настільних проєктів Windows, таких як програми WinForms, WPF і ASP.NET WebForms. [8]

Одним з компонентів пакету Microsoft .NET Framework є CLR (Common Language Runtime — «загальномовне виконуюче середовище») – це середовище для байт-коду CIL (MSIL), в який компілюються програми, написані на .NET-сумісних мовах програмування.

Середовище CLR було створено компанією Microsoft, як реалізація VES (Virtual Execution System – система віртуального виконання), згідно специфікації Microsoft CLI (англ. Common Language Infrastructure – специфікація загальномовної інфраструктури). Код, який виконується в Common Language Runtime, називається керованим кодом. Іншими словами, можна сказати, що CLR забезпечує кероване середовище виконання для .NET програм шляхом покращення безпеки, включаючи міжмовну інтеграцію та багатий набір бібліотек класів тощо. CLR присутній у кожній версії .NET Framework.

Переваги CLR:

* він покращує продуктивність, забезпечуючи багату взаємодію між програмами під час виконання;
* підвищує мобільність, усунувши необхідність перекомпілювати програму в будь-якій операційній системі, яка її підтримує;
* безпека також підвищується, оскільки він аналізує, наскільки безпечні інструкції MSIL. Крім того, використання делегатів замість покажчиків на функції підвищує безпеку типів;
* підтримка автоматичного керування пам'яттю за допомогою Garbage Collector;
* забезпечує міжмовну інтеграцію, оскільки CTS усередині CLR забезпечує загальний стандарт, який дає різним мовам можливість для розширення та спільного використання бібліотек одна одної;
* надає підтримку для використання компонентів, розроблених іншими мовами програмування .NET;
* забезпечення незалежності від мови, платформи та архітектури;
* дозволяє легко створювати масштабовані та багатопотокові програми, оскільки розробнику не потрібно думати про керування пам’яттю та питання безпеки.

CLR є основним компонентом віртуальної машини .NET Framework. Це середовище виконання в .NET Framework, яке запускає коди та допомагає полегшити процес розробки, надаючи різні служби. По суті, він відповідає за керування виконанням програм .NET незалежно від мови програмування .NET .

Загалом, CLR є критично важливим компонентом .NET Framework і відповідає за безпечне, надійне та ефективне виконання програм .NET, що робить його основним аспектом програмування на C#. [9]

Під час компіляції програми C# отриманий виконуваний код містить проміжну мову, Common Intermediate Language (CIL) або Microsoft Intermediate Language (MSIL). Цей код не залежить від комп’ютера, і він може працювати на будь-якій платформі, на якій встановлено CLR. Коли код CIL виконується, CLR компілює його в машинний код, який може бути виконаний процесором.

Для написання програмного забезпечення була обрана мова програмування С#. Мова C# не розповсюджується як окремий продукт – вона є частиною платформи Microsoft .NET Framework.

C# — це сучасна об’єктно-орієнтована мова програмування загального призначення , створена та розроблена компанією Microsoft разом із платформою .NET. Існує дуже різноманітне програмне забезпечення, розроблене на C# та на платформі .NET: офісні програми, веб-програми, веб-сайти, настільні програми, мобільні програми, ігри та багато іншого.

C# є мовою високого рівня, яка схожа на Java і C++ і, певною мірою, на такі мови, як Delphi, VB.NET і C. Усі програми на C# об’єктно-орієнтовані. Вони складаються з набору визначень у класах, які містять методи, а методи містять логіку програми – інструкції, які виконує комп’ютер.

C# вже давно займає своє місце серед найпопулярніших мов програмування на ринку, та лише укріплює свої позиції з часом. Його використовують мільйони розробників у всьому світі. Оскільки C# розроблено Microsoft як частину їхньої сучасної платформи для розробки та виконання додатків .NET Framework, ця мова широко поширена серед орієнтованих на Microsoft компаній, організацій та окремих розробників. Мова C# і платформа .NET повністю обслуговуються та управляються корпорацією Майкрософт і не є відкритими для третіх осіб.

Розпочавшись на платформі Windows, C# розширив свою сферу застосування і зараз доступний на різних операційних системах, включаючи Mac OS, Linux, iOS та Android. Відкритий початковий код C# призвів до активного розвитку та розширення, зменшивши кількість обмежень на використання. Він рекомендується як одна з провідних мов для розробників у будь-якій галузі. [10]

Мова C# має потужний та універсальний інструментарій, що дозволяє вирішувати широкий спектр завдань: його часто розробляються веб-додатків, ігор, мобільних додатків для Android та iOS, а також додатків для платформи Windows. Завдяки багатству інструментів та засобів, можливості для розробки на C# практично не мають обмежень. У той час як інші мови програмування, деякі з яких є більш спеціалізованими і вимагають додаткових сторонніх інструментів, можуть вирішувати свої завдання, C# пропонує швидші, простіші і більш ресурсоефективні рішення для більш широкого спектру завдань.

За стандартом Ecma дизайн C# виконує такі цілі:

* він є простою, сучасною, об’єктно-орієнтованою мовою програмування загального призначення;
* забезпечує підтримку принципів розробки програмного забезпечення, таких як сильна перевірка типів, перевірка меж масиву,  виявлення спроб використання неініціалізованих змінних і автоматичне збирання сміття;
* мова призначена для використання в розробці програмних компонентів, придатних для розгортання в розподілених середовищах;
* підтримка інтернаціоналізації.

Основним поняттям C# є об'єктно-орієнтоване програмування (ООП). Методика ООП не відокремлена від С#, і тому всі програми на C# є об'єктно-орієнтованими хоча б найменшою мірою. Зараз ООП вважається найефективнішим підходом до програмування.

Об’єктно-орієнтовані програми організовані навколо даних, з головного принципу: "дані керують доступом до коду". В об'єктно-орієнтованій мові програмування визначаються дані та код, яким дозволяється впливати на ці дані. Отже, тип даних точно визначає операції, які можуть бути виконані над даними.

Для підтримки принципів ООП усі об'єктно-орієнтовані мови програмування, у тому числі і С#, повинні мати три загальні властивості: інкапсуляцію, поліморфізм і спадкування. [11]

Інкапсуляція – це механізм програмування, що поєднує разом код і дані, якими він маніпулює, виключаючи як втручання ззовні, так і неправильне використання даних. У ООП інкапсуляція здійснюється за допомогою об’єктів. Об’єкт – це елемент, що має певні властивості та поведінку. Форму об’єкта у C# визначає клас. Він описує дані, а також код, який ними оперуватиме. У C# опис класу служить для подальшої побудови об'єктів, які є екземплярами класу.

Поліморфізм – це властивість, що дозволяє одному інтерфейсу отримувати доступ до загального класу дій. Поняття поліморфізму у більш загальному сенсі виражається так: "один інтерфейс - безліч методів". Це означає, що з групи взаємозалежних процесів можна розробити загальний інтерфейс. Поліморфізм допомагає спростити програму, дозволяючи використовувати один і той же інтерфейс для опису загального класу дій. Вибрати конкретну дію (тобто метод) для кожного окремого випадку допомагає компілятор, що ще сильніше спрощує роботу програміста.

Спадкування являє собою процес, в ході якого один об'єкт набуває властивостей іншого об'єкта, що забезпечує ієрархію об’єктів. Це дозволяє визначити для об’єкта лише ті властивості, які роблять об'єкт особливим у його класі.

Метод у C# є членом класу, який може бути викликаний як функція. Як і в інших синтаксично подібних мовах, сигнатура методу - це оголошення, що містить: будь-які додаткові ключові слова доступності (такі як private), явну специфікацію його типу повернення (наприклад int), ім’я методу та послідовність специфікацій параметрів, розділених комами, у дужках, кожна з яких складається з типу параметра, його назви та, за бажанням, значення за замовчуванням.

У C# покажчики адреси пам’яті можна використовувати лише в межах блоків, спеціально позначених як небезпечні, а програми з небезпечним кодом потребують відповідних дозволів для запуску. Більшість доступу до об’єктів здійснюється через безпечні посилання на об’єкти, які завжди або вказують на «живий» об’єкт, або мають чітко визначене нульове значення.

Керовану пам'ять не можна явно звільнити; натомість автоматично збирається сміття. Збирання сміття вирішує проблему витоків пам’яті , звільняючи програміста від відповідальності за звільнення пам’яті, яка в більшості випадків більше не потрібна. [12]

Враховуючи всі перелічені пункти для написання програмного забезпечення було обрано мову програмування С#, яка є найбільш підходящою мовою програмування для розробки програмного продукту.

**1.4 Обґрунтування вибору середовища функціонування системи**

Середовищем функціонування програмного продукту обрана операційна система Wіndоws 10.

Windows 10 в порівнянні з іншими ОС має багато технічних переваг, що роблять цю операційну систему досить потужною і функціональною для своїх користувачів. До цих переваг відносяться:

* звичний та інтуїтивно зрозумілий дизайн;
* підтримка кількох робочих столів;
* підтримка операційної системи на різних типах ПК та переносних пристроях;
* адекватні системні вимоги;
* підтримка нових технологій;
* можливість легко перемикатися між різними вікнами та програмами, працювати з ними одночасно та зручно організовувати робочий стіл;
* покращена безпека, включаючи удосконалений захист від шкідливих програм та шифрування даних;
* покращений центр діяльності, що групує всю важливу інформацію та повідомлення у одному місці. [13]

Windows 10 в порівнянні з іншими своїми версіями має наступні переваги:

* удосконалена продуктивність. Windows 10 має багато оптимізована, що поліпшує продуктивність під час розробки програм;
* широкий набір інтегрованих інструментів розробки. Windows 10 сумісна з багатьма найкращими інструментами розробки, такими як Microsoft Visual Studio, який є одним з найпоширеніших ІDE для розробки програмного забезпечення. Інші популярні інструменти, такі як PowerShell, Git, SSH тощо, також доступні для розробників;
* легкість розгортання. Windows 10 має простий процес розгортання програм на пристроях з цією операційною системою. Розробники можуть легко розповсюджувати та установлювати свої програми на Windows Store;
* сумісність з попередніми версіями. Windows 10 залишається сумісним з багатьма існуючими програмами та додатками, розробленими для попередніх версій Windows. Це означає, що програміст можете легко розробляти та тестувати нові програми на Windows 10, не хвилюючись про сумісність зі старими версіями операційної системи;
* підтримка громадських стандартів. Windows 10 активно підтримує громадські стандарти, такі як HTML, CSS, JavaScript та інші. Це дозволяє програмістам використовувати відкриті технології та стандарти для розробки програм на Windows 10. [13]

Таким чином, Windows 10 – потужна і функціональна операційна система, яка має багато технічних переваг та володіє усіма функціями, що є необхідними для розробки програмного продукту, та відповідає усім поставленим вимогам.

2. **СПЕЦІАЛЬНИЙ РОЗДІЛ**

**2.1 Основні рішення щодо реалізації компонентів системи**

При запуску програми завантажується головний модуль програми Main. В ньому вмикаються візуальні стилі, встановлюється значення сумісного відтворення тексту за замовченням, та завантажується головна форма.

Користувацький інтерфейс – це спосіб взаємодії між користувачем і пристроєм. Графічний інтерфейс передбачає відображення інформації та елементів керування за допомогою графічних об’єктів, які користувач може бачити та з якими він може взаємодіяти. Інтерфейс забезпечує зручність і ефективність роботи, та впливає на задоволення користувачів під час взаємодії з програмним забезпеченням. Для забезпечення гарного користувацького досвіду розробник при створенні інтерфейсу має дотримуватися таких принципів:

* інтерфейс не має бути перевантажений надлишковою інформацією і функціями;
* інтерфейс повинен бути зрозумілим і забезпечити можливість користування для будь-якого користувача;
* інтерфейс повинен бути легким у сприйнятті і зрозумілим для користувачів різного рівню знань і досвіду;
* інтерфейс має бути виконаний в єдиному стилі та мати єдину структуру;
* інтерфейс повинен забезпечувати швидке і ефективне виконання завдання користувачем;
* інтерфейс повинен мати привабливий і естетичний зовнішній вигляд;
* інтерфейс повинен виконувати усі поставлені задачі і задовольняти усі потреби, що можуть з’явитись під час роботи користувача, надаючи всі необхідні функції і опції.

Інтерфейс програми був виконаний у стандартному стилі програм Windows. Завдяки використанню зрозумілої для всіх користувачів Windows логічної структури програма має інтуїтивно зрозумілий інтерфейс. Такий стиль є зручним і простим для сприйняття.

Компоненти середовища розробки, що були використані при побудові інтерфейсу розробленого програмного продукту приведено в таблиці 2.1.

Таблиця 2.1– Перелік використовуваних компонентів C#

|  |  |
| --- | --- |
| Назва компоненту | Призначення компоненту |
| Button | Компонент, який забезпечує функції кнопки, що широко  застосовується в діалогових вікнах. Процедура обробки  задається в події Click |
| ComboBox | Являє елемент керування у вигляді поля зі списком Windows. Властивість Items – колекція елементів списку |
| Panel | Являє собою елемент керування для Windows, який відокремлює елементи у групу |
| TextBox | Призначений для введення і виведення текстової інформації |
| Label | Напис на формі |
| ListBox | Компонент, який виконує функцію відображення даних у вигляді списку |
| Chart | Відображає дані у виді точок на графіках |
| MenuStrip | Являє собою елемент керування у вигляді горизонтального меню з випадаючими списками. |
| ContextMenuStrip | Являє собою контекстне меню, яке з’являється при натисканні правою кнопкою миші |
| SaveFileDialog | Елемент керування, призначений для збереження даних у файлі |
| NumericUpDown | Призначений для введення і виведення числової інформації з можливістю обмеження діапазону значень |
| TabControl | Являє собою елемент керування у вигляді меню з можливістю перемикання вкладок |
| GroupBox | Відокремлює елементи у групу та дозволяє задати заголовок групи |
| FlowLayoutPanel | Відокремлює елементи у групу та дозволяє прокручувати елементи панелі |
| FolderBrowserDialog | Елемент керування, призначений для вибору папки |
| TreeView | Відображає ієрархічну колекцію об’єктів |
| RichBox | Забезпечує додаткові можливості вводу та редагування тексту |

Розроблена програма складається з наступних файлів.

Призначення кожного з файлів представлено у таблиці 2.2

Таблиця 2.2 – Опис файлів програми

|  |  |
| --- | --- |
| Модуль | Опис модулю |
| Diff.sln | Скомпонований файл додатку |
| App.config | Конфігураційний файл програми |
| Program.cs | Забезпечує процес побудови та впровадження |
| Form1.cs | Логіка роботи головної форми |
| Form1.Designer.cs | Інтерфейс головної форми |
| FormModalMethods.cs | Логіка роботи форми модального вікна для введення кількості методів для одночасного обчислення |
| FormModalMethods. Designer.cs | Інтерфейс форми модального вікна |
| FormChart.cs | Логіка роботи форми для графіка |
| FormChart.Designer.cs | Інтерфейс форми для графіка |
| FormHelp.cs | Логіка роботи форми довідки |
| FormHelp. Designer.cs | Інтерфейс форми довідки |
| AboutBox.cs | Логіка роботи форми про программу |
| AboutBox.Designer.cs | Інтерфейс форми |

**2.2 Алгоритми вирішення диференціальних рівнянь**

Перед рішенням формулюється задача Коші. Дано диференціальне рівняння:

і початкова умова . Потрібно знайти функцію на відрізку від і , яка задовольняє як вказаному рівнянню, так і початковій умові (при цьому завжди передбачається, що існує єдине рішення на всьому відрізку). [2]

Іншими словами, потрібно знайти інтегральну криву , що проходить через задану точку .

В основі чисельних методів розв'язання диференціальних рівнянь лежить розкладання функції y ряд Тейлора в округах вихідної точки :

де h - крок між вихідною точкою і точкою , в якій шукається рішення.

Причому в різних методах враховується різна кількість членів розкладання, що визначає точність обчислень.

Ідея методу Ейлера заснована на тому, що шукана інтегральна крива , яка проходить через точку , відновлюється у вигляді кусочно-лінійної ламаної з вершинами . Кожен відрізок цієї ламаної має напрям, що співпадає з напрямом тієї інтегральної кривої рівняння, яка проходить через точку . [3]

Тоді . Тангенс кута нахилу дотичної до в точці дорівнює , а значить, дорівнює , звідки і випливає формула методу Ейлера.

Таким чином, у методі Ейлера значення обчислюються рекурентно за формулою:

Погрішність методу Ейлера, що оцінюється для величини , має порядок O(h), тобто існує деяка константа така, що:

Порівняння йде в крайній (правій) точці відрізка , оскільки саме в ній погрішність теоретично буде максимальною, відносно проміжних точок .

Метод Ейлера діє невисоку точність при розв’язанні задачі Коші, через що його рідко застосовують на практиці, тому слід розглянути його модифікації.

У модифікації методу Ейлера значення обчислюється рекурентно за формулами :

Ідея цієї модифікації методу Ейлера полягає в тому, що спочатку обчислюється “грубе наближення” і за ним обчислюється значення , а потім отримується остаточне наближення , в якому фігурує середнє значення .

Погрішність модифікації методу Ейлера, що оцінюється для величини , також має порядок , тобто існує деяка константа така, що Значення функції з правої частини диференціального рівняння також обчислюється двічі на кожному кроці цієї модифікації.

Ідея методу Рунге-Кутта заснована на застосуванні, на відміну від методу Ейлера, кривих вищого порядку для відновлення значень шуканої функції. на відрізку . [4]

Найчастіше при розв’язанні практичних задач, використовується метод Рунге-Кутта четвертого порядку. Згідно з цим методом значення обчислюється рекурентно за формулою:

де

.

Погрішність методу Рунге-Кутта 4-го порядку, що оцінюється для величини , має порядок . Погрішність при використанні цього методу нижча, ніж при використанні методу Ейлера і його модифікацій, але така точність потребує додаткових обчислень. Функція з правої частини диференціального рівняння обчислюється на кожному кроці методу Рунге-Кутта 4-го порядку чотири рази, що у випадках, коли обчислення значення функції є дуже трудомістким, може істотно уповільнити пошук розв’язку.

В основі методу Адамса-Бешфорта лежить формула Ньютона-Лейбніца:

де - будь-яка первісна для підінтегральної функції на відрізку .

Застосування формули Ньютона – Лейбніца до рівняння на будь-якому відрізку приводить до рівняння:

Тоді, якщо вже були отримані значення в попередніх точках (вузлах) , то підінтегральну функцію можна замінити інтерполяційним поліномом Ньютона , побудованим за цими вузлами. Таким чином, отримується загальна рекурентна формула методу Адамса-Бешфорта порядку: [5]

Оскільки - поліном, то інтеграл від нього береться аналітично і таким чином отримується різницева схема роз’вязання задачі Коші для будь-якого . Наприклад, якщо , то і з цієї формули витікає:

де .

Таким чином, при метод Адамса-Бешфорта співпадає з методом Ейлера.

Найчастіше застосовують метод Адамса-Бешфорта 4-го порядку, який має таку різницеву схему:

де

Перш ніж застосовувати метод Адамса-Бешорта 4-го порядку, необхідно спочатку обчислити значення , , для перших 3-х кроків. Зазвичай для цього застосовують метод Рунге-Кутта, бо він має достатньо велику точність.

Погрішність методу Адамса-Бешфорта 4-го порядку, що оцінюється для величини , має порядок Тому для розв’язання задачі Коші метод Рунге-Кутта 4-го порядку і метод Адамса-Бешорта 4-го порядку дають майже однакові результати.

**2.3 Методика роботи користувача з системою**

**2.3.1 Керівництво програміста**

Програма призначена для чисельного рішення диференціальних рівнянь.

Програма розроблена за допомогою середовища програмування   
Visual Studio 2019 на мові програмування C#.

Для нормальної роботи програми необхідно мати персональний комп’ютер з такими характеристиками:

* процесор з тактовою частотою не менш 1 ГГц;
* оперативна пам'ять 1 Гб та більше;
* вільний дисковий простір від 100 Мб.

Окрім цього повинні бути у наявності монітор з мінімальним розширенням 854х480 пікселів, клавіатура, та маніпулятор «миша».

На всі пристрої, що використовуються в системі, потрібні драйвери цих пристроїв.

Програма повинна нормально функціонувати під керуванням операційної системи Microsoft Windows 10.

Створюване програмне забезпечення реалізує наступні функції:

* зручне введення досліджуваного рівняння;
* вибір методу розв’язання;
* швидке обчислення за обраним методом;
* зміна початкових умов для рівняння;
* перегляд результату обчислень;
* графічне відображення результату у вигляді графіків;
* можливість експорту результатів обчислень в окремий файл.

Файл проєкту має назву програми має назву Diff.sln.

Налаштування користувача програми зберігаються у файлі: С:\Users\[user name]\AppData\Local\[(Project Name)]\[name project\_cashBuild]\[AssemblyVersion]\ user.config.

Налаштування за замовчуванням програми зберігаються у файлі: Properties\Settings.settings у папці проєкту.

Після корегування модулів проекту необхідно перекомпілювати програму.

В програмі використовується подійний метод передачі керування та даних.

Приклади звітів наведені в додатку Б.

**2.3.2 Керівництво оператора**

Програма призначена для чисельного рішення диференціальних рівнянь.

Для нормальної роботи програми необхідно мати персональний комп’ютер з такими характеристиками:

* процесор з тактовою частотою не менш 1 ГГц;
* оперативна пам'ять 1 Гб та більше;
* вільний дисковий простір від 100 Мб.

Окрім цього повинні бути у наявності монітор з мінімальним розширенням 854х480 пікселів, клавіатура, та маніпулятор «миша».

На всі пристрої, що використовуються в системі, потрібні драйвери цих пристроїв.

Програма повинна нормально функціонувати під керуванням операційної системи Microsoft Windows 10.

Для встановлення програмного продукту необхідно скопіювати файли програми на комп’ютер користувача в окрему папку.

Запуск програми відбувається із середовища Microsoft Windows шляхом відкриття файлу diff.exe.

Основні обчислення відбуваються на головній формі. Для розрахунку точок необхідно ввести початкові умови і натиснути кнопку «Розрахунок». Розраховуються значення точок і виводяться у вигляді списку та відмічаються на графіку при відповідних налаштуваннях (рисунок 2.1).

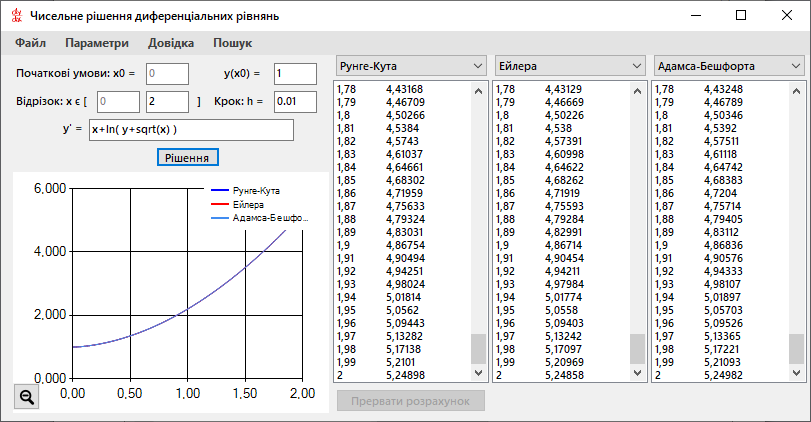


Рисунок 2.1 – Головна форма програми

При натисненні на кнопку «Довідка» або натисканні клавіші F1 відкривається форма, представлена на рисунку 2.2.



Рисунок 2.2 – Форма для перегляду довідки користувача

При натисненні меню «Файл>Експортувати графік» або «Файл>Експортувати точки» відкривається вікно збереження файлу (відповідно рисунки 2.3, 2.4).

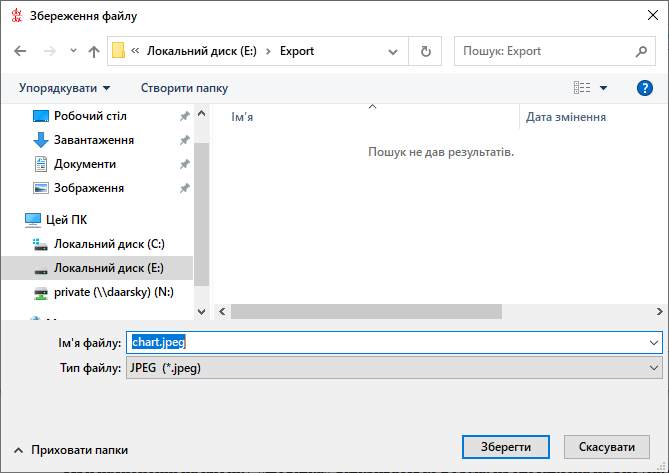


Рисунок 2.3 – Вікно збереження файлу графіку

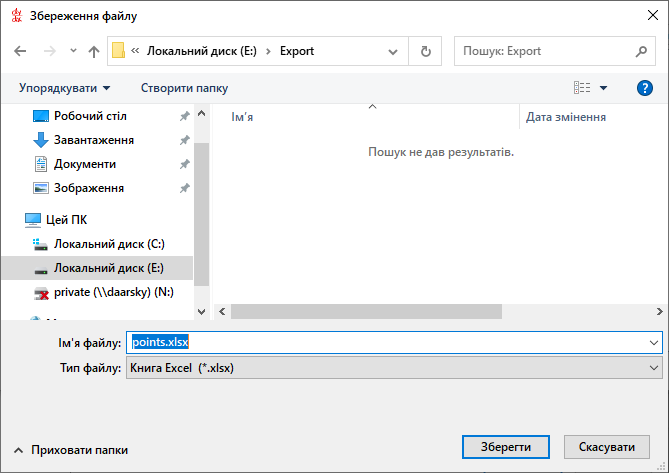


Рисунок 2.4 – Вікно збереження файлу точок

Приклади звітів наведені в Додатку Б.

При натисненні меню «Параметри>Налаштування» відкривається форма налаштувань програми (відповідно рисунки 2.5, 2.6, 2.7, 2.8).

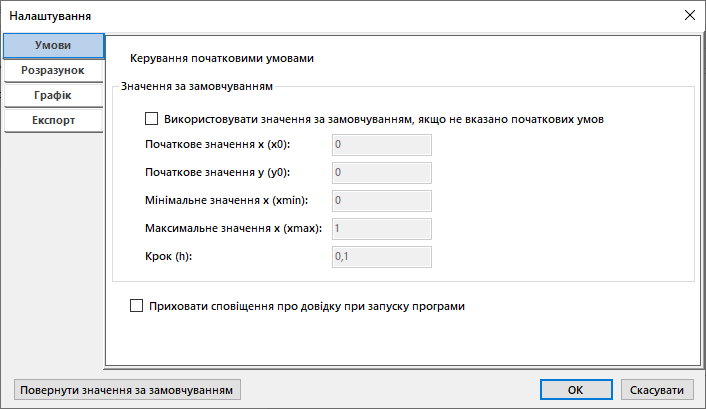


Рисунок 2.5 – Форма «Налаштування>Умови»

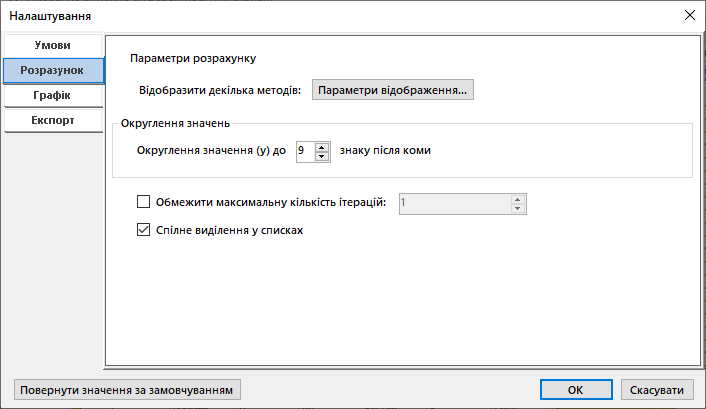


Рисунок 2.6 – Форма «Налаштування>Розрахунок»

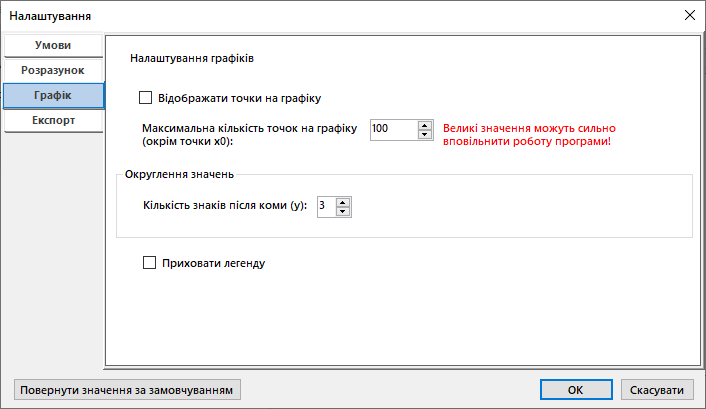


Рисунок 2.7 – Форма «Налаштування>Графік»

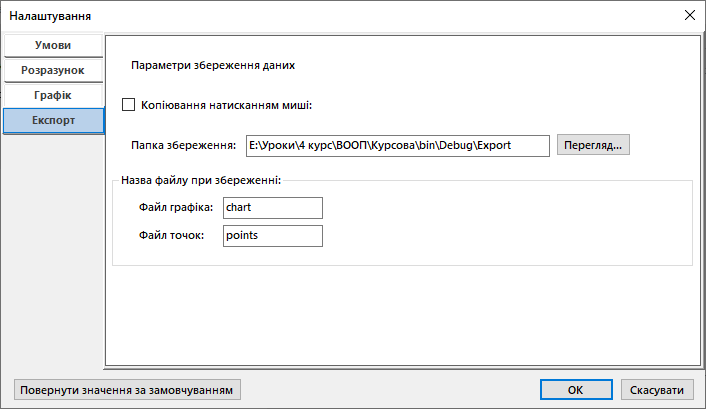


Рисунок 2.8 – Форма «Налаштування>Експорт»

При натисненні «Пошук» відкривається меню пошуку, де можна обрати ввести потрібне значення. Пошук відбувається миттєво. У результаті відмічається перша точка у кожному списку, що підпадає під умову (рисунок 2.9).

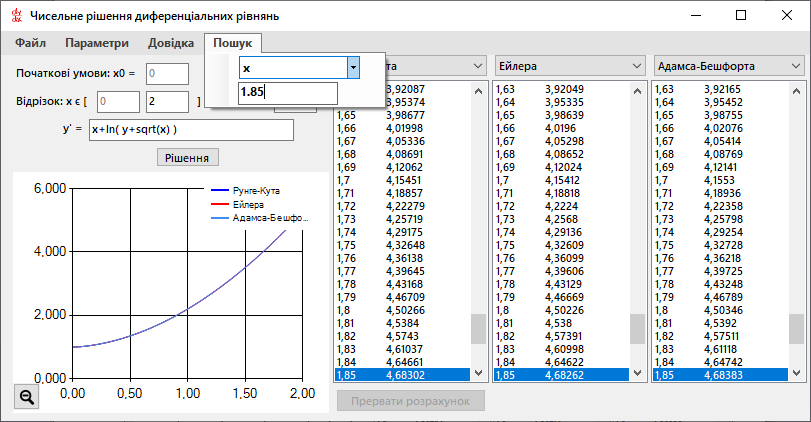


Рисунок 2.9 – Результат пошуку

**ВИСНОВКИ**

В результаті виконання курсової роботи була розроблена програма, призначена для чисельного рішення диференціальних рівнянь. Програма має наступні функції:

* зручне введення досліджуваного рівняння;
* вибір методу розв’язання;
* швидке обчислення за обраним методом;
* зміна початкових умов для рівняння;
* перегляд результату обчислень;
* графічне відображення результату у вигляді графіків;
* можливість експорту результатів обчислень в окремий файл.

Програмне забезпечення розроблене за допомогою візуального середовища розробки Visual Studio 2019 та платформи .NET Framework 4.7.

Програмний продукт працює на персональному комп’ютері під керуванням операційної системи Windows 10. Програма має зручний, максимально орієнтований на кінцевого користувача інтерфейс.

Програма була протестована та виявилася працездатною, має достатньо велику швидкість роботи.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. https://uk.wikipedia.org/wiki/Диференціальні\_рівняння
2. https://uk.wikipedia.org/wiki/Чисельні\_методи
3. https://uk.wikipedia.org/wiki/Метод\_Ейлера
4. https://uk.wikipedia.org/wiki/Метод\_Рунге\_—\_Кутти
5. https://uk.wikipedia.org/wiki/Метод\_Адамса
6. https://visualstudio.microsoft.com
7. https://dotnet.microsoft.com/en-us/
8. https://visualstudiomagazine.com/articles/2018/05/01/net-framework-update.aspx
9. https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp
10. Основи комп’ютерного програмування на C#, Світлін Наков, 2019
11. [Книга C# 4.0: повний посібник, Герберт Шилдт](https://www.amazon.com/4-0-Complete-Reference-Herbert-Schildt-ebook/dp/B003QP3XIM/ref=sr_1_1?crid=15D67V9VE0XUE&keywords=C%23+4.0+The+Complete+Reference+Herbert+Schildt&qid=1663588552&sprefix=c+4.0+the+complete+reference+herbert+schildt%2Caps%2C131&sr=8-1), 2019
12. https://en.wikipedia.org/wiki/C\_Sharp\_(programming\_language)
13. https://en.wikipedia.org/wiki/Windows\_10

**ДОДАТОК А**

**(обов’язковий)**

**Текст програми**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсова

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

using System.Reflection;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

[assembly: AssemblyTitle("Diff")]

[assembly: AssemblyDescription("Програма розроблена для спрощення рішення диференціальних рівнянь за допомогою чисельних методів")]

[assembly: AssemblyProduct("Програма для числового рішення диференціальних рівнянь")]

[assembly: ComVisible(false)]

[assembly: Guid("8af459c2-2f34-4b4f-aa96-37a7aae068f1")]

[assembly: AssemblyVersion("1.0.0.0")]

[assembly: AssemblyFileVersion("1.0.0.0")]

using ClosedXML.Excel;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using static System.Math;

namespace Курсова

{

public partial class Form1 : Form

{

int finished = 1;

bool stopFlag = false;

bool limitStopFlag = false;

bool ignoreLimitFlag = false;

ManualResetEvent \_event = new ManualResetEvent(false);

List<List<double[]>> points = new List<List<double[]>>();

MySettings settings = new MySettings();

bool limitReachedExeption = false;

bool initConditionsError = false;

private Conditions conditions;

public Form1()

{

InitializeComponent();

settings.init(this);

}

private void Form1\_Load(object sender, EventArgs e)

{

settings.apply();

//panel1.Location = new Point(611, 44);

comboBox1.SelectedIndex = 0;

comboBox2.SelectedIndex = 0;

comboBox3.SelectedIndex = 0;

searchToolStripComboBox.SelectedIndex = 0;

textBox2.GotFocus += textBoxAutoSizeOn; textBox2.LostFocus += textBoxAutoSizeOff; textBox2.TextChanged += textBoxAutoSizeOn;

textBox3.GotFocus += textBoxAutoSizeOn; textBox3.LostFocus += textBoxAutoSizeOff; textBox3.TextChanged += textBoxAutoSizeOn;

textBox4.GotFocus += textBoxAutoSizeOn; textBox4.LostFocus += textBoxAutoSizeOff; textBox4.TextChanged += textBoxAutoSizeOn;

textBox5.GotFocus += textBoxAutoSizeOn; textBox5.LostFocus += textBoxAutoSizeOff; textBox5.TextChanged += textBoxAutoSizeOn;

textBox6.GotFocus += textBoxAutoSizeOn; textBox6.LostFocus += textBoxAutoSizeOff; textBox6.TextChanged += textBoxAutoSizeOn;

textBox2.GotFocus += removePlaceHolder;

textBox3.GotFocus += removePlaceHolder;

textBox4.GotFocus += removePlaceHolder;

textBox5.GotFocus += removePlaceHolder;

textBox6.GotFocus += removePlaceHolder;

textBox2.LostFocus += addPlaceHolder;

textBox3.LostFocus += addPlaceHolder;

textBox4.LostFocus += addPlaceHolder;

textBox5.LostFocus += addPlaceHolder;

textBox6.LostFocus += addPlaceHolder;

chart1.ChartAreas[0].CursorX.Interval = 0;

chart1.ChartAreas[0].CursorY.Interval = 0;

chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;

chart1.ChartAreas[0].AxisY.ScaleView.Zoomable = true;

chart1.ChartAreas[0].AxisX.ScrollBar.IsPositionedInside = false;

chart1.ChartAreas[0].AxisY.ScrollBar.IsPositionedInside = false;

chart1.ChartAreas[0].CursorX.LineColor = Color.Black;

chart1.ChartAreas[0].CursorX.LineWidth = 1;

chart1.ChartAreas[0].CursorY.LineColor = Color.Black;

chart1.ChartAreas[0].CursorY.LineWidth = 1;

saveFileDialog1.RestoreDirectory = true;

listSearchToolStripTextBox.TextChanged += listSearch;

listSearchToolStripTextBox.LostFocus += (sndr, ev) => { listSearchToolStripTextBox.Text = ""; };

}

private void Form1\_Shown(object sender, EventArgs e)

{

DialogResult result = DialogResult.None;

if (!Properties.Settings.Default.hideHelpWindowOnStart)

result = MessageBox.Show("Перед початком роботи з програмою рекомендуємо прочитати довідку з використання. " +

"Перейти до довідки?", "Information", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (result == DialogResult.Yes)

{

FormHelp formHelp = new FormHelp();

formHelp.ShowDialog();

}

}

private class Conditions

{

public string equation;

public double xmin, xmax, y0, x0, h;

public int methodNumber;

public Conditions(string equation, double x0, double y0, double xmin, double xmax, double h, int methodNumber)

{

this.equation = equation;

this.xmin = xmin;

this.xmax = xmax;

this.y0 = y0;

this.x0 = x0;

this.h = h;

this.methodNumber = methodNumber;

}

}

public class MySettings

{

Form1 f;

public void init(Form1 form)

{

f = form;

}

public void apply()

{

f.allowCopyByClick();

f.saveFileDialog1.InitialDirectory = Path.GetFullPath(Properties.Settings.Default.exportPath);

if (Properties.Settings.Default.exportPath.Contains("./"))

Properties.Settings.Default.exportPath = Path.GetFullPath(Properties.Settings.Default.exportPath);

if (Properties.Settings.Default.showMarker)

foreach (var series in f.chart1.Series)

series.MarkerStyle = MarkerStyle.Circle;

else

foreach (var series in f.chart1.Series)

series.MarkerStyle = MarkerStyle.None;

f.chart1.ChartAreas[0].AxisY.LabelStyle.Format = "0.";

for (int i = 0; i < Properties.Settings.Default.decimalPlacesChartY; i++)

f.chart1.ChartAreas[0].AxisY.LabelStyle.Format = f.chart1.ChartAreas[0].AxisY.LabelStyle.Format + "0";

if (f.chart1.Series[0].Points.Count != 0)

f.chart1.Legends[0].Enabled = !Properties.Settings.Default.hideLegend;

foreach (Form f in Application.OpenForms)

{

if (f.Name == "FormChart")

{

FormChart fc = (FormChart)f;

if (Properties.Settings.Default.showMarker)

foreach (var series in fc.chartF1.Series)

series.MarkerStyle = MarkerStyle.Circle;

else

foreach (var series in fc.chartF1.Series)

series.MarkerStyle = MarkerStyle.None;

fc.chartF1.ChartAreas[0].AxisY.LabelStyle.Format = "0.";

for (int i = 0; i < Properties.Settings.Default.decimalPlacesChartY; i++)

fc.chartF1.ChartAreas[0].AxisY.LabelStyle.Format = fc.chartF1.ChartAreas[0].AxisY.LabelStyle.Format + "0";

if (fc.chartF1.Series[0].Points.Count != 0)

fc.chartF1.Legends[0].Enabled = !Properties.Settings.Default.hideLegend;

}

}

f.showMethods();

f.initConditions();

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "")

{

textBox1.Focus();

MessageBox.Show("Введіть рівняння", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (textBox6.Text.Contains('-'))

{

textBox6.Focus();

MessageBox.Show("Крок не може бути від'ємним", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

double xmin;

double xmax;

double y0;

double x0;

double h;

button3.Enabled = true;

button3.Focus();

panelConditions.Enabled = false;

showManyMethodsToolStripMenuItem.Enabled = false;

settingsToolStripMenuItem.Enabled = false;

listContextMenuStrip.Enabled = false;

finished = 2 - Properties.Settings.Default.countOfMethods;

\_event.Reset();

limitStopFlag = false;

ignoreLimitFlag = false;

try

{

if (Properties.Settings.Default.useDefaultConditions)

{

if (Properties.Settings.Default.defaultX0 != "" && String.IsNullOrEmpty(textBox2.Text))

x0 = double.Parse(Properties.Settings.Default.defaultX0.Replace('.', ','));

else

x0 = double.Parse(textBox2.Text.Replace('.', ','));

if (Properties.Settings.Default.defaultY0 != "" && String.IsNullOrEmpty(textBox3.Text))

y0 = double.Parse(Properties.Settings.Default.defaultY0.Replace('.', ','));

else

y0 = double.Parse(textBox3.Text.Replace('.', ','));

if (Properties.Settings.Default.defaultXmin != "" && String.IsNullOrEmpty(textBox4.Text))

xmin = double.Parse(Properties.Settings.Default.defaultXmin.Replace('.', ','));

else

xmin = double.Parse(textBox4.Text.Replace('.', ','));

if (Properties.Settings.Default.defaultXmax != "" && String.IsNullOrEmpty(textBox5.Text))

xmax = double.Parse(Properties.Settings.Default.defaultXmax.Replace('.', ','));

else

xmax = double.Parse(textBox5.Text.Replace('.', ','));

if (Properties.Settings.Default.defaultH != "" && String.IsNullOrEmpty(textBox6.Text))

h = double.Parse(Properties.Settings.Default.defaultH.Replace('.', ','));

else

h = double.Parse(textBox6.Text.Replace('.', ','));

}

else

{

x0 = double.Parse(textBox2.Text.Replace('.', ','));

y0 = double.Parse(textBox3.Text.Replace('.', ','));

xmin = double.Parse(textBox4.Text.Replace('.', ','));

xmax = double.Parse(textBox5.Text.Replace('.', ','));

h = double.Parse(textBox6.Text.Replace('.', ','));

}

}

catch

{

MessageBox.Show("Помилка в початкових умовах", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

button3.Enabled = false;

panelConditions.Enabled = true;

showManyMethodsToolStripMenuItem.Enabled = true;

settingsToolStripMenuItem.Enabled = true;

listContextMenuStrip.Enabled = true;

return;

}

string equation = textBox1.Text;//x+cos(y/sqrt(0.7))

chart1.Series[0].Points.Clear();

chart1.Series[1].Points.Clear();

chart1.Series[2].Points.Clear();

chart1.Legends[0].Enabled = !Properties.Settings.Default.hideLegend;

chart1.Series[0].IsVisibleInLegend = true;

chart1.ChartAreas[0].CursorX.IsUserEnabled = true;

chart1.ChartAreas[0].CursorX.IsUserSelectionEnabled = true;

chart1.ChartAreas[0].CursorY.IsUserEnabled = true;

chart1.ChartAreas[0].CursorY.IsUserSelectionEnabled = true;

int dpx;

if (GetDecimalDigitsCount(h) < GetDecimalDigitsCount(x0))

dpx = GetDecimalDigitsCount(x0);

else

dpx = GetDecimalDigitsCount(h);

chart1.ChartAreas[0].AxisX.LabelStyle.Format = "0.";

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

chart1.ChartAreas[0].AxisX.LabelStyle.Format = chart1.ChartAreas[0].AxisX.LabelStyle.Format + "0";

for (int i = 0; i < 3; i++)

{

if (i < Properties.Settings.Default.countOfMethods) chart1.Series[i].IsVisibleInLegend = true;

else chart1.Series[i].IsVisibleInLegend = false;

}

foreach (Form f in Application.OpenForms)

if (f.Name == "FormChart")

{

FormChart fc = (FormChart)f;

fc.chartF1.Series[0].Points.Clear();

fc.chartF1.Series[1].Points.Clear();

fc.chartF1.Series[2].Points.Clear();

fc.chartF1.Legends[0].Enabled = true;

fc.chartF1.Series[0].IsVisibleInLegend = true;

for (int i = 0; i < 3; i++)

{

if (i < Properties.Settings.Default.countOfMethods) fc.chartF1.Series[i].IsVisibleInLegend = true;

else fc.chartF1.Series[i].IsVisibleInLegend = false;

}

}

listBox1.Items.Clear();

listBox2.Items.Clear();

listBox3.Items.Clear();

listBox1.Items.Add("x\t y");

listBox2.Items.Add("x\t y");

listBox3.Items.Add("x\t y");

points.Clear();

for (int i = 0; i < Properties.Settings.Default.countOfMethods; i++)

points.Add(new List<double[]>());

for (int i = 0; i < Properties.Settings.Default.countOfMethods; i++)

{

ComboBox comboBox = comboBox1;

switch (i)

{

case 0: comboBox = comboBox1; break;

case 1: comboBox = comboBox2; break;

case 2: comboBox = comboBox3; break;

}

string method = "";

ParameterizedThreadStart pThread = new ParameterizedThreadStart(euler);

switch (comboBox.SelectedIndex)

{

case 0:

method = "Ейлера";

pThread = new ParameterizedThreadStart(euler);

break;

case 1:

method = "Рунге-Кута";

pThread = new ParameterizedThreadStart(rk);

break;

case 2:

method = "Адамса-Бешфорта";

pThread = new ParameterizedThreadStart(adams);

break;

}

chart1.Series[i].LegendText = method;

foreach (Form f in Application.OpenForms)

if (f.Name == "FormChart")

{

FormChart fc = (FormChart)f;

fc.chartF1.Series[i].LegendText = method;

}

conditions = new Conditions(equation, x0, y0, xmin, xmax, h, i + 1);

Thread thread = new Thread(pThread);

thread.IsBackground = true;

thread.Start(conditions);

}

}

string calculate(string equation)//When expression contains "()", "sin()", "sqrt()" etc.

{

equation = equation.Replace('.', ',')

.Replace(" ", "")

.Replace("abs", "a")

.Replace("sin", "s")

.Replace("cos", "c")

.Replace("tg", "t")

.Replace("ctg", "1/t")

.Replace("sqrt", "q")

.Replace("cbrt", "b")

.Replace("rt", "r")

.Replace("log", "l")

.Replace("ln", "n")

.Replace("e", E.ToString())

.Replace("pi", PI.ToString());

for (int i = 0; i < equation.Length; i++)

{

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

{

string toBeReplace = "";

int bracketsCount = 0;

for (int j = i + 1; j < equation.Length; j++)

{

if (equation[j] == '(')

{

bracketsCount++;

toBeReplace += equation[j];

}

else if (equation[j] == ')' && bracketsCount > 0)

{

toBeReplace += equation[j];

bracketsCount--;

}

else if (equation[j] != ')')

{

toBeReplace += equation[j];

}

else break;

}

if (i == 0)

{

equation = equation.Replace('(' + toBeReplace + ')', calculate(toBeReplace));

continue;

}

string newBase = "";

for (int j = i - 1; j >= 0; j--)//l10( //r10(

{

if (equation[j] == 'l')

{

newBase = new string(newBase.Reverse().ToArray());

equation = equation.Replace("l" + newBase + "(" + toBeReplace + ")", Log(double.Parse(calculate(toBeReplace)), double.Parse(newBase)).ToString());

break;

}

else if (equation[j] == 'r')

{

newBase = new string(newBase.Reverse().ToArray());

equation = equation.Replace("r" + newBase + "(" + toBeReplace + ")", Pow(double.Parse(calculate(toBeReplace)), 1.0 / double.Parse(newBase)).ToString());

break;

}

else if (char.IsDigit(equation[j]))

{

newBase += equation[j];

}

else

{

switch (equation[i - 1])

{

case 's':

equation = equation.Replace("s(" + toBeReplace + ")", Sin(double.Parse(calculate(toBeReplace))).ToString());

break;

case 'c':

equation = equation.Replace("c(" + toBeReplace + ")", Cos(double.Parse(calculate(toBeReplace))).ToString());

break;

case 't':

equation = equation.Replace("t(" + toBeReplace + ")", Tan(double.Parse(calculate(toBeReplace))).ToString());

break;

case 'q':

equation = equation.Replace("q(" + toBeReplace + ")", Sqrt(double.Parse(calculate(toBeReplace))).ToString());

break;

case 'b':

equation = equation.Replace("b(" + toBeReplace + ")", Pow(double.Parse(calculate(toBeReplace)), 1.0 / 3.0).ToString());

break;

case 'n':

equation = equation.Replace("n(" + toBeReplace + ")", Log(double.Parse(calculate(toBeReplace))).ToString());

break;

case 'a':

equation = equation.Replace("a(" + toBeReplace + ")", Abs(double.Parse(calculate(toBeReplace))).ToString());

break;

default:

equation = equation.Replace('(' + toBeReplace + ')', calculate(toBeReplace));

break;

}

break;

}

}

}

}

for (int i = 0; i < equation.Length; i++)

{

if (equation[i] == '^')

{

equation = calculateSimple(equation, i);

i = 0;

}

}

for (int i = 0; i < equation.Length; i++)

{

if (equation[i] == '\*' || equation[i] == '/')

{

equation = calculateSimple(equation, i);

i = 0;

}

}

for (int i = 0; i < equation.Length; i++)

{

if (!char.IsDigit(equation[i]) && equation[i] != ',' && i != 0 && equation[i - 1].ToString().ToLower() != "e" && equation[i].ToString().ToLower() != "e" && equation[i + 1].ToString() != Convert.ToString(Double.NegativeInfinity))

{

equation = calculateSimple(equation, i);

i = 0;

}

}

return equation;

}

string calculateSimple(string expression, int operatorPos)//When expression doesn`t contain "()", "sin()", "sqrt()" etc.

{

double numLeft;

double numRight;

string numLeftStr = "";

string numRightStr = "";

bool numRightIsNegative = false;

if (expression.Contains(Convert.ToString(Double.NegativeInfinity)))

return Convert.ToString(Double.NegativeInfinity);

else if (expression.Contains(Convert.ToString(Double.PositiveInfinity)))

return Convert.ToString(Double.PositiveInfinity);

for (int i = operatorPos - 1; i >= 0; i--)//Finds left number around the symbol

{

if (char.IsDigit(expression[i]) || expression[i] == ',')

{

numLeftStr += expression[i];

}

else if ((expression[i] == '-' || expression[i] == '+') && i != 0 && expression[i - 1].ToString().ToLower() == "e")

{

numLeftStr += expression[i] + "E";

i--;

continue;

}

else if (expression[i] == '-')

{

numLeftStr += "-";

break;

}

else break;

}

numLeftStr = new string(numLeftStr.Reverse().ToArray());

if (expression[operatorPos + 1] == '-')

{

numRightStr = "-";

numRightIsNegative = true;

}

for (int i = operatorPos + 1 + Convert.ToInt32(numRightIsNegative); i < expression.Length; i++)//Finds right number around the symbol

{

if (char.IsDigit(expression[i]) || expression[i] == ',')

{

numRightStr += expression[i];

}

else if (i != expression.Length - 1 && (expression[i + 1] == '-' || expression[i + 1] == '+') && (expression[i] == 'E' || expression[i] == 'e'))

{

numRightStr += "E" + expression[i + 1];

i++;

continue;

}

else break;

}

double res = 0;

numRight = double.Parse(numRightStr.ToUpper());

if (numLeftStr.Length != 0)

{

numLeft = double.Parse(numLeftStr.ToUpper());

switch (expression[operatorPos])

{

case '+':

res = numLeft + numRight;

break;

case '-':

res = numLeft - numRight;

break;

case '\*':

res = numLeft \* numRight;

break;

case '/':

res = numLeft / numRight;

break;

case '^':

res = Pow(numLeft, numRight);

break;

}

}

else

{

res = -numRight;

}

expression = expression.ToUpper().Replace(numLeftStr.ToUpper() + expression[operatorPos] + numRightStr.ToUpper(), res.ToString());

return expression;

}

double der(string f, double x, double y)

{

return double.Parse(calculate(f.ToLower().Replace("x", Convert.ToString(x)).Replace("y", Convert.ToString(y))));

}

void euler(string f, double x0, double y0, double xmin, double xmax, double h, int methodNumber)

{

List<double[]> points = new List<double[]>();

ListBox listBox = listBox1;

switch (methodNumber)

{

case 1: listBox = listBox1; break;

case 2: listBox = listBox2; break;

case 3: listBox = listBox3; break;

}

int dpx;

if (GetDecimalDigitsCount(h) < GetDecimalDigitsCount(x0))

dpx = GetDecimalDigitsCount(x0);

else

dpx = GetDecimalDigitsCount(h);

int dpy = Properties.Settings.Default.decimalPlacesListY;

int xminDC = GetDecimalDigitsCount(xmin);

if (xminDC < GetDecimalDigitsCount(h))

xminDC = GetDecimalDigitsCount(h);

if (xminDC < GetDecimalDigitsCount(x0))

xminDC = GetDecimalDigitsCount(x0);

if (Round(x0, xminDC) >= Round(xmin, xminDC))

{

Action action0 = () => listBox.Items.Add(Convert.ToString(Round(x0, dpx).ToString("G6")) + "\t " + Round(y0, dpy).ToString("G6"));

listBox.Invoke(action0);

points.Add(new double[] { x0, y0 });

}

if ((Abs(x0) > 1.0E-28 || x0 == 0) && (Abs(y0) > 1.0E-28 || y0 == 0) && Abs(x0) < 1.0E+28 && Abs(y0) < 1.0E+28)

{

Action actionP0 = () => addPointToChart(x0, y0, methodNumber);

Invoke(actionP0);

}

else

if (!limitReachedExeption)

{

limitReachedExeption = true;

MessageBox.Show("Деякі значення точок занадто великі або малі. Ці значення не будуть відображені на графіку",

"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

int xmaxDC = GetDecimalDigitsCount(xmax);

if (xmaxDC < GetDecimalDigitsCount(h))

xmaxDC = GetDecimalDigitsCount(h);

try

{

int i = 1; //iteration number

int j = 0;

double k = 0;

double pointsCount = (xmax - xmin) / h;

double newStep = pointsCount / (double)Properties.Settings.Default.maxPointsAtChart;

while (Round(x0, xminDC) + h <= Round(xmax, xmaxDC))

{

Thread.Sleep(new TimeSpan(10000));

double df = der(f, x0, y0);

double dfn = der(f, x0 + h, df \* h + y0);

double y = y0 + (df + dfn) \* h / 2; //x\*y+y^5-r3(x)

x0 += h;

x0 = Round(x0, 9);

y0 = y;

if (Round(x0, xminDC) >= Round(xmin, xminDC))

{

Action action1 = () =>

{

listBox.Items.Add(Convert.ToString(Round(x0, dpx).ToString("G6")) + "\t " + Round(y0, dpy).ToString("G6"));

listBox.TopIndex = i - 23;

};

listBox.Invoke(action1);

points.Add(new double[] { x0, y0 });

if ((Abs(x0) > 1.0E-28 || x0 == 0) && (Abs(y0) > 1.0E-28 || y0 == 0) && Abs(x0) < 1.0E+28 && Abs(y0) < 1.0E+28)

{

if (pointsCount > (double)Properties.Settings.Default.maxPointsAtChart)

{

j++;

if (j > k || j == pointsCount)

{

k += newStep;

Action actionP1 = () => addPointToChart(x0, y0, methodNumber);

Invoke(actionP1);

}

}

else

{

Action actionP1 = () => addPointToChart(x0, y0, methodNumber);

Invoke(actionP1);

}

}

else

{

if (!limitReachedExeption)

{

limitReachedExeption = true;

MessageBox.Show("Деякі значення точок занадто великі або малі. Ці значення не будуть відображені на графіку",

"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

if (Properties.Settings.Default.isIterationLimited && i == Properties.Settings.Default.iterationLimit)

{

DialogResult result = DialogResult.None;

if (methodNumber != 1)

\_event.WaitOne();

if (limitStopFlag)

break;

else if (!ignoreLimitFlag)

{

result = MessageBox.Show("Ліміт ітерацій досягнено. Продовжити розрахунок?", "Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

}

if (result == DialogResult.No)

{

limitStopFlag = true;

\_event.Set();

break;

}

else if (result == DialogResult.Yes)

{

ignoreLimitFlag = true;

\_event.Set();

}

}

if (Double.IsInfinity(y0))

break;

if (stopFlag) break;

i++;

}

}

catch

{

if (!initConditionsError)

{

initConditionsError = true;

MessageBox.Show("Помилка в початкових умовах", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

Action actionEnd = () => listBox.TopIndex = listBox.Items.Count - 1;

listBox.Invoke(actionEnd);

this.points[methodNumber - 1].AddRange(points);

if (++finished == 2)

{

Action actionFinish = () =>

{

panelConditions.Enabled = true;

button1.Focus();

button3.Enabled = false;

panelConditions.Enabled = true;

showManyMethodsToolStripMenuItem.Enabled = true;

settingsToolStripMenuItem.Enabled = true;

listContextMenuStrip.Enabled = true;

};

Invoke(actionFinish);

limitReachedExeption = false;

initConditionsError = false;

stopFlag = false;

}

}

void euler(Object obj)

{

Conditions cond = obj as Conditions;

euler(cond.equation, cond.x0, cond.y0, cond.xmin, cond.xmax, cond.h, cond.methodNumber);

}

void rk(string f, double x0, double y0, double xmin, double xmax, double h, int methodNumber)

{

List<double[]> points = new List<double[]>();

ListBox listBox = listBox1;

switch (methodNumber)

{

case 1: listBox = listBox1; break;

case 2: listBox = listBox2; break;

case 3: listBox = listBox3; break;

}

int dpx;

if (GetDecimalDigitsCount(h) < GetDecimalDigitsCount(x0))

dpx = GetDecimalDigitsCount(x0);

else

dpx = GetDecimalDigitsCount(h);

int dpy = Properties.Settings.Default.decimalPlacesListY;

int xminDC = GetDecimalDigitsCount(xmin);

if (xminDC < GetDecimalDigitsCount(h))

xminDC = GetDecimalDigitsCount(h);

if (xminDC < GetDecimalDigitsCount(x0))

xminDC = GetDecimalDigitsCount(x0);

if (Round(x0, xminDC) >= Round(xmin, xminDC))

{

Action action0 = () => listBox.Items.Add(Convert.ToString(Round(x0, dpx).ToString("G6")) + "\t " + Round(y0, dpy).ToString("G6"));

listBox.Invoke(action0);

points.Add(new double[] { x0, y0 });

if ((Abs(x0) > 1.0E-28 || x0 == 0) && (Abs(y0) > 1.0E-28 || y0 == 0) && Abs(x0) < 1.0E+28 && Abs(y0) < 1.0E+28)

{

Action actionP0 = () => addPointToChart(x0, y0, methodNumber);

Invoke(actionP0);

}

else

if (!limitReachedExeption)

{

limitReachedExeption = true;

MessageBox.Show("Деякі значення точок занадто великі або малі. Ці значення не будуть відображені на графіку",

"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

int xmaxDC = GetDecimalDigitsCount(xmax);

if (xmaxDC < GetDecimalDigitsCount(h))

xmaxDC = GetDecimalDigitsCount(h);

try

{

int i = 1; //iteration number

int j = 0;

double k = 0;

double pointsCount = (xmax - xmin) / h;

double newStep = pointsCount / (double)Properties.Settings.Default.maxPointsAtChart;

while (Round(x0, xminDC) + h <= Round(xmax, xmaxDC))

{

Thread.Sleep(new TimeSpan(10000));

double df = der(f, x0, y0);

double dk2 = der(f, x0 + h / 2, y0 + h / 2 \* df);

double dk3 = der(f, x0 + h / 2, y0 + h / 2 \* dk2);

double dk4 = der(f, x0 + h, y0 + h \* dk3);

double y = y0 + h / 6 \* (df + 2 \* dk2 + 2 \* dk3 + dk4);

x0 += h;

y0 = y;

if (Round(x0, xminDC) >= Round(xmin, xminDC))

{

Action action1 = () =>

{

listBox.Items.Add(Convert.ToString(Round(x0, dpx).ToString("G6")) + "\t " + Round(y0, dpy).ToString("G6"));

listBox.TopIndex = i - 23;

};

listBox.Invoke(action1);

points.Add(new double[] { x0, y0 });

if ((Abs(x0) > 1.0E-28 || x0 == 0) && (Abs(y0) > 1.0E-28 || y0 == 0) && Abs(x0) < 1.0E+28 && Abs(y0) < 1.0E+28)

{

if (pointsCount > (double)Properties.Settings.Default.maxPointsAtChart)

{

j++;

if (j > k || j == pointsCount)

{

k += newStep;

Action actionP1 = () => addPointToChart(x0, y0, methodNumber);

Invoke(actionP1);

}

}

else

{

Action actionP1 = () => addPointToChart(x0, y0, methodNumber);

Invoke(actionP1);

}

}

else

{

if (!limitReachedExeption)

{

limitReachedExeption = true;

MessageBox.Show("Деякі значення точок занадто великі або малі. Ці значення не будуть відображені на графіку",

"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

if (Properties.Settings.Default.isIterationLimited && i == Properties.Settings.Default.iterationLimit)

{

DialogResult result = DialogResult.None;

if (methodNumber != 1)

\_event.WaitOne();

if (limitStopFlag)

break;

else if (!ignoreLimitFlag)

{

result = MessageBox.Show("Ліміт ітерацій досягнено. Продовжити розрахунок?", "Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

}

if (result == DialogResult.No)

{

limitStopFlag = true;

\_event.Set();

break;

}

else if (result == DialogResult.Yes)

{

ignoreLimitFlag = true;

\_event.Set();

}

}

if (Double.IsInfinity(y0))

break;

if (stopFlag) break;

i++;

}

}

catch

{

if (!initConditionsError)

{

initConditionsError = true;

MessageBox.Show("Помилка в початкових умовах", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

Action actionEnd = () => listBox.TopIndex = listBox.Items.Count - 1;

listBox.Invoke(actionEnd);

this.points[methodNumber - 1].AddRange(points);

if (++finished == 2)

{

Action actionFinish = () =>

{

panelConditions.Enabled = true;

button1.Focus();

button3.Enabled = false;

panelConditions.Enabled = true;

showManyMethodsToolStripMenuItem.Enabled = true;

settingsToolStripMenuItem.Enabled = true;

listContextMenuStrip.Enabled = true;

};

Invoke(actionFinish);

initConditionsError = false;

limitReachedExeption = false;

stopFlag = false;

}

}

(double, bool) rk(string f, double x0, double y0, double xmin, double h, int methodNumber)

{

double y = 0;

bool error = false;

try

{

Thread.Sleep(new TimeSpan(10000));

int i = 1; //iteration number

double df = der(f, x0, y0);

double dk2 = der(f, x0 + h / 2, y0 + h / 2 \* df);

double dk3 = der(f, x0 + h / 2, y0 + h / 2 \* dk2);

double dk4 = der(f, x0 + h, y0 + h \* dk3);

y = y0 + h / 6 \* (df + 2 \* dk2 + 2 \* dk3 + dk4);

x0 += h;

y0 = y;

ListBox listBox = listBox1;

switch (methodNumber)

{

case 1: listBox = listBox1; break;

case 2: listBox = listBox2; break;

case 3: listBox = listBox3; break;

}

int dpx;

if (GetDecimalDigitsCount(h) < GetDecimalDigitsCount(x0))

dpx = GetDecimalDigitsCount(x0);

else

dpx = GetDecimalDigitsCount(h);

int dpy = Properties.Settings.Default.decimalPlacesListY;

int xminDC = GetDecimalDigitsCount(xmin);

if (xminDC < GetDecimalDigitsCount(h))

xminDC = GetDecimalDigitsCount(h);

if (xminDC < GetDecimalDigitsCount(x0))

xminDC = GetDecimalDigitsCount(x0);

if (Round(x0, xminDC) >= Round(xmin, xminDC))

{

Action action1 = () => listBox.Items.Add(Convert.ToString(Round(x0, dpx).ToString("G6")) + "\t " + Round(y0, dpy).ToString("G6"));

listBox.Invoke(action1);

if ((Abs(x0) > 1.0E-28 || x0 == 0) && (Abs(y0) > 1.0E-28 || y0 == 0) && Abs(x0) < 1.0E+28 && Abs(y0) < 1.0E+28)

{

Action actionP0 = () => addPointToChart(x0, y0, methodNumber);

Invoke(actionP0);

}

else

if (!limitReachedExeption)

{

limitReachedExeption = true;

MessageBox.Show("Деякі значення точок занадто великі або малі. Ці значення не будуть відображені на графіку",

"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

if (Properties.Settings.Default.isIterationLimited && i == Properties.Settings.Default.iterationLimit)

{

DialogResult result = DialogResult.None;

if (methodNumber != 1)

\_event.WaitOne();

if (limitStopFlag)

stopFlag = true;

else if (!ignoreLimitFlag)

{

result = MessageBox.Show("Ліміт ітерацій досягнено. Продовжити розрахунок?", "Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

}

if (result == DialogResult.No)

{

limitStopFlag = true;

\_event.Set();

stopFlag = true;

}

else if (result == DialogResult.Yes)

{

ignoreLimitFlag = true;

\_event.Set();

}

}

if (Double.IsInfinity(y0))

error = true;

}

catch

{

error = true;

if (!initConditionsError)

{

initConditionsError = true;

MessageBox.Show("Помилка в початкових умовах", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

return (y, error);

}

void rk(Object obj)

{

Conditions cond = obj as Conditions;

rk(cond.equation, cond.x0, cond.y0, cond.xmin, cond.xmax, cond.h, cond.methodNumber);

}

void adams(string f, double x0, double y0, double xmin, double xmax, double h, int methodNumber)

{

List<double[]> points = new List<double[]>();

ListBox listBox = listBox1;

switch (methodNumber)

{

case 1: listBox = listBox1; break;

case 2: listBox = listBox2; break;

case 3: listBox = listBox3; break;

}

int dpx;

if (GetDecimalDigitsCount(h) < GetDecimalDigitsCount(x0))

dpx = GetDecimalDigitsCount(x0);

else

dpx = GetDecimalDigitsCount(h);

int dpy = Properties.Settings.Default.decimalPlacesListY;

int xminDC = GetDecimalDigitsCount(xmin);

if (xminDC < GetDecimalDigitsCount(h))

xminDC = GetDecimalDigitsCount(h);

if (xminDC < GetDecimalDigitsCount(x0))

xminDC = GetDecimalDigitsCount(x0);

if (Round(x0, xminDC) >= Round(xmin, xminDC))

{

Action action0 = () => listBox.Items.Add(Convert.ToString(Round(x0, dpx).ToString("G6")) + "\t " + Round(y0, dpy).ToString("G6"));

listBox.Invoke(action0);

points.Add(new double[] { x0, y0 });

if ((Abs(x0) > 1.0E-28 || x0 == 0) && (Abs(y0) > 1.0E-28 || y0 == 0) && Abs(x0) < 1.0E+28 && Abs(y0) < 1.0E+28)

{

Action actionP0 = () => addPointToChart(x0, y0, methodNumber);

Invoke(actionP0);

}

else

if (!limitReachedExeption)

{

limitReachedExeption = true;

MessageBox.Show("Деякі значення точок занадто великі або малі. Ці значення не будуть відображені на графіку",

"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

int xmaxDC = GetDecimalDigitsCount(xmax);

if (xmaxDC < GetDecimalDigitsCount(h))

xmaxDC = GetDecimalDigitsCount(h);

try

{

int i = 2; //iteration number

int j = 0;

double k = 0;

double pointsCount = (xmax - xmin) / h;

double newStep = pointsCount / (double)Properties.Settings.Default.maxPointsAtChart;

(double y1, bool error) = rk(f, x0, y0, xmin, h, methodNumber);

if (Round(x0, xminDC) >= Round(xmin, xminDC))

points.Add(new double[] { x0 + h, y1 });

//x0 += h;

double x1 = x0 + h;

if (!error && !stopFlag)

while (Round(x1, xminDC) + h <= Round(xmax, xmaxDC))

{

Thread.Sleep(new TimeSpan(10000));

double dp = der(f, x0, y0);

double dc = der(f, x1, y1);

double y = y1 + 3.0 / 2.0 \* h \* dc - 1.0 / 2.0 \* h \* dp;

x0 += h;

x1 += h;//?

y0 = y1;

y1 = y;

if (Round(x1, xminDC) >= Round(xmin, xminDC))

{

Action action1 = () =>

{

listBox.Items.Add(Convert.ToString(Round(x1, dpx).ToString("G6")) + "\t " + Round(y1, dpy).ToString("G6"));

listBox.TopIndex = i - 23;

};

listBox.Invoke(action1);

points.Add(new double[] { x1, y1 });

if ((Abs(x1) > 1.0E-28 || x1 == 0) && (Abs(y1) > 1.0E-28 || y1 == 0) && Abs(x1) < 1.0E+28 && Abs(y1) < 1.0E+28)

{

if (pointsCount > (double)Properties.Settings.Default.maxPointsAtChart)

{

j++;

if (j > k || j == pointsCount)

{

k += newStep;

Action actionP1 = () => addPointToChart(Round(x1, dpx), y1, methodNumber);

Invoke(actionP1);

}

}

else

{

Action actionP1 = () => addPointToChart(Round(x1, dpx), y1, methodNumber);

Invoke(actionP1);

}

}

else

{

if (!limitReachedExeption)

{

limitReachedExeption = true;

MessageBox.Show("Деякі значення точок занадто великі або малі. Ці значення не будуть відображені на графіку",

"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

if (Properties.Settings.Default.isIterationLimited && i == Properties.Settings.Default.iterationLimit)

{

DialogResult result = DialogResult.None;

if (methodNumber != 1)

\_event.WaitOne();

if (limitStopFlag)

break;

else if (!ignoreLimitFlag)

{

result = MessageBox.Show("Ліміт ітерацій досягнено. Продовжити розрахунок?", "Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

}

if (result == DialogResult.No)

{

limitStopFlag = true;

\_event.Set();

break;

}

else if (result == DialogResult.Yes)

{

ignoreLimitFlag = true;

\_event.Set();

}

}

if (Double.IsInfinity(y0))

break;

if (stopFlag) break;

i++;

}

}

catch

{

if (!initConditionsError)

{

initConditionsError = true;

MessageBox.Show("Помилка в початкових умовах", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

Action actionEnd = () => listBox.TopIndex = listBox.Items.Count - 1;

listBox.Invoke(actionEnd);

this.points[methodNumber - 1].AddRange(points);

if (++finished == 2)

{

Action actionFinish = () =>

{

panelConditions.Enabled = true;

button1.Focus();

button3.Enabled = false;

panelConditions.Enabled = true;

showManyMethodsToolStripMenuItem.Enabled = true;

settingsToolStripMenuItem.Enabled = true;

listContextMenuStrip.Enabled = true;

};

Invoke(actionFinish);

initConditionsError = false;

limitReachedExeption = false;

stopFlag = false;

}

}

void adams(Object obj)

{

Conditions cond = obj as Conditions;

adams(cond.equation, cond.x0, cond.y0, cond.xmin, cond.xmax, cond.h, cond.methodNumber);

}

static int GetDecimalDigitsCount(double number)

{

bool containsE = false;

string numberStr = Convert.ToString(number);

string countStr = "";

for (int i = 0; i < numberStr.Length; i++)

{

if (containsE)

{

countStr += numberStr[i];

if (i == numberStr.Length - 1)

return Convert.ToInt32(countStr);

}

if (numberStr[i] == 'E')

{

if (numberStr[++i] == '+') return 0;

containsE = true;

}

}

string str = number.ToString(new System.Globalization.NumberFormatInfo() { NumberDecimalSeparator = "." });

return str.Contains(".") ? str.Remove(0, Math.Truncate(number).ToString().Length + 1).Length : 0;

}

void addPointToChart(double x, double y, int methodNumber)

{

foreach (Form f in Application.OpenForms)

if (f.Name == "FormChart")

{

FormChart fc = (FormChart)f;

fc.chartF1.Series[methodNumber - 1].Points.AddXY(x, y);

}

this.chart1.Series[methodNumber - 1].Points.AddXY(x, y);

}

private void showManyMethodsToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormModalMethods modal = new FormModalMethods();

modal.ShowDialog();

showMethods();

}

private void showMethods()

{

switch (Properties.Settings.Default.countOfMethods)

{

case 3:

panel3.Visible = true;

panel2.Visible = true;

panel1.Visible = true;

this.Width = 666 + panel3.Width;

panel1.Width = 159;

adjustSizes();

break;

case 2:

panel3.Visible = false;

panel2.Visible = true;

panel1.Visible = true;

this.Width = 666;

panel1.Width = 159;

adjustSizes();

break;

case 1:

panel3.Visible = false;

panel2.Visible = false;

panel1.Visible = true;

this.Width = 666;

panel1.Width = 159 \* 2;

adjustSizes();

break;

}

}

private void Form1\_SizeChanged(object sender, EventArgs e)

{

adjustSizes();

}

private void adjustSizes()

{

int additionalWidth = 0;

if (panel2.Visible) additionalWidth += panel2.Width;

if (panel3.Visible) additionalWidth += panel3.Width;

panelChart.Width = this.Width - panel1.Width - additionalWidth - 12 - 20;

panelChart.Height = this.Height - panelConditions.Height - menuStrip1.Height - 12 - 40;

}

private void listBoxSelectedItem(object sender, EventArgs e)

{

if (Properties.Settings.Default.multiSelect)

{

try

{

ListBox listBox = sender as ListBox;

switch (listBox.Name)

{

case "listBox1":

listBox2.SelectedIndex = listBox1.SelectedIndex;

listBox3.SelectedIndex = listBox1.SelectedIndex;

break;

case "listBox2":

listBox1.SelectedIndex = listBox2.SelectedIndex;

listBox3.SelectedIndex = listBox2.SelectedIndex;

break;

case "listBox3":

listBox1.SelectedIndex = listBox3.SelectedIndex;

listBox2.SelectedIndex = listBox3.SelectedIndex;

break;

}

}

catch { }

}

}

private void showChartToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (showChartToolStripMenuItem.Checked)

{

floatableChartToolStripMenuItem.Enabled = false;

showChartToolStripMenuItem.Checked = false;

panelChart.Visible = false;

Form fc = null;

foreach (Form f in Application.OpenForms)

if (f.Name == "FormChart")

{

fc = f;

}

if (fc != null) fc.Close();

}

else

{

showChartToolStripMenuItem.Checked = true;

//chart1.Series[0].Points.Clear();

//chart1.Series[1].Points.Clear();

panelChart.Visible = true;

floatableChartToolStripMenuItem.Enabled = true;

}

}

private void floatableChartToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (floatableChartToolStripMenuItem.Checked)

{

Form fc = null;

foreach (Form f in Application.OpenForms)

if (f.Name == "FormChart")

{

fc = f;

}

if (fc != null) fc.Close(); //floatableChartToolStripMenuItem.Checked and hideFloatableChartToolStripMenuItem.Checked = false

panelChart.Visible = true;

}

else

{

floatableChartToolStripMenuItem.Checked = true;

hideFloatableChartToolStripMenuItem.Checked = true;

FormChart formChart = new FormChart(chart1.Series);

formChart.Show();

panelChart.Visible = false;

}

}

private void hideChartToolStripMenuItem\_Click(object sender, EventArgs e)

{

showChartToolStripMenuItem\_Click(sender, e);

}

private void hideFloatableChartToolStripMenuItem\_Click(object sender, EventArgs e)

{

floatableChartToolStripMenuItem\_Click(sender, e);

}

private void removePlaceHolder(object sender, EventArgs e)

{

TextBox textBox = (TextBox)sender;

if (textBox.ForeColor != TextBox.DefaultForeColor)

{

if (textBox.Name == "textBox2")

textBox.Text = "";

else if (textBox.Name == "textBox3")

textBox.Text = "";

else if (textBox.Name == "textBox4")

textBox.Text = "";

else if (textBox.Name == "textBox5")

textBox.Text = "";

else if (textBox.Name == "textBox6")

textBox.Text = "";

}

textBox.ForeColor = TextBox.DefaultForeColor;

}

private void addPlaceHolder(object sender, EventArgs e)

{

TextBox textBox = (TextBox)sender;

if (string.IsNullOrWhiteSpace(textBox.Text) && Properties.Settings.Default.useDefaultConditions)

{

if (textBox.Name == "textBox2")

textBox.Text = Properties.Settings.Default.defaultX0;

else if (textBox.Name == "textBox3")

textBox.Text = Properties.Settings.Default.defaultY0;

else if (textBox.Name == "textBox4")

textBox.Text = Properties.Settings.Default.defaultXmin;

else if (textBox.Name == "textBox5")

textBox.Text = Properties.Settings.Default.defaultXmax;

else if (textBox.Name == "textBox6")

textBox.Text = Properties.Settings.Default.defaultH;

textBox.ForeColor = Color.Gray;

}

}

private void initConditions()

{

if (Properties.Settings.Default.useDefaultConditions)

{

if ((textBox2.Text == "" || textBox2.ForeColor == Color.Gray) && !textBox2.ContainsFocus)

{

textBox2.Text = Properties.Settings.Default.defaultX0;

textBox2.ForeColor = Color.Gray;

}

if ((textBox3.Text == "" || textBox3.ForeColor == Color.Gray) && !textBox3.ContainsFocus)

{

textBox3.Text = Properties.Settings.Default.defaultY0;

textBox3.ForeColor = Color.Gray;

}

if ((textBox4.Text == "" || textBox4.ForeColor == Color.Gray) && !textBox4.ContainsFocus)

{

textBox4.Text = Properties.Settings.Default.defaultXmin;

textBox4.ForeColor = Color.Gray;

}

if ((textBox5.Text == "" || textBox5.ForeColor == Color.Gray) && !textBox5.ContainsFocus)

{

textBox5.Text = Properties.Settings.Default.defaultXmax;

textBox5.ForeColor = Color.Gray;

}

if ((textBox6.Text == "" || textBox6.ForeColor == Color.Gray) && !textBox6.ContainsFocus)

{

textBox6.Text = Properties.Settings.Default.defaultH;

textBox6.ForeColor = Color.Gray;

}

textBox2.Focus();

}

else

{

if (textBox2.ForeColor != TextBox.DefaultForeColor) textBox2.Text = "";

if (textBox3.ForeColor != TextBox.DefaultForeColor) textBox3.Text = "";

if (textBox4.ForeColor != TextBox.DefaultForeColor) textBox4.Text = "";

if (textBox5.ForeColor != TextBox.DefaultForeColor) textBox5.Text = "";

if (textBox6.ForeColor != TextBox.DefaultForeColor) textBox6.Text = "";

textBox2.ForeColor = textBox3.ForeColor = textBox4.ForeColor = textBox5.ForeColor = textBox6.ForeColor = TextBox.DefaultForeColor;

}

}

private void textBoxAutoSizeOn(object sender, EventArgs e)

{

TextBox textBox = (TextBox)sender;

textBox.BringToFront();

int textWidth = TextRenderer.MeasureText(textBox.Text, textBox.Font).Width;

if (textWidth < 43)

textBox.Width = 43;

else

textBox.Width = textWidth + 4;

textBox.BringToFront();

System.Drawing.Drawing2D.GraphicsPath gr = new System.Drawing.Drawing2D.GraphicsPath();

foreach (Control c in panelConditions.Controls)

{

gr.AddRectangle(new Rectangle(c.Location, c.Size));

}

panelConditions.Region = new Region(gr);

}

private void textBoxAutoSizeOff(object sender, EventArgs e)

{

TextBox textBox = (TextBox)sender;

textBox.Width = 43;

textBox.BringToFront();

}

private void chart1\_MouseMove(object sender, MouseEventArgs e)

{

if (finished == 2)

{

chart1.ChartAreas[0].CursorX.LineDashStyle = ChartDashStyle.Dot;

chart1.ChartAreas[0].CursorY.LineDashStyle = ChartDashStyle.Dot;

Point mousePoint = new Point(e.X, e.Y);

chart1.ChartAreas[0].CursorX.SetCursorPixelPosition(mousePoint, true);

chart1.ChartAreas[0].CursorY.SetCursorPixelPosition(mousePoint, true);

}

}

private void chart1\_MouseLeave(object sender, EventArgs e)

{

chart1.ChartAreas[0].CursorX.LineDashStyle = ChartDashStyle.NotSet;

chart1.ChartAreas[0].CursorY.LineDashStyle = ChartDashStyle.NotSet;

}

private void button2\_Click(object sender, EventArgs e)

{

chart1.ChartAreas[0].AxisX.ScaleView.ZoomReset(0);

chart1.ChartAreas[0].AxisY.ScaleView.ZoomReset(0);

chart1.ChartAreas[0].CursorX.IsUserSelectionEnabled = true;

chart1.ChartAreas[0].CursorY.IsUserSelectionEnabled = true;

}

private void chart1\_AxisViewChanged(object sender, ViewEventArgs e)

{

if (chart1.ChartAreas[0].AxisX.ScaleView.Size <= 1E-05 || chart1.ChartAreas[0].AxisY.ScaleView.Size <= 1E-05)

{

chart1.ChartAreas[0].CursorX.IsUserSelectionEnabled = false;

chart1.ChartAreas[0].CursorY.IsUserSelectionEnabled = false;

}

}

private void button3\_Click(object sender, EventArgs e)

{

\_event.Set();

stopFlag = true;

}

private void exportChartToolStripMenuItem\_Click(object sender, EventArgs e)

{

saveFileDialog1.FileName = Properties.Settings.Default.chartFileName;

saveFileDialog1.Filter = "JPEG |\*.jpeg|PNG |\*.png|Точковий малюнок |\*.bmp|TIFF |\*.tiff|GIF |\*.gif";

if (finished == 2)

try

{

DialogResult result = saveFileDialog1.ShowDialog();

if (result == DialogResult.OK && floatableChartToolStripMenuItem.Checked)

{

foreach (Form f in Application.OpenForms)

{

if (f.Name == "FormChart")

{

FormChart fc = (FormChart)f;

fc.chartF1.SaveImage(saveFileDialog1.FileName, (ChartImageFormat)saveFileDialog1.FilterIndex);

}

}

}

else if (result == DialogResult.OK)

{

chart1.SaveImage(saveFileDialog1.FileName, (ChartImageFormat)saveFileDialog1.FilterIndex);

}

}

catch

{

MessageBox.Show("Помилка збереження", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else MessageBox.Show("Неможливо експортувати до розрахунку", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void exportListToolStripMenuItem\_Click(object sender, EventArgs e)

{

saveFileDialog1.FileName = Properties.Settings.Default.pointsFileName;

saveFileDialog1.Filter = "Книга Excel |\*.xlsx|Книга Excel з підтримкою макросів |\*.xlsm|Шаблон Excel |\*.xltx|Шаблон Excel з підтримкою макросів |\*.xltm|Текстовий файл |\*.txt";

if (finished == 2)

try

{

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

if (saveFileDialog1.FilterIndex == 5)

{

var sw = new StreamWriter(saveFileDialog1.FileName);

int j = 0; // Number of method

sw.WriteLine("Методом " + comboBox1.Text);

sw.WriteLine("x\ty");

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

{

if (i == points[j].Count)

{

i = 0;

if (++j == Properties.Settings.Default.countOfMethods) break;

switch (j)

{

case 1:

sw.WriteLine("Методом " + comboBox2.Text);

sw.WriteLine("x\ty");

break;

case 2:

sw.WriteLine("Методом " + comboBox3.Text);

sw.WriteLine("x\ty");

break;

}

}

sw.WriteLine(points[j][i][0] + "\t" + points[j][i][1]);

}

sw.Close();

}

else

{

var workbook = new XLWorkbook();

var sheet = workbook.Worksheets.Add("Результат рішення Диф.рівнянь");

sheet.Cell(1, 1).SetValue(comboBox1.Text);

sheet.Cell(2, 1).SetValue("x");

sheet.Cell(2, 2).SetValue("y");

int j = 0; // Number of method

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

{

if (i == points[j].Count)

{

i = 0;

if (++j == Properties.Settings.Default.countOfMethods) break;

switch (j)

{

case 1:

sheet.Cell(1, j \* 2 + 1).SetValue(comboBox2.Text);

sheet.Cell(2, j \* 2 + 1).SetValue("x");

sheet.Cell(2, j \* 2 + 2).SetValue("y");

break;

case 2:

sheet.Cell(1, j \* 2 + 1).SetValue(comboBox3.Text);

sheet.Cell(2, j \* 2 + 1).SetValue("x");

sheet.Cell(2, j \* 2 + 2).SetValue("y");

break;

}

}

sheet.Cell(i + 3, j \* 2 + 1).SetValue(points[j][i][0]);

sheet.Cell(i + 3, j \* 2 + 2).SetValue(points[j][i][1]);

}

workbook.SaveAs(saveFileDialog1.FileName);

}

}

}

catch

{

MessageBox.Show("Помилка збереження", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else MessageBox.Show("Неможливо експортувати до розрахунку", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void listBox\_Click(object sender, EventArgs e)

{

ListBox listBox = sender as ListBox;

try

{

Clipboard.SetText(listBox.Text, TextDataFormat.UnicodeText);

}

catch { }

}

private void copyPointToolStripContextMenuItem\_Click(object sender, EventArgs e)

{

try

{

ToolStripMenuItem menuItem = sender as ToolStripMenuItem;

ContextMenuStrip contextMenu = menuItem.Owner as ContextMenuStrip;

ListBox listBox = contextMenu.SourceControl.Controls[1] as ListBox;

Clipboard.SetText(listBox.Text, TextDataFormat.UnicodeText);

}

catch { }

}

private void copyAllPointsToolStripContextMenuItem\_Click(object sender, EventArgs e)

{

ToolStripMenuItem menuItem = sender as ToolStripMenuItem;

ContextMenuStrip contextMenu = menuItem.Owner as ContextMenuStrip;

ListBox listBox = contextMenu.SourceControl.Controls[1] as ListBox;

StringBuilder text = new StringBuilder();

try

{

foreach (var item in listBox.Items)

{

text.AppendLine(item.ToString());

}

Clipboard.SetText(text.ToString(), TextDataFormat.UnicodeText);

}

catch { }

}

private void allowCopyByClick()

{

if (Properties.Settings.Default.copyByClick)

{

listBox1.Click += listBox\_Click;

listBox2.Click += listBox\_Click;

listBox3.Click += listBox\_Click;

}

else

{

listBox1.Click -= listBox\_Click;

listBox2.Click -= listBox\_Click;

listBox3.Click -= listBox\_Click;

}

}

private void settingsToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormSettings f = new FormSettings();

if (f.ShowDialog() == DialogResult.OK)

settings.apply();

}

private void listSearch(object sender, EventArgs e)

{

try

{

ToolStripTextBox menuItem = sender as ToolStripTextBox;

if (menuItem.Text != "")

{

int searchInd = searchToolStripComboBox.SelectedIndex;

foreach (var listBox in new ListBox[] { listBox1, listBox2, listBox3 })

{

try

{

for (int i = 0; i < listBox.Items.Count; i++)

{

string itemText;

string searchText;

if (menuItem.Text.Replace('.', ',').Contains(','))

{

itemText = listBox.Items[i].ToString().Split('\t')[searchInd];

searchText = menuItem.Text.Replace('.', ',');

}

else

{

itemText = listBox.Items[i].ToString().Split('\t')[searchInd].Split(',')[0];

searchText = menuItem.Text;

}

if (itemText.Contains(searchText))

{

listBox.SelectedItem = listBox.Items[i];

break;

}

}

}

catch { }

}

}

}

catch { }

}

private void aboutProgramToolStripMenuItem\_Click(object sender, EventArgs e)

{

AboutBox about = new AboutBox();

about.ShowDialog();

}

private void helpF1ToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormHelp formHelp = new FormHelp();

formHelp.ShowDialog();

}

}

}

using System;

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;

using System.Windows.Forms.DataVisualization.Charting;

namespace Курсова

{

public partial class FormChart : Form

{

SeriesCollection ownerSeries;

public FormChart()

{

InitializeComponent();

}

public FormChart(SeriesCollection series)

{

InitializeComponent();

ownerSeries = series;

Form1 f1 = new Form1();

this.Size = new Size(f1.chart1.Width + 16, f1.chart1.Height + 39);

this.StartPosition = FormStartPosition.Manual;

foreach (Form f in Application.OpenForms)

if (f.Name == "Form1")

{

Form1 fm = (Form1)f;

chartF1.ContextMenuStrip = fm.chartContextMenuStrip1;

chartF1.Legends[0].Enabled = fm.chart1.Legends[0].Enabled;

for (int i = 0; i < fm.chart1.Series.Count; i++)

chartF1.Series[i].MarkerStyle = fm.chart1.Series[i].MarkerStyle;

chartF1.ChartAreas[0].AxisX.LabelStyle.Format = fm.chart1.ChartAreas[0].AxisX.LabelStyle.Format;

this.Owner = fm;

}

this.Location = new Point(this.Owner.Location.X + f1.chart1.Location.X, this.Owner.Location.Y + f1.chart1.Location.Y);

chartF1.ChartAreas[0].AxisY.LabelStyle.Format = "0.";

for (int i = 0; i < Properties.Settings.Default.decimalPlacesChartY; i++)

chartF1.ChartAreas[0].AxisY.LabelStyle.Format = chartF1.ChartAreas[0].AxisY.LabelStyle.Format + "0";

}

private void FormChart\_FormClosing(object sender, FormClosingEventArgs e)

{

foreach (Form f in Application.OpenForms)

if (f.Name == "Form1")

{

Form1 fm = (Form1)f;

if (fm.showChartToolStripMenuItem.Checked == true)

{

fm.panelChart.Visible = true;

}

fm.floatableChartToolStripMenuItem.Checked = false;

fm.hideFloatableChartToolStripMenuItem.Checked = false;

fm.chart1.Series[0].Points.Clear();

foreach (DataPoint dp in chartF1.Series[0].Points.ToArray<DataPoint>())

fm.chart1.Series[0].Points.Add(dp);

fm.chart1.Series[1].Points.Clear();

foreach (DataPoint dp in chartF1.Series[1].Points.ToArray<DataPoint>())

fm.chart1.Series[1].Points.Add(dp);

fm.chart1.Series[2].Points.Clear();

foreach (DataPoint dp in chartF1.Series[2].Points.ToArray<DataPoint>())

fm.chart1.Series[2].Points.Add(dp);

fm.chart1.ChartAreas[0].AxisX.ScaleView.Position = chartF1.ChartAreas[0].AxisX.ScaleView.Position;

fm.chart1.ChartAreas[0].AxisX.ScaleView.Size = chartF1.ChartAreas[0].AxisX.ScaleView.Size;

fm.chart1.ChartAreas[0].AxisY.ScaleView.Position = chartF1.ChartAreas[0].AxisY.ScaleView.Position;

fm.chart1.ChartAreas[0].AxisY.ScaleView.Size = chartF1.ChartAreas[0].AxisY.ScaleView.Size;

}

chartF1.Series.Clear();

}

private void FormChart\_Load(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Maximized;

chartF1.Series[0].Points.Clear();

foreach (DataPoint dp in ownerSeries[0].Points.ToArray<DataPoint>())

chartF1.Series[0].Points.Add(dp);

chartF1.Series[1].Points.Clear();

foreach (DataPoint dp in ownerSeries[1].Points.ToArray<DataPoint>())

chartF1.Series[1].Points.Add(dp);

chartF1.Series[2].Points.Clear();

foreach (DataPoint dp in ownerSeries[2].Points.ToArray<DataPoint>())

chartF1.Series[2].Points.Add(dp);

if (ownerSeries[1].IsVisibleInLegend) chartF1.Series[1].IsVisibleInLegend = true;

if (ownerSeries[2].IsVisibleInLegend) chartF1.Series[2].IsVisibleInLegend = true;

chartF1.Series[0].LegendText = ownerSeries[0].LegendText;

chartF1.Series[1].LegendText = ownerSeries[1].LegendText;

chartF1.Series[2].LegendText = ownerSeries[2].LegendText;

foreach (Form f in Application.OpenForms)

if (f.Name == "Form1")

{

Form1 fm = (Form1)f;

chartF1.ChartAreas[0].AxisX.ScaleView.Position = fm.chart1.ChartAreas[0].AxisX.ScaleView.Position;

chartF1.ChartAreas[0].AxisX.ScaleView.Size = fm.chart1.ChartAreas[0].AxisX.ScaleView.Size;

chartF1.ChartAreas[0].AxisY.ScaleView.Position = fm.chart1.ChartAreas[0].AxisY.ScaleView.Position;

chartF1.ChartAreas[0].AxisY.ScaleView.Size = fm.chart1.ChartAreas[0].AxisY.ScaleView.Size;

}

chartF1.ChartAreas[0].CursorX.IsUserEnabled = true;

chartF1.ChartAreas[0].CursorX.IsUserSelectionEnabled = true;

chartF1.ChartAreas[0].CursorY.IsUserEnabled = true;

chartF1.ChartAreas[0].CursorY.IsUserSelectionEnabled = true;

chartF1.ChartAreas[0].CursorX.Interval = 0;

chartF1.ChartAreas[0].CursorY.Interval = 0;

chartF1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;

chartF1.ChartAreas[0].AxisY.ScaleView.Zoomable = true;

chartF1.ChartAreas[0].AxisX.ScrollBar.IsPositionedInside = false;

chartF1.ChartAreas[0].AxisY.ScrollBar.IsPositionedInside = false;

chartF1.ChartAreas[0].CursorX.LineColor = Color.Black;

chartF1.ChartAreas[0].CursorX.LineWidth = 1;

chartF1.ChartAreas[0].CursorX.LineDashStyle = ChartDashStyle.Dot;

chartF1.ChartAreas[0].CursorY.LineColor = Color.Black;

chartF1.ChartAreas[0].CursorY.LineWidth = 1;

chartF1.ChartAreas[0].CursorY.LineDashStyle = ChartDashStyle.Dot;

}

private void chartF1\_MouseMove(object sender, MouseEventArgs e)

{

chartF1.ChartAreas[0].CursorX.LineDashStyle = ChartDashStyle.Dot;

chartF1.ChartAreas[0].CursorY.LineDashStyle = ChartDashStyle.Dot;

Point mousePoint = new Point(e.X, e.Y);

chartF1.ChartAreas[0].CursorX.SetCursorPixelPosition(mousePoint, true);

chartF1.ChartAreas[0].CursorY.SetCursorPixelPosition(mousePoint, true);

}

private void chartF1\_MouseLeave(object sender, EventArgs e)

{

chartF1.ChartAreas[0].CursorX.LineDashStyle = ChartDashStyle.NotSet;

chartF1.ChartAreas[0].CursorY.LineDashStyle = ChartDashStyle.NotSet;

}

private void button2\_Click(object sender, EventArgs e)

{

chartF1.ChartAreas[0].AxisX.ScaleView.ZoomReset(0);

chartF1.ChartAreas[0].AxisY.ScaleView.ZoomReset(0);

}

private void chartF1\_AxisViewChanged(object sender, ViewEventArgs e)

{

if (chartF1.ChartAreas[0].AxisX.ScaleView.Size <= 1E-05 || chartF1.ChartAreas[0].AxisY.ScaleView.Size <= 1E-05)

{

chartF1.ChartAreas[0].CursorX.IsUserSelectionEnabled = false;

chartF1.ChartAreas[0].CursorY.IsUserSelectionEnabled = false;

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсова

{

public partial class FormHelp : Form

{

int searchNumber = 1;

int scrollTop;

int scrollBottom;

public FormHelp()

{

InitializeComponent();

this.richTextBox1.HideSelection = false;

}

private void FormHelp\_Load(object sender, EventArgs e)

{

richTextBox1.SelectAll();

richTextBox1.SelectionIndent += 15;

richTextBox1.SelectionRightIndent += 15;

richTextBox1.SelectionFont = new Font("Segoe UI", 10, FontStyle.Regular);

for (int i = 0; i < richTextBox1.Lines.Length; i++)

{

if (richTextBox1.Lines[i] == richTextBox1.Lines[i].ToUpper())

{

int start = richTextBox1.Find(richTextBox1.Lines[i]);

if (start != -1)

{

richTextBox1.Select(start, richTextBox1.Lines[i].Length);

richTextBox1.SelectionFont = new Font("Segoe UI", 13, (FontStyle)1);

richTextBox1.SelectionColor = Color.DarkRed;

}

}

if (i == 0 || i == 49 || i == 69)

{

int start = richTextBox1.Find(richTextBox1.Lines[i]);

if (start != -1)

{

richTextBox1.Select(start, richTextBox1.Lines[i].Length);

richTextBox1.SelectionFont = new Font("Segoe UI", 16, (FontStyle)7);

}

}

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

try

{

label2.Text = 1.ToString();

int index = richTextBox1.Text.ToLower().IndexOf(textBox1.Text.ToLower());

if (index >= 0)

{

richTextBox1.Select(richTextBox1.Text.ToLower().IndexOf(textBox1.Text.ToLower()), textBox1.Text.Length);

if (scrollTop > richTextBox1.SelectionStart || richTextBox1.SelectionStart > scrollBottom)

richTextBox1.ScrollToCaret();

}

else

{

richTextBox1.SelectionLength = 0;

}

string[] splitted = richTextBox1.Text.ToLower().Split(new string[] { textBox1.Text.ToLower() }, StringSplitOptions.RemoveEmptyEntries);

}

catch { }

}

private void buttonNext\_Click(object sender, EventArgs e)

{

try

{

string[] splitted = richTextBox1.Text.ToLower().Split(new string[] { textBox1.Text.ToLower() }, StringSplitOptions.None);

searchNumber = splitted.Length == 2 ? 1 : searchNumber == splitted.Length - 1 ? 1 : searchNumber + 1;

label2.Text = searchNumber.ToString();

int index = 0;

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

{

index += splitted[i].Length;

if (i > 0) index += textBox1.Text.Length;

}

if (index > 0 || searchNumber == 1)

{

if (richTextBox1.Text.Remove(0, textBox1.Text.Length) == textBox1.Text) index += 2;

richTextBox1.Select(index, textBox1.Text.Length);

if (scrollTop > richTextBox1.SelectionStart || richTextBox1.SelectionStart > scrollBottom)

richTextBox1.ScrollToCaret();

}

else

{

richTextBox1.SelectionLength = 0;

}

}

catch { }

}

private void treeView1\_AfterSelect(object sender, TreeViewEventArgs e)

{

switch (treeView1.SelectedNode.FullPath)

{

case "Синтаксис":

case "Синтаксис\\Базовий синтаксис":

richTextBox1.SelectionStart = 0;

break;

case "Синтаксис\\Математичні функції":

richTextBox1.SelectionStart = richTextBox1.Find("СИНТАКСИС МАТЕМАТИЧНИХ ФУНКЦІЙ");

break;

case "Синтаксис\\Константи":

richTextBox1.SelectionStart = richTextBox1.Find("КОНСТАНТИ");

break;

case "Базове керування":

case "Базове керування\\Методи":

richTextBox1.SelectionStart = richTextBox1.Find("МЕТОДИ РОЗВ'ЯЗКУ");

break;

case "Базове керування\\Способи експорта даних":

richTextBox1.SelectionStart = richTextBox1.Find("СПОСОБИ ЕКСПОРТУ ФУНКЦІЙ");

break;

case "Просунуте керування":

case "Просунуте керування\\Умови":

richTextBox1.SelectionStart = richTextBox1.Find("КЕРУВАННЯ ПОЧАТКОВИМИ УМОВАМИ");

break;

case "Просунуте керування\\Розрахунок":

richTextBox1.SelectionStart = richTextBox1.Find("ПАРАМЕТРИ РОЗРАХУНКУ");

break;

case "Просунуте керування\\Графік":

richTextBox1.SelectionStart = richTextBox1.Find("НАЛАШТУВАННЯ ГРАФІКІВ");

break;

case "Просунуте керування\\Експорт":

richTextBox1.SelectionStart = richTextBox1.Find("ПАРАМЕТРИ ЗБЕРЕЖЕННЯ ДАНИХ");

break;

}

richTextBox1.SelectionLength = 0;

richTextBox1.ScrollToCaret();

}

[DllImport("user32.dll")]

private static extern bool GetScrollInfo(IntPtr hWnd, int nBar, ref SCROLLINFO ScrollInfo);

[StructLayout(LayoutKind.Sequential)]

public struct SCROLLINFO

{

public int cbSize;

public int fMask;

public int nMin;

public int nMax;

public int nPage;

public int nPos;

public int nTrackPos;

}

private const int SB\_VERT = 0x1;

private const int SB\_HORZ = 0x0;

private const int SIF\_PAGE = 0x2;

private const int SIF\_POS = 0x4;

private const int SIF\_RANGE = 0x1;

private const int SIF\_TRACKPOS = 0x10;

private const int SIF\_ALL = (SIF\_RANGE | SIF\_PAGE | SIF\_POS | SIF\_TRACKPOS);

private void richTextBox1\_VScroll(object sender, EventArgs e)

{

SCROLLINFO si = new SCROLLINFO();

si.cbSize = System.Runtime.InteropServices.Marshal.SizeOf(si);

si.fMask = SIF\_ALL;

GetScrollInfo(richTextBox1.Handle, 1, ref si);

scrollTop = si.nPos;

scrollBottom = si.nMax;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсова

{

partial class AboutBox : Form

{

public AboutBox()

{

InitializeComponent();

this.Text = String.Format("Про программу");

this.labelProductName.Text = AssemblyProduct;

this.labelVer.Text = AssemblyVersion;

this.labelDesc.Text = AssemblyDescription;

}

#region Assembly Attribute Access Methods

public string AssemblyTitle

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute), false);

if (attributes.Length > 0)

{

AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)attributes[0];

if (titleAttribute.Title != "")

{

return titleAttribute.Title;

}

}

return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);

}

}

public string AssemblyVersion

{

get

{

return Assembly.GetExecutingAssembly().GetName().Version.ToString();

}

}

public string AssemblyDescription

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyDescriptionAttribute)attributes[0]).Description;

}

}

public string AssemblyProduct

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyProductAttribute)attributes[0]).Product;

}

}

public string AssemblyCopyright

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;

}

}

public string AssemblyCompany

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCompanyAttribute)attributes[0]).Company;

}

}

#endregion

}

}

<?xml version='1.0' encoding='utf-8'?>

<SettingsFile xmlns="http://schemas.microsoft.com/VisualStudio/2004/01/settings" CurrentProfile="(Default)" GeneratedClassNamespace="Курсова.Properties" GeneratedClassName="Settings">

<Profiles />

<Settings>

<Setting Name="copyByClick" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">False</Value>

</Setting>

<Setting Name="exportPath" Type="System.String" Scope="User">

<Value Profile="(Default)">./Export</Value>

</Setting>

<Setting Name="chartFileName" Type="System.String" Scope="User">

<Value Profile="(Default)">chart</Value>

</Setting>

<Setting Name="pointsFileName" Type="System.String" Scope="User">

<Value Profile="(Default)">points</Value>

</Setting>

<Setting Name="showMarker" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">False</Value>

</Setting>

<Setting Name="maxPointsAtChart" Type="System.Int32" Scope="User">

<Value Profile="(Default)">100</Value>

</Setting>

<Setting Name="decimalPlacesChartY" Type="System.Int32" Scope="User">

<Value Profile="(Default)">3</Value>

</Setting>

<Setting Name="hideLegend" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">False</Value>

</Setting>

<Setting Name="decimalPlacesListY" Type="System.Int32" Scope="User">

<Value Profile="(Default)">9</Value>

</Setting>

<Setting Name="isIterationLimited" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">False</Value>

</Setting>

<Setting Name="iterationLimit" Type="System.Int32" Scope="User">

<Value Profile="(Default)">1</Value>

</Setting>

<Setting Name="countOfMethods" Type="System.Int32" Scope="User">

<Value Profile="(Default)">1</Value>

</Setting>

<Setting Name="useDefaultConditions" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">False</Value>

</Setting>

<Setting Name="defaultX0" Type="System.String" Scope="User">

<Value Profile="(Default)">0</Value>

</Setting>

<Setting Name="defaultY0" Type="System.String" Scope="User">

<Value Profile="(Default)">0</Value>

</Setting>

<Setting Name="defaultXmin" Type="System.String" Scope="User">

<Value Profile="(Default)">0</Value>

</Setting>

<Setting Name="defaultXmax" Type="System.String" Scope="User">

<Value Profile="(Default)">1</Value>

</Setting>

<Setting Name="defaultH" Type="System.String" Scope="User">

<Value Profile="(Default)">0,1</Value>

</Setting>

<Setting Name="multiSelect" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">True</Value>

</Setting>

<Setting Name="hideHelpWindowOnStart" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">False</Value>

</Setting>

</Settings>

</SettingsFile>

**ДОДАТОК Б**

**(обов’язковий)**

**Приклади звітів**

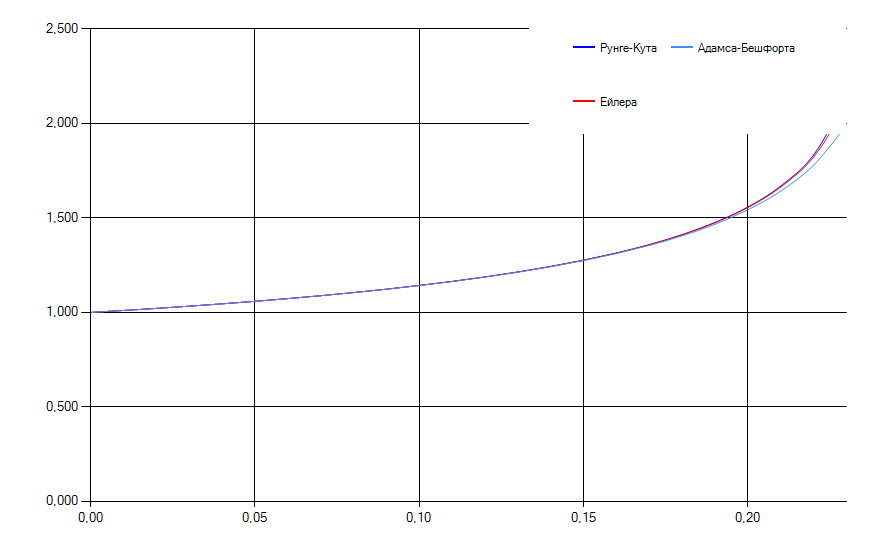
****

Рисунок Б.1 – Звіт «Графік»

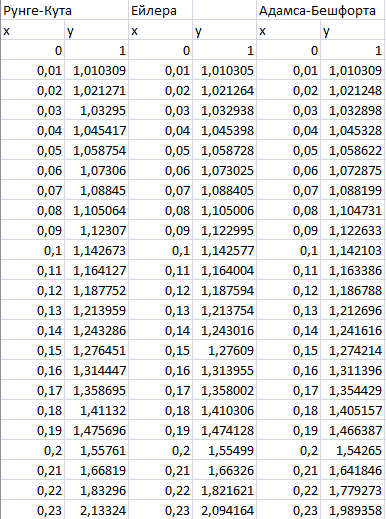
****

Рисунок Б.2 – Звіт «Значення точок»