Зміст

[Вступ 4](#_Toc470597518)

[1 Постановка задачі 5](#_Toc470597519)

[2 Теоретичні відомості 6](#_Toc470597520)

[3 Опис алгоритмів 10](#_Toc470597521)

[3.1 Алгоритм синтаксичного аналізу математичних виразів 10](#_Toc470597522)

[3.2 Алгоритм сортувальної станції 11](#_Toc470597523)

[3.3 Алгоритм зміни графа таблиці 13](#_Toc470597524)

[3.4 Алгоритм топологічного сортування 14](#_Toc470597525)

[3.5 Рекурсивний алгоритм обходу графу у глибину 14](#_Toc470597526)

[4 Опис програмного забезпечення 15](#_Toc470597527)

[4.1 Функціональна структура програмного забезпечення 15](#_Toc470597528)

[4.2 Опис функцій частин програмного забезпечення 20](#_Toc470597529)

[4.2.1 Користувацькі функції 20](#_Toc470597530)

[4.2.2 Стандартні функції 32](#_Toc470597531)

[5 Тестування програмного забезпечення 37](#_Toc470597532)

[5.1 Теорія 37](#_Toc470597533)

[5.2 План тестування 38](#_Toc470597534)

[5.3 Приклади тестування 38](#_Toc470597535)

[5.3.1 Тестування правильності введених значень 38](#_Toc470597536)

[5.3.1.1 Тестування при введенні некоректних посилань 38](#_Toc470597537)

[5.3.1.2 Тестування при введенні некоректних чисел 40](#_Toc470597538)

[5.3.1.3 Тестування при введенні неіснуючої функції 41](#_Toc470597539)

[5.3.2 Тестування коректності роботи 42](#_Toc470597540)

[5.3.2.1 Перевірка коректності збереження файлу 42](#_Toc470597541)

[5.3.2.2 Перевірка коректності відкриття файлу 43](#_Toc470597542)

[5.3.2.3 Перевірка коректності роботи при створенні таблиці малих розмірів 43](#_Toc470597543)

[5.3.2.4 Перевірка коректності роботи при створенні таблиці великих розмірів 44](#_Toc470597544)

[5.3.2.5 Перевірка поведінки програми при закритті файлу, що містить не збережені зміни 45](#_Toc470597545)

[5.3.3 Перевірка вірності роботи програми, коли користувачем вірно введені всі дані та для обраного методу рішення сходиться 45](#_Toc470597546)

[6 Інструкція користувача 48](#_Toc470597547)

[6.1 Робота з програмою 48](#_Toc470597548)

[6.2 Формат вхідних та вихідних даних 53](#_Toc470597549)

[6.3 Системні вимоги 53](#_Toc470597550)

[Висновки 54](#_Toc470597551)

[Перелік посилань 55](#_Toc470597552)

[Додаток А Тексти програмного коду 57](#_Toc470597553)

Вступ

Шахи – гра, що відома здавна та має досить велику популярніть в більшості країн серед людей будь-якого віку. Звичайно, актуальність її залишається на досить високому рівні, адже для когось вона стимулює мислену діяльність, для когось є просто методом відпочити з друзями, або ж самому.

Останнім часом набули популярність ПК, що мають високу обчислювальну швидкість та багатий функціонал, що дозволяє використовувати їх в багатьох цілях, в тому числі і розважальних. Не дивно, що шахи можуть бути також комп’ютерною грою, що не зменшую інтересу до неї, а навіть збільшує.

З час свого існування було створено декілька методів для пошуку правильного рішення, що призведе до перемоги в грі, також вони з часом поєднувались, що з давало кращий результат.

Комп’ютерна програма – бот для гри в шахів – одна з можливостей урізноманітнити цю старовинну гру за допомогою різноманітного моделювання графічної та алгоритмічної частини. Тому комп’ютерні шахи – досить актуальна гра, що допоможе людині провести час та потренувати або ж тримати в тонусі мізки.

# Постановка задачі

Розробити програмне забезпечення, що буде реалізовувати комп’ютерні шахи.

Програма повинна забезпечувати можливість виконання нижче наведених функцій:

1. Відображення шахової дошки в форматі 3D;
2. Виконання алгоритму гри в шахи для бота(автоматичної програми)
3. Реалізація підключення до віддаленого гравця напряму.
4. Зберігати статистику ігор та дані користувача.

Вхідними даними програми є дані користувача, що зберігаються на диску та його дії в грі.

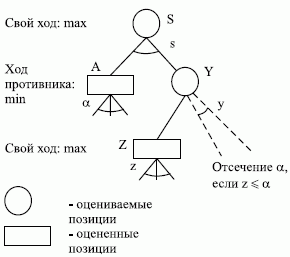
Вихідними даними є статистика ігор гравця та моделювання процесу гри.

# Теоретичні відомості

Важливим механіхмом в шазматних програмах є алгоритми пошуку оптимального ходу. Є декілька алгоритмів пошуку, кожен з яких є покращенням іншого. Серед них є мінімакс та альфа-бета відсікання. Розглянемо алгоритм цей алгоритм.

Теоретично, це еквівалентна мінімаксу процедура, за допомогою якої завжди виходить такий же результат, але помітно швидше, так як цілі частини дерева виключаються без проведення аналізу. В основі цієї процедури лежить ідея Дж. Маккарті про використання двох змінних, позначених α і β.

Основна ідея методу полягає в порівнянні найкращих оцінок, отриманих для повністю вивчених гілок, з найкращими передбачуваними оцінками для решти. Можна показати, що при певних умовах деякі обчислення є зайвими. Розглянемо ідею відсікання на прикладі рис. 3.6. Припустимо, позиція А повністю проаналізовано і знайдено значення її оцінки α. Припустимо, що один хід з позиції Y призводить до позиції Z, оцінка якої за методом минимакса дорівнює z. Припустимо, що z ≤ α. Після аналізу вузла Z, коли справедливо співвідношення y ≤ z ≤ α ≤ s, гілки дерева, що виходять з вузла Y, можуть бути відкинуті (альфа-відсікання).



Мал. 3.6. - відсікання

Якщо ми захочемо опуститися до вузла Z, лежачого на рівні довільної глибини, що належить тій же стороні, що і рівень S, то необхідно враховувати мінімальне значення оцінки β., одержуваної на ходах супротивника.

Відсікання типу β можна виконати щоразу, коли оцінка позиції, що виникає після ходу противника, перевищує значення β. Алгоритм пошуку будується так, що оцінки своїх ходів і ходів противника порівнюються при аналізі дерева з величинами α і β відповідно. На початку обчислень цим величинам присвоюються значення +∞ і -∞, а потім, у міру просування до кореня дерева, знаходиться оцінка початкової позиції і найкращий хід для одного з супротивників.

Правила обчислення α і β в процесі пошуку рекомендуються наступні:

а) у MAX вершини значення α одно найбільшому в даний момент значенням серед остаточних повернутих значень для її дочірніх вершин;

б) у MIN вершини значення β одно найменшому в даний момент значенням серед остаточних повернутих значень для її дочірніх вершин.

Правила припинення пошуку:

а) Можна не проводити пошуку на поддереве, що росте з будь-якої MIN вершини, у якій значення β не перевищує значення α всіх її батьківських MAX вершин;

б) можна не проводити пошуку на поддереве, що росте з будь-якої MAX вершини, у якій значення α не менш значення β всіх її батьківських MIN вершин.

На рис. 3.7 показані α - β відсікання для конкретного прикладу. Таким чином, α - β - алгоритми дає той же результат, що і метод мінімакса, але виконується швидше.

Використання алгоритмів евристичного пошуку для пошуку на графі виграшної стратегії в складніших завданнях і іграх (шашки, шахи) не реальний. За деякими оцінками ігрове дерево гри в шашки містить 1040 вершин, в шахах 10120 вершин. Якщо при грі в шашки для однієї вершини потрібно 1/3 наносекунди, то всього ігрового часу буде потрібно 1021 століть. У таких випадках вводяться штучні умови зупинки, засновані на таких факторах, як найбільша допустима глибина вершин в дереві пошуку або обмеження на час і обсяг пам'яті.

# Опис алгоритмів

## Алгоритм альфа-бета відсікання

1. ПОЧАТОК
2. Перевести вхідну строку з інфіксної в постфіксну нотацію за допомогою алгоритму сортувальної станції (підрозділ 3.2).
3. ПОКИ не закінчилася вхідна строка:
   1. ЯКЩО поточний символ цифра АБО поточний символ знак мінус ТА наступний символ цифра, ТО починаючи з поточної позиції перевести символи в число та помістити його в стек
   2. ЯКЩО поточний символ «$» АБО поточний символ знак мінус та наступний символ «$», ТО:
      1. Знайти координати на які вказує посилання.
      2. ЯКЩО комірка з заданими координатами існує ТА даній комірці зберігається число, ТО помістити дане число в стек, ІНАКШЕ видати повідомлення про помилку.
   3. ЯКЩО поточний символ є елементарною операцією, ТО:
      1. ЯКЩО стек містить менше 2-х значень, ТО видати повідомлення про помилку.
      2. Витягнути зі стеку два числа та виконати над ними задану операцію. Результат операції помістити в стек.
   4. ЯКЩО поточний символ є літерою, ТО:
      1. ЯКЩО стек пустий, то видати повідомлення про помилку.
      2. ЯКЩО існує задана в формулі функція ТА верхній елемент стеку належить області допустимих значень даної функції, ТО виконати дану функцію з цим значенням та результат помістити в стек. ІНАКШЕ видати повідомлення про помилку.
4. КІНЕЦЬ

## Алгоритм сортувальної станції

1. ПОЧАТОК
2. ПОКИ не кінець вхідної строки:
   1. ПОКИ поточний символ є пробілом, ТО перейти до наступного символа.
   2. ЯКЩО досягнутий кінець вхідної строки, ТО перейти до пункту 3.
   3. ЯКЩО поточний символ цифра, ТО:
      1. ПОКИ поточний символ є цифрою, ТО вставити поточний символ в вихідну строку та перейти до наступного символу.
      2. ЯКЩО поточний символ є точкою, ТО помістити її вихідну строку та перейти до пункту 2.3.3. ІНАКШЕ перейти до пункту 2.3.4.
      3. ПОКИ поточний символ є цифрою, ТО вставити поточний символ в вихідну строку та перейти до наступного символу.
      4. ЯКЩО поточний символ є точкою, ТО видати повідомлення про помилку.
      5. Вставити пробільний символ в вихідну строку.
      6. Перейти до пункту 2.
   4. ЯКЩО поточний символ відкриваюча дужка, ТО помістити її в стек та перейти до пункту 2.
   5. ЯКЩО поточний символ ТО:
      1. ПОКИ стек не пустий ТА поточний елемент стеку не відкриваюча дужка, ТО помістити верхній елемент стеку в вихідну строку та видалити даний елемент зі стеку.
      2. Вставити пробільний символ в вихідну строку.
      3. Перейти до пункту 2.
   6. ЯКЩО поточний символ є символом елементарної операції, ТО:
      1. ЯКЩО поточний символ є знаком мінус ТА вихідна строка пуста АБО верхній елемент стеку є відкриваючою дужкою, ТО помістити знак мінус як унарний операнд в вихідну строку ТА перейти до пункту 2.
      2. ПОКИ стек не пустий ТА на вершині стеку знаходиться операція з пріоритетом не нижчим, ніж дана операції, ТО помістити операцію зі стеку в вихідну строку та видалити її зі стеку.
      3. Помістити поточну операцію у стек.
      4. Вставити пробільний символ в вихідну строку.
      5. Перейти до пункту 2.
   7. ЯКЩО поточний символ «$», ТО:
      1. ПОКИ поточний символ літера, ТО помістити поточну літеру в вихідну строку та перейди до наступного символу.
      2. ЯКЩО поточний символ не дорівнює «$», ТО видати повідомлення про помилку.
      3. ПОКИ поточний символ цифра, ТО помістити цю цифру в вихідну строку та перейти до наступного символу.
      4. Помістити пробіл в вихідну строку.
      5. Перейти до пункту 2.
   8. ЯКЩО поточний символ є літерою, ТО:
      1. ПОКИ поточний символ є літерою, ТО помістити її вихідну строку та перейти до наступного символу.
      2. ПОКИ поточний символ є пробілом, ТО перейти до наступного символу.
      3. ЯКЩО досягнуто кінець строки АБО поточний символ не є відкриваючою дужкою, ТО видати повідомлення про помилку.
      4. Добавити назву функції в стек.
      5. Добавити відкриваючу дужку в стек.
      6. Добавити пробільний символ в вихідну строку.
      7. Перейти до пункту 2.
   9. Вивести повідомлення про помилку.
3. ПОКИ стек не пустий, ТО помісти операцію з вершини стеку в вихідну строку та видалити її зі стеку.
4. КІНЕЦЬ

## Алгоритм зміни графа таблиці

1. ПОЧАТОК
2. ЯКЩО змінена комірка містила вірну формулу, ТО:
   1. Знайти список всіх комірок на які посилається стара формула зміненої комірки.
   2. Видалити з графу відповідні ребра між зміненою коміркою та всіма комірками, що знайдені у пункті 2.1.
3. ЯКЩО у змінену комірку введено вірно формулу, ТО:
   1. Знайти список всіх комірок на які посилається нова формула зміненої комірки.
   2. Добавити у граф відповідні ребра між зміненою коміркою та всіма комірками, що знайдені у пункті 3.1.
4. Виконати топологічне сортування графу, починаючи зі зміненої комірки (підрозділ 3.4) та помістити результат сортування у список.
5. Видалити змінену комірку зі списку знайденому у пункті 4.
6. ПОКИ список, що знайдений у пункті 5, містить елементи, ТО:
   1. Обчислити значення комірки, що містить у хвості списку.
   2. Видалити комірку зі хвосту списку.
7. КІНЕЦЬ

## Алгоритм топологічного сортування

1. ПОЧАТОК
2. Стоворити масив відвіданих вершин.
3. Виконати обхід в глибину графа (підрозділ 3.5) починаючи з комірки, що була змінена.
4. КІНЕЦЬ

## Рекурсивний алгоритм обходу графу у глибину

1. ПОЧАТОК
2. Помітити поточну вершину, як ту, що знаходиться в процесі обробки.
3. ЦИКЛ по всім ребрам, що виходять з поточної вершини:
   1. ЯКЩО вершина в яку веде поточне ребро ще не відвідана, ТО рекурсивний виклик алгоритму обходу графу у глибину   
      (підрозділ 3.5).
   2. ЯКЩО поточно вершина знаходить в процесі обробки, ТО помітити поточну вершину як вершину, що входить до складу циклу.
4. Помітити поточну вершину як оброблену.
5. Помістити поточну вершину у чергу, що є топологічно відсортованою.
6. КІНЕЦЬ

# Опис програмного забезпечення

## Функціональна структура програмного забезпечення

На рисунку 4.1 Ви можете бачити загальну структуру додатку, де наявні усі класи програми та стрілками позначено наслідування класів (стрілка напрямлена до Базового класу).



Рисунок 4.1 – Загальна структура класів проекту

На рисунку 4.2 зображено загальну структуру основного класу додатку – Excel, який безпосередньо пов'язує всі компоненти програми між собою та надає інтерфейс користувачеві.



Рисунок 4.2 – Загальна структура класу Excel

На рисунку 4.3 зображено загальну структуру форми збереження таблиці у файл – Save.



Рисунок 4.3 – Загальна структура класу Save

На рисунку 4.4 зображено загальну структуру форми створення нової   
таблиці – New.



Рисунок 4.4 – Загальна структура класу New

На рисунку 4.5 зображено загальну структуру форми спраки – Help.



Рисунок 4.5 – Загальна структура класу Help

На рисунку 4.6 зображено загальну структуру класа синтаксичного аналізатора – Parser.



Рисунок 4.6 – Загальна структура класу Parser

На рисунку 4.7 зображено загальну структуру класа графу – Graph.



Рисунок 4.7 – Загальна структура класу Graph

На рисунку 4.8 зображено загальну структуру класа представлення комірки таблиці – Cell.



Рисунок 4.8 – Загальна структура класу Cell

На рисунку 4.9 зображено загальну структуру класа представлення   
таблиці – Table.



Рисунок 4.9 – Загальна структура класу Table

На рисунку 4.10 зображено загальну структуру класа представлення   
таблиці – About.



Рисунок 4.10 – Загальна структура класу Table

## Опис функцій частин програмного забезпечення

В ході виконання поставленого завдання було створено наступні модулі та бібліотеки:

а) Cell.h – реалізує комірку таблиці;

б) Graph.h – реалізує побудову та проведення дій з графом таблиці;

в) Number.h – реалізує тип число, що є основним контейнером результатів;

г) Parser.h – реалізує механізм синтаксичного розбору математичних виразів;

д) Table.h – реалізує таблицю;

е) resource.h – містить унікальні ідентифікатори ресурсів проекту;

є) Excel.h - реалізує головне вікно програми;

ж) New.h - реалізує вікно створення нової таблиці;

з) About.h - реалізує вікно About;

и) Help.h - реалізує вікно зі справою;

і) Save.h - реалізує вікно запитання про збереження незбереженого файлу.

### Користувацькі функції

Користувацькі функції, які використані в даній курсовій роботі, описані в таблиці 4.1.

Таблиця 4.1 – Опис користувацьких функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | Config | Get | Отримання налаштувань | - | - |
| 2 | Config | Set | Установка налаштувань | Нове значення поля value | - |
| 3 | Converter | ConvertChessPosition | Конвертування позиції | - | Поточне значення поля value |
| 4 | Converter | ConvertFigureToPieces | Конвертування позиції | - | Поточне значення поля result |
| 5 | Serializer | Load | Отримання даних з файлу | Нове значення поля result | - |
| 6 | Serializer | LoadFromStream | Отримання даних з потоку | - | Поточне значення поля isFormula |
| 7 | Serializer | Save | Збереження до файлу | Нове значення поля isFormula | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметра |
| 8 | Serializer | SaveString | Серіалізація в строку | Координати поточної комірки, посилання на таблицю, посилання на масив результату сортування та посилання на масив  циклів | - |
| 9 | GameFactory | GetGame | Отримання гри з її типом | Номер поточної вершини, посилання на масив відвіданих вершин, посилання на масив результату обходу та посилання на масив циклів | - |
| 10 | GameState | GetWithMove | Отримання стану гри разом з ходом | Строка, що містить лінк (посилання), індекс з якої починати розбір Y-координату, ширина таблиці | Число, Y-координата лінку |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 11 | GameState | Parse | Отримання стану гри зі строки | Строка, що містить лінк (посилання), індекс з якої починати розбір X-координату, висота таблиці | Число, X-координата лінку |
| 12 | ChessAI | CostPosition | Оцінити поточну позицію | Посилання на таблицю, посилання на поточне представлення таблиці, координати зміненої комірки та чи містила раніше поточна комірка формулу | - |
| 13 | ChessAI | GetMove | Рекурсивно обрахувати ходи | Строка С# | Строка С++ |
| 14 | ChessAI | GetMoveAsync | Асинзронно отримати хід | Формула, яку необхідно підрахувати та посилання на таблицю | Значення формули |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 15 | ChessToolValidate | CanMove | Перевірка чи можна ходити | Формула в інфіксному форматі | Формула в постфіксному форматі |
| 16 | ChessToolValidate | GetMoves | Отримання можливих ходів | Символ операції | Пріоритет операції |
| 17 | ChessToolValidate | MakeMove | Зробоити хід | Строка, що містить лінк, індекс звідки починається розбір, висота таблиці | Число, X-координата лінку |
| 18 | ChessToolValidate | QueryCancelMove | Відминити хід | Строка, що містить лінк, індекс звідки починається розбір, ширина таблиці | Число, Y-координата лінку |
| 19 | App | Main | Вхідна точка програми | Строка, що містить назву функції та індекс її першого символу назви, індекс назви функції з бібліотеки | True, якщо назви співпадають, інашкше False |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 20 | ChessPiece | AnimateCaptureMove | Анімація руху | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці | - |
| 21 | ChessPiece | PerformAnimatedTranslation | Анімація перетворення | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці, стек операцій | - |
| 22 | ChessPiece | ReadInt32Collection | Зчитати дані про сітку | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці, стек операцій | - |
| 23 | ChessPiece | ReadPoint3DCollection | Зчитати дані про сітку | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 24 | ChessPiece | ReadPointCollection | Зчитати дані про сітку | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці, стек операцій | - |
| 25 | ChessPiece | ReadVector3DCollection | Зчитати дані про сітку | Вхідна строка, стек чисел, індекс значення в вхідній строці | - |
| 26 | ChessPiece | RemoveOldTransform | Видалення анмації | Посилання на таблицю, вхідна строка, стек чисел, індекс значення в вхідній строці | - |
| 27 | ChessPiece | Transformation\_changed | Подія зміни трансформації | Вхідна строка, стек чисел, індекс значення в вхідній строці | - |
| 28 | ChessPiece | calculateFunctions | Обчислення функції | Вхідна строка, стек чисел, індекс значення в вхідній строці | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 29 | MainPresenter | AddChatMessage | Додавання повідомлення в чат | Вхідна строка | Індекс символу присвоєння |
| 30 | MainPresenter | ClearChat | Очистка чату | Індекс функції у бібліотеці та її аргумент | Значення функції |
| 31 | MainPresenter | HideChat | Скриття чату | Вхідна строка та індекс з якого починається пошук | Значення єдиного числа |
| 32 | MainPresenter | ShowChat | Зображення чату | Розміри таблиці | - |
| 33 | MainPresenter | QueryForSeupView | Запрос на установку елемента | - | Поточне значення поля height |
| 34 | MainPresenter | ReleaseControl | Установка стнадратного елемента | - | Поточне значення поля width |
| 35 | MainPresenter | SetupView | Вимушена установка елемента | Значення на яке треба змінити поле height | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 36 | MainPresenter | ShowInfo | Вивести повідомленгня | Значення на яке треба змінити поле width | - |
| 37 | MainPresenter | ShowError | Вивести повідолення про помилку | - | - |
| 38 | MainPresenter | StartGameAgainstPC | Почати гру проти ПК | - | - |
| 39 | MainPresenter | StartLocalGame | Почати гру з самим собою | - | - |
| 40 | MainPresenter | StartSettings | Запустити вікно налаштувань | Шлях до файлу | - |
| 41 | MainPresenter | StartStatistics | Запустити вікно статистик | Номер стовпця | Строка - заголовок |
| 42 | MainPresenter | Get | Отримати екземпляр | Рядок | Рядок |

### Стандартні функції

Стандартні функції, які використані в даній курсовій роботі, описані в   
таблиці 4.2.

Таблиця 4.2 – Опис стандартних функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Назва класу |
| 1 | Add | Додавання елементу до списку | Елемент, що слід додати | - | List |
| 2 | Count | Отримання розміру списку | Список | Число | List |
| 3 | Debug | Запис інфомації в вікно налагодження | Повідомлення | - | Write |
| 4 | ToInt32 | Конвертування в ціле число | Змінна простого типу | Ціле число | Convert |
| 5 | ToString | Конвертування в строку | Параметр функції | Значення функції для заданого параметру | Convert |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Назва класу |
| 6 | Invoke | Виклик функції без повернаного значення | Параметр функції | Значення функції для заданого параметру | Action |
| 7 | Invoke | Виклик функції | Параметр функції | Значення функції для заданого параметру | Func |
| 8 | Run | Запуск нової задачі в іншому потоці | Параметр функції | Значення функції для заданого параметру | Task |
| 9 | Fetch | Отримання даних з БД | Параметр функції | Значення функції для заданого параметру | Database |
| 10 | Select | Маппінг | Параметр функції | Значення функції для заданого параметру | IEnumerable |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Назва класу |
| 11 | Where | Фільтр | Параметр функції | Значення функції для заданого параметру | IEnumerable |
| 12 | ContainsKey | Перевірка вхрдження ключа | Параметр функції | Значення функції для заданого параметру | Dictonary |
| 13 | Serialize | Серіалізує об’єкт | Параметр функції | Значення функції для заданого параметру | XmlSerializer |
| 14 | ShowDialog | Показує діалогове вікно | - | - | MessageBox |
| 15 | Strcpy | Копіює рядок | 2 рядки | - | string.h |
| 16 | Append | Заклеює рядки | 2 рядки | - | StringBuilder |
| 17 | CreateDirectory | Створює каталог | Рядок з адресою нового каталогу | Код помилки | Directory |
| 18 | GetBytes | Перетворює вхідні данні на масив байтів | Данні | Масив байтів | Encoding |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Назва класу |
| 19 | Write | Записує масив байтів у файл | Масив байтів, зміщення, розмір | - | Stream |
| 20 | WriteByte | Записує байт у файл | Байт данних | - | Stream |
| 21 | GetSpecialFolder | Повертає папку в системі | Ім’я змінної | Тип папки | Environment |
| 22 | Read | Зчитує масив байт з файлу | зміщення, розмір | Масив байтів | Stream |
| 23 | ToDouble | Перетворює масив байтів у int | Масив байтів | Число int | Convert |
| 24 | GetType | Змінює розмір масиву | Масив, новий розмів | - | object |
| 25 | ReadByte | Зчитує байт з файлу | - | Байт даних | FileStream |
| 26 | Add | Додає елемент до колекції | Елемент | - | - |
| 27 | IsNullOrEmpty | Порівнює рядки | 2 рядки | Індикатор рівності рядків | string |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Назва класу |
| 28 | ToCharArray | Переводить рядок .net в масив символів | Рядок | Масив символів | string |
| 29 | Remove | Видаляє елемент з колекції | Елемент | - | List |
| 30 | Add | Додає стовпець | Ім’я стовпця, заголовок стовпця | - | - |
| 31 | Add | Додає рядки | Кількість рядків | - | - |
| 32 | Now | Повертає час | - | Час | DateTime |
| 33 | OpenFile | Відкриває файл вибраний в діалозі вибору файла | - | - | FileStream |
| 34 | ToString | Перетворює об’єкт у рядок | Об’єкт | Рядок | int |
| 35 | Close | Закриває вікно | - | - | Window |

# Тестування програмного забезпечення

## Теорія

Тестування програмного забезпечення — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:

а) відповідність вимогам, якими керувалися проектувальники та розробники;

б) правильна відповідь для усіх можливих вхідних даних;

в) виконання функцій за прийнятний час;

г) практичність;

д) сумісність з програмним забезпеченням та операційними системами;

е) відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення багів (помилок або інших дефектів).

Тестування ПЗ може надавати об'єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів так і для замовників.

Тестування може проводитись, як тільки створено виконуваний код (навіть частково завершено). Процес розробки зазвичай передбачає коли та як буде відбуватися тестування. Наприклад, при поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На противагу цьому, відповідно до вимог гнучкої розробки ПЗ, програмування і тестування часто відбувається одночасно.

## План тестування

Усі можливі випадки виникнення помилок у програмі залежать від вхідних даних, тобто тої інформації, що отримується від користувача. Тому тестування програми полягає у виявленні правильності та коректності обробки програмою різних вхідних даних.

Задля виявлення усіх помилок у роботі програми потрібно запустити її на виконання в таких умовах:

1. Тестування правильності введених значень.
   1. Тестування при введенні некоректних налаштувань гри
   2. Тестування при спробі зробити некоректний хід
2. Тестування коректності роботи
   1. Перевірка коректності роботи програми-бота
   2. Перевірка коректності керування ігровим процесом в програмі
   3. Перевірка коректності завантаження та збереження даних
   4. Перевірка коректності переходів по пунктам меню

Для вирішення відповідних помилкових ситуацій слід вдосконалити алгоритми роботи програми та обробити всі виключні ситуації.

Далі слід упевнитись, що усі методи програми коректно працюють на усіх наборах вхідних значень.

## Приклади тестування

### Тестування правильності введених значень

#### Тестування при введенні некоректних посилань

Якщо під час введення інформації в таблицю користувач введе посилання на комірку, що не існує (Рисунок 5.1 - 5.3) або введе посилання у невірному форматі (Рисунок 5.4), то в відповідну комірку програма видасть попередження.

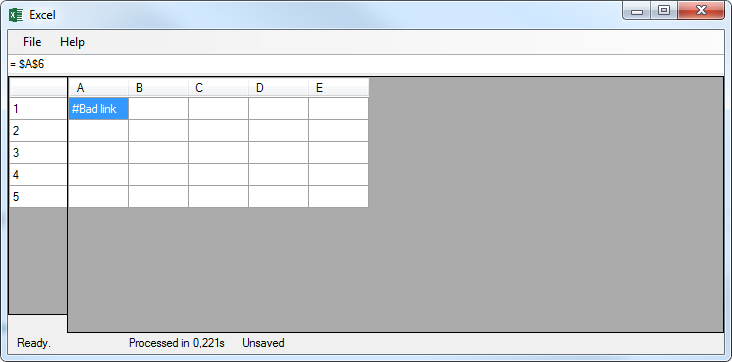


Рисунок 5.1 – Вікно програми, після введення посилання на неіснуючий рядок

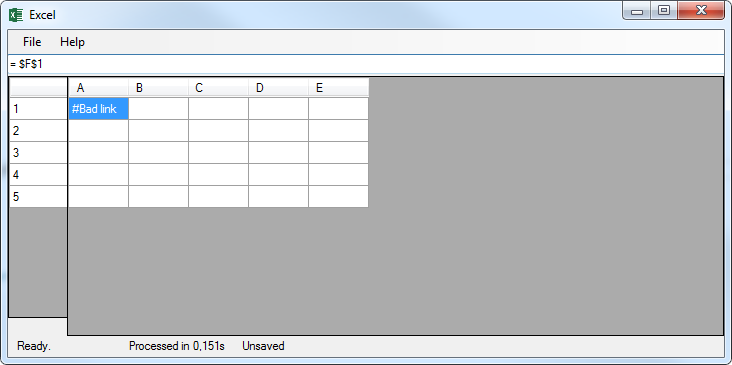


Рисунок 5.2 – Вікно програми, після введення посилання на неіснуючю колонку

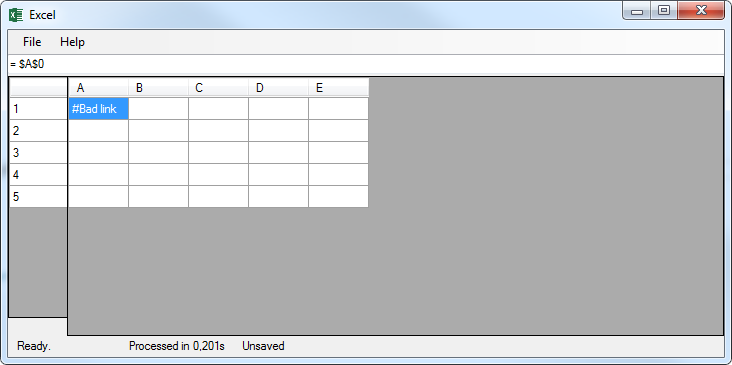


Рисунок 5.3 – Вікно програми, після введення посилання на неіснуючий рядок

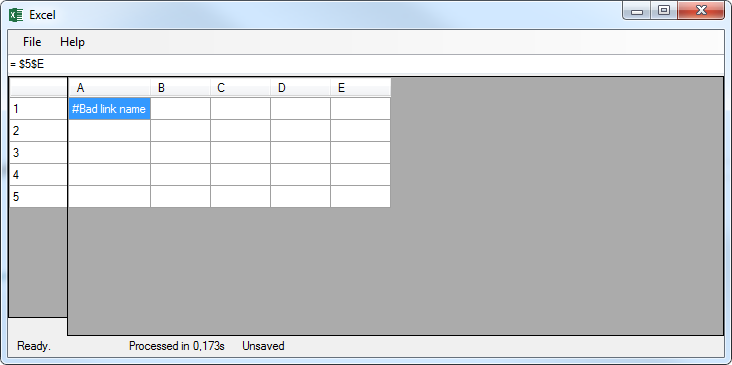


Рисунок 5.4 – Вікно програми, після введення посилання у некоректному форматі

#### Тестування при введенні некоректних чисел

Якщо під час введення інформації в таблицю користувач введе число у некоректному форматі (Рисунок 5.5), то в відповідну комірку програма видасть попередження.

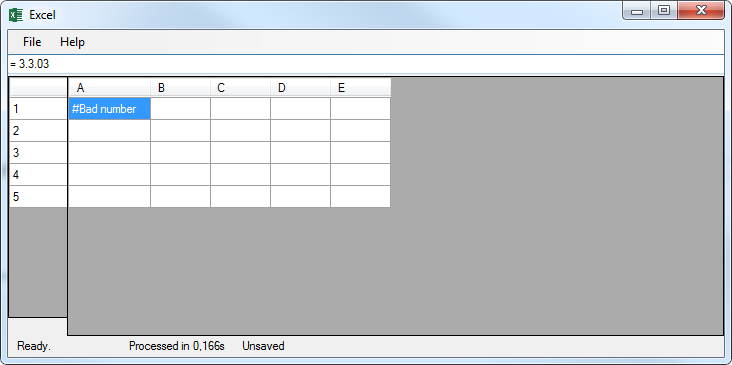


Рисунок 5.5 – Вікно програми, після введення числа у некоректному форматі

#### Тестування при введенні неіснуючої функції

Якщо під час введення інформації в таблицю користувач введе назву неіснуючої функції (Рисунок 5.6), то в відповідну комірку програма видасть попередження.

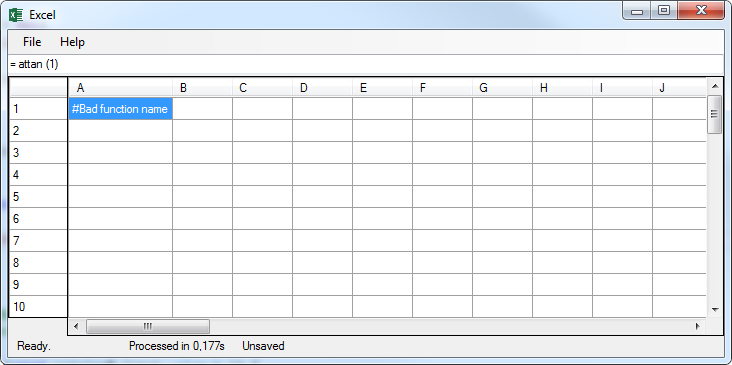


Рисунок 5.6 – Вікно програми, після введення неіснуючої функції

### Тестування коректності роботи

#### Перевірка коректності збереження файлу

Якщо під час користування програмним продуктом буде винесено якісь зміни у файл, то програма виведе відповідне повідомлення в рядку статусу (Рисунок 5.7). Відповідні зміни можна зберегти у новий файл або поточний. Тоді відповідно дане повідомлення зникне (Рисунок 5.8).

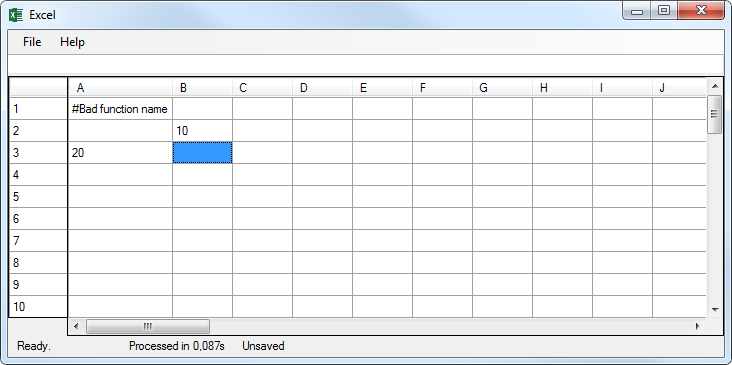


Рисунок 5.7 – Вікно програми, після внесення змін у файл

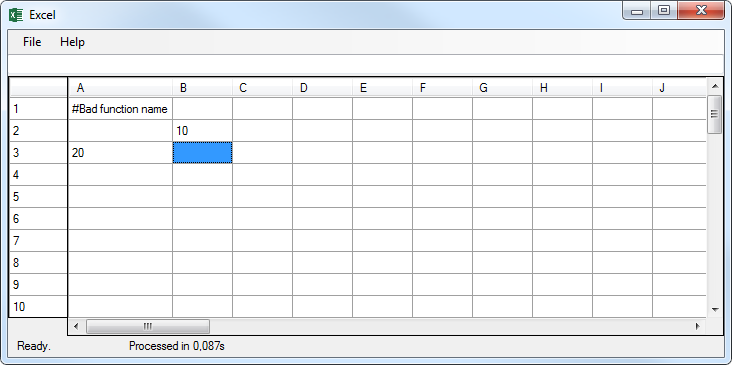


Рисунок 5.8 – Вікно програми, після збереження у файл

#### Перевірка коректності відкриття файлу

Відкриття збереженого файла в пункті 5.3.2.1 зображено на рисунок 5.9.

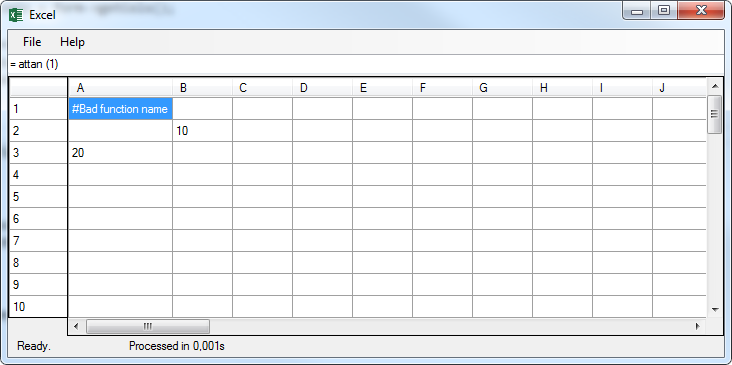


Рисунок 5.9 – Вікно програми, після відкриття таблиці з файлу

#### Перевірка коректності роботи при створенні таблиці малих розмірів

Якщо користувач створить таблицю мінімального можливого розміру (1х1) та введе в неї які-небудь дані, то програма спрацює коректно (Рисунок 5.10).

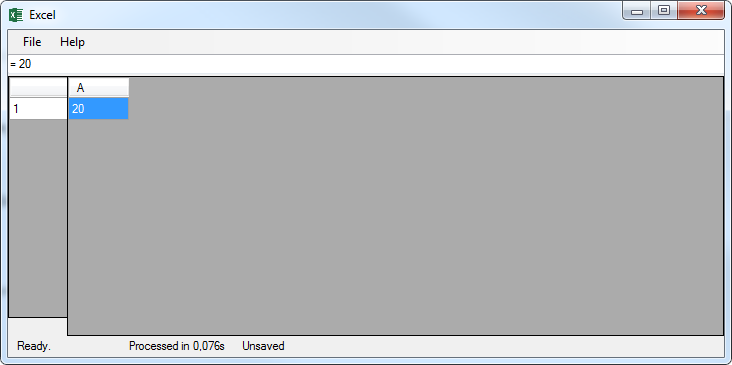


Рисунок 5.10 – Вікно програми, після створення таблиці мінімального розміру

#### Перевірка коректності роботи при створенні таблиці великих розмірів

Якщо користувач створить таблицю максимально можливого розміру (500х500) та введе в неї будь-які дані, то програма спрацює коректно   
(Рисунок 5.11).

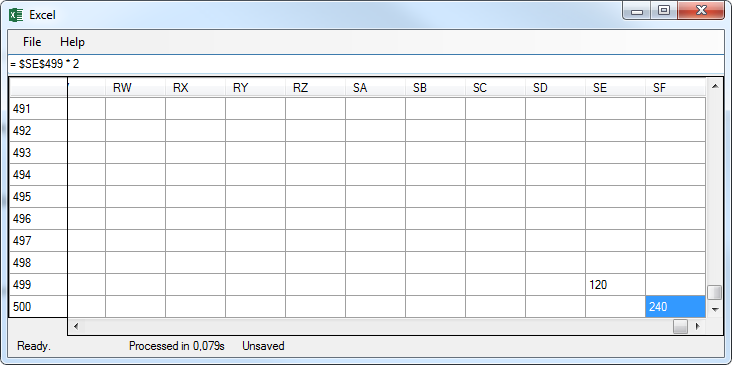


Рисунок 5.11 – Вікно програми, після створення таблиці максимального розміру

#### Перевірка поведінки програми при закритті файлу, що містить не збережені зміни

Якщо користувач внесе певні зміни у файл та спробує його закрити без збереження вмісту, то він отримає повідомлення про те, що файл містить не збережені зміни (Рисунок 5.12).

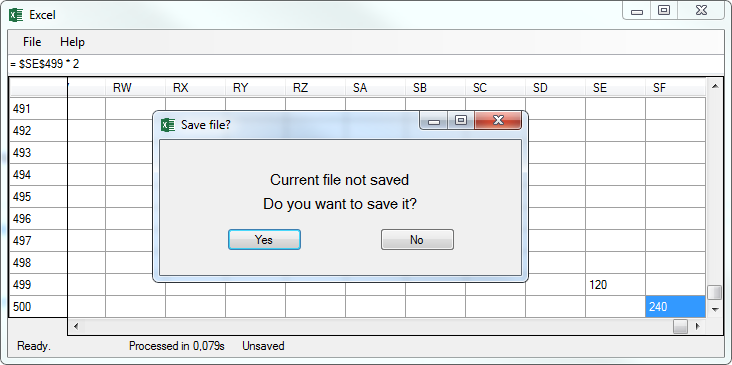


Рисунок 5.12 – Вікно програми, після спроби закрити файл, що містить незбережені зміни

### Перевірка вірності роботи програми, коли користувачем вірно введені всі дані та для обраного методу рішення сходиться

Результати тестування програми на коретних даних наведена у   
таблицях 5.1 - 5.3.

Таблиця 5.1 – Тестування обробки посилань

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані посилань у формулі |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір таблиці: 50 на 50 |
| Схема проведення тесту | Введення даних та натискання кнопки «Enter» |
| Очікуваний результат | Виведення рішення формули |
| Стан програми після проведення випробувань | Виведено рішення формули |

Таблиця 5.2 – Тестування обробки функцій

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані функцій у формулі |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір таблиці: 50 на 50 |
| Схема проведення тесту | Введення даних та натискання кнопки «Enter» |
| Очікуваний результат | Виведення рішення формули |
| Стан програми після проведення випробувань | Виведено рішення формули |

Таблиця 5.3 – Тестування обробки складних формул з багатьма посиланнями та функціями

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані обробки складних формул з багатьма посиланнями та функціями |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір таблиці: 50 на 50 |
| Схема проведення тесту | Введення даних та натискання кнопки «Enter» |
| Очікуваний результат | Виведення рішення формули |
| Стан програми після проведення випробувань | Виведено рішення системи |

Критичні ситуації у роботі програми виявлені не були. Під час тестування було виявлено, що більшість помилок виникало тоді, коли користувачем вводилися не коректні вхідні дані. Тому всі дані, які вводить користувач, ретельно перевіряються на валідність і лише потім подаються на обробку програмі.

# Інструкція користувача

## Робота з програмою

Після запуску виконавчого файлу з розширенням \*.exe, відкривається головне вікно програми (Рисунок 6.1).

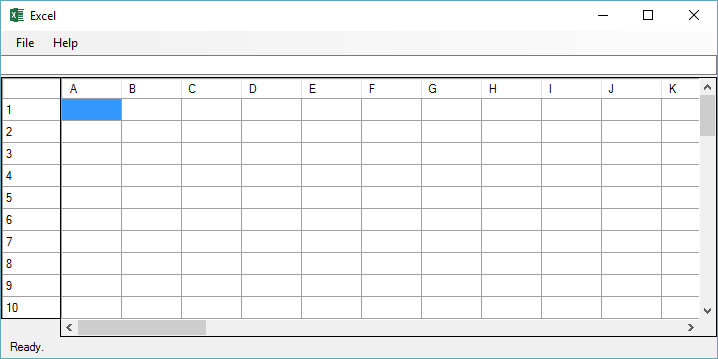


Рисунок 6.1 – Головне вікно програми

Далі за допомогою меню File → New можна створити нову таблицю з заданою висотою та шириною (Рисунок 6.2).

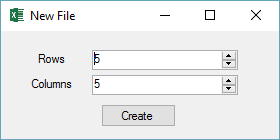


Рисунок 6.2 – Вибір необхідного розміру таблиці

Далі можливо редагувати данні в таблиці (Рисунок 6.3).

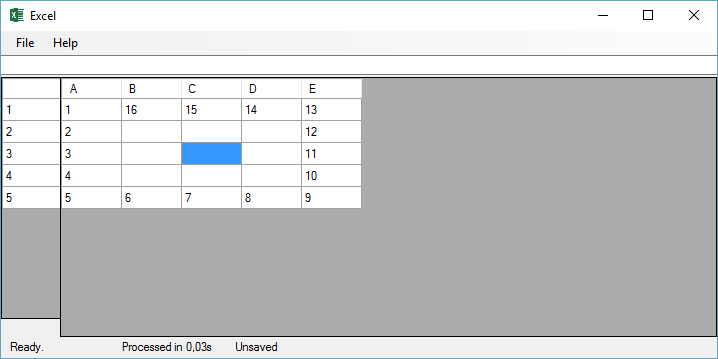


Рисунок 6.3 – Введення вхідних даних

Якщо якісь вхідні дані є формулами вони автоматично будуть підраховані(Рисунок 6.4).

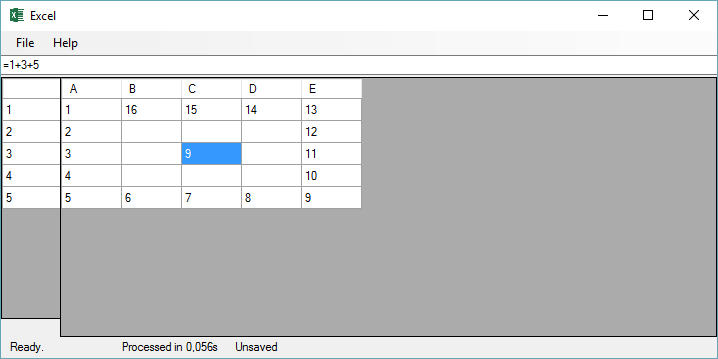


Рисунок 6.4 – Приклад формули

Якщо буде занесена неправильна формула програма сповістить про помилку (Рисунок 6.5).

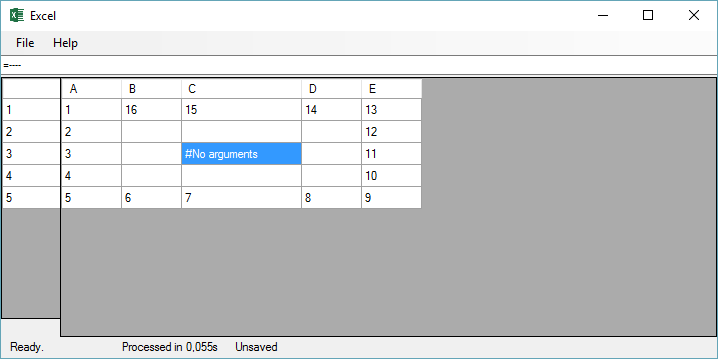


Рисунок 6.5 – Приклад повідомлення про помилку

В формулах можна посилатись на інші комірки (Рисунок 6.6).

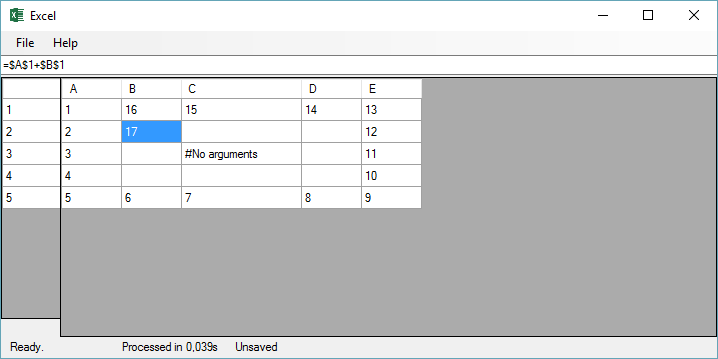


Рисунок 6.6 – Приклад складної формули

За допомогою меню File → Save можна зберегти таблицю (Рисунок 6.7).

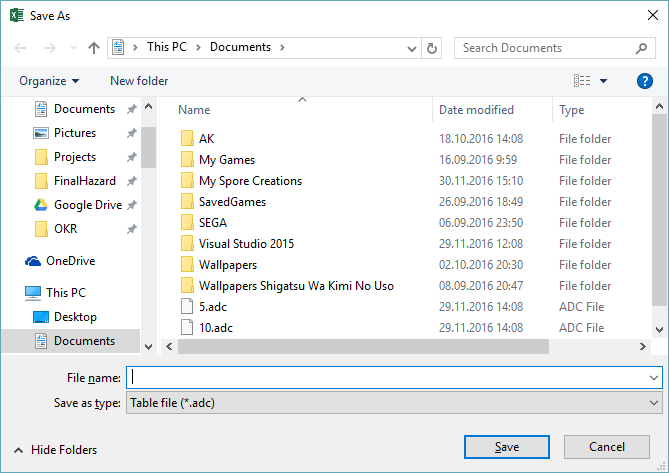


Рисунок 6.7 – Збереження таблиці

За допомогою меню File → Open можна відкрити таблицю (Рисунок 6.8).

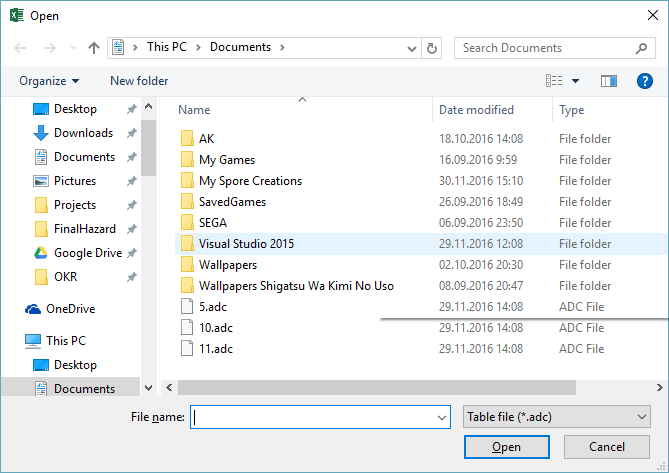


Рисунок 6.8 – Приклад відкриття таблиці

За допомогою меню Latest можна відкрити останні використані файли   
(Рисунок 6.9).

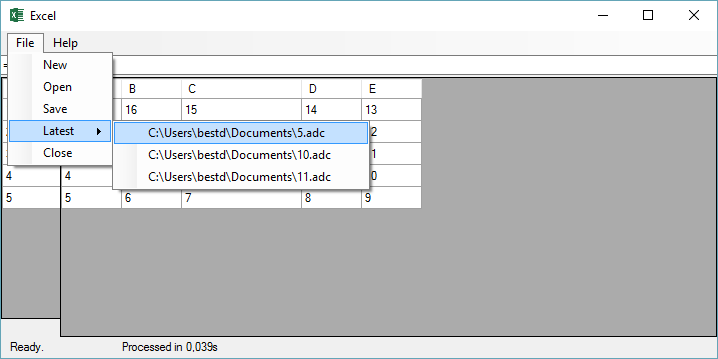


Рисунок 6.9 – Приклад останніх файлів

Також можливо закрити програму за допомогою меню File → Close. Якщо користувач спробує закрити змінений файл програма запропонує зберегти поточний файл.

## Формат вхідних та вихідних даних

Вхідними та вихідними данними є дані користувача та моделювання процесу гри в шахи.

## Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1.

Таблиця 6.1 – Системні вимоги програмного забезпечення

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows 7/ Windows 8/Windows 10 | |
| Процесор | Intel® Pentium® IV  3.0 GHz або  AMD Athlon™ 64 3000+ 2.0 GHz | Intel® Pentium® D або AMD Athlon™ 64 X2 |
| Оперативна пам'ять | 1 GB RAM | 2 GB RAM |
| Відеоадаптер | Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог) | |
| Жорсткий диск | 512Мб | 1Гб |
| Дисплей | 800х600 | 1024х768 або краще |
| Прилади введення | Клавіатура, комп’ютерна миша | |
| Додаткове програмне забезпечення | Microsoft .Net Framework 4.5.2 або вище | |

Висновки

Етап ознайомлення з теоретичною частиною завдання даної курсової роботи включав пошук необхідних механізмів для вирішення поставленої задачі та знаходження алгоритмів реалізації, проаналізовано складність виконання курсової, особливості та доцільність використання в межах реальних проектів, що було доведено реалізацією програмного забезпечення, яке розглядається в данні курсовій роботі.

Етап проектування включав проведення об’єктно-орієнтованого аналізу, будування базової структури програми, налагодження моделі зв’язків між модулями, створення основних алгоритмів курсової. Також було передбачено всі проблемні ситуації та ситуації що потребують «гнучкості» програмного забезпечення.

Етап кодування включав в себе реалізацію всіх алгоритмів та операцій в програмі, було створено графічний інтерфейс для спрощення всзаємодії з користувачем. Крім того було реалізовано 3D-сцени.

На етапі тестування було визначено, що програма коректно виконує всі операції та видає очікуваний результат.

Результатом курсової роботи є створена програма, яку можна використовувати у розважальних цілях з метою пограти в настільні ігри.

Перелік посилань

1. Альфа-Бета отсечение [Електронний ресурс] // en.wikipedia.org – Режим доступу:

https://en.wikipedia.org/wiki/Alpha%E2%80%93beta\_pruning

1. НОУ ИНТУИТ | Лекция | Методы поиска решений [Електронний ресурс] // intuit.ru – Режим доступу:

http://www.intuit.ru/studies/courses/46/46/lecture/1372?page=3

1. Шахматы [Електронний ресурс] // ru.wikipedia.org – Режим доступу:

https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D1%85%D0%BC%D0%B0%D1%82%D1%8B

1. MSDN [Електронний ресурс] // msdn.microsoft.com – Режим доступу:

https://msdn.microsoft.com/ru-ru/default.aspx

1. Введение в WPF [Електронний ресурс] // msdn.microsoft.com – Режим доступу:

https://msdn.microsoft.com/ru-ru/library/aa970268(v=vs.110).aspx

Додаток А Тексти програмного коду

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення*

Комп’ютені шахи

(Вид носія даних)

*CD-RW*

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

*21 арк, 16461 Кб*

*студентів групи ІП-52 ІІ курсу*

Ващенка Юрія Олександровича

Ільїної Марії Дмитрівни

*“Excel.h”*

#pragma once

#include "Table.h"

#include "Parser.h"

#include "Graph.h"

#include "New.h"

#include "About.h"

#include "Help.h"

#include "Save.h"

#include <string>

#include <ctime>

using std::string;

namespace Excel {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Text;

using namespace System::Drawing;

using namespace System::IO;

/// <summary>

/// Summary for Excel

/// </summary>

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

{

public:

Excel(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~Excel()

{

if (components)

{

delete components;

}

}

private: Table^ table;

private: Graph^ graph;

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

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

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

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

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

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

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

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

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

private: System::Windows::Forms::ToolStripMenuItem^ toolStripMenuItem2;

private: System::Windows::Forms::DataGridView^ dataGridView1;

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

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

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

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

private: System::Windows::Forms::DataGridView^ dataGridView2;

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

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

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

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

/// </summary>

void InitializeComponent(void)

{

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

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

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

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

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

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

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

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

this->toolStripMenuItem2 = (gcnew System::Windows::Forms::ToolStripMenuItem());

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

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

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

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

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

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

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

this->dataGridView2 = (gcnew System::Windows::Forms::DataGridView());

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

this->menuStrip1->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->BeginInit();

this->SuspendLayout();

//

// textBox1

//

this->textBox1->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

this->textBox1->Enabled = false;

this->textBox1->Location = System::Drawing::Point(-1, 24);

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

this->textBox1->Size = System::Drawing::Size(717, 20);

this->textBox1->TabIndex = 1;

this->textBox1->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Excel::textBox1\_KeyPress);

//

// menuStrip1

//

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->fileToolStripMenuItem,

this->helpToolStripMenuItem

});

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

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

this->menuStrip1->Size = System::Drawing::Size(716, 24);

this->menuStrip1->TabIndex = 2;

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

//

// fileToolStripMenuItem

//

this->fileToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(5) {

this->newToolStripMenuItem,

this->openToolStripMenuItem, this->saveToolStripMenuItem, this->latestToolStripMenuItem, this->closeToolStripMenuItem

});

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

this->fileToolStripMenuItem->Size = System::Drawing::Size(37, 20);

this->fileToolStripMenuItem->Text = L"File";

//

// newToolStripMenuItem

//

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

this->newToolStripMenuItem->Size = System::Drawing::Size(105, 22);

this->newToolStripMenuItem->Text = L"New";

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

//

// openToolStripMenuItem

//

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

this->openToolStripMenuItem->Size = System::Drawing::Size(105, 22);

this->openToolStripMenuItem->Text = L"Open";

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

//

// saveToolStripMenuItem

//

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

this->saveToolStripMenuItem->Size = System::Drawing::Size(105, 22);

this->saveToolStripMenuItem->Text = L"Save";

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

//

// latestToolStripMenuItem

//

this->latestToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->toolStripMenuItem2 });

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

this->latestToolStripMenuItem->Size = System::Drawing::Size(105, 22);

this->latestToolStripMenuItem->Text = L"Latest";

//

// toolStripMenuItem2

//

this->toolStripMenuItem2->Enabled = false;

this->toolStripMenuItem2->Name = L"toolStripMenuItem2";

this->toolStripMenuItem2->Size = System::Drawing::Size(114, 22);

this->toolStripMenuItem2->Text = L"No files";

this->toolStripMenuItem2->Click += gcnew System::EventHandler(this, &Excel::toolStripMenuItem2\_Click);

//

// closeToolStripMenuItem

//

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

this->closeToolStripMenuItem->Size = System::Drawing::Size(105, 22);

this->closeToolStripMenuItem->Text = L"Close";

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

//

// helpToolStripMenuItem

//

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

this->functionsToolStripMenuItem,

this->aboutToolStripMenuItem

});

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

this->helpToolStripMenuItem->Size = System::Drawing::Size(44, 20);

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

//

// functionsToolStripMenuItem

//

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

this->functionsToolStripMenuItem->Size = System::Drawing::Size(107, 22);

this->functionsToolStripMenuItem->Text = L"Docs";

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

//

// aboutToolStripMenuItem

//

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

this->aboutToolStripMenuItem->Size = System::Drawing::Size(107, 22);

this->aboutToolStripMenuItem->Text = L"About";

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

//

// dataGridView1

//

this->dataGridView1->AllowUserToAddRows = false;

this->dataGridView1->AllowUserToDeleteRows = false;

this->dataGridView1->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom)

| System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Location = System::Drawing::Point(59, 46);

this->dataGridView1->MultiSelect = false;

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->RowHeadersVisible = false;

this->dataGridView1->Size = System::Drawing::Size(657, 260);

this->dataGridView1->TabIndex = 0;

this->dataGridView1->CellBeginEdit += gcnew System::Windows::Forms::DataGridViewCellCancelEventHandler(this, &Excel::dataGridView1\_CellBeginEdit);

this->dataGridView1->CellEndEdit += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &Excel::dataGridView1\_CellEndEdit);

this->dataGridView1->CellValueChanged += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &Excel::dataGridView1\_CellValueChanged);

this->dataGridView1->CurrentCellChanged += gcnew System::EventHandler(this, &Excel::dataGridView1\_CurrentCellChanged);

this->dataGridView1->Scroll += gcnew System::Windows::Forms::ScrollEventHandler(this, &Excel::dataGridView1\_Scroll);

//

// label1

//

this->label1->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Bottom | System::Windows::Forms::AnchorStyles::Left));

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(6, 309);

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

this->label1->Size = System::Drawing::Size(35, 13);

this->label1->TabIndex = 3;

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

//

// label2

//

this->label2->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Bottom | System::Windows::Forms::AnchorStyles::Left));

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(118, 309);

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

this->label2->Size = System::Drawing::Size(35, 13);

this->label2->TabIndex = 4;

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

//

// dataGridView2

//

this->dataGridView2->AllowUserToAddRows = false;

this->dataGridView2->AllowUserToDeleteRows = false;

this->dataGridView2->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom)

| System::Windows::Forms::AnchorStyles::Left));

this->dataGridView2->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView2->Location = System::Drawing::Point(0, 46);

this->dataGridView2->Name = L"dataGridView2";

this->dataGridView2->RowHeadersVisible = false;

this->dataGridView2->ScrollBars = System::Windows::Forms::ScrollBars::None;

this->dataGridView2->Size = System::Drawing::Size(60, 242);

this->dataGridView2->TabIndex = 5;

this->dataGridView2->CurrentCellChanged += gcnew System::EventHandler(this, &Excel::dataGridView2\_CurrentCellChanged);

//

// label3

//

this->label3->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Bottom | System::Windows::Forms::AnchorStyles::Left));

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(231, 309);

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

this->label3->Size = System::Drawing::Size(0, 13);

this->label3->TabIndex = 6;

//

// Excel

//

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

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

this->ClientSize = System::Drawing::Size(716, 327);

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

this->Controls->Add(this->dataGridView2);

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

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

this->Controls->Add(this->dataGridView1);

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

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

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

this->MainMenuStrip = this->menuStrip1;

this->Name = L"Excel";

this->Text = L"Excel";

this->Closed += gcnew System::EventHandler(this, &Excel::Excel\_Closed);

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

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

/\* Convert System::String to std::wstring

\*/

const int startRows = 50, startCols = 50;

array <String^>^ lastFiles;

int lastFilesNumber = 0;

bool Initialized = 0;

void SaveLastFiles();

void LoadLastFiles();

void UpdateLastFiles(String^ s);

String^ CollumnHeader(int a);

wchar\_t\* toStdWstring(String^ str);

bool WantSave();

void DeleteTable();

void CreateTable(int RowCount, int ColumnCount);

void ReCreateTable(int RowCount, int ColumnCount);

void UpdateText(String^ newString, unsigned int RowIndex, unsigned int CollumnIndex);

private: System::Void Excel\_Closed(Object^ sender, EventArgs^ e);

private: System::Void Excel\_Load(System::Object^ sender, System::EventArgs^ e);

private: System::Void dataGridView1\_CurrentCellChanged(System::Object^ sender, System::EventArgs^ e);

private: System::Void dataGridView1\_CellValueChanged(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e);

private: System::Void textBox1\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e);

private: System::Void dataGridView1\_CellEndEdit(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e);

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

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

private: System::Void dataGridView1\_CellBeginEdit(System::Object^ sender, System::Windows::Forms::DataGridViewCellCancelEventArgs^ e);

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

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

private: System::Void toolStripMenuItem2\_Click(System::Object^ sender, System::EventArgs^ e);

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

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

private: System::Void dataGridView2\_CurrentCellChanged(System::Object^ sender, System::EventArgs^ e);

private: System::Void dataGridView1\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e);

};

};

*“Excel.cpp”*

#include "Excel.h"

#include <direct.h>

namespace Excel

{

bool Excel::WantSave()

{

Save^ form = gcnew Save;

form->ShowDialog();

return (form->Selection == 1);

}

void Excel::SaveLastFiles()

{

char\* appdata = getenv("APPDATA");

char SettingsFile[260];

strcpy(SettingsFile, appdata);

strcat(SettingsFile, "\\Excel");

mkdir(SettingsFile);

strcat(SettingsFile, "\\Settings.dat");

try

{

Stream^ saveFile = gcnew FileStream(gcnew String(SettingsFile), System::IO::FileMode::Create);

saveFile->Write(BitConverter::GetBytes(lastFilesNumber),0,sizeof(lastFilesNumber));

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

{

int length = lastFiles[i]->Length;

saveFile->Write(BitConverter::GetBytes(length), 0, sizeof(int));

for (int k = 0; k < length; k++)

{

saveFile->WriteByte(lastFiles[i][k] >> 8);

saveFile->WriteByte(lastFiles[i][k]);

}

}

}

catch(...)

{

}

}

void Excel::LoadLastFiles()

{

char\* appdata = getenv("APPDATA");

char SettingsFile[260];

strcpy(SettingsFile, appdata);

strcat(SettingsFile, "\\Excel\\Settings.dat");

try

{

Stream^ saveFile = gcnew FileStream(gcnew String(SettingsFile), System::IO::FileMode::Open);

cli::array <unsigned char>^ temp = gcnew cli::array <unsigned char>(sizeof(int));

saveFile->Read(temp, 0, sizeof(int));

lastFilesNumber = BitConverter::ToInt32(temp, 0);

lastFiles->Resize(lastFiles, lastFilesNumber);

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

{

saveFile->Read(temp, 0, sizeof(int));

int length = BitConverter::ToInt32(temp, 0);

wchar\_t\* str = new wchar\_t[length + 1];

for (int k = 0; k < length; k++)

{

wchar\_t s1, s2;

s1 = saveFile->ReadByte();

s2 = saveFile->ReadByte();

s2 = s2 + (s1 << 8);

str[k] = s2;

}

str[length] = L'\0';

lastFiles[i] = gcnew String(str);

}

if (lastFilesNumber>0)

{

latestToolStripMenuItem->DropDownItems[0]->Enabled = true;

}

for (int i = 1; i < lastFilesNumber;i++)

{

System::Windows::Forms::ToolStripMenuItem^ newItem = gcnew System::Windows::Forms::ToolStripMenuItem();

newItem->Click += gcnew System::EventHandler(this, &Excel::toolStripMenuItem2\_Click);

latestToolStripMenuItem->DropDownItems->Add(newItem);

}

int i = lastFilesNumber, j = 0;

while (i > 0)

latestToolStripMenuItem->DropDownItems[j++]->Text = lastFiles[--i];

}

catch (...)

{

}

}

void Excel::UpdateLastFiles(String^ s)

{

int sIndex = -1;

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

{

if (s->Equals(lastFiles[i]))

{

sIndex = i;

break;

}

}

if (sIndex==-1)

{

lastFiles->Resize(lastFiles, ++lastFilesNumber);

lastFiles[lastFilesNumber - 1] = s;

latestToolStripMenuItem->DropDownItems[0]->Enabled = true;

if (lastFilesNumber != 1)

{

System::Windows::Forms::ToolStripMenuItem^ newItem = gcnew System::Windows::Forms::ToolStripMenuItem();

newItem->Click += gcnew System::EventHandler(this, &Excel::toolStripMenuItem2\_Click);

latestToolStripMenuItem->DropDownItems->Add(newItem);

}

}

else

{

String^ temp=lastFiles[0];

int i = sIndex;

while (i<lastFilesNumber-1)

{

lastFiles[i] = lastFiles[i + 1];

i++;

}

lastFiles[lastFilesNumber - 1] = s;

}

int i = lastFilesNumber, j = 0;

while (i > 0)

latestToolStripMenuItem->DropDownItems[j++]->Text = lastFiles[--i];

}

String^ Excel::CollumnHeader(int a)

{

a--;

int l = 1, a2 = a;

while (a2 /= 26) l++;

char\* temp = new char[l + 1];

temp[l] = '\0';

for (int i = l - 1; i >= 0; i--)

{

temp[i] = 'A' + a % 26;

a /= 26;

a--;

}

String^ t = gcnew String(temp);

delete[] temp;

return t;

}

wchar\_t\* Excel::toStdWstring(String^ str) {

wchar\_t\* s = new wchar\_t[str->Length + 1];

array <wchar\_t>^ temp = str->ToCharArray();

for (size\_t i = 0; i < str->Length; i++)

s[i] = temp[i];

s[str->Length] = 0;

return s;

}

void Excel::DeleteTable()

{

dataGridView2->Columns->Remove(dataGridView2->Columns[0]);

int cnt1 = dataGridView1->Columns->Count;

for (int i = 0; i < cnt1; i++)

{

dataGridView1->Columns->Remove(dataGridView1->Columns[0]);

}

int cnt2 = dataGridView1->Rows->Count;

for (int i = 0; i < cnt2; i++)

{

dataGridView1->Rows->Remove(dataGridView1->Rows[0]);

}

}

void Excel::CreateTable(int RowCount, int ColumnCount)

{

Initialized = 0;

int rows = RowCount, cols = ColumnCount;

table = gcnew Table(rows + 1, cols + 1);

graph = gcnew Graph(rows + 1, cols + 1);

dataGridView2->Columns->Add(" ", " ");

dataGridView2->Rows->Add(rows);

dataGridView2->Columns[0]->SortMode = DataGridViewColumnSortMode::NotSortable;

dataGridView1->Columns->Add("", "");

dataGridView1->Columns[0]->Visible = false;

dataGridView1->Rows->Add(rows);

char s1[2] = "A";

dataGridView1->Columns[0]->Width = 60;

dataGridView1->Columns[0]->SortMode = DataGridViewColumnSortMode::NotSortable;

for (int i = 1; i <= rows; i++)

{

dataGridView2->Rows[i - 1]->Cells[0]->Value = Convert::ToString(i);

dataGridView2->Rows[i - 1]->Cells[0]->ReadOnly = true;

}

for (int i = 1; i <= cols; i++)

{

//dataGridView1->Columns->Add(gcnew String(s1), gcnew String(s1));

dataGridView1->Columns->Add(CollumnHeader(i), CollumnHeader(i));

dataGridView1->Columns[i]->SortMode = DataGridViewColumnSortMode::NotSortable;

dataGridView1->Columns[i]->Width = 60;

s1[0]++;

}

dataGridView2->ClearSelection();

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

for (int z = 0; z <= cols; z++)

table[i][z]->setValue(Convert::ToString(dataGridView1->Rows[i]->Cells[z]->Value));

Initialized = 1;

}

void Excel::ReCreateTable(int RowCount, int ColumnCount)

{

DeleteTable();

CreateTable(RowCount, ColumnCount);

}

void Excel::UpdateText(String^ newString, unsigned int RowIndex, unsigned int CollumnIndex)

{

label1->Text = "Processing...";

label2->Text = "";

clock\_t begin = clock();

if (newString == nullptr) {

label1->Text = "Ready.";

label2->Text = "Processed in " + (clock() - begin) / static\_cast <double> (1000) + "s";

return;

}

if (newString->Length == 0)

{

table[RowIndex][CollumnIndex]->setIsFormula(false);

table[RowIndex][CollumnIndex]->setValue(newString);

table[RowIndex][CollumnIndex]->setResult(0.0);

label1->Text = "Ready.";

label2->Text = "Processed in " + (clock() - begin) / static\_cast <double> (1000) + "s";

return;

}

wchar\_t\* value = toStdWstring(newString);

bool isWasFormula = table[RowIndex][CollumnIndex]->getIsFormula();

try {

table[RowIndex][CollumnIndex]->setResult(Parser::parse(wstring(value), table));

table[RowIndex][CollumnIndex]->setIsFormula(true);

graph->changeGraph(table, dataGridView1, RowIndex, CollumnIndex, isWasFormula);

dataGridView1->Rows[RowIndex]->Cells[CollumnIndex]->Value = Convert::ToString(table[RowIndex][CollumnIndex]->getResult());

}

catch (char\* str) {

table[RowIndex][CollumnIndex]->setIsFormula(false);

graph->changeGraph(table, dataGridView1, RowIndex, CollumnIndex, isWasFormula);

String^ temp = gcnew String(str);

dataGridView1->Rows[RowIndex]->Cells[CollumnIndex]->Value = temp;

}

catch (int value) {

table[RowIndex][CollumnIndex]->setIsFormula(false);

graph->changeGraph(table, dataGridView1, RowIndex, CollumnIndex, isWasFormula);

}

delete[] value;

table[RowIndex][CollumnIndex]->setValue(newString);

label1->Text = "Ready.";

label2->Text = "Processed in " + (clock() - begin) / static\_cast <double> (1000) + "s";

}

System::Void Excel::Excel\_Closed(Object^ sender, EventArgs^ e) {

if (label3->Text->Length != 0)

{

if (WantSave())

{

saveToolStripMenuItem\_Click(sender, e);

}

label3->Text = "";

}

SaveLastFiles();

}

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

label1->Text = "Ready.";

label2->Text = "";

CreateTable(startRows, startCols);

dataGridView1->RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode::DisableResizing;

dataGridView2->RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode::DisableResizing;

dataGridView1->AllowUserToResizeRows = false;

dataGridView2->AllowUserToResizeRows = false;

LoadLastFiles();

}

System::Void Excel::dataGridView1\_CurrentCellChanged(System::Object^ sender, System::EventArgs^ e) {

if (!Initialized) return;

int RowIndex = dataGridView1->CurrentCell->RowIndex;

int CollumnIndex = dataGridView1->CurrentCell->ColumnIndex;

try

{

textBox1->Text = table[RowIndex][CollumnIndex]->getValue();

}

catch (...)

{

textBox1->Text = "";

}

if (!dataGridView1->CurrentCell)

{

textBox1->Enabled = false;

textBox1->Text = "";

return;

}

if (dataGridView1->CurrentCell->ColumnIndex == 0)

{

textBox1->Enabled = false;

textBox1->Text = "";

return;

}

textBox1->Enabled = true;

}

System::Void Excel::dataGridView1\_CellValueChanged(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e) {

if (!Initialized) return;

if (!dataGridView1->CurrentCell)

{

textBox1->Enabled = false;

return;

}

if (dataGridView1->CurrentCell->ColumnIndex == 0)

{

textBox1->Enabled = false;

textBox1->Text = "";

}

int RowIndex = dataGridView1->CurrentCell->RowIndex;

int CollumnIndex = dataGridView1->CurrentCell->ColumnIndex;

textBox1->Text = Convert::ToString(dataGridView1->Rows[RowIndex]->Cells[CollumnIndex]->Value);

label3->Text = "Unsaved";

}

System::Void Excel::textBox1\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

if (e->KeyChar == (char)13)

{

int RowIndex = dataGridView1->CurrentCell->RowIndex;

int CollumnIndex = dataGridView1->CurrentCell->ColumnIndex;

dataGridView1->CurrentCell->Value = textBox1->Text;

UpdateText(textBox1->Text, RowIndex, CollumnIndex);

textBox1->Text = Convert::ToString(table[RowIndex][CollumnIndex]->getValue());

label3->Text = "Unsaved";

}

}

System::Void Excel::dataGridView1\_CellEndEdit(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e) {

int RowIndex = dataGridView1->CurrentCell->RowIndex;

int CollumnIndex = dataGridView1->CurrentCell->ColumnIndex;

UpdateText(Convert::ToString(dataGridView1->CurrentCell->Value), RowIndex, CollumnIndex);

textBox1->Text = table[RowIndex][CollumnIndex]->getValue();

}

System::Void Excel::saveToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

SaveFileDialog saveFileDialog1;

saveFileDialog1.Filter = "Table file (\*.adc)|\*.adc";

if (saveFileDialog1.ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

clock\_t begin = clock();

Stream^ saveFile = saveFileDialog1.OpenFile();

if (saveFile != nullptr)

{

try

{

int RowCount = dataGridView1->Rows->Count;

int ColumnCount = dataGridView1->Columns->Count;

saveFile->Write(BitConverter::GetBytes(RowCount), 0, sizeof(int));

saveFile->Write(BitConverter::GetBytes(ColumnCount - 1), 0, sizeof(int));

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

for (int j = 1; j < ColumnCount; j++)

{

Cell^ cell = table[i][j];

String^ str = cell->getValue();

int length = str->Length;

saveFile->Write(BitConverter::GetBytes(length), 0, sizeof(int));

for (int k = 0; k < length; k++)

{

saveFile->WriteByte(str[k] >> 8);

saveFile->WriteByte(str[k]);

}

}

saveFile->Close();

UpdateLastFiles(saveFileDialog1.FileName);

label3->Text = "";

label1->Text = "Ready.";

label2->Text = "Processed in " + (clock() - begin) / static\_cast <double> (1000) + "s";

}

catch (...)

{

label1->Text = "Error.";

label2->Text = "Got error in " + (clock() - begin) / static\_cast <double> (1000) + "s";

MessageBox::Show("Error Processing file", "Error");

}

}

else

{

label1->Text = "Error.";

label2->Text = "Got error in " + (clock() - begin) / static\_cast <double> (1000) + "s";

MessageBox::Show("Error opening file", "Error");

}

}

}

System::Void Excel::openToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

if (label3->Text->Length != 0)

{

if (WantSave())

{

saveToolStripMenuItem\_Click(sender, e);

}

label3->Text = "";

}

Initialized = false;

OpenFileDialog openFileDialog1;

openFileDialog1.Filter = "Table file (\*.adc)|\*.adc";

if (openFileDialog1.ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

clock\_t begin = clock();

Stream^ saveFile = openFileDialog1.OpenFile();

if (saveFile != nullptr)

{

try

{

cli::array <unsigned char>^ temp = gcnew cli::array <unsigned char>(sizeof(int));

saveFile->Read(temp, 0, sizeof(int));

int RowCount = BitConverter::ToInt32(temp, 0);

saveFile->Read(temp, 0, sizeof(int));

int ColumnCount = BitConverter::ToInt32(temp, 0);

int rows = RowCount, cols = ColumnCount;

ReCreateTable(rows, cols);

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

for (int z = 0; z <= cols; z++)

table[i][z]->setValue(Convert::ToString(dataGridView1->Rows[i]->Cells[z]->Value));

textBox1->Width = Width + 1;

Initialized = 1;

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

for (int j = 1; j <= ColumnCount; j++)

{

saveFile->Read(temp, 0, sizeof(int));

int length = BitConverter::ToInt32(temp, 0);

wchar\_t\* str = new wchar\_t[length + 1];

for (int k = 0; k < length; k++)

{

//saveFile->Read(temp, 0, sizeof(wchar\_t));

wchar\_t s1, s2;

s1 = saveFile->ReadByte();

s2 = saveFile->ReadByte();

s2 = s2 + (s1 << 8);

str[k] = s2;

}

str[length] = L'\0';

String^ newString = gcnew String(str);

dataGridView1->Rows[i]->Cells[j]->Value = newString;

try

{

UpdateText(newString, i, j);

}

catch (...)

{

}

delete[] str;

}

saveFile->Close();

UpdateLastFiles(openFileDialog1.FileName);

label1->Text = "Ready.";

label2->Text = "Processed in " + (clock() - begin) / static\_cast <double> (1000) + "s";

}

catch (...)

{

label1->Text = "Error.";

label2->Text = "Got error in " + (clock() - begin) / static\_cast <double> (1000) + "s";

MessageBox::Show("Error Processing file", "Error");

}

}

else

{

label1->Text = "Error.";

label2->Text = "Got error in " + (clock() - begin) / static\_cast <double> (1000) + "s";

MessageBox::Show("Error opening file", "Error");

}

}

}

System::Void Excel::dataGridView1\_CellBeginEdit(System::Object^ sender, System::Windows::Forms::DataGridViewCellCancelEventArgs^ e) {

int RowIndex = e->RowIndex;

int CollumnIndex = e->ColumnIndex;

dataGridView1->CurrentCell->Value = table[RowIndex][CollumnIndex]->getValue();

}

System::Void Excel::newToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

if (label3->Text->Length!=0)

{

if (WantSave())

{

saveToolStripMenuItem\_Click(sender, e);

}

label3->Text = "";

}

Initialized = false;

New^ form = gcnew New;

form->ShowDialog();

if (!form->good) return;

int rows = form->getRows(), cols = form->getCols();

ReCreateTable(rows, cols);

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

for (int z = 0; z <= cols; z++)

table[i][z]->setValue(Convert::ToString(dataGridView1->Rows[i]->Cells[z]->Value));

textBox1->Width = Width + 1;

Initialized = 1;

}

System::Void Excel::aboutToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

About^ form = gcnew About;

form->ShowDialog();

}

System::Void Excel::toolStripMenuItem2\_Click(System::Object^ sender, System::EventArgs^ e) {

if (label3->Text->Length != 0)

{

if (WantSave())

{

saveToolStripMenuItem\_Click(sender, e);

}

label3->Text = "";

}

clock\_t begin = clock();

Initialized = false;

String^ FileName = ((System::Windows::Forms::ToolStripMenuItem^)sender)->Text;

Stream^ saveFile;

try

{

saveFile = gcnew FileStream(FileName, System::IO::FileMode::Open);

}

catch (...)

{

MessageBox::Show("Error Opening File", "Error");

}

if (saveFile != nullptr)

{

try

{

cli::array <unsigned char>^ temp = gcnew cli::array <unsigned char>(sizeof(int));

saveFile->Read(temp, 0, sizeof(int));

int RowCount = BitConverter::ToInt32(temp, 0);

saveFile->Read(temp, 0, sizeof(int));

int ColumnCount = BitConverter::ToInt32(temp, 0);

int rows = RowCount, cols = ColumnCount;

ReCreateTable(rows, cols);

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

for (int z = 0; z <= cols; z++)

table[i][z]->setValue(Convert::ToString(dataGridView1->Rows[i]->Cells[z]->Value));

textBox1->Width = Width + 1;

Initialized = 1;

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

for (int j = 1; j <= ColumnCount; j++)

{

saveFile->Read(temp, 0, sizeof(int));

int length = BitConverter::ToInt32(temp, 0);

wchar\_t\* str = new wchar\_t[length + 1];

for (int k = 0; k < length; k++)

{

//saveFile->Read(temp, 0, sizeof(wchar\_t));

wchar\_t s1, s2;

s1 = saveFile->ReadByte();

s2 = saveFile->ReadByte();

s2 = s2 + (s1 << 8);

str[k] = s2;

}

str[length] = L'\0';

String^ newString = gcnew String(str);

dataGridView1->Rows[i]->Cells[j]->Value = newString;

try

{

UpdateText(newString, i, j);

}

catch (...)

{

}

delete[] str;

}

saveFile->Close();

UpdateLastFiles(((System::Windows::Forms::ToolStripMenuItem^)sender)->Text);

label1->Text = "Ready.";

label2->Text = "Processed in " + (clock() - begin) / static\_cast <double> (1000) + "s";

}

catch (...)

{

label1->Text = "Error.";

label2->Text = "Got error in " + (clock() - begin) / static\_cast <double> (1000) + "s";

MessageBox::Show("Error Processing file", "Error");

}

}

else

{

MessageBox::Show("Error opening file", "Error");

}

}

System::Void Excel::closeToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Close();

}

System::Void Excel::functionsToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Help^ form = gcnew Help;

form->ShowDialog();

}

System::Void Excel::dataGridView2\_CurrentCellChanged(System::Object^ sender, System::EventArgs^ e) {

dataGridView2->ClearSelection();

}

System::Void Excel::dataGridView1\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e) {

dataGridView2->FirstDisplayedScrollingRowIndex = dataGridView1->FirstDisplayedScrollingRowIndex;

dataGridView1->FirstDisplayedScrollingRowIndex = dataGridView1->FirstDisplayedScrollingRowIndex;

}

}

*“About.h”*

#pragma once

namespace Excel {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for About

/// </summary>

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

{

public:

About(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~About()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::PictureBox^ pictureBox1;

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

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

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

/// </summary>

void InitializeComponent(void)

{

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

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

this->SuspendLayout();

//

// pictureBox1

//

this->pictureBox1->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox1.Image")));

this->pictureBox1->InitialImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox1.InitialImage")));

this->pictureBox1->Location = System::Drawing::Point(0, 0);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(645, 484);

this->pictureBox1->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureBox1->TabIndex = 0;

this->pictureBox1->TabStop = false;

this->pictureBox1->Click += gcnew System::EventHandler(this, &About::pictureBox1\_Click);

//

// About

//

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

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

this->ClientSize = System::Drawing::Size(644, 481);

this->Controls->Add(this->pictureBox1);

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

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

this->Name = L"About";

this->Text = L"Authors";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

this->ResumeLayout(false);

}

#pragma endregion

private: System::Void pictureBox1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

};

}

*“Save.h”*

#pragma once

namespace Excel {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for Save

/// </summary>

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

{

public:

Save(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~Save()

{

if (components)

{

delete components;

}

}

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

protected:

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

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

private: System::Windows::Forms::Button^ button2;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

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

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

/// </summary>

void InitializeComponent(void)

{

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

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

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

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

this->button2 = (gcnew System::Windows::Forms::Button());

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label1->Location = System::Drawing::Point(107, 30);

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

this->label1->Size = System::Drawing::Size(147, 18);

this->label1->TabIndex = 0;

this->label1->Text = L"Current file not saved";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label2->Location = System::Drawing::Point(100, 54);

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

this->label2->Size = System::Drawing::Size(162, 18);

this->label2->TabIndex = 1;

this->label2->Text = L"Do you want to save it\?";

//

// button1

//

this->button1->Location = System::Drawing::Point(67, 88);

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

this->button1->Size = System::Drawing::Size(75, 23);

this->button1->TabIndex = 2;

this->button1->Text = L"Yes";

this->button1->UseVisualStyleBackColor = true;

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

//

// button2

//

this->button2->Location = System::Drawing::Point(220, 88);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(75, 23);

this->button2->TabIndex = 3;

this->button2->Text = L"No";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &Save::button2\_Click);

//

// Save

//

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

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

this->ClientSize = System::Drawing::Size(361, 135);

this->Controls->Add(this->button2);

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

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

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

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

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

this->Name = L"Save";

this->Text = L"Save file\?";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

public:

int Selection = 2;

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e);

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

};

}

*“New.h”*

#pragma once

namespace Excel {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for New

/// </summary>

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

{

public:

New(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~New()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::NumericUpDown^ numericUpDown1;

protected:

private: System::Windows::Forms::NumericUpDown^ numericUpDown2;

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

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

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

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

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

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

/// </summary>

void InitializeComponent(void)

{

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

this->numericUpDown1 = (gcnew System::Windows::Forms::NumericUpDown());

this->numericUpDown2 = (gcnew System::Windows::Forms::NumericUpDown());

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

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

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

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown2))->BeginInit();

this->SuspendLayout();

//

// numericUpDown1

//

this->numericUpDown1->Location = System::Drawing::Point(91, 19);

this->numericUpDown1->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 500, 0, 0, 0 });

this->numericUpDown1->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->numericUpDown1->Name = L"numericUpDown1";

this->numericUpDown1->Size = System::Drawing::Size(146, 20);

this->numericUpDown1->TabIndex = 0;

this->numericUpDown1->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 5, 0, 0, 0 });

//

// numericUpDown2

//

this->numericUpDown2->Location = System::Drawing::Point(91, 44);

this->numericUpDown2->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 500, 0, 0, 0 });

this->numericUpDown2->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->numericUpDown2->Name = L"numericUpDown2";

this->numericUpDown2->Size = System::Drawing::Size(146, 20);

this->numericUpDown2->TabIndex = 1;

this->numericUpDown2->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 5, 0, 0, 0 });

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(34, 21);

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

this->label1->Size = System::Drawing::Size(34, 13);

this->label1->TabIndex = 2;

this->label1->Text = L"Rows";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(28, 46);

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

this->label2->Size = System::Drawing::Size(47, 13);

this->label2->TabIndex = 3;

this->label2->Text = L"Columns";

//

// button1

//

this->button1->Location = System::Drawing::Point(100, 73);

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

this->button1->Size = System::Drawing::Size(75, 23);

this->button1->TabIndex = 4;

this->button1->Text = L"Create";

this->button1->UseVisualStyleBackColor = true;

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

//

// New

//

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

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

this->ClientSize = System::Drawing::Size(278, 108);

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

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

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

this->Controls->Add(this->numericUpDown2);

this->Controls->Add(this->numericUpDown1);

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

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

this->Name = L"New";

this->Text = L"New File";

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

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown2))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

public:

bool good;

int getRows();

int getCols();

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

private: System::Void New\_Load(System::Object^ sender, System::EventArgs^ e);

};

}

*“Help.h”*

#pragma once

namespace Excel {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for Help

/// </summary>

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

{

public:

Help(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~Help()

{

if (components)

{

delete components;

}

}

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

protected:

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

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

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

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

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

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

private: System::Windows::Forms::Label^ label8;

private: System::Windows::Forms::Label^ label9;

private: System::Windows::Forms::Label^ label10;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

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

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

/// </summary>

void InitializeComponent(void)

{

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

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

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

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

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

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

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

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

this->label8 = (gcnew System::Windows::Forms::Label());

this->label9 = (gcnew System::Windows::Forms::Label());

this->label10 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(19, 21);

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

this->label1->Size = System::Drawing::Size(212, 13);

this->label1->TabIndex = 0;

this->label1->Text = L"To enter formula start it from \"=\" sign (=1+2)";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(19, 34);

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

this->label2->Size = System::Drawing::Size(220, 13);

this->label2->TabIndex = 1;

this->label2->Text = L"You can use arithmetical operations (+,-,\*,/,^)";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(19, 47);

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

this->label3->Size = System::Drawing::Size(149, 13);

this->label3->TabIndex = 2;

this->label3->Text = L"You can use built-in functions:";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(19, 60);

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

this->label4->Size = System::Drawing::Size(203, 13);

this->label4->TabIndex = 3;

this->label4->Text = L"sin,cos,tan,ctg,ln,exp,asin,aocs,atan,actg";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(19, 73);

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

this->label5->Size = System::Drawing::Size(149, 13);

this->label5->TabIndex = 4;

this->label5->Text = L"All angles should be in radians";

//

// label6

//

this->label6->AutoSize = true;

this->label6->Location = System::Drawing::Point(19, 86);

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

this->label6->Size = System::Drawing::Size(146, 13);

this->label6->TabIndex = 5;

this->label6->Text = L"You can reference other cells";

//

// label7

//

this->label7->AutoSize = true;

this->label7->Location = System::Drawing::Point(19, 99);

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

this->label7->Size = System::Drawing::Size(213, 13);

this->label7->TabIndex = 6;

this->label7->Text = L"To do it use $Cell\_Column$Cel\_Row ($A$1)";

//

// label8

//

this->label8->AutoSize = true;

this->label8->Location = System::Drawing::Point(19, 112);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(142, 13);

this->label8->TabIndex = 7;

this->label8->Text = L"You can save and open files";

//

// label9

//

this->label9->AutoSize = true;

this->label9->Location = System::Drawing::Point(19, 125);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(127, 13);

this->label9->TabIndex = 8;

this->label9->Text = L"Max table size = 500x500";

//

// label10

//

this->label10->AutoSize = true;

this->label10->Location = System::Drawing::Point(19, 169);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(186, 13);

this->label10->TabIndex = 9;

this->label10->Text = L"Khuda, Bulatov && Zarichkoviy © 2016";

//

// Help

//

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

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

this->ClientSize = System::Drawing::Size(257, 194);

this->Controls->Add(this->label10);

this->Controls->Add(this->label9);

this->Controls->Add(this->label8);

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

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

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

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

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

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

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

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

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

this->Name = L"Help";

this->Text = L"Help";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

};

}

*“Cell.h”*

#pragma once

#include "Number.h"

using namespace System;

/\* Class for represenation cell of table

\*/

public ref class Cell {

private:

// Value that contains cell

String^ value;

// Result of formula calculation

Number result;

// Variable that shows is cell contains formula

bool isFormula;

public:

Cell(void);

void setValue(Object^ str);

String^ getValue(void);

Number getResult(void);

void setResult(Number res);

void setIsFormula(bool flag);

bool getIsFormula(void);

};

*“Cell.cpp”*

#include "Cell.h"

Cell::Cell(void) : value(L""), result(0), isFormula(true) {}

void Cell::setValue(Object^ str) {

value = String::Copy(Convert::ToString(str));

}

String^ Cell::getValue(void) {

if (value)

return value;

return gcnew String("");

}

Number Cell::getResult(void) {

if (isFormula) return result;

else throw "#Unable to calc expresion";

}

void Cell::setResult(Number res) {

result = res;

}

void Cell::setIsFormula(bool flag) {

isFormula = flag;

}

bool Cell::getIsFormula(void) {

return isFormula;

}

*“Graph.h”*

#pragma once

#include <cliext\vector>

#include <cliext\set>

#include <cliext\map>

#include "Table.h"

#include "Parser.h"

using cliext::vector;

using cliext::set;

using cliext::map;

using namespace System::Windows::Forms;

using System::String;

/\* Class for representation information in form of graph

\*

\* If some two vertex connected by oriented edge then

\* one of them depend from other (in formula). Orientation

\* show who depend.

\*

\*/

public ref class Graph {

private:

vector < vector<set <unsigned \_\_int64> ^> ^> graph; // Array that contain graph in list form

/\* Return`s list of table cells that used in formula

\*/

void getListOfCells(String^ str, unsigned int height, unsigned int width, vector <unsigned \_\_int64>^ res) {

for (int i = 0, size = str->Length; i < size; i++)

if (str[i] != L'$') continue;

else {

unsigned int y = getY(str, ++i, width);

unsigned int x = getX(str, ++i, height);

res->push\_back(x \* 1ll \* height + y);

}

return;

}

/\* Make topological sort of cells that depend from current cell

\*/

void topologicalSort(unsigned int row, unsigned int column, Table^ table, vector <unsigned \_\_int64>^ res, set <unsigned \_\_int64>^ circles) {

unsigned int height = table->getHeight();

// Set of visited cells

map <unsigned \_\_int64, char>^ visited = gcnew map <unsigned \_\_int64, char>;

dfs(row \* 1ll \* height + column, res, visited, circles, height);

return;

}

/\* Release dfs algorithm for topological sort

\*/

void Graph::dfs(unsigned \_\_int64 v, vector <unsigned \_\_int64>^ res, map <unsigned \_\_int64, char>^ visit, set <unsigned \_\_int64>^ circles, unsigned int& height) {

// Setting cell state as in process

visit[v] = 1;

unsigned int x = v / height;

unsigned int y = v % height;

// Flag that shows statet of processing cell

register char flag;

for (auto iter = graph[x]->at(y)->begin(), end = graph[x]->at(y)->end(); iter != end; iter++) {

flag = visit[\*iter];

switch (flag) {

case 0: dfs(\*iter, res, visit, circles, height); continue;

case 1: circles->insert(\*iter); continue;

case 2: continue;

}

}

// Setting cell state as processed

visit[v] = 2;

// Pushing current cell in order

res->push\_back(v);

}

/\* Conver Y\_index of table cell to digit

\*/

int Graph::getY(String^ str, int& index, unsigned int max) {

unsigned int res = 0;

unsigned int i = index;

while (i < str->Length && str[i] != '$') i++;

if (i == str->Length) throw "#Bad link";

if (i - index > 7) throw "#Index out of range";

while (str[index] != '$')

if (isalpha(str[index]) && toupper(str[index]) == str[index]) res = res \* 26 + str[index++] - 'A' + 1;

else throw "#Bad link";

if (res > max) throw "#Bad link";

return res;

}

/\* Conver X\_index of table cell to digit

\*/

int Graph::getX(String^ str, int& index, unsigned int max) {

int res = 0;

int i = index;

while (i < str->Length && isdigit(str[i])) i++;

if (i - index > 8) throw "#Index out of range";

while (index < str->Length && isdigit(str[index])) res = res \* 10 + str[index++] - '0';

if (res - 1 == -1 || res > max) throw "#Bad link";

return res - 1;

}

public:

Graph(unsigned int height, unsigned int width) {

graph.resize(height);

for (size\_t i = 0; i < height; i++) {

graph[i] = gcnew vector < set <unsigned \_\_int64>^ >;

graph[i]->resize(width);

for (size\_t z = 0; z < width; z++)

graph[i]->at(z) = gcnew set <unsigned \_\_int64>;

}

}

/\* Analiz changes in current cell of table and recalculete values

\* in dependent cells

\*/

void changeGraph(Table^ table, DataGridView^ view, unsigned int row, unsigned int column, bool isWasFormula) {

// Getting all indexs of all cells that used in old and new formula

vector <unsigned \_\_int64>^ oldList = gcnew vector <unsigned \_\_int64>;

if(isWasFormula)

getListOfCells(table[row][column]->getValue(), table->getHeight(), table->getWidth(), oldList);

vector <unsigned \_\_int64>^ newList = gcnew vector <unsigned \_\_int64>;

if(table[row][column]->getIsFormula())

getListOfCells(view->Rows[row]->Cells[column]->Value->ToString(), table->getHeight(), table->getWidth(), newList);

// Delete old depends

for (int i = 0, size = oldList->size(); i < size; i++)

graph[oldList[i] / table->getHeight()]->at(oldList[i] % table->getHeight())->erase(row\*table->getHeight() + column);

// Adding new depends

for (int i = 0, size = newList->size(); i < size; i++)

graph[newList[i] / table->getHeight()]->at(newList[i] % table->getHeight())->insert(row\*table->getHeight() + column);

// Getting dependent cells in topological order

vector <unsigned \_\_int64>^ order = gcnew vector <unsigned \_\_int64>;

// Adding cells where start and end circle

set <unsigned \_\_int64>^ circles = gcnew set <unsigned \_\_int64>;

topologicalSort(row, column, table, order, circles);

// Recalculation of dependent cells

for (int i = order->size() - 1; i >= 0; i--) {

unsigned int x = order[i] / table->getHeight();

unsigned int y = order[i] % table->getHeight();

// If we have a circle in this point

if (circles->find(order[i]) != circles->end()) {

view->Rows[x]->Cells[y]->Value = Convert::ToString(L"#Unable to calculate");

table[x][y]->setIsFormula(false);

continue;

}

// We already caclculate value of this (start) cell

if (i == order->size() - 1) continue;

wchar\_t\* input = toStdWstring(table[x][y]->getValue());

try {

table[x][y]->setResult(Parser::parse(input, table));

table[x][y]->setIsFormula(true);

view->Rows[x]->Cells[y]->Value = Convert::ToString(table[x][y]->getResult());

}

catch (char\* str) {

String^ temp = gcnew String(str);

view->Rows[x]->Cells[y]->Value = temp;

table[x][y]->setIsFormula(false);

}

catch (int value) {

table[x][y]->setIsFormula(false);

}

}

}

/\* Convert System::String to std::wstring

\*/

wchar\_t\* toStdWstring(String^ str) {

wchar\_t\* s = new wchar\_t[str->Length + 1];

array <wchar\_t>^ temp = str->ToCharArray();

for (size\_t i = 0; i < str->Length; i++)

s[i] = temp[i];

s[str->Length] = 0;

return s;

}

};

*“Parser.h”*

#pragma once

#include "Number.h"

#include "Table.h"

#include <string>

#include <stack>

using std::wstring;

using std::stack;

class Parser {

private:

static wstring shuntingYard(const wstring& input);

inline static unsigned int opPrior(const wchar\_t& ch);

inline static bool isElemOper(const wchar\_t& ch);

inline static int getX\_index(wstring& str, int index, unsigned int max);

inline static int getY\_index(wstring& str, int index, unsigned int max);

inline static bool strcmp(wstring& str, int index, int element);

static void processNumbers(const wstring& input, wstring& output, unsigned int& i);

static void processCloseBracket(const wstring& input, wstring& output, stack<wstring>& aStack, unsigned int& i);

static void processElemOperations(const wstring& input, wstring& output, stack<wstring>& aStack, unsigned int& i);

static void processLink(const wstring& input, wstring& output, unsigned int& i);

static void processFunctions(const wstring& input, wstring& output, stack<wstring>& aStack, unsigned int& i);

static void calculateNumber(wstring& output, stack<Number>& aStack, size\_t& i);

static void calculateLink(Table^ table, wstring& output, stack<Number>& aStack, size\_t& i);

static void calculateElemOperations(wstring& output, stack<Number>& aStack, size\_t& i);

static void calculateFunctions(wstring& output, stack<Number>& aStack, size\_t& i);

static int searchAssigmentSymbol(const wstring& input);

static Number caseFuction(Number& top, int index);

static Number isOnlyOneDigit(const wstring& input, size\_t i);

public:

static Number parse(const wstring& input, Table^ table);

};

*“Parser.cpp”*

#include "Parser.h"

#include <math.h>

#define CODE\_OF\_STRING 0 // Defining code of simple string (no formula)

#define CODE\_OF\_EMPTY\_STRING 1 // Defining code of empty input string

#define pression 4 // Number of symbols after point

#define eps 1e-7 // Number zero

#define M\_PI 3.14159265358979323846 // pi

#define numOfFunc sizeof(func)/sizeof(char \*)

const wstring func[] = {

L"sin",

L"cos",

L"tan",

L"ctg",

L"ln",

L"exp",

L"asin",

L"acos",

L"atan",

L"actg"

};

using std::stack;

using System::Convert;

/\* Rewrite expresion in postfix notation

\* using Shunting-yard algorithm

\*/

wstring Parser::shuntingYard(const wstring& input) {

// Stack for operations and functions

stack<wstring> aStack;

// Output string in postfix notation

wstring output(L"");

unsigned int length = input.length();

unsigned int i = 0;

while(i < length) {

// Skip space symbols

while (isspace(input[i])) i++;

if (i == length) break;

// Process number

if (isdigit(input[i]))

processNumbers(input, output, i);

// Process open bracket

else if (input[i] == L'(')

aStack.push(wstring(L"("));

// Process close bracket

else if (input[i] == L')')

processCloseBracket(input, output, aStack, i);

// If elementary operation +, -, \*, /, ^

else if (isElemOper(input[i]))

processElemOperations(input, output, aStack, i);

// Process link

else if (input[i] == L'$')

processLink(input, output, i);

// Process functions

else if (isalpha(input[i]))

processFunctions(input, output, aStack, i);

// invalid input format

else throw "#Invalid input format";

i++;

}

while (!aStack.empty()) {

output += aStack.top() += L" ";

aStack.pop();

}

return output;

}

/\* Cheaching elementary operation priority

\*/

unsigned int Parser::opPrior(const wchar\_t& ch) {

switch (ch){

case '^':

return 3;

case '\*':

case '/':

return 2;

case '+':

case '-':

return 1;

}

}

/\* Function for parsing math expresion

\*/

Number Parser::parse(const wstring& input, Table^ table) {

int index;

try {

index = searchAssigmentSymbol(input);

}

catch (Number x) {

return x;

}

// Deleting all symbols in [0; index]

wstring simpleInput = input.substr(index, input.size() - index + 1);

wstring output = shuntingYard(simpleInput);

stack <Number> aStack;

size\_t i = 0;

size\_t length = output.size();

while (i < length) {

if (isspace(output[i])) i++;

if (i == length) break;

// Calculate number and push it in the stack

else if (isdigit(output[i]) || (i + 1 < length && output[i] == L'-' && isdigit(output[i+1])))

calculateNumber(output, aStack, i);

// Select from table number from link and push it in the stack

else if (output[i] == L'$' || (i + 1 < length && output[i] == L'-' && output[i+1] == L'$'))

calculateLink(table, output, aStack, i);

// Release elementary operations with stack elements

else if (isElemOper(output[i]))

calculateElemOperations(output, aStack, i);

else calculateFunctions(output, aStack, i);

}

if (aStack.empty()) throw "#Bad expresion";

// Cat more that pression symbols after point

return roundl(aStack.top() \* pow(static\_cast<Number> (10), pression)) / pow(static\_cast<Number> (10), pression);

}

/\* Cheacking is operation is elementary (+, -, \*, /, ^)

\*/

bool Parser::isElemOper(const wchar\_t& ch) {

return (ch == '+') || (ch == '-') || (ch == '\*') || (ch == '/') || (ch == '^');

}

/\* Conver Y\_index of table cell to digit

\*/

int Parser::getY\_index(wstring& str, int index, unsigned int max) {

int res = 0;

int i = index;

while (i < str.length() && str[i] != '$') i++;

if (i == str.length()) throw "#Bad link";

if (i - index > 7) throw "#Index out of range";

while (str[index] != '$')

if(isalpha(str[index]) && toupper(str[index]) == str[index]) res = res \* 26 + str[index++] - 'A' + 1;

else throw "#Bad link";

if(res > max) throw "#Bad link";

return res;

}

/\* Conver X\_index of table cell to digit

\*/

int Parser::getX\_index(wstring& str, int index, unsigned int max) {

int res = 0;

int i = index;

while (i < str.length() && isdigit(str[i])) i++;

if (i - index > 8) throw "#Index out of range";

while (index < str.length() && isdigit(str[index])) res = res \* 10 + str[index++] - '0';

if(res - 1 == -1 || res > max) throw "#Bad link";

return res - 1;

}

/\* Compare two definishions of functions

\*/

bool Parser::strcmp(wstring& str, int index, int element) {

int i = 0;

for (int size = str.length(); func[element][i] && i + index <= size; i++)

if (func[element][i] != str[i + index]) return false;

if (func[element][i]) return false;

else return true;

}

/\* Processing nubers literals in input string

\*/

void Parser::processNumbers(const wstring& input, wstring& output, unsigned int& i) {

unsigned int length = input.size();

// Adding sign if this nessasary

if (input[i] == '-') {

output += L'-';

while(i < length && !isdigit(input[i])) i++;

if (i == length) throw "#Bad number";

}

// Variable that show is point read already

bool isPoint = false;

while (i < length && (isdigit(input[i]) || input[i] == L'.')) {

if (input[i] == '.') {

// If adding point for this number not at first time

if (isPoint) throw "#Bad number";

output += '.';

isPoint = true;

}

else output += input[i];

i++;

}

output += ' ';

i--;

}

/\* Processing close bracket

\*/

void Parser::processCloseBracket(const wstring& input, wstring& output, stack<wstring>& aStack, unsigned int& i) {

while (!aStack.empty() && aStack.top()[0] != '(') {

output += aStack.top() += ' ';

aStack.pop();

}

if (aStack.empty()) throw "#No close bracket";

else aStack.pop();

}

/\* Porcessing elementary operations

\*/

void Parser::processElemOperations(const wstring& input, wstring& output, stack<wstring>& aStack, unsigned int& i) {

if (input[i] == L'-')

if (output.empty() || (!aStack.empty() && aStack.top()[0] == L'(')) {

output += L'-';

return;

}

while (!aStack.empty() && isElemOper(aStack.top()[0]) && (opPrior(aStack.top()[0]) >= opPrior(input[i]))) {

output += aStack.top() += ' ';

aStack.pop();

}

aStack.push(wstring(L"") += input[i]);

}

/\* Processing links

\*/

void Parser::processLink(const wstring& input, wstring& output, unsigned int& i) {

size\_t length = input.size();

output += '$';

while (++i < length && input[i] != '$')

if (input[i] < 'A' || input[i] > 'Z')

throw "#Bad link name";

else output += input[i];

if (i == length) throw "#Bad link name";

else output += input[i++];

while (i < length && isdigit(input[i]))

output += input[i++];

output += ' ';

i--;

}

/\* Processing functions

\*/

void Parser::processFunctions(const wstring& input, wstring& output, stack<wstring>& aStack, unsigned int& i) {

wstring func;

size\_t length = input.size();

while (i < length && isalpha(input[i]))

func += input[i++];

// Skip space symbols

while (i < length && isspace(input[i])) i++;

if (i == length || input[i] != '(') throw "#No args for function";

aStack.push(func);

aStack.push(wstring(L"("));

}

/\* Calulate number and pushing it in the

\*/

void Parser::calculateNumber(wstring& output, stack<Number>& aStack, size\_t& i) {

size\_t index = 0;

aStack.push(std::stod(output.substr(i), &index));

i += index;

}

/\* Select from table number from link and push it in the stack

\*/

void Parser::calculateLink(Table^ table, wstring& output, stack<Number>& aStack, size\_t& i) {

int koef;

if (output[i] == L'-') {

koef = -1;

i++;

}

else koef = 1;

int yIndex = getY\_index(output, ++i, table->getWidth());

while (output[i] != '$') i++;

int xIndex = getX\_index(output, ++i, table->getHeight());

while (isdigit(output[i])) i++;

if (table->getHeight() - 1 <= xIndex || table->getWidth() <= yIndex) throw "#Bad link";

aStack.push(koef \* table[xIndex][yIndex]->getResult());

}

/\* Release elementary operations with stack elements

\*/

void Parser::calculateElemOperations(wstring& output, stack<Number>& aStack, size\_t& i) {

if (aStack.size() < 2) throw "#No arguments";

Number y = aStack.top();

aStack.pop();

Number x = aStack.top();

aStack.pop();

switch (output[i]) {

case '+': aStack.push(x + y); break;

case '-': aStack.push(x - y); break;

case '\*': aStack.push(x \* y); break;

case '/': {

if (abs(y - 0) < eps)

throw "#Division by zero";

else

aStack.push(x / y);

break;

}

case '^': aStack.push(pow(x, y)); break;

}

i++;

}

/\* Calculate funtions

\*/

void Parser::calculateFunctions(wstring& output, stack<Number>& aStack, size\_t& i) {

if (aStack.size() < 1) throw "#No arguments";

bool flag = false;

for (int z = 0; z < numOfFunc; z++)

if (strcmp(output, i, z)) {

flag = true;

Number t = caseFuction(aStack.top(), z);

// Deliting argument of function

aStack.pop();

// Pusing result

aStack.push(t);

// Set index pointer after funtion name

i += func[z].size();

break;

}

if (!flag) throw "#Wrong function name";

}

/\* Searching index of assigment symbol in input string

\*/

int Parser::searchAssigmentSymbol(const wstring& input) {

if (input[0] == '\0') throw CODE\_OF\_EMPTY\_STRING;

try {

Number x = isOnlyOneDigit(input, 0);

throw x;

}

catch (int x) {

// Noting to do

}

// Searching sybol '='

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

if (!isspace(input[i]))

if (input[i] == L'=') return i + 1;

else throw CODE\_OF\_STRING; // if first symbol is not space or '=' then this is a general string

}

/\* Computing function value

\*/

Number Parser::caseFuction(Number& top, int index) {

switch (index) {

// sin

case 0: return sin(top);

// cos

case 1: return cos(top);

// tan

case 2: {

if (abs(top - M\_PI / 2) < eps)

throw "#Infinity";

else

return tan(top);

}

// ctg

case 3: {

if (abs(top - 0) < eps)

throw "#Infinity";

else

return static\_cast<Number> (1) / tan(top);

}

// ln

case 4: {

// if in segment (- inf; 0]

if (top < eps)

throw "#Ln invalid argument";

else

return log(top);

}

// exp

case 5: return exp(top);

// Asin

case 6: {

if (abs(top) - 1 > eps)

throw "#Asin indvalid arguments";

else

return asin(top);

}

// Acos

case 7: {

if (abs(top) - 1 > eps)

throw "#Acos indvalid arguments";

else

return acos(top);

}

// Atan

case 8: return atan(top);

// Actg

case 9: return M\_PI/static\_cast<Number> (2) - atan(top);

default: throw "#Bad function name";

}

}

/\* Definition is input string consist from one number

\* Return number or throw exception if not one number

\*/

Number Parser::isOnlyOneDigit(const wstring& input, size\_t i) {

size\_t index = 0;

size\_t length = input.size();

while (i < length && isspace(input[i])) i++;

if (input[i] != L'-' && !isdigit(input[i])) throw 0;

int koef;

if (input[i] == L'-') {

koef = -1;

i++;

}

else koef = 1;

while (i < length && isspace(input[i])) i++;

if (!isdigit(input[i])) throw 0;

Number t = std::stod(input.substr(i), &index);

i += index;

while (i < length && isspace(input[i])) i++;

if (i != length) throw 0;

else return t \* koef;

}

*“Table.h”*

#pragma once

#include "Cell.h"

#include <cliext\vector>

using cliext::vector;

/\* Class for representation of table

\*/

public ref class Table {

private:

// Demensions of table

unsigned int height;

unsigned int width;

// Table

vector < vector <Cell^>^ > table;

public:

Table(unsigned int height, unsigned int width);

// Getters of values

unsigned int getHeight(void);

unsigned int getWidth(void);

vector <Cell^>^ operator[] (unsigned int index);

// Changers of demensions

void changeHeight(int delta);

void changeWidth(int delta);

};

*“Table.cpp”*

#include "Table.h"

/\* Constructor of class Table

\* Input: Takes height and width of created/loaded table

\* Allocate memory for table

\*/

Table::Table(unsigned int height, unsigned int width) {

this->height = height;

this->width = width;

table.resize(height);

for (unsigned int i = 0; i < height; i++) {

table[i] = gcnew vector <Cell^>(width);

for (unsigned int z = 0; z < width; z++)

table[i][z] = gcnew Cell;

}

}

/\* Getter for height atribute

\*/

unsigned int Table::getHeight(void) {

return height;

}

/\* Getter for width atribute

\*/

unsigned int Table::getWidth(void) {

return width;

}

/\* Overload operation [] for direct access to elements

\*/

vector <Cell^>^ Table::operator[] (unsigned int index) {

return table[index];

}

/\* Change height to some delta

\*/

void Table::changeHeight(int delta) {

height += delta;

}

/\* Change width to some delta

\*/

void Table::changeWidth(int delta) {

width += delta;

}