НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматики та управляння в технічних системах

Реалізація програми «Гра лабіринт “Maze”»

Курсова робота

З дисципліни «Об’єктно-орієнтоване програмування»

|  |  |
| --- | --- |
| Керівник  Хмелюк М.С.  «Допущений до захисту»    (Особистий підпис керівника)  « \_\_\_\_ » 2018 р.  Захищений з оцінкою    (оцінка)  Члени комісії:    (Особистий підпис)    (Особистий підпис) | Виконавець  ст. Макіян С.А.  зал. Книжка № ЗПІ-зп6115  гр. ЗПІ-ЗП 61    (Особистий підпис виконавця)  « \_\_\_\_ » 2018 р.    (Розшифровка підпису)    (Розшифровка підпису) |

Київ - 2018

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра АВТОМАТИКИ ТА УПРАВЛЯННЯ В ТЕХНІЧНИХ СИСТЕМАХ

Дисципліна «Об’єктно-орієнтоване програмування»

Курс 2 Група ЗПІ-зп61 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Макіян Смбат Артурович |
| (прізвище, ім’я, по батькові) |

|  |  |
| --- | --- |
| 1. Тема роботи | Гра лабіринт «Maze» |
|  | |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи |  |

|  |
| --- |
| 3. Вихідні дані до роботи: |
| Операційна система Windows 8.1 та вище, платформа x86\_64. Мова прогрумавння та фреймворк C#, .Net v4, Windows Forms. База даних SQLite v3. |
|  |
|  |

|  |
| --- |
| 4. Зміст розрахунково-пояснювальної записку (перелік питань, які підлягають розробці) |
| *1.Вступ. 2. Постановка задачі. 3. Огляд існуючих рішень та опис моделі.* |
| *4. Режим роботи та опис діалогу. 5. Опис програми. 6. Інструкція програміста.* |
| *7. Керівництво користувача 8. Висновки.* |
| *9. Список використаних джерел.* |
| ***Додатки:*** |
| *10.Текст програми. 11. Результати працездатності програми.* |

|  |
| --- |
| 5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) |
| Діаграм класів, діаграма зв’язків підмодулей програми |
|  |
|  |
|  |

|  |  |
| --- | --- |
| 6. Дата видачі завдання | 21.10.2017 |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Назва етапів виконання курсової роботи** | **Строк виконання етапів роботи** | **Підписи або примітки** |
| 1. | Отримання та узгодження теми курсової роботи | 21.10.2017 |  |
| 2. | Огляд існуючих рішень з тематики роботи | 01.11.2017 |  |
| 3. | Розробка моделі та структури програми | 15.11.2017 |  |
| 4. | Вибір алгоритмів розв’язання задачі | 01.12.2017 |  |
| 5. | Розробка діалогового інтерфейсу програми | 15.12.2017 |  |
| 6. | Кодування програми | 25.12.2017 |  |
| 7. | Налагодження та перевірка програми | 10.01.2018 |  |
| 8. | Оформлення пояснювальної записки | 01.02.2018 |  |
| 9. | Захист курсової роботи | 01.03.2018 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент** |  |  |  |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Керівник** |  |  |  |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2018 р.

# АНОТАЦІЯ

В курсовій роботі розглянуто та описано роботу гри лабіринт з точки зору об’єктно-орієнтованого програмування. Основні сутності дослідженої предметної області описані як класи, між якими встановлені відповідні зв’язки. Програма розроблена об’єктно-орієнтованою мовою програмування C# в середовищі розробки Microsoft Visual Studio 2017. Використано шаблон Windows Forms Application, що дало змогу розробити дружній інтерфейс інформаційної системи. Вхідні та вихідні дані зберігаються в XML-файл та в базу даних SQLite v3.

Пояснювальна записка виконана на 48-х сторінках, містить 2 додатки, 8 рисунків.

ЗМІСТ

[АНОТАЦІЯ 4](#_Toc507663192)

[ВСТУП 7](#_Toc507663193)

[1 ПОСТАНОВКА ЗАДАЧІ 8](#_Toc507663194)

[2 МАТЕМАТИЧНАЯ МОДЕЛЬ ТА СТРУКТУРА ПРОГРАМИ 10](#_Toc507663195)

[3 ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ 11](#_Toc507663196)

[4 СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ 12](#_Toc507663197)

[4.1 Клас MazeInfo. 12](#_Toc507663198)

[4.2 Клас BaseNode. 12](#_Toc507663199)

[4.3 Клас FinishNode 13](#_Toc507663200)

[4.4 Клас UserNode 13](#_Toc507663201)

[4.5 Клас MazeLogic 13](#_Toc507663202)

[4.6 Клас MazeNode 14](#_Toc507663203)

[4.7 Клас MazeLink 14](#_Toc507663204)

[4.8 DB\_logics 15](#_Toc507663205)

[5 ОПИС ПРОГРАМИ 16](#_Toc507663206)

[5.1 Модульна структура 16](#_Toc507663207)

[5.2 Виклик і завантаження 16](#_Toc507663208)

[5.3 Повідомлення 17](#_Toc507663209)

[6 ІНСТРУКЦІЯ ПРОГРАМІСТА 18](#_Toc507663210)

[6.1 Загальні відомості 18](#_Toc507663211)

[6.2 Умови проектування програми 18](#_Toc507663212)

[6.3 Керівництво розробника програми 18](#_Toc507663213)

[6.4 Повідомлення підчас розробки 19](#_Toc507663214)

[6.4.1 Група повідомлень: 19](#_Toc507663215)

[6.4.2 Повідомлення: 20](#_Toc507663216)

[6.4.3 Повідомення: 20](#_Toc507663217)

[7 КЕРІВНИЦТВО КОРИСТУВАЧА 21](#_Toc507663218)

[7.1 Загальні відомості 21](#_Toc507663219)

[7.2 Умови застосування 21](#_Toc507663220)

[7.3 Характеристики програми 21](#_Toc507663221)

[7.4 Послідовність дій користувача програмою 21](#_Toc507663222)

[ВИСНОВКИ 26](#_Toc507663223)

[ПЕРЕЛІК ВИКОРИСТОВАНОЇ ЛІТЕРАТУРИ 27](#_Toc507663224)

[ДОДАТОК А. ТЕКСТ ПРОГРАМИ 28](#_Toc507663225)

[ДОДАТОК Б. РЕЗУЛЬТАТ ПРАЦЕЗДАТНОСТІ ПРОГРАМИ 43](#_Toc507663226)

# ВСТУП

Мета роботи – зробити гру лабіринт, котру користувач може пройти з клавіатури за допомогою стрілок вверх, вниз, вліво, вправо. Якщо користувач бажає покинути гру, то рівень має бути збережений до xml файлу. На прикинці гри користувач має мати право зберегти результати своєї гри в базі. Рівень має представляти з себе матрицю 29х29 елементів, де по периметру мають бути стіни, також на старті та на кінці має бути клітині місцезнаходження користувача та клітина фінішу.

При програмуванні застосовувалися основні методи об’єктно орієнтовного програмування [1]. Для збереження результатів в xml доцільно серіалізувати/десеріалізувати рівні через System.Runtime.Serialization[2]. Також програма має обробляти різні сітуації, в котрих користувач може зробити помилку: пусте ім’я рівня, невірні координати для елементу. Програма має бути написана на мові C# та використовувати фреймворка .Net версії 4.0. Для цієї мови доцільно вести розробку в середовищі Visual Studio (остання версія на момент написання 2017)[3]. Для зберігання результатів проходження доцільно використать базу даних формату SQLite v3. [4]

# ПОСТАНОВКА ЗАДАЧІ

Для створення графічного інтерфейсу користувача потрібно використати framework Windows Forms. Програма має складатись заставки; головного вікна, в котрому можна почати гру, або подивитись статистику. Якщо користувач відкрив вікно початку гри, то він або продовжує попередню гру, котру біло збережено, або починає нову гру. В вікні статистики користувач може переглянути 5 найкращів результатів з проходження гри (див. Рисунок 1.1). При проходженні гри користучав має чути звуки відповідно своїм діям: звук кінця гри, або звук переходу на сосідню клітину.

Також доцільно розділити програму на UI та бізнес логіку: мати окрему бібліотеку з класами для елементів, а графічний інтерфейс скомпілювати до виконавчого файлу.



Рисунок 1.1 – Блок-схема роботи програми

# МАТЕМАТИЧНАЯ МОДЕЛЬ ТА СТРУКТУРА ПРОГРАМИ

Результатом роботи програми має бути гра, з формою рівня, в котрої дані про рівень мають бути згенеровані програмним алгоритмом, та зберігатись в виді матриці елементів 29х29. Також рівень має зберігати поточні координати користувача та час, котрий користувач вже використав на проходження рівня .

При генерації рівня будуть використані допоміжні класи: з’єднання між пустими клітинами та елементи клітини. На цьому етапі рівень буде розглядатися як матриці 14х14. Пуста клітин представляє з себе структуру, котра зберігає в собі з’єднання з сусідніми елементами, до котрих користувач може потрапити з цієї клітини. Для підвищення швидкості, після генерації лабіринту дані потрапляють то матриці 29х29 де пусти клітини зберігаються як 0, а стіни як 1. Дані про місцезнаходження користувача доцільно зберігати в окремому масиві з двох елементів (див. Рисунок 2.1). Зв’язок допоміжних класів (див. Рисунок 2.2).



Рисунок 2.1 – Зв’язок один до багатьох



Рисунок 2.2 – Зв’язок один до багатьох

# ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ

Програма починається з заставки (з назвою програми, картинкою в супроводі зі звуковим привітанням), після котрої користувач бачить  діалогу старту програми. В цьому діалозі має бути наступний функціонал:

* Кнопка почати гру (Button для відкриття діалогу гри);
* Кнопка статистику (Button для відкриття діалогу статистики);

Діалог гри:

* кнопка перезапуску гри (Button);
* кнопка виходу з гри(стандартна кнопка діалогу).

Перегляд статистики елементу має бути в окремому діалозі статистики, де користувач нічого не може редагувати. В цьому діалозі користувачу має бути доступна наступна інформація:

* Ім’я користувача та час проходження (label);

Діалог гри:

* Кнопка перезапуску рівня (Button);
* Панель що зберігає елементи рівня (Panel);
* Підказку як керувати грой (Label).

Діалог вводу даних на прикінці:

* заголовок вікна (Label);
* поле для вводу імені (TextBox);
* кнопка збереження (Button);
* інформація про пройдений час (Label).

# СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ

Дані програми можна розділити на дві категорії класів: проміжні класи для генерації масиву, та класи рівня. Також окремо можна поставити клас для роботи с базою Даних. Серед проміжних класів MazeLink, Maze Node. Класи рівня представлені: BaseNode, FinishNode, UserNode, MazeInfo, MazeLogic.

## Клас MazeInfo.

Призначен для зберігання даних про рівень. Зберігає ім’я рівня (string Name), його унікальний ідентифікатор (Guid Id), та список елементів (List< BaseGameElement> elements), також має статичну змінну static Level tempLevel для збереження тимчасових рівней. Зберігає статус проходження(Boolean completed), координати користувача(int[] userPos), матрицю рівня(int [][]gridFinal), час затрачений на проходження рівня (TimeSpan timespent)

Не має методів, але інкапсуляція реалізована зкриттям для MazeMain.

## Клас BaseNode.

Призначен для збереження даних про стіни. Є базовим класом для всіх елементів (FinishNode, UserNode). Зберігає в собі ім’я елементу (string Name), тип статичності (Boolean IsStatic), начальну позицію (Point InitialPoint), та ідентифікатор рівня (Guid LevelId). Зберігає в собі ссилку на батьківський елемент форми (Panel m\_parent), ссилку на елемент стіни (Label m\_label), та координати стіни в матриці ( int m\_x, my).

Має наступні методи:

* конструктор BaseNode(Panel parent, int x, int y);
* публічний getter для полей m\_x, m\_y (Point GetXY());
* публічний віртуальний getter для m\_label (Label GetLabel());
* публічний віртуальний метод SetNewLocation (int x, int y), котрий генерує помилку, якщо його визвати (реалізован для наслідників).

## Клас FinishNode

Є нащадком BaseNode, та батьком для классів UserNode. Не має ніяких додаткових полів.

Має наступні методи:

* конструктор FinishNode (Panel parent, int x, int y).

## Клас UserNode

Є нащадком FinishNode.

Має наступні методи:

* конструктор UserNode (Panel parent, int x, int y);
* перегружний метод метод SetNewLocation (int x, int y).

## Клас MazeLogic

Призначен для керування логікою керування лабіринтом. Має наступні поля: публічні статичні NORTH, EAST,SOUTH,WEST (для зберігання напрямку до сусдньої клітини); константний приватний int GRID\_SIZE (розмір сітки), логічні приватні HORIZONTAL, VERTICAL (потрібні для роботи алгоритму генерації); поле часу початка гри (DateTime startTime); логічне приватне поле показчик чи вдалося відновити рівень з xml файлу (Boolean loadSuccessFully); приватну ссилку на панель, в котру треба добавляти елементи (Panel m\_parent), приватні силки на клітину користувача та кінця (BaseNode m\_finish, UserNode m\_panel); приватну ссилку на MazeInfo mazeInfo.

Має наступні методи:

* public MazeLogic(Panel panel) (конструктор);
* public Boolean IsLoadSuccessfully() (публічний getter на loadSuccessFully);
* private object Load(string fileName, Type t) (метод для дереалізації рівня);
* public void Save() (метод для серілізації тазбереження рівня);
* public void loadMaze() (загрузити десеріалізованний рівень);
* public void createMaze() (згенерувати рівень з нуля);
* public bool checkIsEnd() (метод на перевірку кінця гри);
* public int GetSpentTime() (метод для отрімання часу гри);
* public Boolean makeMove(string direction) (метод для переміщення користувача на сітці);
* private void drawMaze() (методя дл вібрісовки);
* private void genMazeNewWay() (метод генерації);
* private MazeNode[,] MakeNodes(int wid, int hgt) (метод для генерації допоміжних об’єктів);
* private void FindSpanningTree(MazeNode root) (метод для створення звв’язків в лабіринті);
* private void makeDraftGrid() (метод для генерації пустого остаточного ігрового поля);
* private void DisplayMaze(MazeNode[,] nodes) (метод для відрі совки стін);
* private void DrawWalls(MazeNode node, int x, int y) (метод безпосередню відрі совки стін в вузла);
* private void DrawWall(MazeNode node, int side, int x, int y) (метод відрісовки однієї стіни з вузла).

## Клас MazeNode

Допоміжний клас для генерації лабіринту. Має наступні поля: масив сусідів (public MazeNode[] Neighbors), ссилку на попередній елемент (public MazeNode Predecessor).

Має власний пустий конструктор.

## Клас MazeLink

Допоміжний клас для генерації лабіринту. Має наступні поля: силку між MazeNode FromNode, ToNode.

Має власний пустий конструктор в котрому зберігає ссилки на сусідні вузли.

## DB\_logics

Об’єкт за допомогою котрого відбувається звернення до бази даних. Має єдине поле string dbName в котрому зберігається ім’я бази даних.

Має наступні методи:

* Публічний конструктор, в котрому створюється база даних, якщо її не має;
* Публічний мето insertWinner для запису користувача до списку переможців;
* Публічний метод string ReadTop5Winners для того щоби отримати перших 5 переможців.

## Структура бази даних.

База даних представлена SQLite базой даних “” єдиною таблицею «highscores», з текстовим полем name (максимальна довжина 20 символів), та числового поля time.

# ОПИС ПРОГРАМИ

## Модульна структура

Програма створена з двох окремих компнентів Makiyan\_Cursovaya\_sem2 та Makiyan\_cursovaya\_sem2.Data. В першому компоненті знаходяться форми, у другому класи та бізнес логіка (див. Рисунок 5.1).

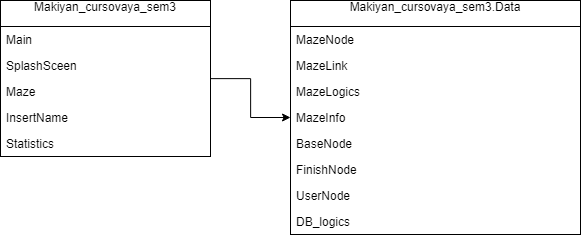


Рисунок 5.1 – Взаємодія модулів

## Виклик і завантаження

Точкою завантаження програми є функція public FormMain(), з класу FormMain з FormMain.cs. Програма завантажується як люба стандартна програма Windows: по подвійному кліку лівої кнопки миши або поклавіши Enter.

Вхідні дані

Програма не потребує вхідних даних.

Вихідні дані

Програма на виході генерує файл “MyDatabase.sqlite” то вона ії згенерує та добавить таблицю highscore зтекстовим полем name, та числовим полем time.

Відкриття форми Maze.cs.

Вхідні дані.

Якщо існує “maze.xml”, то програма загрузить звідти статус рівня, його час, координати користувача, та матрицю стін.

Вихідні дані.

На виході ми отримуємо файл “maze.xml”, з даними про координати користувача, матрицею рівня та часом гри.

## Повідомлення

Підчас роботи з програмою користувач може получити наступні повідомлення:

* при закінченні гри, користувачу буде запропоновано зберегти свої дані списку переможців.

# ІНСТРУКЦІЯ ПРОГРАМІСТА

## Загальні відомості

Гра лабіринт “Maze”, назва програми Makiyan\_cursovaya\_sem3.

## Умови проектування програми

Для написання програми потрібно ПК з наступними компонентами:

* середа програмування Microsoft Visual Studio 2017, з пакетом для роботи з C#;
* пакет EFTools 6 та пакет SQLite з пакет менеджеру NuGet.
* Windows 8.1 (або новіша), архітектура х86/x64, мінімальна конфігурація згідно з вимогами середовища VS 2017 (CPU 2x 1.8 GHz, 4 GB Ram, 40 GB hard disk space, video card minimal support resolution 720p and higher);
* стандартна комп’ютерна периферія: миша, клавіатура, монітор від 720p 32 bit.

## Керівництво розробника програми

Встановіть Microsoft Visual Studio 2017; при інсталяції виберіть пакет для роботи Desktop Visual C# .

Запустіть VS2017, та створіть новий проект Visual C#/Windows Forms. Вкажіть ім’я для цього проекту. Створіть в цьому проекті ще один проект Visual C#/Class library .Net Framework [6][7]. Задайте ім’я цьому проекту. Перейдіть до першого проекту з формами, та додайте reference на проект бібліотеки. Розробку бізнес логіки треба вести у проекті з бібліотекою, а формами в головному проекті. Для додання нової форми, перейдіть до головного проекту, натисніть на його ім’я правою кнопкою миші, виберіть Add/New Item/Visual C#/Windows form. Для додавання нового класу, перейдіть до проекту з класами, натисніть правою кнопкою миші, на його імені, та виберіть Add/New Item/Visual C#/Class file.

Для того щоби додати нові елементи до форми, двічи натисніть на файл форми поки не відкриється редактор форми. Відкрийте Toolbox, знайдіть потрібний вам компонент, та перетягніть його на форму. В Properties ви можете редагувати, ім’я та візуальні характеристики об’єкту. В Properties/Events ви можете редагувати системні події, наприклад, зміну фокусу або натиск кнопками миші.

При розробці програми для збереження результатів, нам знадобиться Serializer, з System.Runtime.Seiralization. Для цього натисніть правою кнопкою миші на імені проекті Add/Reference/Framework/System.Runtime.Serialization. Тим самим образом треба добавити System.Drawing. Також буде потрібна база даних SQLite, котра автоматично підтягнеться після установки пакету SQLite з NuGet.

Для настройки типу зборки перейдіть до Build/Configuration Manager, та застосуйте стандартні налаштування для створення конфігурації Release x86.

Після написання бізнес логіки, і створення форм, запустіть Build/Build Solution. Якщо сборка пройшла вдало, в папці проекту має з’явитись папка bin/x86/Release, з виконавчим файлом та бібліотекою, та декілька допоміжних файлів (важливі файли MainExec.exe та MazeMain.Data.dll). Якщо скопіювати цю папку в файлами до іншого комп’ютера з Windows 8.1 та новіших версій (зі всіма встановленими оновленнями), то можна користуватися програмою, без будь яких додаткових дій.

## Повідомлення підчас розробки

### Група повідомлень:

* error CS0246: The type or namespace name 'DataContractAttribute' could not be found (are you missing a using directive or an assembly reference?)
* error CS0246: The type or namespace name 'DataContract' could not be found (are you missing a using directive or an assembly reference?)
* error CS0246: The type or namespace name 'KnownTypeAttribute' could not be found (are you missing a using directive or an assembly reference?)
* error CS0246: The type or namespace name 'KnownType' could not be found (are you missing a using directive or an assembly reference?)

Ця група повідомлень з’являється коли розробник, хоче скористуватись системною серіалізацією, але забув підключити бібліотеку System.Runtime.Serialization.

### Повідомлення:

* System.FormatException**:** 'Input string was not in a correct format.' Тіло повідомлення: {"Input string was not in a correct format."}

Це повідомлення з’являлось підчас зчитування з бази данни, та свідчить про те, що з об’єкту SQLiteDataReader неможливо зчитати поле з наданним ім’ям. Вирішення проблеми: брати з цього об’кту дані за номером.

### Повідомення:

* Error CS0122 'MazeLogic' is inaccessible due to its protection level Makiyan\_Cursovaya\_sem3 Maze.cs 1

Це повідомлення свідчить про те, що розробник питається получити доступ до класу, котрий недоступний в цієї області бачимості. Вирішення проблеми: зробити клас публічним.

# КЕРІВНИЦТВО КОРИСТУВАЧА

## Загальні відомості

Гра лабіринт “Maze”.

## Умови застосування

Для застосування програми нам знадобиться комп’ютер з операційною системою Windows 8.1 та новіша, та платформа x86 або x64, з усіма встановленими оновленнями та встановленим .Net framework рівня 4.0. Також потрібно вільне місце на жорсткому дисці більше ніж 25 Mb. Також бажано мати роздільну здатність екрана 1280х720 та більш високу для комфортної роботи.

Комп’ютер має бути обладнаним стандартною комп’ютерною периферією: миша, клавіатура, монітор.

## Характеристики програми

Для коректного запуску програми потрібно, щоб в папці з програмою був файл “maze.xml” згенерований самою програмою, або його відсутність.

Інсталяція та видалення

Програма встановлюється методом копіювання папки з програмою на жорсткий діск (або роботу з флеш накопичувача). Для того щоби видалити програму, достатньо видалити папку програми.

## Послідовність дій користувача програмою

Для початку роботи, нічого не треба налаштовувати. Якщо файл “maze.xml”, був пошкоджен, або модифікован із зовні, то для коректної роботи програми може знадобиться його видалити, з папці з програмою.

При відкритті користувач бачить перед собою заставку (зі звуком початку гри Maze, див рисунок 7.1) після чого відкриється головне вікно (див рисунок 7.2). В котрому йому користувач може або почати гру, або передивитись статистику.



Рисунок 7.1 – Заставка загрузки програми

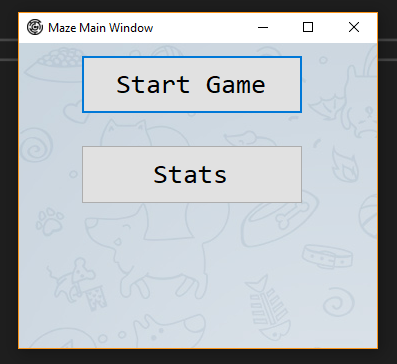


Рисунок 7.2 – Діалог старту прогарами

Вікно гри має кнопку “Play new game” для перезапуску рівня. (див. рисунок 7.3). Після цього користувачу треба за допомогою клавіатури клавішами вверх, вниз, вліво, або вправо перевести зелений курсор гравця в синій курсор виходу з лабіринту. Весь цей час, котрий користувач використає для цього записується системою. Для зручності в правому нижньому куті діалогу є підказка як пройти гру. Підчас вводу з клавіатури ходи гравця мають звуковий супровід, а в кінці користувач отримає діалог перемогу під звук кінця гри.



Рисунок 7.3 – Діалог проходження гри

Після проходження гри користувач отримує діалог вибору або пройти гру заново, або зберегти свої дані до бази переможців (див. рисунок 7.4). На вибір користувач має кнопки “Ok” та “Cancel”. На діалозі є інструкції що яка кнопка зробить якщо на неї натиснути.

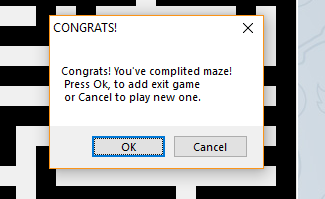


Рисунок 7.4 – Діалог редагування ігрового елементу

Якщо користувач вирішив зберегти свої дані, то він побачить діалог вводу імені. Після натискання кнопки “Save” користувач автоматично переводиться до головного вікна (див. рисунок 7.5).

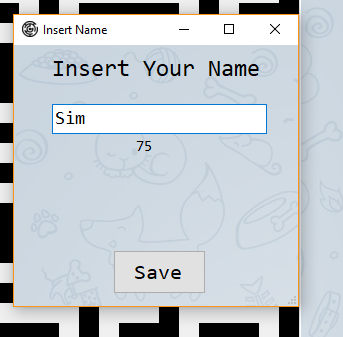


Рисунок 7.5 – Діалог вводу нікнейму

В головному вікні користувач також може переглянути статистику топ 5 найкращіх переможців (див. Малюнок 7.6).

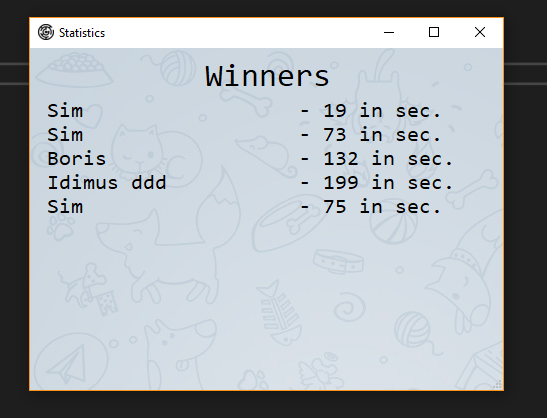


Рисунок 7.6 – Вікно переможців

# ВИСНОВКИ

Під час курсової роботи було отримане вміння писати програми на мові C# з використанням .Net Framework (v. 4.0) та Windows forms. На виході ми отримали повноцінну лабіринт «Maze». Перехід від одного вікна до іншого вікна в програмі відбувається інтуїтивно. В процесі роботи програми дані про рівень можуть бути збережені до xml фалу, та зчитуються з нього в початку при відкритті рівня. Також програма на старті генерує файл бази даних SQLite v3, до котрого можуть бути внесені дані про гравців котри пройшли гру. Переглянути то 5 гравців можна в окремому діалозі, або в SQLite клієнтах.

При розробці програми використовувалися основні методи об’єктно орієнтованого програмування [1] інкапсуляція, поліморфізм, наслідування [5].

Так загалом створено 1 основний та 2 додаткові класи, які реалізуються шляхом наслідування від основного. Для кожного типу об’єкта забезпечено свій окремий клас. Також було створено 5 допоміжних классів

# ПЕРЕЛІК ВИКОРИСТОВАНОЇ ЛІТЕРАТУРИ

1. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4 [Текст] / Э. Троелсен : Москв, ИЛ, 2011. – 331 с.
2. Ярошенко Ф. Проектное управление. Управление инновационными проектами и программами [Текст] / Ф. Ярошенко : 2011. – 268 с.
3. Microsoft corp. Средства и языки разработки [Електронний ресурс] – Режим доступу <https://msdn.microsoft.com/ru-ru/library/aa187916.aspx>
4. Microsoft corp. MSDN Microsoft.Data.Sqlite [Електронний ресурс] – Режим доступу <https://docs.microsoft.com/en-us/dotnet/api/microsoft.data.sqlite?view=msdata-sqlite-2.0.0>
5. Паппас К. Программирование на С и C++ [Текст] : Издание второе / К. Паппас, У. Мюррей : Киев: «Ирина»; BHV, 2000. – 824 с.
6. Microsoft corp. Документация по Visual Studio [Електронний ресурс] – Режим доступу <https://docs.microsoft.com/ru-ru/visualstudio/>
7. Майо Д. С# Builder. Быстрый старт [Текст] / Д. Майо : Бином-Пресс, 2005. – 384 с.

# ДОДАТОК А. ТЕКСТ ПРОГРАМИ

**BaseNode.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Drawing;

namespace MazeMain.Data

{

public class BaseNode

{

protected Panel m\_parent;

protected Label m\_label;

protected int m\_x, m\_y;

public BaseNode(Panel parent, int x, int y)

{

m\_parent = parent;

if (x < 0 || x > 29 || y < 0 || y > 29)

throw new Exception("out of bunds");

m\_label = new Label();

m\_label.Parent = m\_parent;

m\_label.Location = new System.Drawing.Point(x \* 20, y \* 20);

m\_label.Text = "";

m\_label.AutoSize = false;

m\_label.Size = new System.Drawing.Size(20, 20);

m\_label.BackColor = Color.Black;

m\_x = x;

m\_y = y;

}

public virtual Label GetLabel ()

{

return m\_label;

}

public Point GetXY()

{

return new Point(m\_x, m\_y);

}

public virtual void SetNewLocation(int x, int y)

{

throw new Exception("This class has constant location");

}

}

}

**FinishNode.cs**

using System.Windows.Forms;

namespace MazeMain.Data

{

public class FinishNode: BaseNode

{

public FinishNode(Panel parent, int x, int y)

: base (parent, x, y)

{

m\_label.BackColor = System.Drawing.Color.Blue;

m\_x = x;

m\_y = y;

}

}

}

**UserNode.cs**

namespace MazeMain.Data

{

class UserNode : FinishNode

{

public UserNode (Panel parent, int x, int y)

: base(parent, x, y)

{

m\_label.BackColor = System.Drawing.Color.Green;

}

public override void SetNewLocation (int x, int y)

{

m\_x = x;

m\_y = y;

m\_label.Location = new System.Drawing.Point(x \* 20, y \* 20);

}

}

}

**MazeLink.cs**

namespace MazeMain.Data

{

public class MazeLink

{

public MazeNode FromNode, ToNode;

public MazeLink(MazeNode from\_node, MazeNode to\_node)

{

FromNode = from\_node;

ToNode = to\_node;

}

}

}

**MazeNode.cs**

namespace MazeMain.Data

{

public class MazeNode

{

public const int North = 0;

public const int South = North + 1;

public const int East = South + 1;

public const int West = East + 1;

// The node's neighbors in order North, South, East, West.

public MazeNode[] Neighbors = new MazeNode[4];

// The predecessor in the spanning tree.

public MazeNode Predecessor = null;

public MazeNode()

{ }

}

}

**MazeInfo.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

namespace MazeMain.Data

{

[DataContract]

class MazeInfo

{

[DataMember]

public Boolean completed;

[DataMember]

public int[][] gridFinal;

[DataMember]

public int[] userPos;

[DataMember]

public TimeSpan timeSpent;

}

}

**DB\_logics.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Data.SQLite;

namespace MazeMain.Data

{

public class DB\_logics

{

private string dbName = "MyDatabase.sqlite";

public DB\_logics()

{

if (!File.Exists(dbName))

{

SQLiteConnection.CreateFile("MyDatabase.sqlite");

SQLiteConnection dbConnecction;

dbConnecction = new SQLiteConnection("Data Source=MyDatabase.sqlite;Version=3;");

dbConnecction.Open();

string sql = "create table highscores (name varchar(20), time UNSIGNED BIG INT);";

SQLiteCommand command = new SQLiteCommand(sql, dbConnecction);

command.ExecuteNonQuery();

dbConnecction.Close();

}

}

public void insertWinner(string name, int time)

{

SQLiteConnection dbConnecction;

dbConnecction = new SQLiteConnection("Data Source=MyDatabase.sqlite;Version=3;");

dbConnecction.Open();

string sql = String.Format("insert into highscores (name, time) values ('{0}', {1});", name, time);

SQLiteCommand command = new SQLiteCommand(sql, dbConnecction);

command.ExecuteNonQuery();

dbConnecction.Close();

}

public string ReadTop5Winners ()

{

SQLiteConnection dbConnecction;

dbConnecction = new SQLiteConnection("Data Source=MyDatabase.sqlite;Version=3;");

dbConnecction.Open();

string sql = "select \* from highscores ;";

SQLiteCommand command = new SQLiteCommand(sql, dbConnecction);

SQLiteDataReader reader = command.ExecuteReader();

int counter = 0;

string result = "";

while (reader.Read() && reader.HasRows && counter <5)

{

++counter;

result += String.Format("{0,-20} - {1} in sec.\n", reader[0], reader[1]);

//Console.WriteLine("Name: " + reader[0] + "\tScore: " + reader[1]);

}

dbConnecction.Close();

return result;

}

}

}

**MazeLogics.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Runtime.Serialization;

using System.IO;

using System.Xml;

namespace MazeMain.Data

{

public class MazeLogic

{

public static int NORTH = 1;

public static int SOUTH = 2;

public static int EAST = 4;

public static int WEST = 8;

private const bool HORIZONTAL = false;

private const bool VERTICAL = true;

private const int GRID\_SIZE = 14;

private Boolean loadSucessFully;

private Panel m\_panel;

private UserNode m\_user;

private BaseNode m\_finish;

private List<BaseNode> m\_availableNodes;

private MazeInfo mazeInfo;

private DateTime startTime;

public MazeLogic(Panel panel)

{

m\_panel = panel;

try

{

//FormMain.levels = (List<Level>)Load("level.xml", typeof(List<Level>));

mazeInfo = (MazeInfo)Load("maze.xml", typeof(MazeInfo));

}

catch

{

}

if (mazeInfo.gridFinal != null)

{

loadSucessFully = true;

}

else

{

loadSucessFully = false;

}

}

public Boolean IsLoadSuccessfully()

{

return loadSucessFully;

}

private object Load(string fileName, Type t)

{

if (File.Exists(fileName))

{

DataContractSerializer dcs = new DataContractSerializer(t);

XmlReader xmlr = XmlReader.Create(fileName);

object res = dcs.ReadObject(xmlr);

xmlr.Close();

return res;

}

return new MazeInfo();

}

public void Save()

{

mazeInfo.timeSpent += DateTime.Now - startTime;

DataContractSerializer dcs = new DataContractSerializer(mazeInfo.GetType());

XmlWriter xmlw = XmlWriter.Create("maze.xml");

dcs.WriteObject(xmlw, mazeInfo);

xmlw.Close();

}

public void loadMaze()

{

if (mazeInfo.completed)

{

createMaze();

return;

}

startTime = DateTime.Now;

m\_panel.Controls.Clear();

m\_availableNodes = new List<BaseNode>();

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

{

for (int j = 0; j < mazeInfo.gridFinal[i].Length; ++j)

{

if (mazeInfo.gridFinal[i][j] == 1)

{

m\_availableNodes.Add(new BaseNode(m\_panel, i, j));

}

}

}

m\_user = new UserNode(m\_panel, mazeInfo.userPos[0], mazeInfo.userPos[1]);

m\_finish = new FinishNode(m\_panel, GRID\_SIZE \* 2, GRID\_SIZE \* 2 - 1);

}

public void createMaze()

{

mazeInfo.timeSpent = DateTime.Now - DateTime.Now;

startTime = DateTime.Now;

mazeInfo.completed = false;

makeDraftGrid();

genMazeNewWay();

drawMaze();

}

public bool checkIsEnd()

{

mazeInfo.completed = m\_user.GetXY() == m\_finish.GetXY();

if (mazeInfo.completed)

{

mazeInfo.timeSpent += DateTime.Now - startTime;

}

return mazeInfo.completed;

}

public int GetSpentTime()

{

return (int)mazeInfo.timeSpent.TotalSeconds;

}

public Boolean makeMove(string direction)

{

Point point = m\_user.GetXY();

if (direction == "Up")

{

int newY = point.Y - 1;

if (newY > 0 && mazeInfo.gridFinal[point.X][newY] ==0)

{

m\_user.SetNewLocation(point.X, newY);

}

}

else if (direction == "Down")

{

int newY = point.Y + 1;

if (newY < GRID\_SIZE\*2 && mazeInfo.gridFinal[point.X][newY] == 0)

{

m\_user.SetNewLocation(point.X, newY);

}

}

else if (direction == "Left")

{

int newX = point.X - 1;

if (newX >= 0 && mazeInfo.gridFinal[newX][point.Y] == 0)

{

m\_user.SetNewLocation(newX, point.Y);

}

}

else if (direction == "Right")

{

int newX = point.X + 1;

if (newX <= GRID\_SIZE\*2 && mazeInfo.gridFinal[newX][point.Y] == 0)

{

m\_user.SetNewLocation(newX, point.Y);

}

}

else

{

throw new Exception("wrong direction!");

}

mazeInfo.userPos[0] = m\_user.GetXY().X;

mazeInfo.userPos[1] = m\_user.GetXY().Y;

return !(m\_user.GetXY() == point);

}

private void drawMaze()

{

m\_panel.Controls.Clear();

m\_availableNodes = new List<BaseNode>();

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

{

for (int j = 0; j < mazeInfo.gridFinal[i].Length; ++j)

{

if (mazeInfo.gridFinal[i][j] == 1)

{

m\_availableNodes.Add( new BaseNode(m\_panel, i, j));

}

}

}

m\_user = new UserNode(m\_panel, 0, 1);

mazeInfo.userPos[0] = 0;

mazeInfo.userPos[1] = 1;

m\_finish = new FinishNode(m\_panel, GRID\_SIZE \* 2, GRID\_SIZE\*2 - 1);

}

private void genMazeNewWay()

{

// Build the maze nodes.

MazeNode[,] nodes = MakeNodes(14, 14);

// Build the spanning tree.

FindSpanningTree(nodes[0, 0]);

DisplayMaze(nodes);

}

// Make the network of MazeNodes.

private MazeNode[,] MakeNodes(int wid, int hgt)

{

// Make the nodes.

MazeNode[,] nodes = new MazeNode[hgt, wid];

for (int x = 0; x < hgt; ++x)

{

for (int y = 0; y < wid; y++)

{

nodes[x, y] = new MazeNode();

}

}

// Initialize the nodes' neighbors.

for (int r = 0; r < hgt; r++)

{

for (int c = 0; c < wid; c++)

{

if (r > 0)

nodes[r, c].Neighbors[MazeNode.North] = nodes[r - 1, c];

if (r < hgt - 1)

nodes[r, c].Neighbors[MazeNode.South] = nodes[r + 1, c];

if (c > 0)

nodes[r, c].Neighbors[MazeNode.West] = nodes[r, c - 1];

if (c < wid - 1)

nodes[r, c].Neighbors[MazeNode.East] = nodes[r, c + 1];

}

}

// Return the nodes.

return nodes;

}

// Build a spanning tree with the indicated root node.

private void FindSpanningTree(MazeNode root)

{

Random rand = new Random();

// Set the root node's predecessor so we know it's in the tree.

root.Predecessor = root;

// Make a list of candidate links.

List<MazeLink> links = new List<MazeLink>();

// Add the root's links to the links list.

foreach (MazeNode neighbor in root.Neighbors)

{

if (neighbor != null)

links.Add(new MazeLink(root, neighbor));

}

// Add the other nodes to the tree.

while (links.Count > 0)

{

// Pick a random link.

int link\_num = rand.Next(0, links.Count);

MazeLink link = links[link\_num];

links.RemoveAt(link\_num);

// Add this link to the tree.

MazeNode to\_node = link.ToNode;

link.ToNode.Predecessor = link.FromNode;

// Remove any links from the list that point

// to nodes that are already in the tree.

// (That will be the newly added node.)

for (int i = links.Count - 1; i >= 0; i--)

{

if (links[i].ToNode.Predecessor != null)

links.RemoveAt(i);

}

// Add to\_node's links to the links list.

foreach (MazeNode neighbor in to\_node.Neighbors)

{

if ((neighbor != null) && (neighbor.Predecessor == null))

links.Add(new MazeLink(to\_node, neighbor));

}

}

}

private void makeDraftGrid()

{

mazeInfo.gridFinal = new int[GRID\_SIZE \* 2 + 1][];

mazeInfo.userPos = new int[2];

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

mazeInfo.gridFinal[i] = new int[GRID\_SIZE \* 2 + 1];

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

{

for (int j = 0; j < mazeInfo.gridFinal[i].Length; ++j)

mazeInfo.gridFinal[i][j] = 0;

}

}

private void DisplayMaze(MazeNode[,] nodes)

{

for (int r = 0; r < GRID\_SIZE; r++)

{

for (int c = 0; c < GRID\_SIZE; c++)

{

DrawWalls(nodes[r, c], r \* 2 + 1, c \* 2 + 1);

}

}

mazeInfo.gridFinal[0][1] = 0;

mazeInfo.gridFinal[mazeInfo.gridFinal.Length - 1][mazeInfo.gridFinal[mazeInfo.gridFinal.Length - 1].Length - 2] = 0;

}

private void DrawWalls(MazeNode node, int x, int y)

{

for (int side = 0; side < 4; side++)

{

if ((node.Neighbors[side] == null) ||

((node.Neighbors[side].Predecessor != node) &&

(node.Neighbors[side] != node.Predecessor)))

{

DrawWall(node, side, x, y);

}

}

}

// Draw one side of our bounding box.

private void DrawWall(MazeNode node, int side, int x, int y)

{

switch (side)

{

case MazeNode.North:

mazeInfo.gridFinal[x - 1][y] = 1;

mazeInfo.gridFinal[x - 1][y - 1] = 1;

mazeInfo.gridFinal[x - 1][y + 1] = 1;

break;

case MazeNode.South:

mazeInfo.gridFinal[x + 1][y] = 1;

mazeInfo.gridFinal[x + 1][y - 1] = 1;

mazeInfo.gridFinal[x + 1][y + 1] = 1;

break;

case MazeNode.West:

mazeInfo.gridFinal[x][y - 1] = 1;

mazeInfo.gridFinal[x - 1][y - 1] = 1;

mazeInfo.gridFinal[x + 1][y - 1] = 1;

break;

case MazeNode.East:

mazeInfo.gridFinal[x][y + 1] = 1;

mazeInfo.gridFinal[x - 1][y + 1] = 1;

mazeInfo.gridFinal[x + 1][y + 1] = 1;

break;

}

}

}

}

**FormMain.cs**

using MazeMain.Data;

namespace MazeMain

{

public partial class FormMain : Form

{

public FormMain()

{

FormSplashScreen sp = new FormSplashScreen();

sp.ShowDialog();

InitializeComponent();

DB\_logics db = new DB\_logics();

BringToFront();

}

private void FormMain\_FormClosed(object sender, FormClosedEventArgs e)

{

}

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

{

}

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

{

Maze mz = new Maze();

Hide();

mz.ShowDialog();

Show();

}

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

{

Statistics st = new Statistics();

Hide();

st.ShowDialog();

Show();

}

}

}

**Maze.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Media;

using System.Windows.Forms;

using MazeMain.Data;

namespace MazeMain

{

public partial class Maze : Form

{

private MazeLogic m\_logics;

public Maze()

{

InitializeComponent();

m\_logics = new MazeLogic(panel1);

if (m\_logics.IsLoadSuccessfully())

{

m\_logics.loadMaze();

}

else

{

m\_logics.createMaze();

}

}

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

{

m\_logics.createMaze();

}

private void Maze\_KeyDown(object sender, KeyEventArgs e)

{

e.Handled = true;

e.SuppressKeyPress = true;

}

private void Maze\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

private void Maze\_KeyUp(object sender, KeyEventArgs e)

{

Boolean result = false;

if (e.KeyCode == Keys.Up)

{

result = m\_logics.makeMove("Up");

}

else if (e.KeyCode == Keys.Down)

{

result = m\_logics.makeMove("Down");

}

else if (e.KeyCode == Keys.Left)

{

result = m\_logics.makeMove("Left");

}

else if (e.KeyCode == Keys.Right)

{

result = m\_logics.makeMove("Right");

}

e.Handled = true;

e.SuppressKeyPress = true;

if (m\_logics.checkIsEnd())

{

System.IO.Stream str = MazeMain.Properties.Resources.end;

SoundPlayer snd = new SoundPlayer(str);

snd.Play();

DialogResult dialog = MessageBox.Show("Congrats! You've complited maze!" +

"\n Press Ok, to add exit game" +

"\n or Cancel to play new one. ", "CONGRATS!", MessageBoxButtons.OKCancel);

if (dialog == DialogResult.OK)

{

InsertName ins = new InsertName(m\_logics.GetSpentTime());

ins.ShowDialog();

this.Close();

}

else if (dialog == DialogResult.Cancel)

{

m\_logics.createMaze();

}

}

else if (result)

{

System.IO.Stream str = MazeMain.Properties.Resources.move;

SoundPlayer snd = new SoundPlayer(str);

snd.Play();

}

}

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

{

}

private void Maze\_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)

{

}

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

{

panel1.Focus();

}

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

{

panel1.Focus();

}

private void panel1\_Leave(object sender, EventArgs e)

{

panel1.Focus();

}

private void Maze\_FormClosed(object sender, FormClosedEventArgs e)

{

m\_logics.Save();

}

}

}

**Statistics.cs**

using MazeMain.Data;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace MazeMain

{

public partial class Statistics : Form

{

public Statistics()

{

InitializeComponent();

DB\_logics db = new DB\_logics();

string res = db.ReadTop5Winners();

label2.Text = res;

}

}

}

**InsertName.cs**

using MazeMain.Data;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace MazeMain

{

public partial class InsertName : Form

{

private int m\_time;

public InsertName(int time)

{

InitializeComponent();

label2.Text = ""+time;

m\_time = time;

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

string name = textBox1.Text;

if (name.Length > 19)

{

MessageBox.Show("Too long name");

}

else

{

DB\_logics db = new DB\_logics();

db.insertWinner(textBox1.Text, m\_time);

Close();

}

}

}

}

**FormSplashScreen.cs**

namespace MazeMain

{

public partial class FormSplashScreen : Form

{

public FormSplashScreen()

{

InitializeComponent();

System.IO.Stream str = MazeMain.Properties.Resources.start;

SoundPlayer snd = new SoundPlayer(str);

snd.Play();

new Thread(() =>

{

Thread.Sleep(500);

this.Invoke(new MethodInvoker(delegate { Close(); }));

}).Start();

}

}

}

# ДОДАТОК Б. РЕЗУЛЬТАТ ПРАЦЕЗДАТНОСТІ ПРОГРАМИ

**Текст файлу “maze.xml”**

<?xml version="1.0" encoding="UTF-8"?>

<MazeInfo xmlns="http://schemas.datacontract.org/2004/07/MazeMain.Data" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">

<completed>true</completed>

<gridFinal xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays">

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int>

<d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

<d2p1:ArrayOfint>

<d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>1</d2p1:int><d2p1:int>0</d2p1:int><d2p1:int>1</d2p1:int>

</d2p1:ArrayOfint>

</gridFinal>

<timeSpent>PT4M38.659872S</timeSpent>

<userPos xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays">

<d2p1:int>28</d2p1:int>

<d2p1:int>27</d2p1:int>

</userPos>

</MazeInfo>

**Дані з бази “MyDatabase.sqlite”**

SELECT \* FROM HIGHSORES;

"Sim" "19"

"Sim" "73"

"Boris" "132"

"Idimus ddd" "199"

"Sim" "75"