МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»

Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА

за темою

«Створення ігрового додатку Windows Forms на C#»

Студент 4 курсу групи ІПЗ-20к	Керівник дипломної роботи
	_____
	(посада керівника)
Бурлаченко Єгор Сергійович	Добришин Юрій Євгенович
(прізвище, ім'я та по-батькові студента)	(прізвище, ім'я та по-батькові керівника)

До захисту

(резолюція «До захисту»)

Є. Бурл.

(підпис студента)

11.06.2024

(дата)

(підпис викладача)

ЗМІСТ

СКОРОЧЕННЯ ТА ТЕРМІНИ

1. **SQL** (Structured Query Language): мова запитів, яка використовується для взаємодії з реляційними базами даних.

2. **MySQL**: відкрита реляційна система керування базами даних, яка базується на мові SQL.

3. **GUI** (Graphical User Interface): графічний інтерфейс користувача, який надає зручну взаємодію з програмою через візуальні елементи.

4. **MFC** (Microsoft Foundation Classes): набір класів у середовищі програмування Microsoft Visual C++, що використовується для розробки Windows-додатків.

5. **.NET**: фреймворк розробки програмного забезпечення, розроблений компанією Microsoft, який підтримує мови програмування, такі як C#, Visual Basic і F#.

6. **MYISAM**: один з типів таблиць у системі управління базами даних MySQL, який надає швидкий доступ до даних, але не підтримує транзакції.

7. **InnoDB**: інший тип таблиць у системі управління базами даних MySQL, який підтримує транзакції та забезпечує високий рівень надійності даних.

8. **INT** (Integer): тип даних у SQL для зберігання цілих чисел.

9. **FLOAT** (Float): тип даних у SQL для зберігання чисел з плаваючою комою.

10. **BOOLEAN** : тип даних у SQL для зберігання логічних значень TRUE або FALSE.

11. **ОС**(Operating System): операційна система, яка керує ресурсами комп'ютера та надає інтерфейс для користувача.

12. **ASCII**(American Standard Code for Information Interchange): стандарт кодування символів, який використовується для представлення текстової інформації в комп'ютерах.

13. **GITHUB** - веб-платформа для зберігання та спільної розробки програмного забезпечення за допомогою системи контролю версій Git.

14. **C#**(C Sharp): мова програмування, розроблена компанією Microsoft, яка часто використовується для розробки програмного забезпечення під платформу .NET.

ВСТУП

У сучасному цифровому середовищі ігрові застосунки є невід'ємною складовою нашого щоденного життя. Розширення можливостей розваг та розвиток ігрової індустрії вимагають створення відмінних та інноваційних грифічних ігор, які б задовольняли смаки широкого кола користувачів.

Метою даної дипломної роботи є розробка ігрового додатку, який поєднує в собі кілька функціональних можливостей для користувачів. Додаток включає в себе інтуїтивний інтерфейс з можливістю реєстрації користувачів та зберігання їхніх облікових даних у базі даних. Додатковою функціональністю є можливість зміни мови інтерфейсу на усіх формах, що забезпечує зручність використання для користувачів різних країн.

Обрано мову програмування C# та технологію Windows Forms з метою забезпечення швидкого та ефективного процесу розробки. C# є потужною мовою програмування, що забезпечує широкі можливості для створення різноманітних програмних продуктів. Windows Forms надає розробникам зручний інструментарій для створення графічних інтерфейсів користувача з високою швидкодією, що є важливим фактором для ігрових додатків.

Особливий акцент роботи складається на ігрових можливостях додатку. Зокрема, в рамках проекту реалізовано гру "2048", що відома своєю цікавістю та складністю, а також класичну гру "Крестики-нолики". Обидві гри включають в себе можливість зберігання результатів у базі даних, що дозволяє користувачам змагатися та порівнювати свої досягнення.

Реалізація цього ігрового додатку має на меті не лише розважити користувачів, але й продемонструвати навички розробки програмного забезпечення, включаючи роботу з базами даних, локалізацію інтерфейсу та реалізацію різноманітних ігрових механік.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ВИКОРИСТАННЯ WINDOWS FORMS ТА MYSQL.

1.1. Вступ до Windows Forms

1.1.1. Історія та еволюція Windows Forms

Windows Forms, як частина .NET Framework, була вперше представлена в 2002 році Microsoft. Ця технологія призначалася для розробки графічних інтерфейсів користувача (GUI) для операційної системи Windows. До її появи розробники використовували Win32 API або MFC (Microsoft Foundation Classes), що вимагало значних зусиль і знань низькорівневого програмування. Перші версії Windows Forms дозволяли створювати базові форми та додавати до них елементи управління. З часом, з виходом нових версій .NET Framework, Windows Forms розвивалася, додаючи нові компоненти та можливості. Наприклад, в .NET Framework 2.0 були введені такі елементи, як SplitContainer, DataGridView та багато інших, що значно спростило створення складних інтерфейсів.

1.1.2. Основні компоненти Windows Forms

Windows Forms складається з великої кількості компонентів, кожен з яких призначений для вирішення конкретних завдань. Ось деякі з них:

- **Форми (Forms)**: Основний контейнер для інших елементів управління. Вони можуть мати меню, панелі інструментів, статусні рядки та інші елементи.

- **Кнопки (Buttons)**: Елемент управління, який запускає певну дію при натисканні. В Windows Forms існує кілька типів кнопок, включаючи стандартні кнопки, радіокнопки та прапорці.

- **Текстові поля (TextBoxes)**: Використовуються для введення та відображення тексту.

- **Панелі (Panels)**: Контейнери, що дозволяють групувати інші елементи управління.

- **Меню (Menus)**: Елементи, які надають користувачу доступ до команд програми.

1.2. Принципи розробки додатків на Windows Forms

1.2.1. Архітектура додатків на Windows Forms

Типова архітектура додатків на Windows Forms включає кілька шарів:

- **Модель (Model)**: Містить дані та бізнес-логіку програми. Наприклад, для гри Tic-Tac-Toe модель включає логіку гри, стан дошки та правила перемоги.

- **Вид (View):** Відповідає за відображення даних користувачу. Це форми та елементи управління, які користувач бачить на екрані.

- **Контролер (Controller)**: Обробляє взаємодію між моделлю та видом. Контролер отримує вхідні дані від користувача через вид, обробляє їх, оновлює модель та відображає зміни у вигляді.

1.2.2. Подієво-орієнтоване програмування

Windows Forms підтримує подієво-орієнтоване програмування, що дозволяє створювати інтерактивні додатки. Основні концепції включають:

- **Події (Events)**: Дії, які відбуваються в результаті взаємодії користувача з інтерфейсом. Наприклад, натискання кнопки або зміна тексту в текстовому полі.

- **Делегати (Delegates)**: Типи, що представляють методи з певним набором параметрів і типом повернення. Використовуються для оголошення подій і приєднання до них обробників.

```
private void button_Click(object sender, EventArgs e)
{
    MessageBox.Show("Button clicked!");
}
```

Рис 1.1 - Приклад події в Windows Forms

1.3. Вступ до MySQL

1.3.1. Історія та еволюція MySQL

MySQL була створена в середині 1990-х років компанією MySQL AB, яка пізніше була придбана Sun Microsystems, а потім Oracle Corporation. MySQL є системою керування реляційними базами даних з відкритим вихідним кодом і використовується в багатьох веб-додатках, таких як WordPress, Joomla та Drupal.

MySQL підтримує кілька типів зберігання даних, включаючи InnoDB та MyISAM. InnoDB забезпечує транзакційну підтримку та зовнішні ключі, що робить його підходящим для додатків, які потребують надійності та цілісності даних.

1.3.2. Основні концепції баз даних

Основні поняття реляційної бази даних включають:

- **База даних (Database)**: Колекція таблиць, яка зберігає структуровані дані.

- **Таблиця (Table)**: Набір рядків (записів), організованих у стовпці (поля).

- **Рядок (Row)**: Окремий запис у таблиці, що складається з одного або більше полів.

- **Стовпець (Column)**: Атрибут, що описує властивість рядка.

Типи даних у MySQL включають числові типи (INT, FLOAT), текстові типи (VARCHAR, TEXT) та інші спеціальні типи (DATE, BOOLEAN).

1.4. Основні аспекти роботи з MySQL

1.4.1. Встановлення та налаштування MySQL

Процес встановлення MySQL залежить від операційної системи. Ось короткий огляд для різних ОС:

- **Windows**:

- Завантажте інсталятор з офіційного сайту MySQL.

- Запустіть інсталятор та дотримуйтесь інструкцій.

- Налаштуйте кореневий пароль та виберіть бажані опції.

- **Linux**:

- Використовуйте пакетний менеджер для встановлення (наприклад, `apt-get install mysql-server` для Ubuntu).

- Запустіть MySQL сервер і налаштуйте кореневий пароль.

1.4.2. Основи SQL

SQL (*Structured Query Language*) - це мова для роботи з реляційними базами даних. Основні команди SQL включають:

- SELECT: Використовується для вибірки даних з таблиць.

```
SELECT * FROM users;
```

Рис 1.2 – Приклад використання команди SELECT.

- INSERT: Додає новий рядок у таблицю.

```
INSERT INTO users (username, password) VALUES ('user1', 'password1');
```

Рис 1.3 - Приклад використання команди INSERT.

- UPDATE: Оновлює існуючі записи в таблиці.

```
UPDATE users SET password = 'newpassword' WHERE username = 'user1';
```

Рис 1.4 - Приклад використання команди UPDATE.

- DELETE: Видаляє записи з таблиці.

```
DELETE FROM users WHERE username = 'user1';
```

Рис 1.4 - Приклад використання команди DELETE.

1.5. Інтеграція Windows Forms та MySQL

1.5.1. Підключення до бази даних MySQL з Windows Forms

Для підключення до MySQL з додатка на Windows Forms можна використовувати бібліотеку ADO.NET. Основні кроки включають:

1. Додайте посилання на бібліотеку MySQL.Data:

```
using MySql.Data.MySqlClient;
```

Рис 1.5 – Приклад додавання посилання.

2. Створіть з'єднання з базою даних:

```
MySqlConnection connection = new MySqlConnection(connectionString);
connection.Open();
```

Рис 1.6 – Приклад з'єднання з базою даних.

3. Виконуйте SQL-запити:

```
MySqlCommand command = new MySqlCommand("SELECT * FROM users", connection);
MySqlDataReader reader = command.ExecuteReader();
while (reader.Read())
{
    Console.WriteLine(reader["username"].ToString());
}
reader.Close();
```

Рис 1.7 – Приклад SQL-запиту.

1.5.2. Виконання SQL-запитів з Windows Forms

Після встановлення з'єднання з базою даних можна виконувати SQL-запити для взаємодії з даними:

- Відображення даних у DataGridView:

```
MySqlCommand command = new MySqlCommand("SELECT * FROM users", connection);
MySqlDataAdapter adapter = new MySqlDataAdapter(command);
DataTable table = new DataTable();
adapter.Fill(table);
dataGridView1.DataSource = table;
```

Рис 1.8 – Приклад відображення даних у DataGridView.

- Вставка нових даних через форму:

```
private void insertButton_Click(object sender, EventArgs e)
{
    string query = "INSERT INTO users (username, password) VALUES (@username, @password)";
    MySqlCommand command = new MySqlCommand(query, connection);
    command.Parameters.AddWithValue("@username", usernameTextBox.Text);
    command.Parameters.AddWithValue("@password", passwordTextBox.Text);
    command.ExecuteNonQuery();
    MessageBox.Show("User added successfully!");
}
```

Рис 1.9 – Приклад вставки нових даних через форму.

- Оновлення даних:

```
private void updateButton_Click(object sender, EventArgs e)
{
    string query = "UPDATE users SET password = @password WHERE username = @username";
    MySqlCommand command = new MySqlCommand(query, connection);
    command.Parameters.AddWithValue("@username", usernameTextBox.Text);
    command.Parameters.AddWithValue("@password", passwordTextBox.Text);
    command.ExecuteNonQuery();
    MessageBox.Show("Password updated successfully!");
}
```

Рис 1.10 – Приклад оновлення даних.

- Видалення даних:

```
private void deleteButton_Click(object sender, EventArgs e)
{
    string query = "DELETE FROM users WHERE username = @username";
    MySqlCommand command = new MySqlCommand(query, connection);
    command.Parameters.AddWithValue("@username", usernameTextBox.Text);
    command.ExecuteNonQuery();
    MessageBox.Show("User deleted successfully!");
}
```

Рис 1.11 – Приклад видалення даних.

1.6. Безпека та оптимізація

1.6.1. Безпека даних

Забезпечення безпеки даних є критичним аспектом при розробці додатків на Windows Forms, які взаємодіють з MySQL. Основні принципи включають:

- Використання параметризованих запитів: Захищає від SQL-ін'єкцій, які є одним з найпоширеніших видів атак.

```
string query = "SELECT * FROM users WHERE username = @username AND password = @password";
MySqlCommand command = new MySqlCommand(query, connection);
command.Parameters.AddWithValue("@username", usernameTextBox.Text);
command.Parameters.AddWithValue("@password", passwordTextBox.Text);
MySqlDataReader reader = command.ExecuteReader();
```

Рис 1.12 – Приклад використання параметризованих запитів.

- Шифрування даних: Захист конфіденційної інформації, такої як паролі. Використання хешування та сольового значення для зберігання паролів.

```
public static string ComputeSha256Hash(string rawData)
{
    using (SHA256 sha256Hash = SHA256.Create())
    {
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
            builder.Append(bytes[i].ToString("x2"));
        }
        return builder.ToString();
    }
}
```

Рис 1.13 – Приклад шифрування даних.

- Контроль доступу: Використання ролей та прав доступу для обмеження дій, які користувачі можуть виконувати.

```
CREATE USER 'app_user'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT, INSERT, UPDATE ON yourdatabase.* TO 'app_user'@'localhost';
```

Рис 1.14 – Приклад контролю доступу.

1.6.2. Оптимізація продуктивності

Оптимізація продуктивності є важливою для забезпечення швидкої роботи додатків:

- Індексація: Використання індексів для прискорення пошуку даних у великих таблицях.

```
CREATE INDEX idx_username ON users (username);
```

Рис 1.15 – Приклад індексації.

- Оптимізація запитів: Використання EXPLAIN для аналізу та оптимізації SQL-запитів.

```
EXPLAIN SELECT * FROM users WHERE username = 'user1';
```

Рис 1.16 – Приклад оптимізації запита.

- Налаштування конфігурації MySQL: Зміна параметрів конфігурації для покращення продуктивності, таких як `innodb_buffer_pool_size`, `query_cache_size`, та інших.

- Кешування даних: Використання кешування для зменшення навантаження на базу даних та підвищення швидкості доступу до часто використовуваних даних.

РОЗДІЛ 2 РЕАЛІЗАЦІЯ ТА ОГЛЯД ІГРОВОГО ДОДАТКУ.

Зараз на скріншоті ви зможете побачити всі складові проекту і детально розглянемо кожну з них.

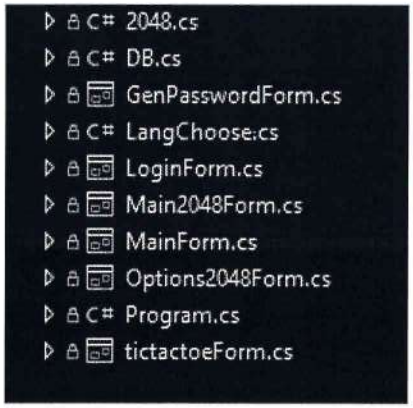

Рис 2.1 – Вигляд структури проекту.

Проект складається з 3 основних частин, а саме форму Авторизації, ігри 2048 та ігри в Хрестики-нуліки.

Перейдем к самому интересному, к рассмотрению каждой складовой проекта.

Я не вставлятиму код, показуватиму дизайн і розповідати з чого він складається, а сам код ви зможете подивитися в моему GitHub.

2.1 Огляд форми LoginForm.cs

Форма авторизації, за допомогою якої ви можете увійти/реєструватися, згенерувати пароль, так само вибрати якою мовою надалі ви користуватиметеся програмою.

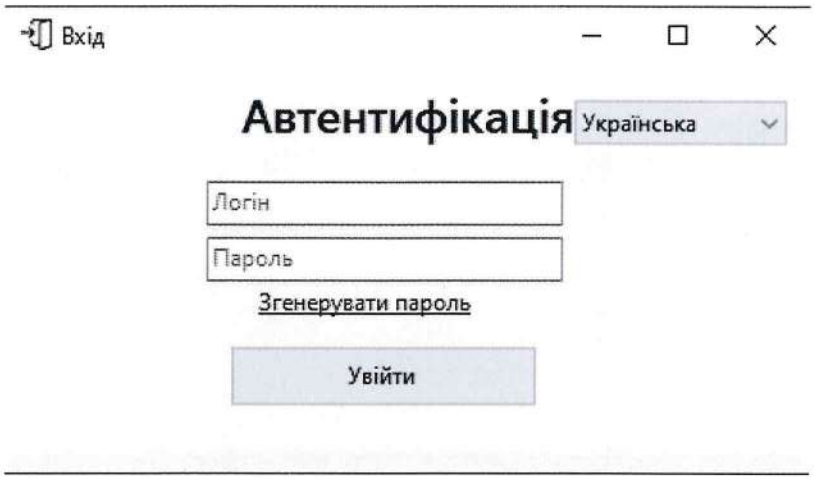

Рис 2.2 – Дизайн форми LoginForm.cs

1. **Конструктор `LoginForm()`:**

- Цей конструктор викликається при створенні екземпляра форми `LoginForm`.

- В ньому відбувається ініціалізація компонентів форми (ініціалізація елементів управління, прив'язка подій і т. д.).

- Вибирається перший елемент у випадаючому списку мов (`LangBox`), а також встановлюється тип його розгортання (ComboBoxStyle.DropDownList).

2. **Метод `LoginButton_Click(object sender, EventArgs e)`:**

- Цей метод викликається при натисканні на кнопку "LoginButton".

- Отримує введений користувачем логін і пароль.

- Формує SQL-запит для вибірки даних з таблиці `users`, де перевіряє наявність користувача з вказаним логіном і паролем.

- Якщо користувач знайдений, відкривається головна форма (`MainForm`), в іншому випадку користувачу запропоновано зареєструватися.

3. **Метод `addNewUser(string loginUser, string PassUser)`:**

- Цей метод додає нового користувача до бази даних.

- Він відкриває з'єднання з базою даних, створює SQL-запит на додавання нового запису в таблицю `users` з вказаним логіном і паролем, виконує цей запит і закриває з'єднання.

4. **Метод `ifUserExist(string loginUser)`:**

- Цей метод перевіряє існування користувача в базі даних.

- Він відкриває з'єднання з базою даних, створює SQL-запит на вибірку даних з таблиці `users`, де перевіряє наявність користувача з вказаним логіном, виконує цей запит, а потім повертає результат перевірки.

5. **Метод `GeneragePasswordLabel_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)`:**

- Цей метод викликається при кліку на посилання "GeneragePasswordLabel".

- Він створює та відображає нову форму `GenPasswordForm`, призначену для генерації пароля.

6. **Метод `LangBox_SelectedIndexChanged(object sender, EventArgs e)`:**

- Цей метод викликається при зміні вибраної мови в випадаючому списку `LangBox`.

- Залежно від обраної мови він змінює текстові значення для всіх елементів управління на формі, включаючи написи, текстові поля та кнопки.

2.2 Огляд форми GenPasswordForm.cs

У цій формі ви можете вибрати довжину, символи, які хочете мати у вашому паролі. Можете не зберігати його, або ж вибрати шлях, де ваш пароль буде збережений.

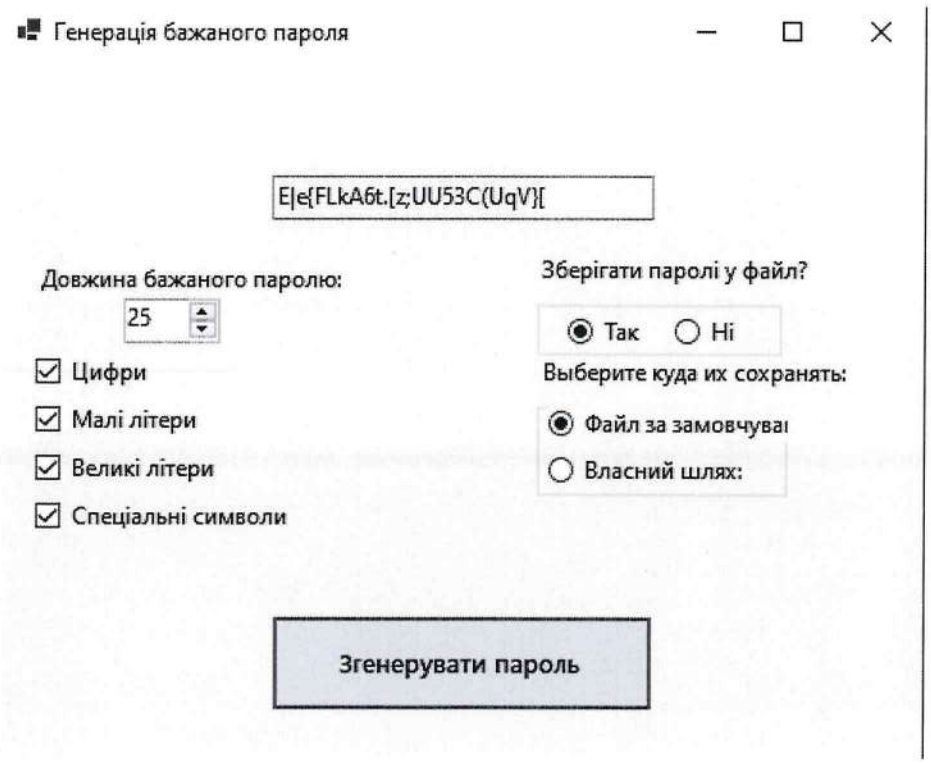

Рис. 2.3 – Дизайн форми GenPassword.cs.

1. **Конструктор `GenPasswordForm()`:**

- Цей конструктор викликається при створенні екземпляру форми `GenPasswordForm`.

- Він ініціалізує компоненти форми, такі як кнопки, радіокнопки, поля введення та інші.

- Викликає метод `changelang()` для зміни текстів на формі залежно від обраної мови.

2. **Метод `changelang()`:**

- Цей метод відповідає за зміну текстів на формі залежно від обраної мови.

- Він встановлює тексти для всіх елементів управління на формі в залежності від обраної мови. Наприклад, змінює надписи кнопок та підказки.

3. **Метод `generateBut_Click(object sender, EventArgs e)`:**

- Цей метод викликається при натисканні на кнопку "generateBut".

- Виконує генерацію пароля на основі вибраних параметрів: наявності цифр, малих і великих літер, спеціальних символів.

- Перевіряє обрані параметри та генерує відповідний пароль.

- Виводить згенерований пароль у вікно введення `outTextBox`.

4. **Метод `ConvertAsciiCodesToString(int[] asciiCodes)`:**

- Цей метод приймає масив ASCII-кодів та конвертує його у рядок символів.

- Проходиться по кожному ASCII-коду та додає відповідний символ у результуючий рядок.

5. **Обробники подій `radioButton1_CheckedChanged`, `radioButton3_CheckedChanged`, `radioButton4_CheckedChanged`:**

- Ці методи викликаються при зміні вибору радіокнопок.

- Вони відповідають за відображення додаткових елементів управління (наприклад, полів введення для шляху до файлу) та збереження пароля у файлі залежно від вибору користувача. Наприклад, якщо обрано збереження у файл, з'являється можливість вибору шляху до файлу або використання файлу за замовчуванням).

2.3 Огляд форми MainForm.cs

Найголовніша форма і одночасно найпростіша форма, в якій ви просто вибираєте в яку гру гратимете. Так само якщо ви закриєте наступні ігри, ви будете перенаправлені в це основне вікно, але при закритті його ви закінчите роботу програми.

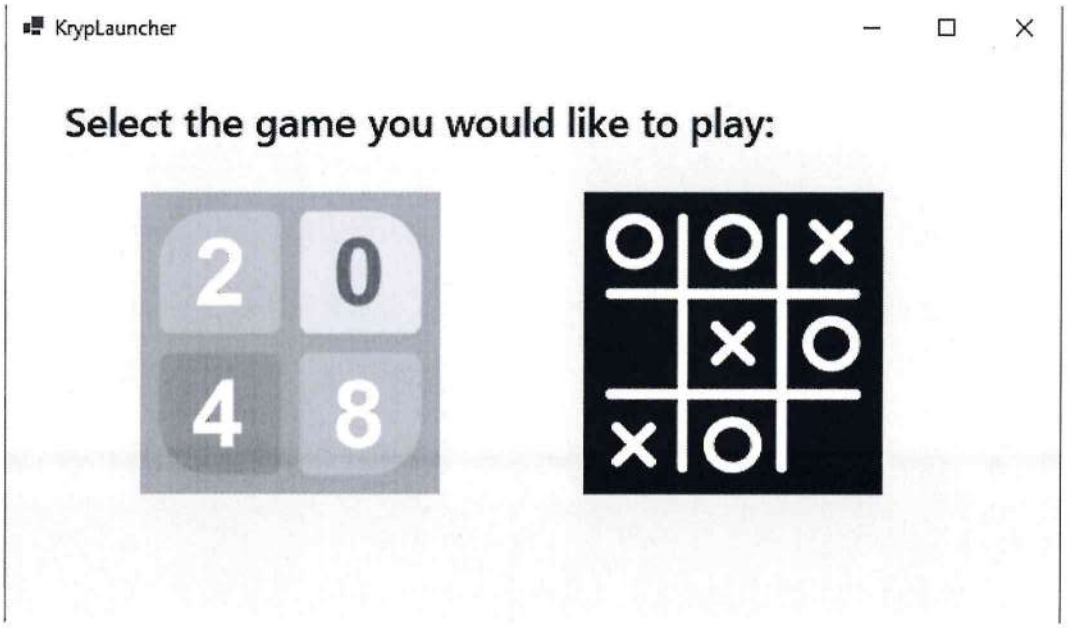

Рис 2.4 – Дизайн форми MainForm.cs

1. **Конструктор `MainForm(string loginUser)`:**

- Цей конструктор викликається при створенні екземпляру форми `MainForm`.

- Ініціалізує компоненти форми.

- Викликає метод `changelang()` для зміни мови і передає ім'я користувача `loginUser`.

2. **Метод `pictureBox2048_Click(object sender, EventArgs e)`:**

- Цей метод викликається при натисканні на зображення "2048".

- Створює новий екземпляр форми `Options2048Form`, передаючи параметри для ініціалізації гри "2048".

- Приховує поточну форму і показує форму для налаштування гри "2048".

3. **Метод `pictureBoxTicTacToe_Click(object sender, EventArgs e)`:**

- Цей метод викликається при натисканні на зображення "Крестики-нолики".

- Створює новий екземпляр форми `tictactoeForm`, передаючи ім'я користувача `loginUser`.

- Приховує поточну форму і показує форму для гри "Крестики-нолики".

4. **Метод `changelang()`:**

- Цей метод відповідає за зміну тексту мітки `chooseLabel` залежно від обраної мови.

- Вибирає текст для мітки згідно з обраною мовою.

5. **Обробник події `MainForm_FormClosed(object sender, FormClosedEventArgs e)`:**

- Цей метод викликається при закритті форми.

- Закриває програму, викликаючи метод `Environment.Exit(0)`.

2.4 Огляд форми tictactoe.cs

Сама гра в хрестики-нуліки, мета гри поставити 3 символи в ряд. У разі збільшення кількості ходів відкриваються нові клітини.

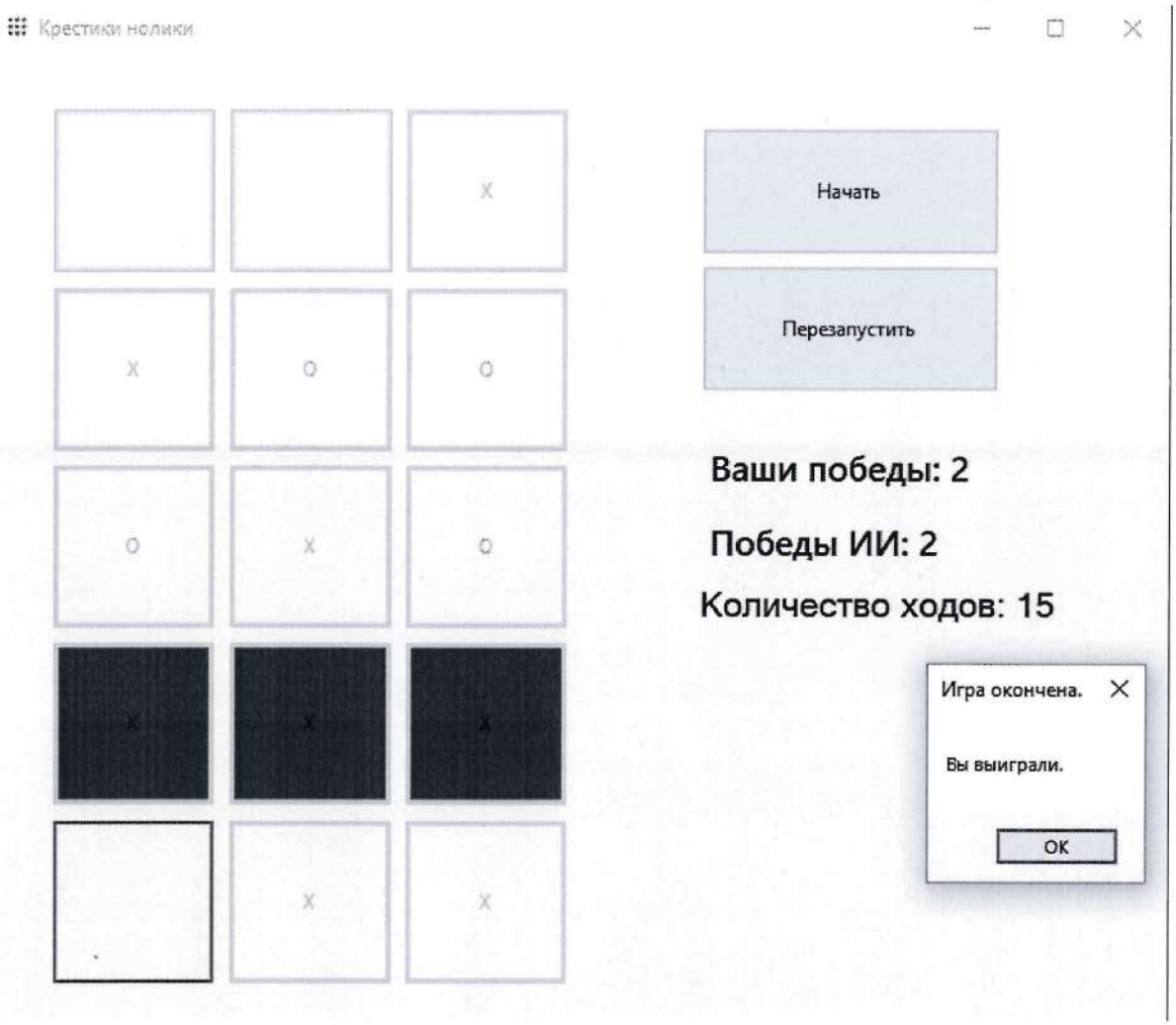

Рис 2.5 – Дизайн форми tictactoe.cs.

1. **Клас `tictactoeForm`:**

- Цей клас є частиною простору імен `KrypLauncher`.

- Він є похідним від класу `Form`, що означає, що це вікно програми.

- Має внутрішній стан, такий як рядок `loginUser` та рядки для текстів повідомлень гри.

- Містить перелік гравців у форматі перерахування (`enum`) з двома значеннями: "X" та "O".

2. **Конструктор `tictactoeForm(string loginUser)`:**

- Ініціалізує компоненти форми за допомогою метода `InitializeComponent()`.

- Приймає аргумент `loginUser` та зберігає його в приватному полі `this.loginUser`.

3. **Метод `Form1_Load(object sender, EventArgs e)`:**

- Викликає метод `Default()`, що налаштовує початковий стан елементів у формі.

- Викликає метод `chooselang()`, який встановлює мову гри.

- Встановлює тексти міток з перемогами користувача та комп'ютера, використовуючи результати з методу `resultUpdate()`.

4. **Метод `Start()`:**

- Показує спливаюче вікно з питанням про вибір першого ходу.

- Якщо користувач обирає "Hi", викликає метод `randomMove()` для ходу комп'ютера.

5. **Метод `resultUpdate(int action)`:**

- Оновлює дані про перемоги та поразки користувача в базі даних.

- Приймає аргумент `action`, який вказує, яку дію слід виконати: оновлення перемог, оновлення поразок, або лише повернення поточних даних.

- Відображає повідомлення про перемогу або поразку користувача.

6. **Метод `CheckMoves(int a)`:**

- Забезпечує логіку для зміни доступності та видимості кнопок у грі в залежності від кількості зроблених ходів.

7. **Метод `ButtonClick(object sender, EventArgs e)`:**

- Викликається при натисканні кнопки на ігровому полі.

- Помічає поточного гравця, встановлює значення кнопки на полі, вимикає кнопку та збільшує лічильник ходів.

- Перевіряє умови для перемоги користувача, викликає метод для ходу комп'ютера, перевіряє умови перемоги комп'ютера та виводить лічильник ходів у консоль.

8. **Метод `CheckWin_AI()`:**

- Перевіряє всі можливі комбінації для перемоги комп'ютера, які відображені на ігровому полі.

- Якщо будь-яка з комбінацій знаходиться у стані "O", функція встановлює відповідні кнопки у червоний колір, щоб підкреслити перемогу комп'ютера.

- Після цього викликає метод `resultUpdate(2)`, який оновлює рахунок поразок комп'ютера у відповідній таблиці користувача.

9. **Метод `CheckWin_USER()`:**

- Аналогічно методу `CheckWin_AI()`, але перевіряє комбінації для перемоги гравця ("X").

- Якщо будь-яка з комбінацій знаходиться у стані "X", функція встановлює відповідні кнопки у зелений колір, щоб підкреслити перемогу гравця.

- Після цього викликає метод `resultUpdate(1)`, який оновлює рахунок перемог користувача у відповідній таблиці користувача.

10. **Метод `randomMove(int a)`:**

- Спочатку створюється список кнопок `buttons`.

- Поточним гравцем встановлюється гравець "O".

- До списку кнопок додаються певні кнопки з форми в залежності від значення аргумента `a`.

- Після цього зі списку видаляються кнопки, які вже були натиснуті.

- Обирається випадкова кнопка з доступних і встановлюється на ній символ поточного гравця.

11. **Метод `Default()`:**

- Проходиться по всім елементам у контейнері форми.

- Якщо зустрічається кнопка з тегом "Play", то вона вимикається та скидається її текст та колір фону.

- Якщо зустрічається кнопка з тегом "Play1", вона приховується та скидаються її властивості.

12. **Обробники подій `restBut_Click` та `startbut_Click`:**

- `restBut_Click`: Скидається лічильник ходів та викликається метод `Default()`.

- `startbut_Click`: Вмикаються кнопки для гри, їх текст та фоновий колір скидаються до початкових значень, а потім запускається гра.

2.5 Огляд форми Options2048Form.cs

Вікно налаштувань для гри "2048" тут можна вибрати різні параметри для подальшої гри.

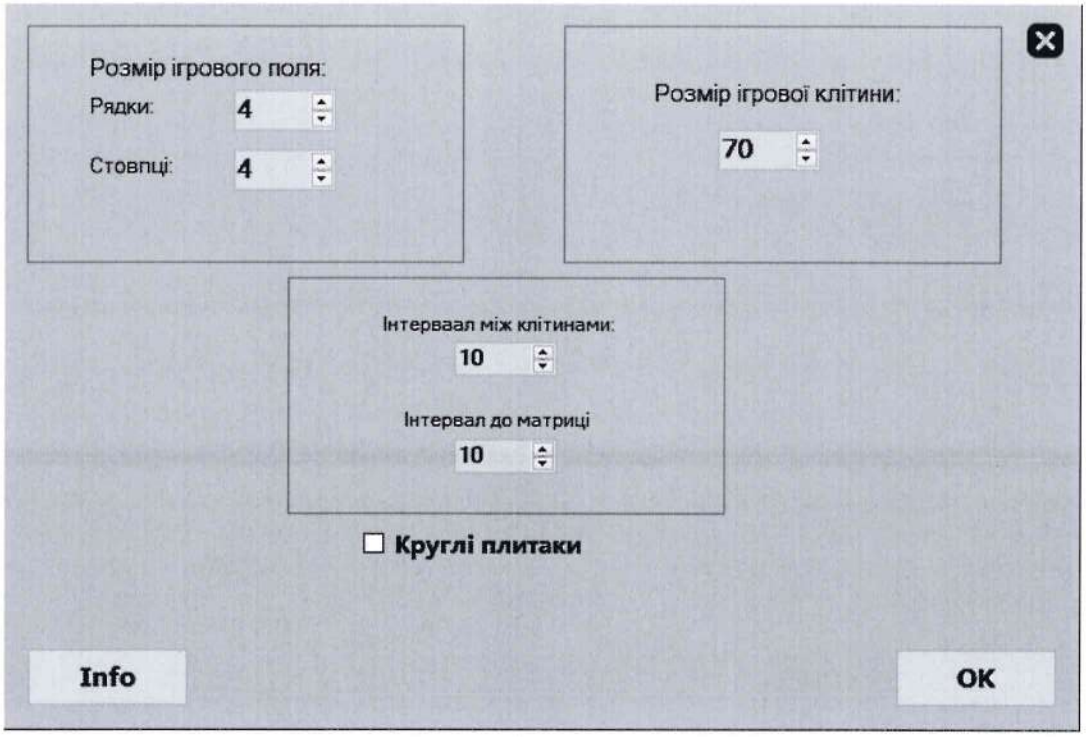

Рис 2.6 – Дизайн форми Options2048Form.cs.

1. **Конструктор `Options2048Form(int matrixRows, int matrixCells, Size tileSize, int Int32ervalBetweenTiles, int borderInt32erval, Color backColor, string loginUser)`:**

- Ініціалізує компонент форми та викликає метод `changelang()`, який встановлює мову інтерфейсу.

- Зберігає ім'я користувача у змінну `loginUser`.

- Встановлює початкові значення для елементів налаштувань (кількість рядків, стовпців, розмір плиток, інтервали) відповідно до переданих параметрів.

2. **Метод `OnOptions()`:**

- Активує режим налаштувань гри.

- Відображає форму налаштувань на екрані.

- Робить головну форму гри неактивною (якщо вона існує).

3. **Метод `ReadSettings()`:**

- Зчитує налаштування гри з файлу `data.2048`.

- Встановлює значення для елементів налаштувань, таких як кількість рядків, стовпців, розмір плиток та інтервали.

- У разі виникнення помилки під час читання файлу, виводить відповідне повідомлення про помилку.

4. **Метод `StartForm_FormClosing`:**

- Обробляє подію закриття форми налаштувань.

- Якщо форма налаштувань активна, вона приховується замість закриття, а головна форма гри знову стає активною.

5. **Метод `StartForm_Load`:**

- Викликається при завантаженні форми налаштувань.

- Викликає метод `ReadSettings()`, який зчитує налаштування з файлу.

6. **Метод `bOK_Click`:**

- Обробляє подію натискання кнопки "OK".

- Зчитує введені користувачем значення налаштувань.

- Створює нову головну форму гри з цими налаштуваннями.

- Приховує форму налаштувань та показує головну форму гри.

- Підписує головну форму гри на подію `OptionsEvent`.

7. **Метод `bClose_Click`:**

- Обробляє подію натискання кнопки "Закрити".

- Приховує форму налаштувань.

- Відкриває головну форму програми.

8. **Метод `OptionsForm_MouseDown`:**

- Дозволяє користувачу переміщувати форму налаштувань по екрану, утримуючи ліву кнопку миші.

9. **Метод `bInfo_Click`:**

- Обробляє подію натискання кнопки "Інформація".

- Показує повідомлення з інформацією про гарячі клавіші в грі.

10. **Метод `changelang`:**

- Налаштовує текстові елементи інтерфейсу відповідно до вибраної мови.

- Змінює текст міток, кнопок та інших елементів форми на відповідну мову.

2.6 Огляд форми Main2048Form.cs

Рис 2.7 – Дизайн форми Main2048Form.cs.

1. **Конструктор `Main2048Form(int _matrixRows, int _matrixCells, Size _tileSize, int _Int32ervalBetweenTiles, int _borderInt32erval, bool _ellipseTile, Color backColor, string loginUser)`:**

- Ініціалізує об'єкт `Main2048Form` з заданими параметрами розмірів матриці, розмірів плиток, інтервалів та кольорів.

- Встановлює властивості об'єкта на основі отриманих параметрів.

2. **Метод `CreateMatrix()`:**

- Створює матрицю плиток (кнопок) за вказаними розмірами та інтервалами.

- Додає створені кнопки до панелі `pMatrix`.

3. **Метод `CreateButtons(int count, Size size, Font font, Color backColor)`:**

- Створює масив кнопок зазначеного розміру з вказаними параметрами: розміром, шрифтом та кольором фону.

4. **Метод `ShowMatrix(int[,] oldMatrix)`:**

- Відображає матрицю на екрані, змінюючи лише елементи, що були змінені в порівнянні з попередньою версією матриці.

- Встановлює текст, колір та розмір кнопок відповідно до значень матриці та налаштувань.

5. **Метод `ShowScore()`:**

- Відображає поточний рахунок гравця на екрані.

6. **Метод `ShowBestScore()`:**

- Відображає найкращий результат гравця на екрані.

7. **Метод `NewGame()`:**

- Розпочинає нову гру, створюючи новий об'єкт гри.

- Призначає обробники подій для об'єкта гри.

- Оновлює відображення матриці та рахунку.

9. **Метод `EndGameHandler()`:**

- Обробляє подію завершення гри, виводячи відповідне повідомлення.

10. **Метод `ScoreOverflowHandler()`:**

- Обробляє подію переповнення рахунку, скидаючи рекорди та виводячи відповідне повідомлення.

11. **Метод `NewTileHandler(int x, int y)`:**

- Обробляє подію появи нового тайлу в матриці, анімуючи його відображення.

12. **Метод `AnimationExtension(Control element, int timeOnAnimation)`:**

- Здійснює анімацію елемента згідно заданого часу виконання.

13. **Метод `AdaptButtonFontSize(Button element, double fontSizePecrent = 0.85)`:**

- Адаптує розмір шрифту кнопки в залежності від розмірів кнопки та вказаного відсотка розміру шрифту.

14. **Метод `ResetAllScores()`:**

- Скидає всі рекорди гравця на нуль.

15. **Метод `ReadColors()`:**

- Зчитує кольори плиток з файлу `colors.2048`, обробляючи винятки.

16. **Метод `ResetCurrentRecord()`:**

- Скидає поточний рекорд гравця на нуль.

- Якщо існує запис про рекорд для поточного розміру матриці, встановлює його значення в 0.

17. **Метод `ReadColors()`:**

- Зчитує кольори плиток з файлу `colors.2048`.

- Ініціалізує словник `colors` з ключами, що відповідають номерам плиток та значеннями кольорів.

- У разі виникнення помилки читання файлу виводить відповідне повідомлення про помилку.

18. **Метод `WriteColors()`:**

- Записує кольори плиток у файл `colors.2048`.

- Ітерується по словнику `colors` та записує номери плиток і відповідні кольори.

19. **Метод `ReadRecords()`:**

- Зчитує рекорди гравця з бази даних.

- Встановлює значення найкращого результату гравця (`bestScore`).

- Виводить найкращий результат гравця на екран.

20. **Метод `WriteRecords()`:**

- Записує рекорди гравця у базу даних.

- Оновлює значення найкращого результату гравця.

21. **Метод `WriteSettings()`:**

- Записує налаштування гри у файл `data.2048`.

- Записує розміри матриці, розмір плиток та інші параметри гри.

22. **Метод `bNewGame_Click`:**

- Викликає функцію `NewGame()`, що починає нову гру.

23. **Метод `bUndo_Click`:**

- Перевіряє, чи можливо виконати дію "Скасувати" в грі.

- Якщо так, то виконує дію "Скасувати" і відображає попередній стан гри.

24. **Метод `bOptions_Click`:**

- Викликає функцію `OptionsEvent()`, яка активує режим налаштувань гри.

25. **Метод `Main2048Form_Load`:**

- Викликає функцію `chooselang()`, яка вибирає мову інтерфейсу.

- Зчитує кольори плиток з файлу.

- Зчитує рекорди гравця з бази даних.

26. **Метод `Main2048Form_KeyDown`:**

- Обробляє натискання клавіш гравцем під час гри.

- Виконує відповідні дії відповідно до натиснутої клавіші: рухає плитки вгору, вниз, вліво або вправо, або виконує інші дії, такі як "Скасувати", "Нова гра" тощо.

27. **Метод `Main2048Form_FormClosing`:**

- Записує кольори плиток у файл.

- Записує рекорди гравця у базу даних.

- Записує налаштування гри у файл.

- Завершує роботу програми.

28. **Метод `pMenu_Paint`:**

- Відповідає за відображення елементів головного меню гри на формі.

2.7 Огляд класу 2048.cs

1. **TrySaveGame():**

- Ця функція спробує зберегти поточний стан гри до файлу. Вона перевіряє, чи існує файл за вказаною шляхом, і якщо так, то відкриває його для запису. Потім вона записує необхідні параметри гри, такі як мінімальне значення блоків, поточний рахунок, розміри матриці та саму матрицю у файл.

2. **TryLoadGame():**

- Ця функція спробує завантажити стан гри з вказаного файлу. Вона перевіряє, чи файл існує, і якщо так, відкриває його для читання. Потім вона

читає з файлу параметри гри, такі як мінімальне значення блоків, поточний рахунок, розміри матриці та саму матрицю, щоб відновити стан гри.

3. **New():**

- Ця функція генерує нове поле гри з випадковими значеннями. Вона створює нову матрицю з вказаними розмірами, а потім заповнює її випадковими значеннями, викликаючи функцію **NewElement** задану кількість разів.

4. **NewElement():**

- Ця функція генерує новий елемент у випадковій пустій клітинці матриці гри. Вона випадковим чином вибирає координати для нового елементу та його значення, додаючи його до матриці гри.

5. **UndoAllowed():**

- Ця функція перевіряє, чи можливо скасувати останню дію в грі. Вона повертає `true`, якщо стек дій не порожній, тобто є що скасовувати.

6. **Undo():**

- Ця функція скасовує останню дію в грі. Вона відновлює попередній стан матриці гри і поточний рахунок.

7. **Score():**

- Ця функція повертає поточний рахунок гри, який відображає кількість очок, набраних гравцем.

8. **RowCount():**

- Ця функція повертає кількість рядків у матриці гри.

9. **ColumnCount():**

- Ця функція повертає кількість стовпців у матриці гри.

10. **GetMatrix():**

- Ця функція повертає копію поточної матриці гри. Вона створює нову матрицю і копіює значення з поточної матриці до нової, яку потім повертає.

11. **TryMoveDown():**

- Ця функція спробує виконати хід униз на поле гри. Вона викликає внутрішню функцію **Move()** з параметром **down**, що означає рух униз.

12. **TryMoveUp():**

- Ця функція аналогічна попередній, але спробує виконати хід угору на поле гри.

13. **TryMoveLeft():**

- Ця функція аналогічна попередній, але спробує виконати хід вліво на поле гри.

14. **TryMoveRight():**

- Ця функція аналогічна попередній, але спробує виконати хід вправо на поле гри.

15. **Move():**

- Ця внутрішня функція виконує фактичний рух блоків у вказаному напрямку. Вона перевіряє можливість руху, виконує рух та оновлює рахунок. Якщо рух був успішним, вона також викликає функцію NewElemen для додавання нового блоку.

16. **UndoAllowed():**

- Ця функція повертає логічне значення, яке показує, чи можна виконати відміну останнього ходу. Вона перевіряє, чи в стеку історії є елементи для відміни.

17. **Undo():**

- Ця функція відмінює останній хід, повертаючи гру до попереднього стану. Вона використовує стек історії для отримання попередньої матриці і рахунку.

18. **TrySaveGame():**

- Ця функція спробує зберегти поточний стан гри у файл. Вона перевіряє, чи існує файл та чи вдалося відкрити його для запису. Потім вона записує в файл мінімальне значення, рахунок, розмір матриці, матрицю та максимальну довжину історії.

19. **TryLoadGame():**

- Ця функція спробує завантажити стан гри з файлу. Вона перевіряє, чи існує файл та чи вдалося відкрити його для читання. Потім вона читає з файлу мінімальне значення, рахунок, розмір матриці, матрицю та максимальну довжину історії.

20. **SetLine():**

- Ця функція встановлює значення для рядка у матриці. Вона приймає вхідні параметри, такі як матриця, рядок, початкова і кінцева точки, і встановлює значення елементів у вказаному рядку згідно з переданим рядком.

21. **LineShiftAttempt():**

- Ця функція виконує спробу здвигу лінії матриці відповідно до правил гри "2048". Вона приймає одновимірний масив (рядок) та спробує виконати зсув цього рядка відповідно до правил гри. Якщо зсув відбувся, функція повертає суму отриманих очок, в іншому випадку -1.

22. **CanBeMoved():**

- Ця функція перевіряє, чи можна здійснити будь-який зсув у матриці в будь-якому напрямку згідно з правилами гри "2048". Вона перевіряє кожен елемент матриці та його сусідів, щоб визначити можливість зсуву.

23. **Push():**

- Якщо стек досягнув максимальної довжини, вона видаляє останній елемент зі стеку перед додаванням нового.

ВИСНОВОК

У цій дипломній роботі я розробив ігровий додаток на платформі Windows Forms із використанням мови програмування C# та системи управління базами даних MySQL. Під час роботи я детально розглянув теоретичні основи використання цих технологій, а також реалізував ключові аспекти розробки ігрових додатків.

Основні досягнення роботи включають:

1. **Розробка функціонального інтерфейсу користувача**: Я створив зручний та інтуїтивно зрозумілий інтерфейс користувача, який дозволяє легко взаємодіяти з додатком. Особливу увагу я приділив можливості зміни мови інтерфейсу, що підвищує зручність використання додатку для користувачів з різних країн.

2. **Інтеграція з базою даних MySQL**: У додатку я реалізував зберігання облікових даних користувачів, а також результатів ігор у базі даних MySQL. Це дозволяє забезпечити високу надійність збереження даних та можливість їх подальшої обробки та аналізу.

3. **Реалізація ігрових механік**: Додаток включає дві гри – "2048" та "Крестики-нолики", кожна з яких має свої особливості та цікаві механіки. Я забезпечив можливість збереження результатів ігор, що дозволяє користувачам змагатися та порівнювати свої досягнення.

Результати цієї роботи можуть бути використані як основа для подальшого розвитку та удосконалення ігрових додатків. Використання сучасних технологій, таких як Windows Forms та MySQL, забезпечує широкі можливості для створення високоякісних програмних продуктів. Отримані знання та навички можуть бути застосовані у різних сферах розробки програмного забезпечення, що робить цю роботу цінним внеском у мій професійний розвиток.

ПОСИЛАННЯ НА ДЖЕРЕЛА

1. docs.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-6.0

2. www.tutorialspoint.com/csharp/csharp_net_windows_forms.htm

3. www.c-sharpcorner.com/ebooks/c-sharp-winforms-programming-in-visual-studio-2019

4. www.youtube.com/watch?v=YGSmROG1TK8

5. www.udemy.com/course/windows-forms-programming-in-csharp/

6. www.dotnetperls.com/winforms

7. www.codeproject.com/Articles/tag/winforms

8. www.c-sharpcorner.com/technologies/winforms

9. www.youtube.com/watch?v=kDhXuRg7bGQ

10. docs.microsoft.com/en-us/dotnet/desktop/winforms/controls/?view=netdesktop-6.0

11. mva.microsoft.com/en-us/training-courses/c-fundamentals-for-absolute-beginners-16169

12. www.guru99.com/c-sharp-windows-forms.html

13. www.youtube.com/watch?v=qQORMR1n6LQ

14. www.tutorialspoint.com/csharp/csharp_net_windows_forms.htm

15. www.codeproject.com/Articles/14062/Windows-Forms-Best-Practices

ДОДАТОК

Використанні технології:

В якості мови програмування я обрав C#.

Для написання самого коду я обрав Microsoft Visual Studio, тому що для мене він є самим зручним.

Як систему управління версіями проекту я обрав GitHub.

Для збереження результату користувачів та їх облікових записів я вибрав MySql.

І як сервер мною був обраний MAMP.