МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЛЬВІВСЬКА ПОЛІТЕХНІКА



**АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ**

**КОМП’ЮТЕРНИХ СИСТЕМ**

Завдання 3: “ Реалізація серверної (HW) і клієнтської (SW) частин гри”

Виконав:

ст. гр. КІ-404

Мох М. П.

Приняв:

Федак П.Р.

Львів – 2024

**ЗАВДАННЯ**

1. Розробити клієнтську і серверну частини гри.

**Варіант - 12**





**ТЕОРИТИЧНИЙ МАТЕРІАЛ**

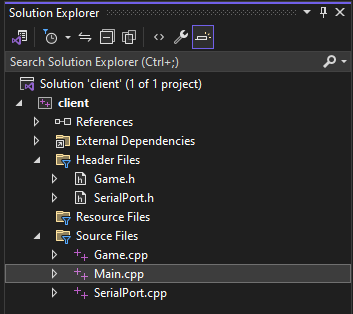
**UART (Universal Asynchronous Receiver-Transmitter)** — це протокол для асинхронної послідовної передачі даних між двома пристроями, наприклад, між комп'ютером і мікроконтролером (таким як Arduino). UART дозволяє пристроям обмінюватися інформацією через два провідники: один для передачі даних (TX) і один для прийому (RX). Головна особливість UART полягає в тому, що він не потребує синхронізуючого сигналу, що робить його простим у налаштуванні.

**CMake** – це кросплатформений інструмент автоматизації збірки, який використовують для генерації конфігураційних файлів (наприклад, Makefile, Ninja) для компіляції програм. Його перевага в тому, що він дозволяє легко створювати збірки на різних системах.

**YML або YAML (Yet Another Markup Language)** — це формат даних, який широко використовується для написання конфігураційних файлів. YAML є зручним через свою просту, читабельну структуру. Він застосовується для опису налаштувань у багатьох системах автоматизації, наприклад, CI/CD-процесах у GitHub Actions, GitLab CI тощо.

**Файл формату `.ini`** — це текстовий файл конфігурації, який часто використовується для зберігання налаштувань програм. Він складається з розділів (секцій), позначених у квадратних дужках, кожен з яких містить параметри у вигляді пар "ключ-значення". Структура `.ini` файлу дозволяє легко читати й редагувати його як вручну, так і програмно, що робить його зручним для збереження налаштувань, конфігурацій користувача або даних гри. Цей формат є універсальним та підтримується багатьма мовами програмування, зокрема C++ через бібліотеки для роботи з конфігураційними файлами.

**ВИКОНАННЯ РОБОТИ**

1. Створіть комунікаційну схему SW (клієнт)<-> UART <-> HW (сервер). 

**Клієнт:** у файлі serialport.h визначено структуру класу, його методи та змінні, а в serialport.cpp реалізовано всі методи. У файлі game.cpp містяться функції для збереження стану гри, відновлення попередньої гри через завантаження її з файлу game.ini, а також для відображення меню, де можна вибрати режим гри та розпочати нову гру. Прототипи функцій визначені у файлі game.h. Головний файл — main.cpp, де викликається функція playGame(), що ініціює гру в консольному додатку та забезпечує взаємодію з сервером, отримуючи результат перемоги гравця.

**Сервер:** сервер містить код, що реалізує логіку гри, зокрема режими гри, визначення переможця, прийом повідомлень від клієнта та надсилання результатів гри клієнту для відображення та подальшого збереження стану гри.

1. Демонстрація функціоналу гри та опис режимів.

У грі реалізовано чотири режими, кожен з яких дозволяє користувачам або програмі взаємодіяти в різних сценаріях. Ось детальний опис кожного режиму:

### **Player vs Player (Гравець проти Гравця)**

Цей режим передбачає гру між двома реальними гравцями. Обидва користувачі по черзі вводять свої ходи (наприклад, "rock", "paper" або "scissors") через консольний інтерфейс. Гра розрахована на чесність і взаємодію між гравцями, тому ходи не відображаються до завершення раунду.  
 Після введення обох ходів система передає дані до мікроконтролера через **SerialPort**, де обчислюється результат. Мікроконтролер повертає результат раунду у вигляді текстового повідомлення.

### **Player vs AI (Random)**

У цьому режимі гравець змагається проти штучного інтелекту, який робить ходи випадково.  
Процес виглядає так:

1. Гравець вводить свій хід через консоль.
2. Хід передається через **SerialPort** разом із зазначенням режиму.
3. AI випадково вибирає хід ("rock", "paper" або "scissors") на основі генератора випадкових чисел, реалізованого в мікроконтролері.
4. Результат раунду повертається через **SerialPort** у програму.

Цей режим забезпечує виклик для гравця, оскільки результат раунду залежить лише від удачі та стратегій гравця.

### **Player vs AI (Win Strategy)**

У цьому режимі гравець протистоїть AI, який використовує просту стратегію для підвищення своїх шансів на перемогу.  
Особливості стратегії AI:

* Аналізує хід гравця з попереднього раунду.
* Підбирає хід, який найімовірніше приведе до виграшу, враховуючи ймовірні патерни в ходах гравця.  
  Наприклад, якщо в попередньому раунді гравець обрав "rock", AI може обрати "paper" як контрхід.

Процес виглядає так:

1. Гравець вводить свій хід через консоль.
2. Хід разом із інформацією про попередні раунди передається до мікроконтролера через **SerialPort**.
3. AI обирає оптимальний хід відповідно до стратегії та повертає результат.

Цей режим випробовує здатність гравця адаптувати свою стратегію, оскільки AI стає більш передбачуваним, якщо гравець зберігає однакову тактику.

### **AI (Random) vs AI (Win Strategy)**

У цьому режимі два штучних інтелекти змагаються між собою, причому один із них приймає випадкові рішення, а інший діє на основі стратегії перемоги.  
Процес:

1. Гравець ініціює раунд, вводячи команду "start" у консоль.
2. Програма передає мікроконтролеру команду, в якій зазначається цей режим.
3. AI (Random) випадково обирає свій хід, а AI (Win Strategy) аналізує результати попереднього раунду та обирає оптимальний хід для перемоги.
4. Мікроконтролер повертає результат раунду через **SerialPort**, включаючи вибрані ходи обох AI та результат.

Цей режим є демонстраційним і підходить для спостереження за поведінкою двох типів AI. Він дозволяє оцінити ефективність стратегії AI (Win Strategy) проти випадкових ходів.

Демонстрація роботи програми у різних режмах гри та локальне тестування проекту за допомогою утиліти cmake:

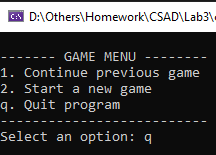


Рис. 1. Головне меню

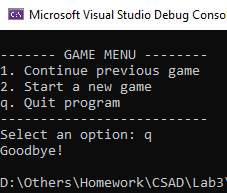


Рис. 2. Вихід з гри

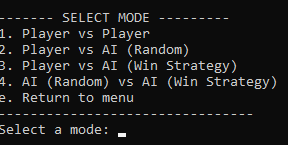


Рис. 3. Перехід в Start a new game

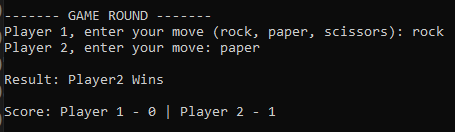


Рис. 4. Режим Player vs Player

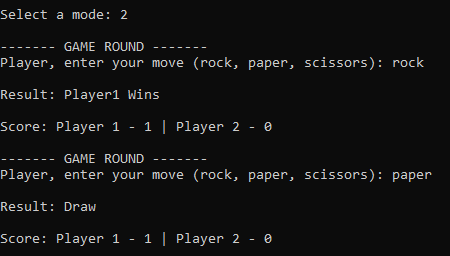


Рис. 5. Режим Player vs AI (Random)

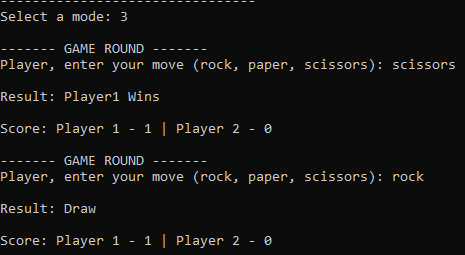


Рис. 6. Режим Player vs AI (Win Strategy)

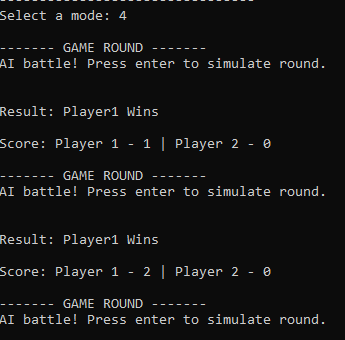


Рис. 7. Режим AI (Random) vs AI (Win Strategy)

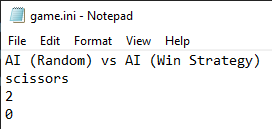


Рис. 8. Збереження гри в форматі .ini

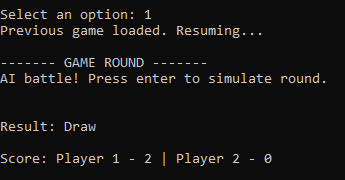


Рис. 9. Продовження цієї гри

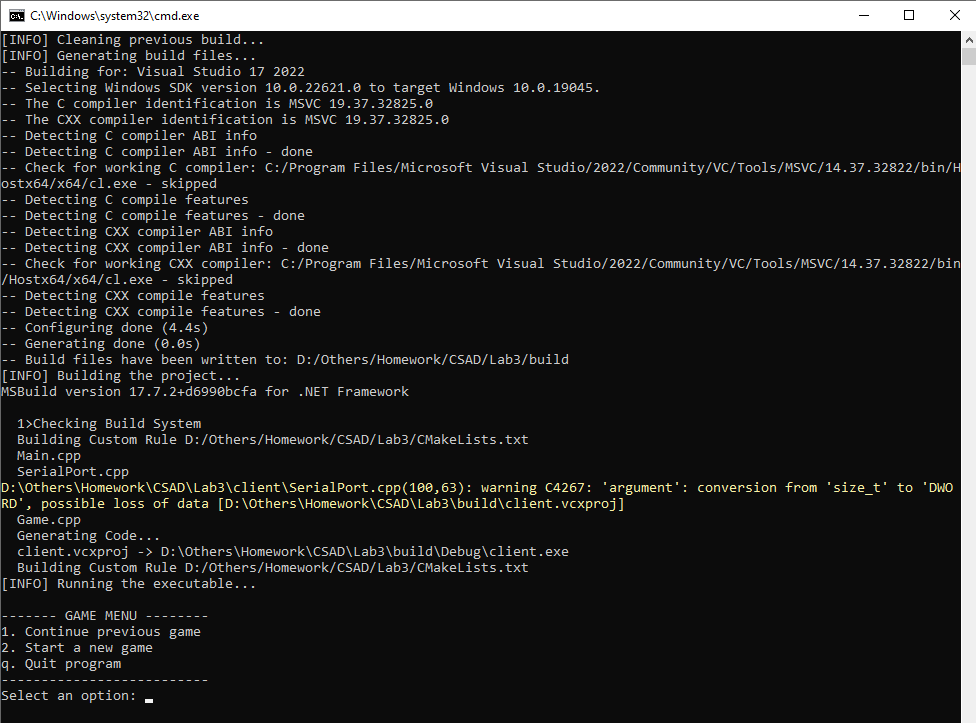


Рис 10. Локальне тестування за допомогою Cmake

**ВИСНОВОК**

Розроблено багатофункціональну гру "Камінь-ножиці-папір" з архітектурою клієнт-сервер (ПК - клієнт, Arduino - сервер). Реалізовано 4 ігрові режими, включаючи гру з AI. Для зручності користувачів передбачено збереження стану гри та продовження з місця зупинки. Ведення статистики перемог дозволяє аналізувати результати гри.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Arduino. "Arduino Documentation." Arduino.cc. <https://docs.arduino.cc/>
2. GitHub. "GitHub Actions Documentation." GitHub Docs. <https://docs.github.com/en/actions>
3. Microsoft Documentation on Batch Scripts: <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490887(v=technet.10)>
4. SparkFun. "UART Communication Basics." Learn.sparkfun.com. <https://learn.sparkfun.com/tutorials/serial-communication>
5. Cplusplus. "C++ Standard Library Documentation." <https://en.cppreference.com/w/cpp>