

# Dokumentacja do projektu

## **Connect Four**

z przedmiotu

# Języki programowania

Kierunek: Elektronika I

Kacper Filipek

czwartek 14:40

Prowadzący: Rafał Frączek

Kraków, 14 czerwca 2022

### 1. Opis projektu

Projekt jest realizcją gry w "Czwórki" (ang. "Connect Four") w C++, z interfejsem TUI (t.j. interfejs używający znaków specjalnych i kolorów do rysowania interfejsu użytkownika w konsoli).

### 2. Project description

The project is a realisation of the game "Connect Four" made in C++ with TUI interface (an interface using special characters for drawing UI in the terminal)

### 3. Instrukcja użytkownika

#### Menu główne

Po włączeniu programu wyświetla się menu główne z następującymi opcjami:

- Play rozpoczyna grę w bieżącymi ustawieniami
- Load game umożliwia wczytanie pliku z zapisem gry
- Settings otwiera menu umożliwiające zmianę ustawień gry
- Quit game wychodzi z programu

Nawigacja pomiędzy opcjami odbywa się za pomocą strzałek w górę i w dół. Wciśnięcie klawisza Enter spowoduje wykonanie akcji związanej z wybranym elementem menu

Wybranie "Load game" wczyta stan gry z pliku z zapisem save.bin. W przypadku błędu w odczycie pliku, wyświetli się komunikat powiadamiający o wystąpieniu błędu.

#### Ustawienia

W menu "Settings" znajdują się opcje do zmienienia ustawień gry. Wokół liczby przy wybranej opcji pojawią się strzałki. Parametry te można zmienić używając strzałek w lewo i w prawo. Nawigacja pomiędzy opcjami odbywa się za pomocą strzałek w górę i w dół.

- Rows liczba wierszy planszy (domyślnie 5)
- Columns liczba kolumn planszy (domyślnie 7)
- Win condition liczba żetonów w sekwencji potrzebna do wygrania gry (domyślnie 4)

W menu ustawień znajdują się również przyciski "Done" i "Cancel". Gdy użytkownik naciśnie Enter mając wybraną opcje "Done", ustawienia zostaną zapisane i włączy się główne menu. Wybranie "Cancel" wyjdzie do głównego menu bez zapisywania zmienionych ustawień.

### Ekran gry

Po rozpoczęciu gry, na ekranie wyświetli okno zostanie podzielone na dwie części. Górna część zawiera elementy interfejsu informujące o aktualnym graczu i wybranej kolumnie, a dolna - planszę do gry Zmiana wybranej kolumny odbywa się przy pomocy strzałek w lewo i w prawo. Po wciśnięciu klawisza Enter, do wybranej kolumny zostanie wrzucony żetonu koloru odpowiadającego aktualnemu graczowi. Po wrzuceniu żetonu następuje tura kolejnego gracza. Gracz który jako pierwszy ułoży odpowiednią liczbę żetonów w prostej linii lub po przekątnej wygrywa.

Jeśli program zostanie zamknięty w czasie gry lub podczas wyświetlania menu pauzy poprzez wysłanie sygnału SIGTERM, aktualny stan gry zostanie zapisany do pliku .autosave.bin. Jeśli program zostanie otwarty ponownie, sprawdzi czy plik o tej nazwie istnieje w aktualnym folderze. Jeśli plik z autozapisem istnieje, zostanie on wczytany bezpośrednio po włączeniu programu, z pominięciem głownego menu.

Niestety nie jest możliwy automatyczny zapis gry przy zamknięciu okna, ponieważ zamknięty terminal wysyła wszystkim procesom w nim otwartym sygnał SIGKILL, którego obsłużenie nie jest możliwe.

#### Menu pauzy

Wciśnięcie klawisza p podczas gry spowoduje pauzę w grze i wyświetlenie się menu z następującymi opcjami:

- Resume wznawia przebieg gry (alternatywnie można ponownie wcisnąć p)
- Save game zapisuje stan gry do pliku save.bin
- Main menu kończy grę i przechodzi do głównego menu bez zapisu

### 4. Kompilacja

Program został napisany na systemy operacyjne z rodziny Linux i nie jest możliwe skompilowanie go na system Windows, ponieważ on Unixowegy systemych sygnałów SIGTERM i SIGWINCH. Prawdopodobnie jest możliwa kompilacja programu na systemy z rodziny BSD, jednak program nie był na nich testowany.

Program można skompilować na dwa sposoby:

- Sposób 1:
  - 1. Wejść do folderu build/
  - 2. Wykonać polecenie cmake .. && make
- Sposób 2:
  - 1. Z katalogu głównego wywołać skrypt ./bld.sh.

Po zbudowaniu powienien się plik wykonywalny ./build/connect-four. Z uwagi na fakt, że ścieżki do zasobów są wpisane w programie relatywnie do głównego katalogu, to program wykonywalny powinien z niego wywoływany.

#### 5. Pliki źródłowe

W tym punkcie należy opisać wszystkie pliki źródłowe (.cpp, .h) w projekcie. Należy podać nazwę każdego pliku oraz informację o tym co się w nim znajduje. Na przykład: Projekt składa się z następujących plików źródłowych:

- board.h, board.cpp deklaracja oraz implementacja klasy Board,
- game.h, game.cpp deklaracja oraz implementacja klasy Game,
- menu.h, menu.cpp deklaracja oraz implementacja klasy Menu.
- extras.h, extras.cpp deklaracja oraz implementacja funkcji pomocniczych.
- color.h definicje procesora nazw kolorów do użycia w funkcjach biblioteki ncurses.
- main.cpp główny plik z implementacją funkcji main.

#### 6. Zależności

W projekcie wykorzystano następujące dodatkowe biblioteki:

 ncurses – biblioteka do interakcji z emulatorem terminala, pozwala tworzyć zaawansowane interfejsy konsolowe:

https://invisible-island.net/ncurses/

### 7. Opis klas

W projekcie utworzono następujące klasy:

- Menu klasa reprezentująca menu główne programu
  - Menu()
  - GameParameters Start() rozpoczyna pętlę klasy Menu i zwraca strukturę z parametrami gry po jej zakończeniu

- void update\_dimensions() ustawia wymiary elementów w zależności od rozmiaru okna
- Game klasa reprezentująca stan i właściwy przebieg rogrywki
  - Game() domyślny konstruktor klasy Game
  - void Start() ustawia parametry i rozpoczyna pętlę gry
  - bool set\_parameters (GameParameters parameters) ustawia parametry gry na podstawie podanego argumentu i zwraca informację o powodzeniu metody
  - bool save\_game(const std::string& path) zapisuje stan gry do pliku podanego przez path
     i zwraca informację o powodzeniu metody
  - void update\_dimensions() ustawia wymiary elementów w zależności od rozmiaru okna
- Board klasa reprezentująca planszę do gry
  - Board() domyślny konstruktor, ustawiający domyślne parametry
  - Board(uint8\_t board\_rows, uint8\_t board\_columns) konstruktor parametryczny ustawia rozmiar planszy na board\_rows × board\_rows
  - uint16\_t get\_columns() zwraca liczbę kolumn planszy
  - uint16\_t get\_rows() zwraca liczbę wierszy planszy
  - uint16\_t get\_win\_condition() zwraca warunek zwycięstwa
  - void set\_win\_condition() ustawia warunek zwycięstwa
  - void set\_dimensions(uint16\_t rows, uint16\_t columns) ustawia wymiary planszy
  - uint8\_t check\_victory() sprawdza czy któryś z graczy wygrał
  - void clear() czyści planszę
  - std::vector<uint8\_t>::iterator operator[] (int index) zwraca iterator do początku podanego wiersza. Metoda ta umożliwia odnoszenie się do elementów planszy dwoma indeksami np. board[row][col]

### 8. Zasoby

W projekcie wykorzystywane są następujące pliki zasobów:

- assets/ katalog zawierający dodatkowe zasoby do gry. Struktura katalogu:
  - logo1.txt, logo2.txt pliki zawierające tekstowe logo pojawiające się na ekranie startowym.

### 9. Dalszy rozwój i ulepszenia

- Mimo, że gra oryginalnie gra została zaprojektowana dla dwóch graczy, to nie ma żadnych powodów dla których w grę nie mogłaby grać większa liczba graczy. Zapis gry już przechowuje liczbę graczy, więc implementacja funkcji wielu graczy sprowadza się do niewielkich zmian w klasie Game i Board oraz dodanie opcji do menu ustawień.
- Aktualnie gra nie posiada żadnej możliwości gry dla jednego gracza. Możliwa jest implementacja sztucznej inteligencji z różnymi poziomami trudności.
- Możliwa jest implementacja logowania ruchów i wyników graczy.
- Przy większym nakładzie pracy można zaimplementować funkcjonalność multi-player poprzez protokoły sieciowe, jednak wymagałoby to znacznej przebudowy programu.
- Gra w obecnym stanie działa w terminalu. Z tego powodu nie jest możliwa obsługa sytuacji w której gracz zamyka okno terminala poprzez zastosowanie sygnałów. Problem ten mógłby zostać rozwiązany poprzez napisanie faktycznego interfejsu okienkowego (np. w Qt lub GTK), dzięki czemu można by obsługiwać zamknięcie okna.
- Alternatywnym rozwiązaniem tego problemu jest zapisywanie gry po każdym ruchu do tymczasowego
  pliku z zapisem. Jeśli program zakończyłby swoje działanie bez problemu, plik mógłby być usunięty. W
  przypadku wysłania sygnału SIGTERM lub SIGKILL, plik z zapisem pozostawałby w folderze.

### 10. Inne

#### 10.1. Format zapisu

Stan gry jest zapisywany do pliku w formie binarnej. Poniżej podane są po kolei bajty oznaczające konkretne parametry gry oraz ich rozmiary w pliku

- 1. Liczba graczy 1 bajt (unsigned)
- 2. Aktualny gracz 1 bajt (unsigned)
- 3. Wymiary planszy 4 bajty (unsigned)
  - 3.1. liczba wierszy w planszy 2 bajty (unsigned)
  - 3.2. liczba kolumn w planszy 2 bajty (unsigned)
- 4. Warunek zwycięstwa 2 bajty (unsigned)
- 5. zawartość planszy każdej komórce odpowiada jeden bajt, więc rozmiar tej sekcji w bajtach, to liczba kolumn × liczba wierszy (unsigned)

Parametry większe niż jeden bajt są zapisywane w formacie little endian (najmniej znaczący bajt na początku).