Projekt PROI 22Z – Szachy

Milan Wróblewski, Miłosz Mizak, Jakub Podrażka

Na czym polegało zadanie?:

Naszym zadaniem była implementacja gry w szachy. Całość implementacji mieliśmy napisać, zgodnie z założeniami przedmiotu, w języku C++. Program miał umożliwiać:

- grę z drugą osobą („vs human”)
- grę z komputerem (algorytm dowolny) („vs computer”)
- zapis przebiegu rozgrywki (do pliku .txt)

Opis:

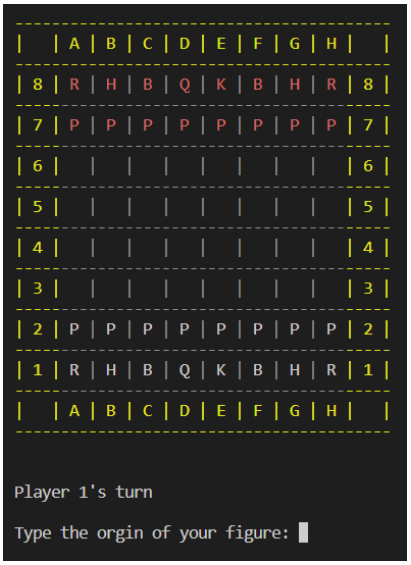
- Program zrealizowaliśmy w języku angielskim, przy czym szachy są grą, w którą (poza wyborem odpowiedniej opcji w menu) da się grać nie patrząc na ew. bariery językowe.
- **Program działa poprawnie na systemie Linux (dowolnej jego dystrybucji czy poprzez wirtualizację).**
- Polecamy wydłużenie okna terminala/konsoli, aby wszystkie informacje oraz szachownica były w pełni widoczne.

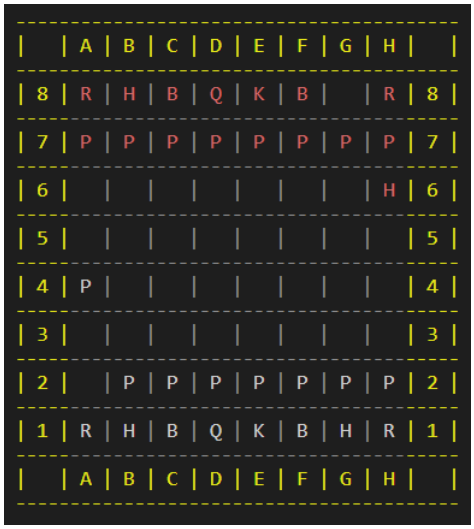
Jak działa program?:

Po uruchomieniu programu, w konsoli systemowej graczowi ukazuje się menu gry. Do wyboru są 2 opcje:

- Human – opcja ta pozwala na grę dwóch osób ze sobą (tryb „Human” -> „vs Human”) lub na grę użytkownika z komputerem (tryb „Human” -> „vs Computer”). Gracz wybierający tryb gry dostaje także możliwość wyboru koloru pionków, którymi będzie się poruszał;
- Ai vs Ai – opcja ta uruchamia rozgrywkę, w której naprzeciw siebie stają 2 boty.

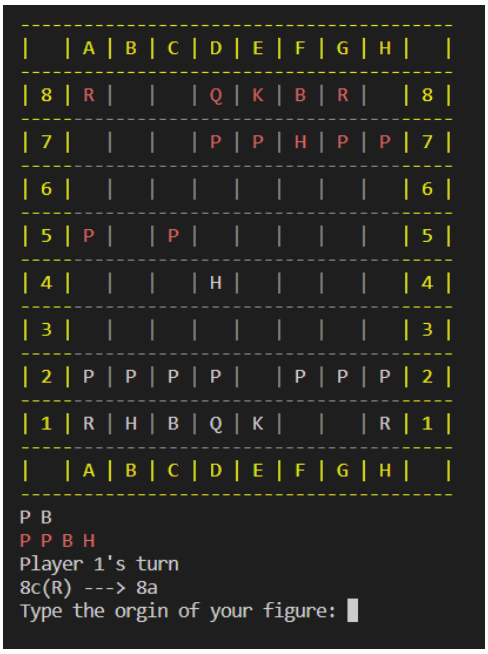
Po wybraniu interesującej użytkownika funkcji, w konsoli wyświetli się plansza do gry z odpowiednio ustawionymi pionkami (czarne figury są koloru czerwonego, a białe – białego), a także informacja o tym, kto wykonuje w danym momencie ruch. Pod nią natomiast program zapyta się obecnie wykonującego turę gracza, jaki ruch pragnie wykonać. W tym celu musi on podać najpierw początkowe położenie figury, którą chce się ruszyć, a następnie pole, na którym chce ją postawić.

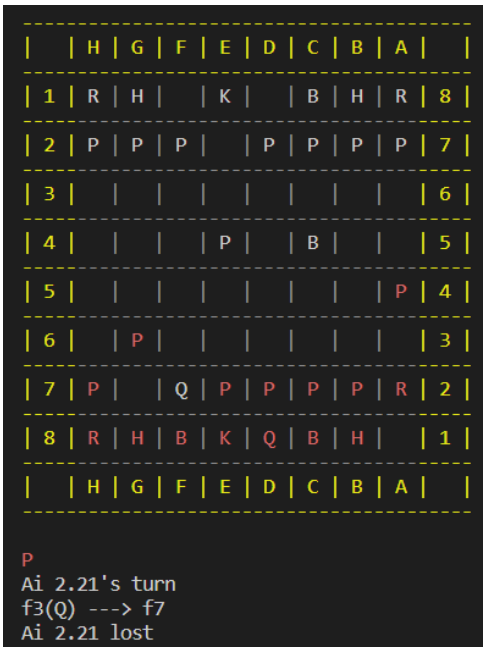




W tym przypadku każdy gracz wykonał po 1 ruchu: gracz 1 ruszył figurą A2 na pole A4, gracz 2 – figurą G8 na pole H6.

Wraz z rozwojem gry dojdzie do zbicia niektórych figur z planszy – zbite figury będą wyświetlały się pod planszą (oddzielnie dla figur czarnych i białych). Ponadto, pod informacją o tym, który gracz wykonuje ruch, znajduje się przedstawienie ostatniego wykonanego przez przeciwnika ruchu. Gdy jeden z graczy przegra, program wyświetla stosowną informację.





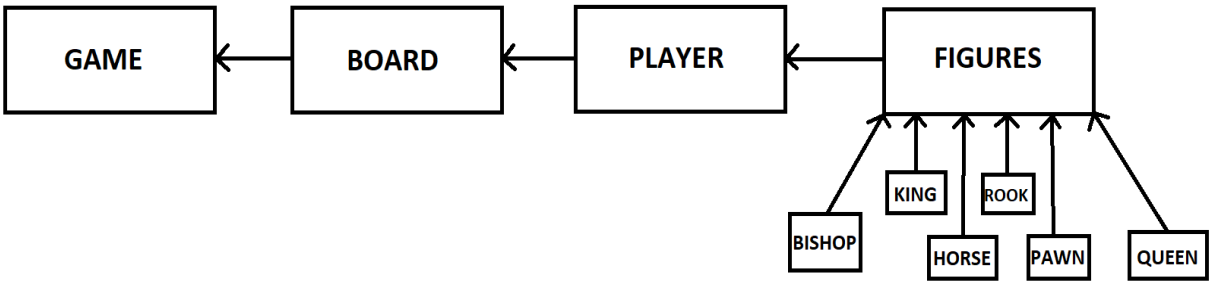
Udało nam się również zaimplementować specjalne ruchy czy innego rodzaju „akcje”, takie jak promocja pionka na dowolną inną figurę, gdy ten dojdzie do „obozu” przeciwnika czy roszada.

Strona techniczna projektu:

Implementację zdecydowaliśmy się podzielić na 4 główne części – Figures, Player, Game oraz Board.

- Figures – każda figura przechowuje swoje oznaczenie (np. dla damy będzie to ‘Q’, od Queen), kolor, zbiór swoich ruchów, a także informację o tym, czy została ona zbita czy też nie.
- Player – każdy gracz przechowuje informację o tym, czy jest człowiekiem (jeśli nie, to jest to komputer/AI), kolor figur, którymi się może ruszać, nazwę, dostępne figury i pozycję króla (potrzebna np. do sprawdzenia, czy nie wystąpił szach i/lub mat). Poza getterami i seterami, dostępna jest również metoda pozwalająca na dodanie nowej figury (wykorzystujemy ją przy promocji pionka).
- Board – przechowuje wskaźniki na obiekty graczy, dwuwymiarową tablicę reprezentującą pola na szachownicy, „cmentarz” (graveyard) dla zbitych figur oraz „czyściec” (pulgatory), wykorzystywany przy symulacji ruchu. Board może przestawić figurę, usunąć ją, wykonać promocję pionka, sprawdzić poprawność ruchu, a także, czy nie ma szacha i/lub mata. Klasa ta odpowiada też za wykonanie roszady, kończenie rundy i zamianę gracza wykonującego ruch oraz za wypisanie na ekran szachownicy.
- Game – stanowi „centrum” całego programu. Przechowuje wskaźniki na obiekty graczy oraz na szachownicę. Odpowiada za wyświetlanie menu, odpowiednich, istotnych dla gracza informacji oraz za przekazywanie otrzymanych od niego odpowiedzi dalej.

Przy tworzeniu klasy Board postanowiliśmy podzielić ją na kilka modułów – plik board.cpp zawiera metody związane z wykonywaniem ruchu, takie jak sprawdzenie czy ruch jest poprawny, bicie, samo wykonanie ruchu itp., check.cpp – metody odpowiadające za sprawdzenie, czy w danej sytuacji nie ma szacha i/lub mata, promocję pionka oraz za wykonanie roszady (zarówno krótkiej, jak i długiej), a print.cpp – za wyświetlanie szachownicy oraz „cmentarza” figur. „Modułowo” stworzyliśmy także klasę Figure. Ponieważ figury znacznie różnią się między sobą wykonywanymi ruchami, każda z nich otrzymała własny plik, w którym zdefiniowane są m.in. jej możliwe ruchy czy wyświetlany na planszy symbol.



Kto nad czym pracował?:

- Milan Wróblewski – logika ruchu, szacha, mata, roszady
- Miłosz Mizak – testy, figury, główna pętla gry, naprawianie błędów
- Jakub Podrażka – wyświetlanie szachownicy, innych informacji/interfejsu, dokumentacja, czyszczenie kodu, obsługa „cmentarza” dla figur (graveyard)