Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki

Laboratorium Programowania Komputerów

Warcaby

autor	Paweł Tomaszewski
prowadzący	mgr inż. Wojciech Dudzik
rok akademicki	2017/2018
kierunek	informatyka
rodzaj studiów	SSI
semestr	2
termin laboratoriów / ćwiczeń	piątek, 10:00 –11:30
grupa	3
termin oddania sprawozdania	2018-06-21
data oddania sprawozdania	2018-06-18
adres do programu	https://github.com/tomaszewskipablo/checkers

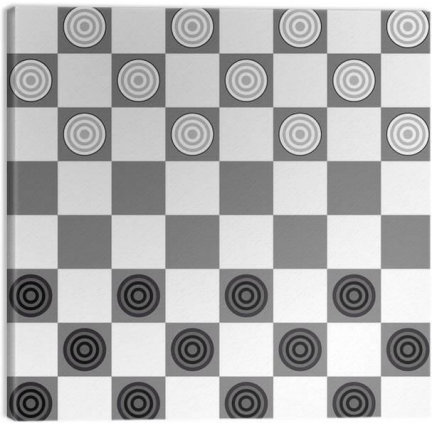
1 Treść zadania

Napisać program realizujący grę warcaby.

2 Ogólny opis działania programu

Program może być uruchamiany z linii poleceń. Akceptowane są 2 różne parametry, wpisywane pojedynczo. Program uruchomiony wraz z parametrem –h uruchomi najpierw krótką pomoc, a później grę. Wywołanie programu z parametrem –i wraz ze ścieżką do pliku spowoduje wywołanie programu z początkowym ustawieniem pionów na szachownicy jak określono w pliku.

Program rozpoczyna się od inicjalizacji tablicy dwuwymiarowej reprezentującej szachownice 8x8 dla początkowego ustawienia pionów.



Funkcja czytająca argumenty wywołania programu sprawdza czy użytkownik chce załadować zapisane w pliku ustawienie pionów na planszy. W main pętla while wykonuje się tak długo aż gracz którego tura nastąpi nie ma żadnego piona na planszy lub nie ma możliwości wykonania ruchu. Jeśli pętla się zakończy gracz, którego tura właśnie miała nastąpić przegrywa. Po każdym przebiegu pętli następuje zmiana gracza dla którego wykonywać będzie następna pętla. Jedyną funkcją jaką wywołuje pętla jest funkcja `YourMove`, która realizuje przebieg ruchu. Najpierw wywołuje funkcje `whichManToMove`, w której użytkownik wybiera piona. Następnie wywoływana jest `funkcja whereToMove` – odpowiedzilna za miejsce w które pion zostanie przesunięty.

3 Funkcje

PLIK administration.h

```
void YourMove(int tab[8][8], int player);
```
Funkcja sterująca przebiegiem ruchu gracza. Wywołuje ona funkcje whichManToMove, aby pozwolić graczowi wybrać piona i whereToMove – wybranie miejsca w które pion ma się ruszyć. Funkcja aktualizuje również statystyki.

```
void whichManToMove(int tab[8][8], int player, char * str1);
```
Funkcja pozwala wybrać piona którym chcemy wykonać ruch. Sprawdza czy na wskazanym przez nas polu znajduje się nasza figurą i czy jest ona wstanie wykonać jakikolwiek ruch, jeśli nie, funkcja nie pozwoli wybrać graczowi danego piona.

```
void whereToMove(int tab[8][8], int player, char * str1);
```
Funkcja odpowiada za finalizację ruchu - wybór pola na które gracz chce się przemieścić pionem wybranym w funkcji whichManToMove. Jeśli pole wybrane przez gracza nie jest legalne dla danego

piona, gracz będzie musiał wybrać właściwe. Jeśli po wykonanym biciu pion ma szansę zbić kolejną figurę przeciwnika musi to zrobić. Co realizowane jest w funkcji haveToBeat.

PLIK displayInformation.h

```
void displayChessboard(int tab[8][8], int player, int x, int y, char * info)
```
Funkcja wyświetlajca szachownicę wraz z pionkami używa ona biblioteki rlutil.h w celu wyświtlania kolorów pionków. Wyświetla ona również komunikaty. Do zbudowania szachownicy zostały użyte znaki extended ASCII.

```
void displayHelp()
```
Funkcja wyświetlająca krótką pomoc wywoływaną jako argument programu parametr –h.

PLIK checking.h

```
int stillOnTheBoard(int tab[8][8], int x, int y, int horizontalMove, int
verticalMove);
```
Funkcja sprawdza czy ruch dla piona nie wykracza po za plansze.

```
int blankSpot(int tab[8][8], int x, int y, int horizontalMove, int verticalMove);
```
Funkcja sprawdza czy wybrane miejsce jest puste.

```
int opponentsMan(int tab[8][8], int x, int y, int horizontalMove, int verticalMove,
int player);
```
Funkcja sprawdza czy na wybranym miejscu znajduje się pion przeciwnika.

```
int possibleNormalMove(int tab[8][8], int x, int y, int player);
```
Funkcja sprawdza czy pion może gdziekolwiek się przemieścić (bez bicia).

```
int possibleBeat(int tab[8][8], int x, int y, int color);
```
Funkcja sprawdza czy pion może bić pion przeciwnika.

```
int PossibleToMoveManToSpot(int tab[8][8], int player, int newX, int newY, int oldX,
int oldY);
```
Funkcja sprawdza czy pion może ruszyć się we wskazane miejsce.

```
int ManCanMoveAnywhere(int tab[8][8], int x, int y, int player);
```
Funkcja sprawdza czy pion może wykonać ruch.

```
int AnyMovePossible(int tab[8][8], int player);
```

Funkcja sprawdza czy jakikolwiek pion gracz może się ruszyć.

PLIK beating.h

```
int jumpOverOpponent(int tab[8][8], int COLOR, int x, int y, int i, int j)
```

Funkcja realizująca zbijanie piona przeciwnika, gdy jest to nie obowiązkowe.

```
void haveToBeat(int tab[8][8], int x, int y, int player)
```

Funkcja realizująca zbijanie piona przeciwnika, gdy bicie jest obowiązkowe

PLIK control.h

```
void updateStatistic(int tab[8][8]);
```
Funkcja aktualizująca statystyki. Liczy ilość białych i czarnych pionów.

```
void ArrowControl(int tab, int COLOR, int * x, int * y, char * str1);
```
Funkcja obsługująca sterowanie strzałkami w grze. Jest interfacem pomiędzy użytkownikiem, a programem. Pozwala również zakończyć grę lub zapisać ustawienie pionów na planszy.

```
void odczytaj_argumenty(int ile, char ** argumenty, int tab[8][8]);
```
Funkcja czytająca argumenty podane przez użytkownika podczas wywołania programu w konsoli. Pozwala uruchomić krótką pomoc oraz załadować ustawienie pionków z poprzedniej gry.

```
void quit(int tab[8][8]);
```
Funkcja obsługująca zakończenie pracy programu, pozwala na zapisanie ustawienia pionów przed zakończeniem.

PLIK files.h

```
void saveArrayToFile(int tab[8][8]);
```
Funkcja zapisuje ustawienie pionów na planszy do pliku.

```
void readArrayFromFile(char * input, int tab[8][8]);
```
Funkcja czyta ustawienie pionów na planszy z pliku.

4 Wnioski

Praca nad projektem była ciekawa. Podczas realizacji natrafiłem na wiele problemów, które były kompletnie nowe w mojej nauce programowania. Nieświadomy problemów jakie mogą wystąpić w późniejszym etapie pisałem projekt zgodnie z zasadą „od szczegółu do ogółu”. Projekt pozwolił mi zrozumieć jak ważne jest mądre rozplanowanie projektu. Zrozumiałem również jak ważne jest odpowiednie korzystanie ze struktur i własnych typów danych, wiele zadań można by rozwiązać w sposób bardziej optymalny opierając się na własnym typie.

Literatura

[1] Jerzy Grębosz. Opus magnum C++11. Wydawnictwo Helion, Gliwice, 2017.
[2] https://pl.wikibooks.org
[3] http://cpp0x.pl/