

# Warcaby - dokumentacja projektu

Jakub Chomiczewski

## Wstęp

Celem projektu było napisanie programu pozwalającego na rozgrywkę dwóch ludzkich graczy lub ludzkiego gracza przeciwko graczowi komputerowemu w warcaby. Rozgrywka może być przeprowadzana jak i w terminalu jak i w trybie okienkowym przy użyciu biblioteki SDL. Zasady meczu będą zgodne z zasadami zamieszczonymi na stan 30.01.2022 roku na stronie [FMJD](#) (World Draughts Federation).

## Instalacja oraz wymagania

Program wymaga od użytkownika zainstalowania następujących bibliotek:

- [SDL 2.0](#)
- [SDL\\_image](#)
- [FreeType](#)
- [SDL\\_TTF](#)

Następnie, należy pobrać plik Checkers\_project-main.zip lub sklonować repozytorium ze strony [https://github.com/MinionJakub/Checkers\\_project](https://github.com/MinionJakub/Checkers_project). Po pobraniu, rozpakowaniu (plik .zip) oraz wejściu do folderu z plikami przez terminal należy wpisać *chmod 755 kompilacja* i uruchomić plik kompilacja komendą *./kompilacja*, a następnie wpisać *./checkers\_project*. Jeśli będą jakieś błędy to należy wpisać następujące komendy:

- `gcc -o checkers_project draw_in_SDL.c checkers_rules_and_logic.c bot.c main.c draw_in_command_console.c -lSDL2 -lSDL2_image -lSDL2_ttf -lm`
- `./checkers_project`

Po pierwszej kompilacji nie trzeba więcej używać ani pliku kompilacja, ani komendy gcc ani chmod.

**Uwagi:** Program był testowany oraz zalecane jest jego uruchamianie na Linuxie (dystrybucja Ubuntu). Warto również sprawdzić, czy terminal obsługuje

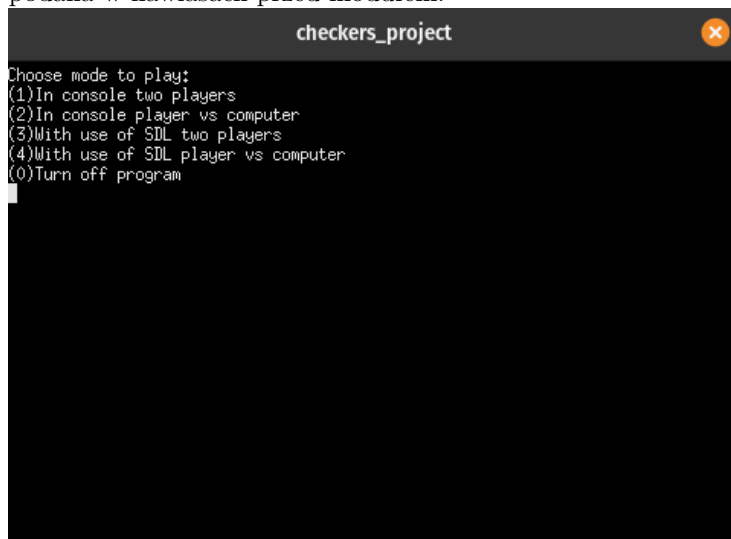
UTF-8. Program należy zawsze uruchamiać poprzez wpierw wejście przez terminal do katalogu z grą i uruchomienie pliku `Uruchom_warcaby`.

**Jeśli użytkownik nie chce instalować biblioteki SDL:** Program pozwala na użytkowanie go bez instalacji wymaganych bibliotek, ale wtedy zamiast `chmod 755 kompilacja` należy wpisać `chmod 755 kompilacja_bez_SDL`. Następnie wpisać `./kompilacja_bez_SDL` i uruchamiać grę komendą `./checkers_without_SDL`.

## Interakcja użytkownika z programem

### Wybór trybu

Użytkownik ma prawo do wyboru między graniem w terminal czy w formacie okienkowym. Podjęcie tej decyzji jest możliwe poprzez wybranie odpowiadającej użytkownikowi opcji w konsoli po przez wpisanie odpowiedniej cyfry, która jest podana w nawiasach przed modulem.



### Obsługa wersji konsolowej

Gracz do dyspozycji ma następujące komendy:

- `surrender` - funkcja pozwalająca na poddanie partii
- `show moves` - funkcja po wpisaniu której są wyświetlane wszystkie legalne ruchy
- `A3B4` - przykład legalnego ruchu po wpisaniu, którego pionek z pola A3 przesunie się na pole B4

- A3C5 - przykład legalnego bicia po wpisaniu, którego pionek z A3 wykona ruch na C5 i zbije wrogiego pionka na B4

**Uwagi:** Po zakończeniu partii w terminalu, wyłączany jest również sam program. Program poprawnie wczyta komendę nawet jeśli użytkownik będzie używał zmiennej wielkości liter czy znaki będą oddzielone spacjami.

## Obsługa wersji okienkowej

Użytkownik ma do dyspozycji następujące komendy:

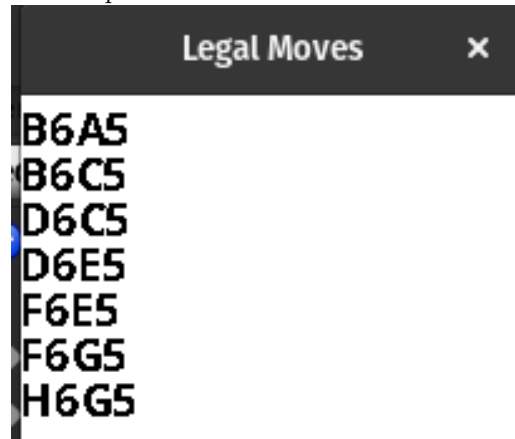
- surrender - funkcja pozwalająca na poddanie partii
- A3B4 - przykład legalnego ruchu po wpisaniu, którego pionek z pola A3 przesunie się na pole B4
- A3C5 - przykład legalnego bicia po wpisaniu, którego pionek z A3 wykona ruch na C5 i zbije wrogiego pionka na B4

jeśli jest wyświetlone okienko:



Zatwierdza się ruch lub komendę przez naciśnięcie przycisku ESC lub Enter na klawiaturze numerycznej.

Nie występuje w tym module funkcja showmoves ze względu, że jest ona realizowana przez oddzielne okienko:



**Uwagi:** W przeciwieństwie do modułu w terminalu zakończenie rozgrywki nie jest równoważne z zakończeniem programu. Program obsługuje jedynie komendy wpisane do okienka Input. Program poprawnie wczyta komendę nawet jeśli użytkownik będzie używał zmiennej wielkości liter czy znaki będą oddzielone spacjami.

## Podział programu

Program składa się z następujących modułów:

- [checkers\\_rules\\_and\\_logic](#) - elementy gry oraz zasady
- [bot](#) - działanie gracza komputerowego
- [draw\\_in\\_command\\_console](#) - rysowanie gry oraz jej przeprowadzanie w terminalu
- [draw\\_in\\_SDL](#) - rysowanie gry oraz jej przeprowadzanie w SDL
- [main](#) - wybór trybu rozgrywki

Powyżej wypisane moduły zostaną opisane w poszczególnych podrozdziałach.

### Checkers rules and logic

- **int \*\*initiate()** Funkcja ta tworzy planszę 8x8 do gry w warcaby i zwraca wskaźnik do niej.
- **void legal\_action(...)** Funkcja ta zapisuje w odpowiednich tablicach (parametry `int legal_attack[ ][4]` oraz `int legal_move[ ][4]`), które są podawane przy jej wywołaniu, legalne ruchy dla podanego gracza (`int whose_turn`) dla zadanego stanu planszy (`int **board`) i jej rozmiaru (`int size_of_board`), wymagane jest również podanie wskaźników na dwie zmienne typu `int` które będą przechowywać ile ruchów danego typu jest (`what_number_legal_moves` oraz `what_number_legal_attacks`).
- **void continue\_attack(...)** - odpowiada za sprawdzenie czy jeżeli było bicie to czy nie jest ono wielokrotne, jako parametry potrzebuje planszy, wiersza oraz kolumny figury która dokonała zbitia, tablicy legalnych ataków oraz wskaźnika na zmienną typu `int` odpowiadającą za ilość legalnych ataków oraz jaki typ ataku to był tzn. czy figura była jak biała figura czy jak czerwona.
- **void promotion (int \*\*board, int size\_of\_board)** - odpowiada za promocję pionka(`man`) w króla(`king`), w odpowiednich modułach jest wykonywana na końcu pętli.
- **void move\_action(int \*\*board, int \*coordinates)** - pozwala na wykonanie ruchu zadaną figurą na planszy, drugi parametr to wskaźnik mający 4 elementy typu `int` pierwsze dwa to kolumna i wiersz figury którą chcemy ruszyć natomiast dwa ostatnie to kolumna oraz wiersz pola docelowego.
- **void attack\_action(int \*\*board, int \*coordinates)** - umożliwia wykonanie bicia, parametr `coordinates` jest tak samo zdefiniowany jak przy wcześniej wymienionej funkcji `move`.

- **int value\_of\_board(int \*\*board, int size\_of\_board)** - determinuje wartość punktową aktualnego stanu planszy.
- **void who\_is\_winner(int \*board, int size\_of\_board, int \*end\_the\_game)**  
- funkcja ta sprawdza czy któraś ze stron rozgrywki przegrała, jeśli tak to zapisuje w zmiennej end\_the\_game odpowiedni stan.
- **int input\_in\_attack(int legal\_attack[][4], int what\_number\_legal\_attacks, int \*input)** - funkcja zwraca wartość boolowską jeśli zadany input (zdefiniowany tak jak coordinates) odpowiada jakiemuś legalnemu atakowi.
- **int input\_in\_move(int legal\_move[][4], int what\_number\_legal\_move, int \*input)** - działanie podobne do input\_in\_attack.
- **int \*read\_move (int legal\_attack[][4], int legal\_move[][4], int what\_number\_legal\_moves, int what\_number\_legal\_attacks, int \*correctness, char \*entry)**  
- służy do określenia czy podane entry jest ruchem, i jeśli jest to czy jest poprawny zgodnie z zasadami, po sprawdzeniu tego zwraca wskaźnik ruch.
- **void dynamic\_array\_of\_chars(char \*array, int \*size)** - funkcja ta zwiększa dwukrotnie rozmiar tablicy char-ów, która jest podawana jako wskaźnik array.
- **void insert\_to\_dynamic\_array(char \*array, int \*last\_element, int \*size, char what\_to\_insert)** - wstawia znak do tablicy char-ów i jeśli nie ma na to miejsca to wywołuje funkcje do zwiększenia zaalokowanej pamięci.
- **int compare\_arrays (char \*array, const char \*with\_what\_to\_compare, int length)** - zwraca wartość boolową jeśli dwa podane ciągi znaków są podobne do określonej długości.
- **char \*read\_entry(int \*operation)** - wczytuje wejście użytkownika i zwraca wartość entry która jest odpowiednio po przetworzeniu, zapisuje również w zmiennej operation odpowiedni stan odpowiadający wczytanej komendzie.
- **char \*read\_entry\_from\_text(int \*operation, char \*text, int last\_element)**  
- jest podobna do funkcji read\_entry, jednakże nie wczytuje tekstu tylko go przetwarza i zwraca odpowiednie elementy.

## Bot

- **int evaluation\_of\_move(int \*\*board, int size\_of\_board, int whose\_turn, int enemy\_value, int my\_value, int deep)** - funkcja ta zwraca relatywną wartość punktową (może zależeć od głębokości) przy założeniu, że gracze perfekcyjnie grają do określonej głębokości (inaczej mówiąc do określonej ilości ruchów w przyszłość), algorytmem odpowiedzialnym głównie za działanie jej jest [algorytm minimax](#).

- **int \*bot\_\_make\_\_decision(int \*\*board,int size\_of\_\_board,int whose\_\_turn,int enemy\_\_value,int my\_\_value,int deep)** - funkcja ta zwraca najlepszy relatywnie ruch na bazie ewaluacji ich przy pomocy wcześniej wymienionej funkcji, jeśli jest kilka równi dobrych ruchów to zwraca losowo jeden z nich.

**Uwagi:** Nie jest zalecane zwiększanie głębokości powyżej 10. Parametry int enemy\_\_value czy int my\_\_value są potrzebne do określenia czy gracz jest na górze czy na dole, dla przykładu jeśli bot jest na górze to jako my\_\_value podajesz 1 natomiast enemy\_\_value = -1.

## Draw in command console

- **void draw\_\_map\_\_of\_\_game(int \*\*map\_\_of\_\_game,int length\_\_of\_\_map, int size\_\_of\_\_line)** - funkcja służy do wypisania stanu rozgrywki do konsoli, rekomendowane jest podawanie tej samej wielkości do length\_\_of\_\_map oraz do size\_\_size\_\_of\_\_line.
- **void show\_\_legal\_\_moves(int legal\_\_move[][4], int legal\_\_attack[][4],int what\_\_number\_\_legal\_\_moves, int what\_\_number\_\_legal\_\_attacks)** - pozwala na wypisanie wszystkich możliwych ruchów po wpisaniu komendy showmoves.
- **void draw\_\_end(int winner)** - odpowiada za wypisanie stanu kończącego grę, w programie winner jest równy zmiennej end\_\_the\_\_game.
- **void PvP\_\_console()** - funkcja jest odpowiedzialna za wyświetlanie gry za pomocą funkcji draw\_\_map\_\_of\_\_game oraz za jej przebieg dla dwóch graczy ludzkich.
- **void PvComputer\_\_console()** - funkcja jest odpowiedzialna za wyświetlanie gry za pomocą funkcji draw\_\_map\_\_of\_\_game oraz za jej przebieg dla jednego gracza ludzkiego i bota. Gracz jest zawsze na dole a bot zawsze na górze.

## Draw in SDL

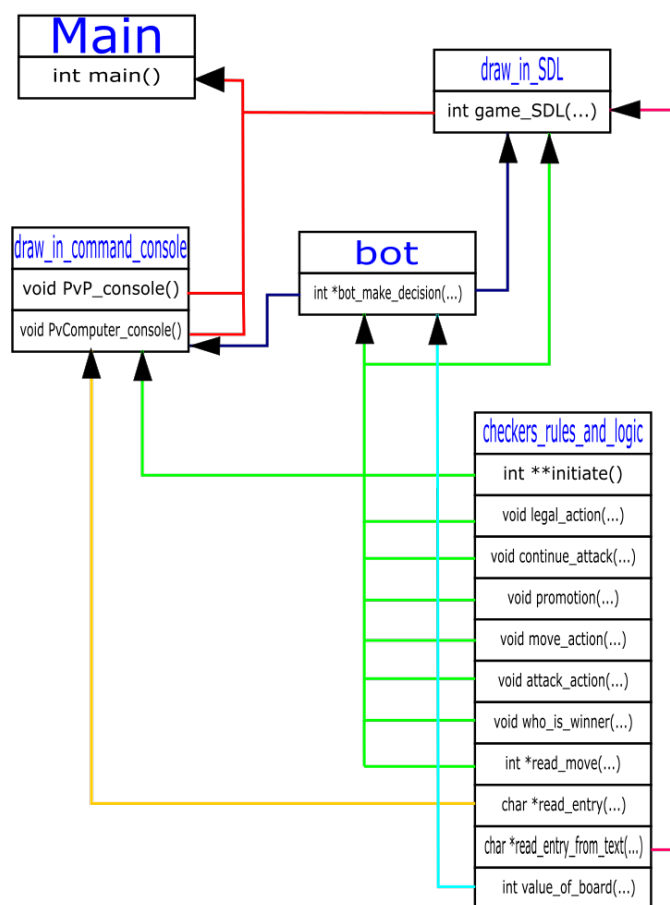
**int game\_\_SDL(int what\_\_type\_\_of\_\_game), int easter\_\_egg** - odpowiada za wyświetlanie wszystkich elementów, trybu okienkowego, jeśli nie będzie wstanie zainicjalizować jakiegokolwiek elementu zwróci odpowiedni błąd oraz zakończy się z nie zerową wartością. Obsługuje grę dwóch ludzkich graczy oraz gracza z botem, odpowiedzialny za to jest parametr what\_\_type\_\_of\_\_game gdzie 0 to gra z botem natomiast 1 to gra między ludzkimi oponentami. Parametr easter\_\_egg jest zmienną która pozwala uruchomić ukrytą w grze niespodziankę przy wyborze trybu.

**Uwagi:** Kolor biały jest równoważny z graczem, który ma turę pierwszy. Może być lekki lag po wykonaniu ruchu jeśli gra się przeciwko komputerowi jest to spowodowane tym, że komputer od razu wykonuje ruch po graczu.

## Main

Składa się z jednej funkcji, której nazwa jest taka sama jak nazwa modułu, odpowiada jedynie za wybór trybu gry.

## Mapa powiązań



## Dodatkowe pliki

Oprócz modułów do gry jest potrzebny folder z plikami graficznymi (Assets rozszerzenie tych plików to .svg) oraz plik font.ttf, pierwszy służy do przecho- wywania, liter, cyfr oraz figur. Lista plików z opisem za co odpowiadają:

- Figury
  - Chess\_klt45.svg - odpowiada za białą figurę króla bądź królowej
  - Chess\_krt45.svg - odpowiada za czerwoną figurę króla bądź królowej
  - Chess\_plt45.svg - odpowiada za białą figurę pionka (ang. men)
  - Chess\_prt45.svg - odpowiada za czerwoną figurę pionka (ang. men)
- Symbole
  - Litera\_\_(znak).svg - odpowiada za odpowiednią literę gdzie znak to litera w zakresie od A do H
  - Number\_\_(znak).svg - odpowiada za odpowiednią cyfrę gdzie znak to cyfra w zakresie od 1 do 8
- Font - należy w głównym folderze z grą i ma nazwę na font.ttf

**Uwagi:** Kształty figur zostały wzięte z [https://en.wikipedia.org/wiki/Chess\\_piece](https://en.wikipedia.org/wiki/Chess_piece).  
 Natomiast obrazki do niespodzianki z <https://www.reddit.com/r/corgi/>