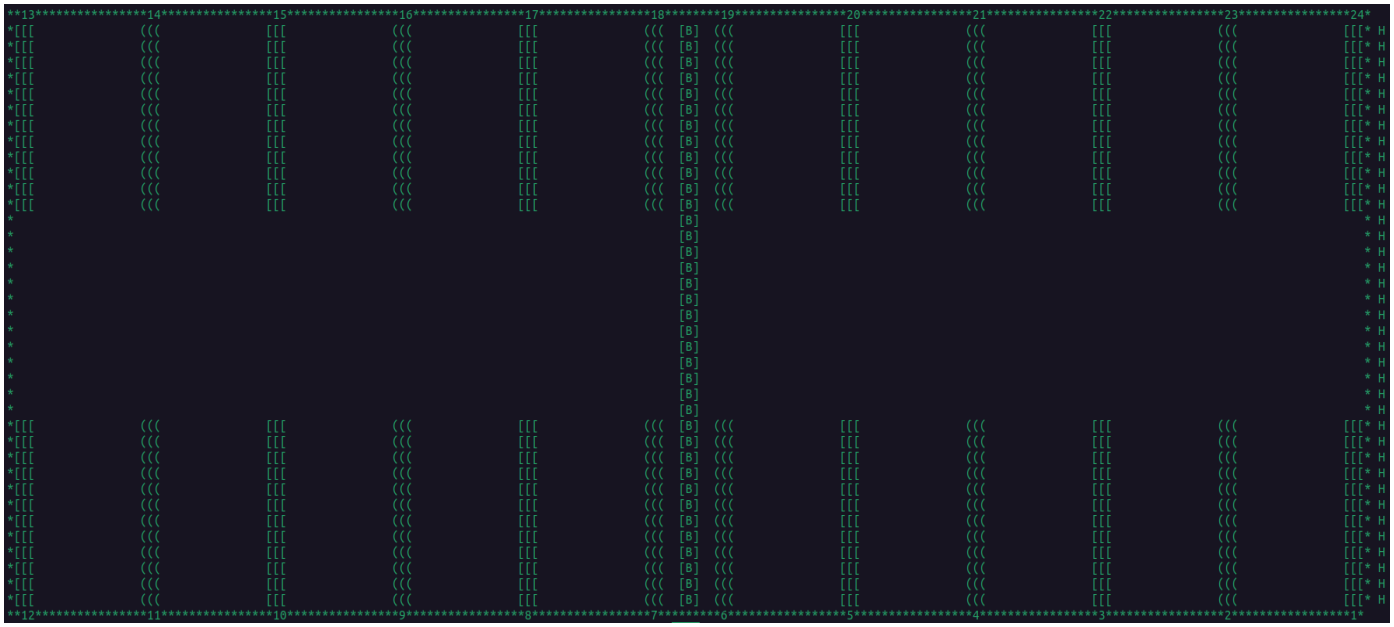
Projekt 1 - Backgammon

Autor: Igor Tomkowicz, s194103

1. Plansza do gry

Plansza do gry, zgodnie z zasadami gry składa się z 24 pól podstawowych (numerowanych od 1 do 24) oraz 4 pól dodatkowych (2 pola do umieszczenia zbijanych pionków oraz 2 pola do umieszczenia pionków schodzących z planszy na dwór).



1.1 Struct Coordinates

Struktura Coordinates to zestaw dwóch zmiennych:

int x: Przechowuje wartość współrzędnej w poziomie, reprezentując położenie w płaszczyźnie poziomej.

int y: Przechowuje wartość współrzędnej w pionie, reprezentując położenie w płaszczyźnie pionowej.

1.2 Struct Pawn

Struktura Pawn składająca się z dwóch zmiennych:

int player: Przechowuje wartość identyfikującą gracza, określającą, do którego gracza należy ten pionek.

Coordinates position: Wykorzystuje strukturę Coordinates do przechowywania współrzędnych pionka. Zawiera informacje o położeniu na planszy za pomocą struktury Coordinates, gdzie x reprezentuje współrzędną w poziomie, a y reprezentuje współrzędną w pionie.

1.3 Struct Board

Struktura Board zawiera tablicę pawns, która jest przeznaczona do przechowywania pionków (Pawn) dla dwóch graczy. Rozmiar tej tablicy wynosi dwukrotność wartości zmiennej TOTAL_PAWNS, ponieważ przewidziano miejsce dla pionków obu graczy.

1.4 Struct Column

Każde pole jest definiowane jako kolumna (Struktura Column). Łącznie mamy 28 kolumn, indeksowanych w następujący sposób:

0 – 23 – Podstawowe pola do gry, 1 pole to indeks 0, 2 pole to indeks 1 itd.,

24 – Pole zarezerwowane do stawiania zbitych czerwonych pionków (Pionków gracza 1) – RED BAR,

25 – Pole zarezerwowane do stawiania zbitych pionków białych (Pionków gracza 2) – WHITE BAR,

26 – Pole zarezerwowane do stawiania czerwonych pionków schodzących z planszy – RED BACKYARD

27 – Pole zarezerwowane do stawiania białych pionków schodzących z planszy – WHITE BACKYARD

Dodatkowo każda kolumna (struktura) zawiera następujące wartości:

- position (Zdefiniowane jako element struktury Coordinates (x,y)) – określa faktyczne współrzędne początku kolumny na ekranie.

- pawns (Zdefiniowane jako int) – liczy ilość pionków znajdujących się aktualnie w danej kolumnie.

1.4.1 Powiązanie współrzędnych pionka z kolumną

Jako, że struktura Column przechowuje wartości X i Y reprezentujące pozycję na ekranie to współrzędne pionków odpowiadają następującym kolumnom:

Pola 1 – 12:

Współrzędna x = 12 dla pola 1, x = 11 dla pola 2 itd., aż do x = 1 dla pola 12.

Współrzędna y = 1.

Pola 13 – 24:

Współrzędna x = 0 dla pola 13, x = 1 dla pola 14 itd., aż do x = 11 dla pola 24.

Współrzędna y = 0.

Pole 24:

Współrzędna x = 13.

Współrzędna y = 1.

Pole 25:

Współrzędna x = 12.

Współrzędna y = 0.

Pole 26:

Współrzędna x = 15.

Współrzędna y = 0.

Pole 27:

Współrzędna x = 15.

Współrzędna y = 1.

2. Biblioteki

Biblioteki wykorzystane przy tworzeniu projektu to:

- **ncurses.h**: Jest odpowiedzialna za interakcję z oknem w terminalu.

- **time.h**: Wykorzystywana w celu generowania liczb pseudolosowych poprzez dostęp do czasu systemowego jako ziarna dla generatora liczb losowych.

- **stdlib.h**: Opowiada za generowanie liczb pseudolosowych, szczególnie używanych w symulowaniu rzutu kostką.

- **cmath**: Jest wykorzystywana do wykonywania operacji matematycznych, w tym obliczania wartości bezwzględnej (abs()) używanej na przykład do określenia nowego położenia pionka na planszy po wykonaniu ruchu.

3. Funkcje

3.1 main():

Odpowiada:

- inicjalizacje nowego okna,

- wyłączenie wyświetlania znaków wpisywanych przez użytkownika,

- wyłącza widoczność kursora na ekranie,

- włącza obsługę klawiszy,

- ustawia kolor tła i tekstu,

- wywołuje funkcje odpowiedzialne za tworzenie ramki, opcji menu oraz sterowania opcji menu.

3.2 optionsSystem():

Odpowiada za sterowanie systemem wyboru opcji w menu.

3.3 play():

Odpowiada za wyczyszczenie ekranu po wybraniu opcji „play” w menu, utworzeniu elementów potrzebnych do gry (wyników, planszy, kolumn itd.). Po kliknięciu enter inicjalizuje rozgrywkę oraz funkcję odpowiadającą za pierwszy rzut kostkami.

3.4 game():

Odpowiada za sterowanie przebiegiem rozgrywki. Na bieżąco zlicza ilość wykonanych ruchów, kontroluje wykonywanie kolejnych posunięć. Pozwala również po wykonaniu ruchu na zapis rozgrywki lub jej wczytanie. Po zakończeniu rozgrywki inicjalizuje funkcję odpowiedzialną za zakończenie rozgrywki i rozpoczęcie nowej.

3.5 prepareGame():

Wywołuje wszystkie funkcje przygotowujące planszę i system do rozgrywki.

3.6 nextPlayer():

Odpowiada za zmianę gracza co turę.

3.7 inGameOptions():

Odpowiada za inicjalizacje funkcji możliwych w trakcie rozgrywki:
- zapis rozgrywki do pliku,

- wczytanie rozgrywki z pliku,

- przesuwanie krokowe.

3.8 endGame():

Odpowiada za wyświetlenie ekranu końcowego gry i możliwość kolejnej rozgrywki.

3.9 pointsAfterWin():

Przydziela, określoną przez kryteria, ilość punktów dla wygranego gracza.

3.10 moveToEnd():

Przesuwa rozgrywkę na koniec (wykorzystywane w ruchu krokowym).

3.11 moveToStart():

Przesuwa rozgrywkę na początek (wykorzystywane w ruchu krokowym).

3.12 gameStatus():

Odpowiada za obsługę opcji zapisania i ładowania gry.

3.13 load():

Odpowiada za ładowanie gry z pliku.

3.14 save():

Odpowiada za zapisanie stanu gry do pliku.

3.15 makeMove():

Inicjalizuje funkcje związane z przebiegiem ruchu. Sprawdza czy na kostce wypadł „dublet”, [rosi gracza o określenie ilości pionków, którymi chce się przesunąć oraz pól, z których zamierza się ruszyć. Dodatkowo kontroluje czy gracz wykonujący ruch ma zbitego pionka, czy wszystkie pionki są już w domu oraz zapisuje aktualny numer ruchu.

3.16 isDoublet():

Sprawdza czy na kostce wypadł „dublet”.

3.17 allPawnsAtHome():

Sprawdza czy wszystkie pionki gracza są w domu. I czy może ruszyć się tymi pionkami.

3.18 areAllPawnsAtBackyard():

Sprawdza czy wszystkie pionki gracza są już na dworze.

3.19 areAllPawnsAtHome():

Wykonuje operacje związaną ze sprawdzeniem czy wszystkie pionki gracza są w domu.

3.20 moveOptions():

Decyduje, który ruch wykonujemy (normalny – czyli o 1 lub 2 pola albo zdublowany – czyli o 3 albo 4 pola).

3.21 doubletVariant():

Inicjalizuje ruch o 3 albo 4 pola.

3.22 normalMove():

Inicjalizuje ruch o 1 albo 2 pola.

3.23 fourPawns():

Inicjalizuje funkcję odpowiedzialną za 4 ruchy. Sprawdza czy gracz faktycznie może wykonać 4 ruchy (czy jest dublet).

3.24 moveFour():

4 razy inicjalizuje funkcję odpowiedzialną za jeden ruch.

3.25 threePawns():
Inicjalizuje funkcje odpowiedzialną za 3 ruchy. Sprawdza czy gracz faktycznie może wykonać 3 ruchy (czy jest dublet).

3.26 moveThree():

3 razy inicjalizuje funkcję odpowiedzialną za jeden ruch.

3.27 makeAgainPawnDecision():
Zmusza gracza do ponownego wyboru ilości pionków, którymi chce się ruszyć.

3.28 twoPawns():
2 razy inicjalizuje funkcje odpowiedzialną za jeden ruch.

3.29 onePawn():
Inicjalizuje 1 ruch i go wykonuje.

3.30 makePawnMove():

Sprawdza wszystkie warunki związane z ruchem pionka.

3.31 updateConditions():
Odpowiedzialna za aktualizacje warunków związanych z ruchem pionka (po pierwszym odrzuceniu, gracz proszony jest o wpisanie nowych pól).

3.32 otherPossibleMoves():

Sprawdza czy gracz może ruszyć się innymi pionkami, jeżeli wybranym nie może.

3.33 validatePawnMoves():

Sprawdza podstawowe warunki ruchu pionka (czy nie wychodzi za planszę, czy nie chce wejść na zajęte pole).

3.34 forceMoveFromBar():
Zmusza gracza do ruszenia się zbitym pionkiem.

3.35 farthestPawn():
Szuka najdalszego pionka od dworu (wykorzystywane przy zakończeniu gry).

3.36 possibleMoves():
Sprawdza możliwość ruchów pionka.

3.37 possibleMoveFromBar():

Sprawdza czy gracz może się ruszyć zbitym pionkiem.

3.38 isOnBar():
Sprawdza czy gracz ma zbitego pionka/pionki.

3.39 moveForward():

Odpowiada za wykonanie ruchu krokowego do przodu.

3.40 moveBackward():
Odpowiada za wykonanie ruchu krokowego w tył.

3.41 loadStepMove():

Ładuje krokowy zapis gry i inicjalizuje funkcję odpowiedzialną za znalezienie ruchu oraz za ruch krokowy.

3.42 stepMove():

Inicjalizuje funkcję odpowiedzialną za konwersję zapisu krokowego na liczbowy i inicjalizuje funkcję odpowiedzialną za ruch krokowy.

3.43 convertFromFile():

Konwertuje dane otrzymane z pliku zapisu krokowego.

3.44 countPositionInFile():

Oblicza pozycję w pliku ruchu krokowego.

3.45 stepSystem():

Wykonuje operacje związane z cofnięciem przebiegu gry do poprzedniego/następnego ruchu.

3.46 pawnMove():
Inicjalizuje funkcje związane z ruchem pionka. Inicjalizuje funkcję związaną z zapisem ruchu krokowego.

3.47 stepSaving():

Inicjalizuje zapis ruchu krokowego.

3.48 pawnMoveVisual():

Odpowiada za wizualne przesunięcie pionka po wykonaniu ruchu.

3.49 getNewPawnIndex():

Znajduje indeks nowego pionka (pionka na pozycji, na którą zamierzamy się ruszyć).

3.50 pawnMoveSystem():

Inicjalizuje funkcje odpowiadające za systemowy ruch pionka.

3.51 pawnMoveSystemPlayer2():

Systemowy ruch pionka gracza 2.

3.52 pawnMoveSystemPlayer1():

Systemowy ruch pionka gracza 1.

3.53 pawnMoveSystemToHome():

Funkcja odpowiedzialna za systemowe zdjęcie pionka z planszy (wejście na dwór).

3.54 catchPawn():

Inicjalizuje funkcje związane ze zbiciem pionka.

3.55 drawPawn():
Inicjalizuje funkcje związane z narysowaniem pionka.

3.56 columnToDrawPawn():

Rysuje pionka w nowej kolumnie.

3.57 moveOnBar():

Systemowe przesunięcie zbitego pionka.

3.58 drawOnBar():

Wizualne przesunięcie zbitego pionka.

3.59 drawPawnOnBackyard():

Wizualne przesunięcie pionka na dwór.

3.60 removePawn():

Inicjalizuje funkcje związane z usunięciem pionka ze starej pozycji.

3.61 getColumnPosition():

Pobiera pozycję początku kolumny na ekranie.

3.62 removePawnDraw():
Wizualne usunięcie pionka ze starej pozycji.

3.63 clearColumn():

Wizualne wyczyszczenie kolumny, jeżeli nie ma na niej żadnych pionków.

3.64 checkPawn():
Główna funkcja odpowiadająca za inicjalizację funkcji związanych z ruchem pionka. Zwraca indeks pionka, który będzie wykonywał ruch, jeżeli może go wykonać.

3.65 checkPawnDetails():
System sprawdzający konkretne warunki związane z ruchem pionka.

3.66 getPawnPosition():

Określa pozycję pionka (x,y) na podstawie pola.

3.67 stepSave():

Wykonuje zapis krokowy do pliku.

3.68 playerTurn():

Wypisuje, który gracz aktualnie wykonuje ruch.

3.69 statusMenu():

System odpowiedzialny za sterowaniem menu zapisu/załadowania gry.

3.70 drawStatusMenu():
Rysuje menu wyboru zapisu/załadowania gry.

3.71 chooseField():

Odpowiada za wywołanie funkcji związanej z wyborem pionków oraz zapisuje pola, z których gracz zamierza się ruszyć. Jako wartość zwraca ilość pól, z których gracz zamierza się ruszyć.

3.72 oneMove():

Pobiera wartość pola, z którego gracz zamierza się ruszyć.

3.73 fieldNumber():

Sprawdza czy gracz wpisuje prawidłowe pola.

3.74 clearNumberField():

Czyści pozycję na ekranie, w której gracz wpisywał wartość pola.

3.75 pawnsOptions():

Funkcja odpowiedzialna za menu wyboru ilości pionków, którymi gracz zamierza się ruszyć.

3.76 findPawnIndex():

Znajduje indeks pionka na określonej pozycji (x,y).

3.77 findPawnIndexes():

Zwraca wszystkie indeksy pionków znajdujących się na określonej pozycji (x,y).

3.78 firstRoll():

Odpowiada za wykonanie pierwszego rzutu kostką i decyzję, o tym, który gracz rozpoczyna grę.

3.79 roll():

Funkcja symulująca rzut kostką.

3.80 clearPart():

Funkcja czyszcząca pozycje na ekranie, odpowiedzialną za wyświetlanie komunikatów.

3.81 board():

Odpowiada za wizualne stworzenie planszy (tworzy obramowanie i inicjalizuje funkcje związane z tworzeniem planszy).

3.82 inscription():

Odpowiada za wizualne stworzenie napisów opisujących planszę oraz wynik.

3.83 findColumn():

Znajduje indeks kolumny na podstawie współrzędnych (x,y).

3.84 isPawnPresent():

Sprawdza czy na współrzędnych (x,y) znajduje się już pionek.

3.85 initializeBoard():

Przypisuje indeksy pionków do graczy, tworzy przykładowe współrzędne dla pionków oraz resetuje ilość pionków na każdym polu.

3.86 drawGameBars():

Rysuje pole „Bar” oraz „Home”.

3.87 gameBarsPosition():

Określa wizualną pozycję:

- Pola, na którym umieszczamy zbitego czerwonego pionka.
- Pola, na którym umieszczamy zbitego białego pionka.
- Dwór czerwonych (gracza 1).
- Dwór białych (gracza 2).

3.88 Columns():

Odpowiada za wizualne stworzenie kolumn. Inicjalizuje funkcje związane z numeracją i określeniem pozycji początku kolumn.

3.89 columnsPosition():

Określa wizualne współrzędne(x,y) początku każdej z kolumn.

3.90 columnsNumbers():

Tworzy wizualną numerację kolumn.

3.91 menuOptions():

Tworzy opcje menu na ekranie.

3.92 menuFrame():

Tworzy obramowanie okna.

4. Definicje:

4.1 BOARD_ROWS 2 – ilość wierszy planszy.

4.2 BOARD_COLS 14 – Ilość kolumn planszy.

4.3 TOTAL_PAWNS 15 – Ilość pionków jednego gracza.

4.4 CORNER_SPACE 2 – Odległość od krawędzi ekranu

4.5 SPACE_FROM_EDGE 5 – Odległość od obramowania.

4.6 BAR_SPACE – Odległość dla pola zbijanych pionków (Bar) oraz dworu.