Główne działanie programu opiera się na wykorzystaniu obiektów klasy plansza czyli pionków do gry, których parametry są zawarte w klasie. Działanie programu jest zawarte w pętli, w której to wykonują się funkcję zależne od wyborów użytkownika.

Funkcja **main** zaczyna się od przypisania pewnych wartości dla cech obiektów do planszy i ustawienie pionków na planszy. Następnie Zostaje wyświetlony napis początkowy, a po nim możliwości z menu. Po wybraniu rozpoczęcia gry należy wybrać tryb gry, który uruchamia odpowiednią funkcję.

Klasy:

plansza – zawiera cechy pionków (pole, kolor, typ i znak), pionki to tablica obiektów tej klasy.

ustawienie_planszy – obiekty klasy służą do wyświetlania odpowiednich znaków na grywalnych polach planszy.

ustawienie_pol – obiekty tej klasy służą do wprowadzania przez użytkownika nazwy pola z planszy oraz sprawdzania czy potencjalny ruch pionka znajdzie się na grywalnym polu planszy.

Funkcje:

void aktualizuj_plansze(); wyświetla plansze z odpowiedniki znakami na grywalnych polach.

void ustawienie_pionkow_na_planszy(); ustawia początkowe wartości pionków: kolor, typ, znak i pole.

void zamiana_pionka_na_damke(); sprawdza czy jakiś pionek znajduje się na końcu planszy odpowiednio dla swojego koloru, jeżeli tak to zamienia go na damke.

int sprawdz_numer_pola(string nazwa_pola_f); sprawdza czy podane przez użytkownika pole (nazwa_pola_f) jest jednym z grywalnych pól, biorąc pod uwagę również wprowadzone nazwy pól z małej litery. Funkcja zwraca (int) numer pola łatwiejszy do dalszych operacji odpowiadający podanemu przez użytkownika polu. W przypadku, gdy użytkownik nie poda nazwy grywalnego pola to funkcja zwróci 0 (int).

int sprawdz_indeks_pionka(int nr_pola); funkcja otrzymuje numer pola i następnie sprawdza pętlą czy na tym polu znajduje się jakiś pionek, jeżeli tak to zwraca indeks tego pionka (int). Jeżeli żaden pionek na znajduje się na podanym polu to funkcja zwraca -1 (int).

string sprawdz_kolor_pionka(int nr_pola); funkcja otrzymuje numer pola i zwraca kolor pionka(string), który się na nim znajduje.

void sprawdz_mozliwy_ruch_pionka_krawedzie_funkcja(int indeks_pionka); funkcja otrzymuje indeks pionka, sprawdza możliwość ruchu pionka na krawędziach planszy. W przypadku gdy podany pionek znajduję się na jednej z krawędzi funkcja ustawia na false zmienne odpowiadające za możliwość wykonania ruchu w danum kierunku.

void sprawdz_mozliwe_bicie_pionka_krawedzie_funkcja(int indeks_pionka); Działa podobnie jak poprzednia funkcja, ale dla warunków zbijania przez pionka.

void sprawdz_mozliwy_ruch_pionka_bialego_funkcja(int pole_pionka); funkcja otrzymuje indeks pionka, sprawdza czy biały pionek może wykonać swoje ruchy, jeżeli nie to ustawia zmienne odpowiedzialne za możliwość ruchu pionka w odpowiednim kierunku na false oraz dla kierunków przeciwnych.

void sprawdz_mozliwy_ruch_pionka_czarnego_funkcja(int pole_pionka); funkcja działa identycznie jak poprzednia ale dla czarnych pionków.

void sprawdz_mozliwe_bicie_pionka_funkcja(int pole_pionka); funkcja otrzymuje pole pionka, jest odpowiedzialna za sprawdzenie możliwych zbić przez pionka. Jako pierwsze sprawdza czy pole, na które ma ruszyć się pionek po zbiciu jest puste. Następnie sprawdza czy potencjalny pionek do zbicia jest przeciwnego koloru. Jeżeli któryś z warunków Jest nie zgodny to funkcja wyłącza możliwość zbicia w danym kierunku.

void sprawdz_mozliwe_ruchy_damki(int indeks_pionka); funkcja otrzymuje indeks pionka, sprawdza liczbę możliwych ruchów damki w każdą stronę jeżeli są one puste. Numery pól na które damka może się ruszyć są zapisywane do tablicy. Funkcja zapisuje do zmiennych indeksy pionków na których kończy się możliwość wykonania ruchu w danej linii.

void sprawdz_mozliwe_bicia_damki(int indeks_pionka); funkcja otrzymuje indeks pionka, wykorzystuje zmienne ustawione w poprzedniej funkcji, w których jest zapisany indeks pionka, na którym zatrzymała się damka. Jeżeli wartość zmiennej wynosi -1 to oznacza to, że jest to krawędź planszy i funkcja wyłącza możliwość zbijania w danym kierunku. Następnie sprawdza czy na polu za zapisanym pionkiem jest puste pole i czy pionek do zbicia jest przeciwnego koloru, jeżeli jakiś warunek nie jest spełniony to funkcja wyłącza możliwość zbicia w danym kierunku ustawiając odpowiednią zmienną na false.

void wykonaj_ruch(string nazwa_pola); funkcja odpowiedzialna za ruch użytkownika.
Funkcja otrzymuje nazwe pola, sprawdza czy podana jest jedną z grywalnych pól funkcją sprawdz_numer_pola, następnie sprawdza czy na wybranym polu znajduje się pionek po przez funkcje sprawdz_indeks_pionka, następnie odpowiednio dla tury funkcja sprawdza czy podany pionek jest odpowiedniego koloru. Na końcu wywołuje funkcje ruch_pionka odpowiedzialną za zrealizowanie ruchu.

void ruch_pionka(int pole_pionka, int indeks_pionka); funkcja realizuje wykonanie ruchu użytkownika. Funkcja otrzymuje pole i indeks pionka, sprawdza po kolei wszystkie warunki wywołując funkcje:

- sprawdz_mozliwy_ruch_pionka_krawedzie_funkcja
- sprawdz_mozliwe_bicie_pionka_krawedzie_funkcja

Dla białego pionka:

- sprawdz_mozliwy_ruch_pionka_bialego_funkcja
- sprawdz_mozliwe_bicie_pionka_funkcja

Dla czarnego pionka:

- sprawdz_mozliwy_ruch_pionka_czarnego_funkcja
- sprawdz_mozliwe_bicie_pionka_funkcja

Dla damki:

- sprawdz_mozliwe_ruchy_damki

sprawdz_mozliwe_bicia_damki

Wyświetla komunikaty o możliwych ruchach i zbiciach. Na podstawie wcześniej wywołanych funkcji ustawia zmienne mozliwy_ruch i mozliwe_bicie. Gdy któryś z nich wynosi true to wykonuje się następna sekwencja. Funkcja wyświetla komunikat gdzie gracz chce ruszyć swojego pionka. Jeżeli podane pole jest zgodne z możliwymi do wykonania przez podanego wcześniej pionka to pole pionka zostaje zmienione, a w przypadku zbijania pole zbitego pionka zostaje ustawione na 0. W przypadku źle podanego pola sekwencja zostaje powtórzona po przez funkcję goto.

void wykonaj_ruch_komputera(int id); Funkcja odpowiedzialna za wykonanie ruchu komputera. Jeżeli podany pionek może wykonać zbicie to je zawsze wykona. Funkcja otrzymuje indeks pionka, sprawdza czy zmienna odpowiadająca za możliwe bicie jest równa true odpowiednio dla koloru podanego pionka lub damki. Jeżeli jest to pionek to zostaje wylosowany jedno z możliwych zbić, gdy nie jest ono możliwe do wykonania to losowanie powtarza się, a gdy jest możliwe to zostaje wykonane zamieniając pole zbitego pionka na 0 i zmieniając pole wybranego pionka. Gdy jest to damka to proces wygląda identycznie. Jeżeli zbijanie nie jest możliwe to zostaje wylowosawny jeden z możliwych ruchów dla pionka lub damki zostaje on wykonany.

void wywolanie_ruchu_komputera(); funkcja ta jest odpowiedzialna za wywołanie funkcji sprawdzających możliwe ruchy i zbijanie. Priorytetem ruchu komputera jest zbijanie. Jeżeli są pionki „drużyny" komputera, które mogą zbijać to zostaje wylosowany jeden z nich. Gdy zbijanie nie jest możliwe to zostaje wylosowany jeden z pionków „drużyny" komputera, który może wykonać ruch. Następnie zostaje wywołana funkcja wykonaj_ruch_komputera z podanym indeksem wylosowanego pionka.

void ile_mozliwych_ruchow_pionkow_funkcja(int indeks_pionka); funkcja otrzymuje indeks pionka, a następnie sprawdza ile może wykonać ruchów podany pionek.

void sprawdz_czy_jest_mozliwy_ruch(); funkcja sprawdza możliwość wykonania i liczbę możliwych do wykonania ruchów przez wszystkie pionki i damki, zapisując te wartości i ich indeksy.

void sprawdz_czy_jest_mozliwe_zbicie(); funkcja sprawdza możliwość wykonania i liczbę możliwych do wykonania zbić przez wszystkie pionki i damki, zapisując te wartości i ich indeksy.

void sprawdz_zakonczenie_gry(); funkcja wywołuje funkcje sprawdzające możliwe ruchy i zbicia i na ich podstawie sprawdza warunki zakończenia gry. Gdy warunki zostaną spełnione to gra zostaje zakończona, zostaje wyświetlony odpowiedni komunikat, plansza zostaje zostaje zresetowana i zostaje wyświetlone menu główne.

void GraczVsGracz(); funkcja odpowiadająca za grę turową graczy. Wyświetla komunikat czyj jest ruch zgodnie z obecną turą. Gracz musi podać pole pionka, którym chce ruszyć, jeżeli zostają spełnione wszystkie warunki to zostaje wykonana funkcja wykonaj_ruch. Po każdym ruchu wywołuje się funkcja sprawdz_zakonczenie_gry i zmienia się tura.

void GraczVsKomputer(); funkcja odpowiada za grę turową gracz na komputer. Gracz gra białymi pionkami, a komputer czarnymi. Na przemian gracz wykonuje swój ruch tak jak w trybie gracz na gracza i swój ruch wykonuje komputer po przez wywołanie funkcji wywolanie_ruchu_komputera. Po każdym ruchu wywołuje się funkcja sprawdz_zakonczenie_gry i zmienia się tura.

void KomputerVsKomputer(); funkcja odpowiadająca za symulacje gry turowej dwóch komputerów. Odpowiednio dla tury komuter wykonuje ruch raz dla białych i raz dla czarnych po przez wywołanie funkcji wywolanie_ruchu_komputera. Po każdym ruchu wywołuje się funkcja sprawdz_zakonczenie_gry i zmienia się tura.