DOKUMENTACJA KOŃCOWA PROJEKTU

POPULARNA GRA ”STATKI”

Celem projektu było napisanie gry konsolowej w technice Programowania obiektowego (ang. object-oriented programming, OOP). Zasady gry:

I. Na jednej planszy gracz zaznacza współrzędne swoich statków, a na drugiej zaznacza nietrafione strzały w statki przeciwnika oraz oznaczenia trafionych okrętów.
II. Statki mogą być narysowane w pionie i poziomie, ale nie mogą stykać się ze sobą bokami ani rogami i nie mogą być „zakręcone” czyli przypominać litery L.
III. Gra przebiega naprzemiennie: najpierw jeden z graczy oddaje strzał, następnie drugi gracz strzela
IV. Przeciwnik „komputer” swoje strzały generuje losowo
V. Zatopienie statku ma miejsce, kiedy wszystkie kratki symbolizujące ten okręt zostaną „trafione” przez drugiego gracza.
VI. O oddaniu nietrafionego strzału gra informuje ze „spudłowaliśmy”. Natomiast przy strzale trafionym gra daje nam znać „trafiony” lub „trafiony-zatopiony”.
VII. Na planszach 10 na 10 przyjmuje się, że każdy z graczy dysponuje flotą złożoną z:

-jednego czteromasztowca o wielkości czterech kratek

-dwóch trójmasztowców o wielkości trzech kratek

-trzech dwumasztowców o wielkości dwóch kratek

-czterech jednomasztowców o wielkości jednej kratki.

OPIS KLAS

- Klasa Gracz – jest klasą abstrakcyjną, która jest odpowiedzialna za przechowywanie informacji o danym graczu
  Atrybuty:
  -string nazwa,
  -int numer_gracza,
  -bool rodzaj_gracza
  Metody:
  sprawdz_wejscie(), sprawdz_int(int, int) dwie metody odpowiadające za kontrole błędów
  - posiada także 6 metod (m.in. set_nazwa) do posługiwania się prywatnymi atrybutami klasy
  - 2 konstruktory i destruktor
- Klasa Komputer – klasa dziedziczny publicznie z klasy Gracz, jest odpowiedzialna za inicjalizacje planszy komputera, generator liczb z biblioteki <random> losuje położenia statków na planszy.
  Atrybuty:
  - stack<int>ostatni_hitx – przechowuje współrzędne **trafionych** strzałów x
  - stack<int>ostatni_hity - przechowuje współrzędne **trafionych** strzałów y
  - vector <int> strzal_x – przechowuje ostatni strzał, nie tylko trafione, ale ogólnie ostatni strzał wykonany

- vector <int>strzal_y - przechowuje ostatni strzał, nie tylko trafione, ale ogólnie ostatni strzał wykonany

- int licznik
- bool ostatni_strzal
- bool hunt_poziom
- bool hunt_pion
- W atrybutach użyłem „stack" stosu, dość rzadko spotykany pojemnik na dane, potrzebny mi był do zapisywania ostatnich trafionych strzałów naszego wirtualnego gracza, atrybuty takie jak hunt_poziom/pion, ostatni_strzal, licznik są czysto techniczne, aby usprawnić działanie dwóch głównych metod

Metody:

- Następny_ruch_auto(Plansza &b) operuje następnymi ruchami gracza, a także używa algorytmu „łowcy", który stara się zatopić nasz okręt
- Ustaw_plansze_auto(Plansza &b, bool) – inicjalizuje plansze wirtualnego gracza

- Klasa Człowiek – publicznie z klasy Gracz, za jej pomocą tworzymy na początku gry swoją plansze do gry, ustawiamy statki, a następie kolejne nasze ruchy także są obsługiwane przez tą klasę.

  Metody:

  -Ustaw_plansze(Plansza& p,string) – metoda odpowiedzialna za początek gry, ustawienia statków tak jak wskazuje gracz

  - Nastepny_ruch(Plansza& p) – metoda zajmuję się każdym następnym ruchem gracza

  - Konstruktor z listą inicjalizacyjną, która tworzy obiekt klasy Gracz z argumentem 1

- Klasa Plansza – jest odpowiedzialna za plansze 10x10 czyli podstawową strukturę gry.

  Atrybuty:

  -vector <vector<char>>plansza_gry - dwuwymiarowy wektor (macierz), która właśnie oddaje naszą plansze.
  -vector <Statki>statki - wektor, który przechowuje obiekty klasy Statki.
  W klasie zawartych jest także kilka „stałych"
  - const int WYMIAR_PLANSZY = 10
  - const char WODA = '~'

  Metody:

  - rysuj_plansze_trafien() -pokazywanie planszy z trafieniami i chybieniami
  - sprawdz_polozenie(int, int, int, bool) – sprawdza czy umiejscowienie statków jest prawidłowe, czy się ze sobą nie stykają, a także czy nie wychodzą za rozmiar planszy
  - Sprawdz_trafienie(int, int, bool)- jeśli umiejscowienie jest zgodne z kryteriami to zmienia znak z planszy na znak statku
  - int ilość_trafien() - liczbę trafień po których zakończy się gra
  - char wartość_polozenia(int, int) – zwraca znak, na danym polu
  - Sprawdz_zatopienie(int, int) – sprawdza czy statek o danych współrzędnych jest zatopiony

- Klasa Statek – odpowiedzialna za wszystkie obiekty na planszy

  Atrybuty:

  - char* strefa_statku – w konstruktorze tworzona jest dynamiczna tablica statku o danym rozmiarze
  - int rozmiar_statku – rozmiary poszczególnych statków

- int pozycja_x, pozycja_y
- bool zatopiony
- bool orientacja
- char znak_statku = 'S'

Metody:

- Statek(int, int, int, bool, bool) - konstruktor

-Statek(int, int) -konstruktor

- void ustaw_pozycje(int, int, bool) – ustawia pozycje statku , pierwsza kratka po lewej lub najbardziej u góry

- bool czy_zatopiony() – zwraca informacje czy statek jest zatopiony

- bool jaka_orientacja() - zwraca informacje o orientacji statku

- bool sprawdz_trafienie(int, int) – sprawdza trafienie na statku

- void print() – funkcja pomocnicza

- int get_x() – zwraca atrybut pozycja_x

- int get_y() - zwraca atrybut pozycja_y

- int get_rozmiar() - zwraca atrybut rozmiar

- bool czy_istnieje(int, int) – sprawdza czy statek o danych współrzędnych istnieje

- **Pozostałe funkcje lub stałe:**
  - const int ILOSC_STATKOW = 10
  - const int ROZMIARY_STATKOW[ILOSC_STATKOW] = { 4,3,3,2,2,2,1,1,1,1 }
  - const int suma_kratek_statkow = 20
  - void odliczanie()
  - const int korekta_liter = 65 - korekta na kod ASCII
  - const int korekta_liczb = 48 - korekta na kod ASCII

  Main :
  - void gra_jednoosobowa()
  - void gra_dwuosobowa()
  - enum status{G1_WIN, G2_WIN, ROZGRYWKA_TRWA} – własna typ zmiennych, określający stan rozgrywki
  - status stan_gry(Plansza & p1, Plansza& p2) – sprawdza czy gra wciąż trwa

OBSŁUGA ORAZ INSTALACJA PROGRAMU

Na https://github.com/BeJakub/Projekt-Gra_Statki w folderze Debug mamy plik wykonywalny Gra_Statki_Projekt.exe. Należy pobrać całe repozytorium, a następnie uruchomić grę za pomocą tego pliku.
Po uruchomieniu ukazuje się nam, krótkie MENU, mamy do wyboru grę jednoosobową lub dwuosobową, a także opis gry i wyjście.