1.Imię i nazwisko wykonującego zadanie: Mateusz Chechłowski.

2.Temat zadania:

Napisać program umożliwiający grę w statki z komputerem.

Założenia:

- rozmiar pola rozgrywania gry: definiowana przez użytkownika

- liczba i rodzaj statków definiowana przez użytkownika

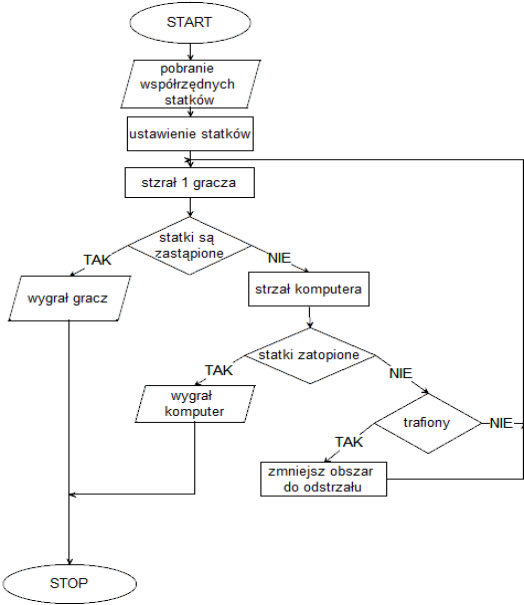
- symulator gracza realizowany przez program powinien mieć wbudowaną „pewną logikę” tzn. np. automatycznie znajdował pola, na których nie mogą znajdować się statki przeciwnika.

3.Dane osoby oceniającej: dr. inż. Marek Niewiński.

4.program, z którym się mierzyłem jest gra w statki. Sposób w jaki rozwiązałem problem jest następujący:

Najpierw gracz ustawia swoje statki, zaczynając od jednego 5-masztowca i 4-masztowca. Później dwa 3-masztowce i jeden dwumasztowiec. Za każdym razem sprawdzana jest poprawność podanych współrzędnych. Następnie losowo generowana jest plansza przeciwnika, wg powyższej zasady. Pierwszeństwo strzału jest losowe W pętli gry sprawdzane jest czy statki są zniszczone. Po strzale wyświetlane są dwie tablice: przeciwnika i gracza. Wyświetlone są współrzędne strzału oraz czy nastąpiło trafienie(Hit)(‘#’) czy pudło(miss)(‘x’). Jeśli na mapie znajduje się fragment statku (znak ‚1;2;3;4;5;’) to zmieniamy reprezentację fragmentu statku na znak ‚#’ i zwiększamy obrażenia. Jeśli obrażenia równają się długości statku, to łódź jest zatopiona. jeśli strzelamy w już ostrzelany obszar, wyświetlony zostaje komunikat o zmarnowanym strzale. Następnie strzela komputer –strzały są losowe dopóki nie trafimy na obiekt. Jeśli komputer trafił w statek to ograniczamy pole do odstrzału do pół sąsiednich . po zatopieniu statku znowu komputer strzela losowo. Gra zostaje przerwana gdy jedna ze stron wystrzelała statki oponenta i następuje ogłoszenie zwycięzcy.

Schemat blokowy:



5.Opis formatu danych wejściowych:

-typedef enum contents {EMPTY=0, FRIGATE, WARSHIP, DESTROYER, SUBMARINE, BOAT} -contents;- nazwy statków(int)

-typedef enum direction {EAST=0, NORTH, WEST, SOUTH} direction - kierunki umieszczania statków(int)

-typedef enum state {PRISTINE=0, HIT, SUNK} state;

typedef struct ship {

contents name;

int location[2];

direction direction;

int health;

} ship; - struktura statków

typedef struct player {

ship ships[5];

bool floating[5];

} player; - struktura gracza

typedef struct storage{

int skill;

state state;

int endpoints;

orientation orientation[;

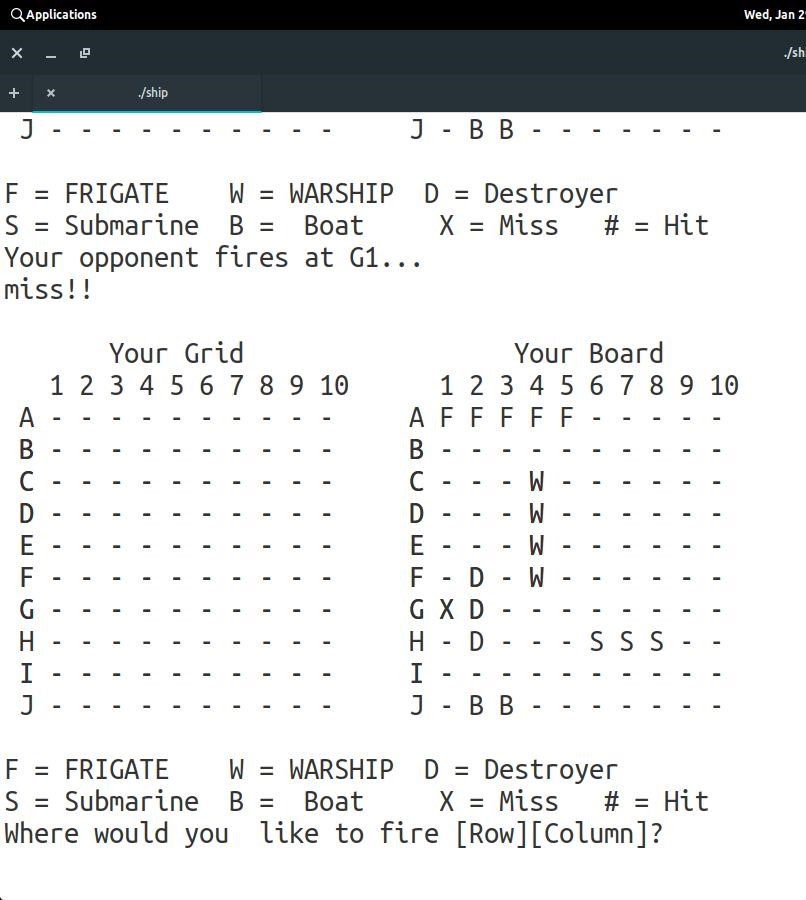
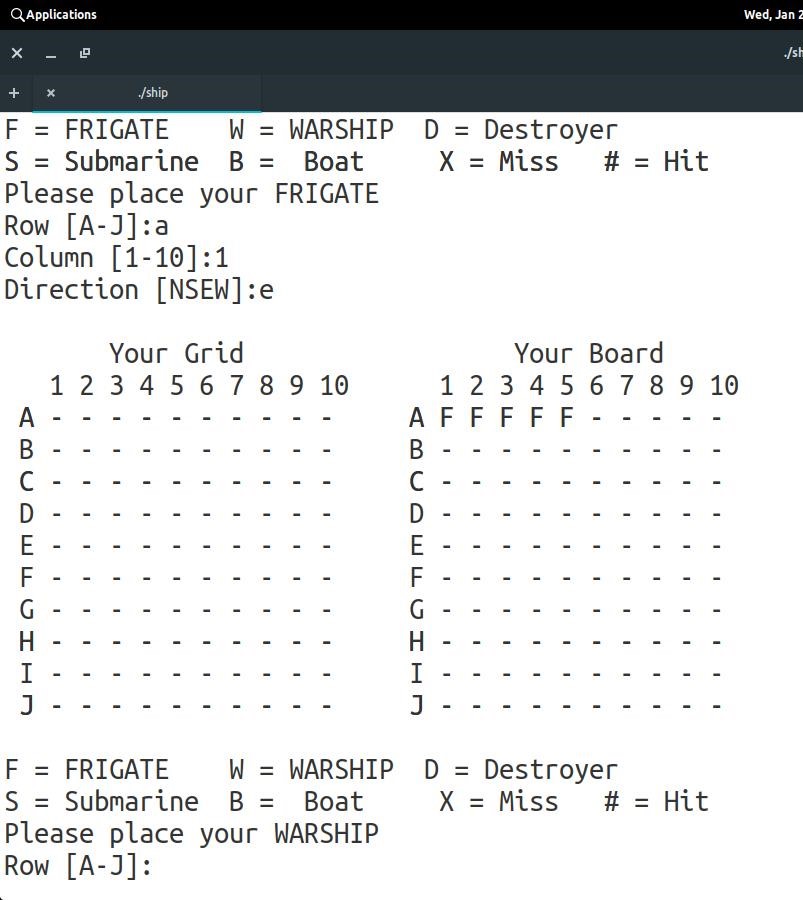
bool patternLikeParity;

} storage; - struktura komputera

Int x,y- współrzędne strzałów.

6.spis funkcji i zmiennych:

|  |  |  |
| --- | --- | --- |
| Nazwa funkcji | Opis funkcji | zmienne |
| void initBoards | Inicjuje tablice | Playerboard,  opponentboard |
| void initPlayer | Inicjuje statki gracza | i |
| int placeShip | Rozmieszczenie statków | ship \* sh, int x, int y, direction dir |
| int checkHit | Sprawdzenie strzału | int x, int y |
| int RNG | Losowanie | int low, int high |
| void place\_Computer\_Ships | inicjowanie statków przez komputer | Int x,y,i |
| void init\_Computer | rozmieszczenie statków przez komputer | int level |
| void Computer\_Fire | Strzał komputera | int \* x, int \* y |
| void initOpponent | Deklaracja komputera | Int Level |
| void drawBoard | Rysowanie pola gry | bool showHidden  int i,j,k |
| void getShipPlacement | Rozmieszczenie statków gracza | int \* x, int \* y, direction \* dir |
| void placeLocalShips | Umieszczenie statków na mapie | int x, y,direction dir  bool err |
| void opponentFire | Strzał przeciwnika | int \* x, int \* y |
| void takeTurn(void) | Rozpoczęcie rundy | int x,y, char a, int result |

7. screeny działającego programu:

