

Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Programowanie Komputerów 2

Konsolowa gra w statki

autor	Mateusz Adamczyk
prowadzący	mgr. inż. Wojciech Łabaj
rok akademicki	2019/2020
kierunek	informatyka
rodzaj studiów	SSI
semestr	2
termin laboratorium	środa, 11.45-13.15
sekcja	71
termin oddania sprawozdania	11.08.2020

1. Temat zadania

Tematem zadania było napisanie konsolowej gry w statki z trybem Gracz na Gracza oraz Gracz na Komputer z wykorzystaniem prostej sztucznej inteligencji.

Oprócz tego program miał zawierać:

- operacje wejścia/wyjścia,
- dynamiczną alokację pamięci
- obsługę plików
- obsługę łańcuchów znakowych c-string
- podział na pliki źródłowe i nagłówkowe

2. Analiza zadania

Zagadnienie obejmuje problem komunikacji z użytkownikiem, walidacji ruchów użytkownika, tak by nie mógł on złamać zasad gry, dynamicznych struktur oraz problem napisania prostej sztucznej inteligencji, która będzie podejmować najbardziej optymalne ruchy dla komputera.

2.1 Struktury danych

Program zawiera dwie główne struktury. Są to **Uzytkownik** oraz **Komputer**.

Struktura **Uzytkownik** zawiera w sobie wskaźnik na listę podwieszaną tworzącą planszę do gry (struktura **Wspolrzedne_y**, która zawiera wskaźnik na koordynaty na osi OX - struktura **Wspolrzedne_x**) oraz informację o aktualnych punktach zdobytych przez gracza jak i informacje o statkach, które gracz może jeszcze ułożyć na planszy do gry.

Strukturą bliźniaczą do struktury **Uzytkownik** jest struktura **Komputer**, która podobnie jak struktura **Uzytkownik** zawiera w sobie wskaźnik na listę podwieszaną tworzącą planszę do gry, informację o punktach zdobytych przez komputer oraz informację o tym, które statki komputer może jeszcze ułożyć na planszy. Poza tym struktura zawiera w sobie wskaźnik na listę strzałów wykonanych przez komputer - struktura **Komputer_strzelanie**, wskaźnik na listę kierunków, w które komputer próbował iść z danego pola - struktura **Lista_kierunkow** oraz zmienne **status_strzalu** i **status_drugiego_strzalu**, które informują o tym, której funkcji komputer ma użyć wykonując następny ruch.

Wcześniej wymieniona struktura **Wspolrzedne_y** to główna lista listy podwieszanej (listy list), która zawiera w sumie **koordynat y** na planszy do gry, wskaźnik na strukturę **Wspolrzedne_x** jak i wskaźniki **pUp** oraz **pDown**, które wskazują na sąsiednie elementy listy **Wspolrzedne_y**.

Struktura **Wspolrzedne_x** to lista, która zawiera w sobie **koordynat x** na planszy do gry, informację o tym jaki **status** ma dane pole (możliwe są statusy: **0** - pole puste, **1** - pole zajęte przez statek, **2** - statek trafiony, **3** - statek zbity, **4** - pudło), wskaźniki **pNext** oraz **pPrev**, które wskazują na sąsiednie pola listy. Dodatkowo, jeżeli na danym polu znajduje się statek zmienna **ustawienie** przyjmie wartość **false** jeżeli statek jest ustawiony horyzontalnie lub **true** jeżeli statek jest ustawiony wertykalnie.

Struktura **Komputer_strzelanie** to lista, która zawiera w sobie **koordynat x** oraz **koordynat y** strzału, informację o **kierunku**, w którym strzelał komputer (możliwe są kierunki: **0** - jeżeli był to pierwszy strzał, **1** - jeżeli komputer strzelał w prawo od ostatniego pola, **2** - jeżeli komputer strzelał na lewo od ostatniego pola, **3** - jeżeli komputer strzelał w dół w stosunku do ostatniego pola, **4** - jeżeli komputer strzelał w górę w stosunku do ostatniego pola) oraz informację o tym czy strzał był trafiony - zmienna **status_strzału** przyjmuje wartość **true** jeżeli strzał był trafiony lub **false** jeżeli był chybiony. Lista zawiera również wskaźniki **pNext** oraz **pPrev**, które są wskaźnikami na sąsiednie elementy listy.

Struktura **Lista_kierunkow** to lista, która zawiera informacje o **kierunkach**, w które próbował strzelać statek (kierunek może przyjąć takie same wartości jak te opisane w strukturze **Komputer_strzelanie** bez wartości **0**) oraz wskaźnik **pNext**, który wskazuje na następny element listy.

Struktura **Lista_ustawien** to lista, która przechowuje wszystkie możliwe ustawienia, które zostały zatwierdzone w funkcji **Komputer_dostaw_statek**, z których potem losowane jest jedno ustawienie statku, który zostaje umieszczony na planszy. Lista zawiera koordynaty początkowe i końcowe na osi X (**xp** - x początkowe, **xk** - x końcowe, **yp** - y początkowe, **yk** - y końcowe) oraz wskaźnik **pNext**, który jest wskaźnikiem na następny element listy.

```
struct Wspolrzedne_x
{
    int x;
    int status;
    struct Wspolrzedne_x* pNext;
    struct Wspolrzedne_x* pPrev;
    bool ustawienie;
};
```

```
struct Uzytkownik
{
    struct Wspolrzedne_y* Tablica;
    int punkty;
    int ilosc_czworek;
    int ilosc_trojek;
    int ilosc_dwojek;
    int ilosc_jedynek;
```

```
struct Wspolrzedne_y
{
    int y;
    struct Wspolrzedne_x* pHeadx;
    struct Wspolrzedne_y* pUp;
    struct Wspolrzedne_y* pDown;
};
```

```
struct Komputer_strzelanie
{
    int x;
    int y;
    struct Komputer_strzelanie* pNext;
    struct Komputer_strzelanie* pPrev;
    int kierunek;
    bool status_strzału;
```

```
};
struct Komputer
{
    struct Wspolrzedne_y* Tablica;
    int punkty;
    int ilosc_czworek;
    int ilosc_trojek;
    int ilosc_dwojek;
    int ilosc_jedynek;
    struct Komputer_strzelanie* Strzaly;
    struct Lista_kierunkow* Kierunki;
    bool status_strzalu;
    bool status_drugiego_strzalu;
};
```

```
};
struct Lista_ustawien
{
    int xp;
    int xk;
    int yp;
    int yk;
    struct Lista_ustawien*pNext;
};

struct Lista_kierunkow
{
    int kierunek;
    struct Lista_kierunkow*pNext;
};
```

Użycie listy list pozwala na szybkie i sprawne poruszanie się po całej planszy gry w celu sprawdzenia, czy gracz nie zamierza swoim ruchem złamać zasad gry. W innych przypadkach listy pozwalają na bardzo łatwe przejście przez całą listę, w celu znalezienia potrzebnych danych.

Wszystkie listy powstają przy użyciu podobnych funkcji. W zależności od tego, która wersja wydawała mi się bardziej optymalna dla danej struktury listy powstają przy pomocy dodania na koniec lub na początek listy.

Przykład funkcji dodającej na koniec listy:

```
void nowy_X(struct Wspolrzedne_x** pHead_ref, int X)
{
    struct Wspolrzedne_x* nowy = (struct Wspolrzedne_x*)malloc(sizeof(struct Wspolrzedne_x));
    nowy->x = X; //przypisanie wartosci koordynatu X nowemu wezlowi
    nowy->status = 0; //ustawienie podstawowego statusu
    nowy->pNext = nowy->pPrev = NULL; //wyzerowanie NULL na wskaznikach pobocznych
    if (!(*pHead_ref)) //jezeli nie ma glowy
        *pHead_ref = nowy; //nowy staje sie glową
    else //jezeli jest glową
    {
        struct Wspolrzedne_x* pomocniczy = *pHead_ref;
        while (pomocniczy->pNext != NULL) //przejdzie na koniec listy
            pomocniczy = pomocniczy->pNext;
        pomocniczy->pNext = nowy; //polaczenie ostatniego elementu listy z nowym
        nowy->pPrev = pomocniczy; //polaczenie nowego z ostatnim elementem listy
    }
}
```

Przykład funkcji dodającej na początek listy:

```
void komputer_dodaj_strzal(struct Komputer_strzelanie** pHead_ref, int x, int y, int kierunek, bool trafienie)
{
    struct Komputer_strzelanie* nowy = (struct Komputer_strzelanie*)malloc(sizeof(struct Komputer_strzelanie));
    nowy->x = x; //przypisanie odpowiednich wartosci nowemu wezlowi listy
    nowy->y = y;
    nowy->kierunek = kierunek;
    nowy->status_strzalu = trafienie;
    nowy->pNext = nowy->pPrev = NULL; //ustawienie wskaznikow pobocznych jako NULL
    if (!(*pHead_ref)) //jezeli nie ma glowy
        *pHead_ref = nowy; //nowy staje sie glową
    else //jezeli jest glową
    {
        struct Komputer_strzelanie* pomocniczy = *pHead_ref; //zapisanie starej glowy
        *pHead_ref = nowy; //nowy staje sie nową glową
        nowy->pNext = pomocniczy; //powiazanie nowej glowy ze stara
        pomocniczy->pPrev = nowy; //powiazanie starej glowy z nowa
    }
}
```

3. Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs od logiki aplikacji.

3.1. Ogólna struktura programu

Program jest podzielony na pliki:

- main.c
- funkcje.c
- funkcje.h
- struktury.h

3.2 Algorytmy i opis kluczowych funkcji

Najważniejszym algorytmem w programie jest algorytm strzelania przez komputer. Algorytm ten wykorzystuje dwie zmienne: **status_strzalu** oraz **status_drugiego_strzalu** jeżeli obie te zmienne przyjmą wartość **false** wywołana zostanie funkcja **komputer_pierwszy_strzal**.

Jeżeli zmienna **status_strzalu** przyjmie wartość **true**, a zmienna **status_drugiego_strzalu** przyjmie wartość **false** wywołana zostanie funkcja **komputer_drugi_strzal**.

Natomiast jeżeli obie zmienne przyjmą wartość **true** wywołana zostanie funkcja **komputer_dalszy_strzal**.

Funkcja **komputer_pierwszy_strzal** losuje koordynaty strzału. Dopóki nie wylosuje koordynatów, które nie zostały wcześniej wykorzystane. Następnie jeżeli strzał jest możliwy do wykonania wywołuje funkcję **komputer_strzal**, która wraca informację o rezultacie ruchu. Jeżeli komputer trafił w statek ale go nie zbił funkcja zwróci wartość **2**, jeżeli komputer zbił statek funkcja zwróci wartość **3**, natomiast jeżeli komputer spudłuje funkcja zwróci wartość **4**. Każdy strzał zostaje zapisany do listy strzałów. Jeżeli komputer wylosował koordynaty pola, w które nie strzelał, ale pole zmieniło swój **status** na **3** ponieważ sąsiadowało ze zbitym statkiem, komputer zapisze te pole do listy strzałów, by nie wylosował go następnym razem.

Po odblokowaniu możliwości korzystania z funkcji **komputer_drugi_strzal** funkcja ta sprawdza, czy ostatni strzał był celny. Jeżeli nie - funkcja będzie losowała nowy **kierunek**, by trafić na kierunek, który nie był wcześniej używany z tego pola. W zależności od wybranego kierunku funkcja odpowiednio dostosuje nowe **koordynaty**. Następnie koordynaty zostaną sprawdzone pod względem tego, czy nie wychodzą poza planszę oraz czy nie były już wcześniej wykorzystane do strzału. Jeżeli strzał zostanie zatwierdzony wywołana zostanie funkcja **komputer_strzal** i tak samo jak w funkcji **komputer_pierwszy_strzal** strzał zostanie zapisany na liście strzałów. Jeżeli komputer trafi, ale nie zbije statku zmienna **status_drugiego_strzalu** przyjmie wartość **true** tak, by następnym razem została wywołana funkcja **komputer_dalszy_strzal**. Jeżeli komputer zbije statek zmienna **status_strzalu** przyjmie wartość **false**, by następnym razem wywołana została funkcja **komputer_pierwszy_strzal**.

Funkcja **komputer_dalszy_strzal** sprawdza, czy ostatni strzał był trafiony. Jeżeli tak to komputer będzie kontynuował strzelanie we wcześniej wybranym kierunku. Jeżeli jednak

ostatni strzał był chybiony komputer wróci do pierwszego trafionego pola z tego statku (będzie szukał w liście strzałów ostatniego razu, gdy strzelał w kierunku **0**) po czym zacznie strzelać w kierunku przeciwnym do kierunku ostatnio oddanego przez siebie strzału. Ponownie każdy strzał zostanie zapisany do listy strzałów. Jeżeli komputer zbije statek zmienne **status_strzalu** oraz **status_drugiego_strzalu** przyjmą wartość **false** tak, by następnym razem została wywołana funkcja **komputer_pierwszy_strzal**.

3.3 Opis innych ważnych algorytmów

Funkcja **walidacja_zbicia** sprawdza, czy w pobliżu pola, w które właśnie trafiono znajdują się inne pola oznaczające statek. Jeżeli zmienna **ustawienie** pola, w które trafiono miała wartość **false** komputer będzie sprawdzał sąsiedztwo na osi OX, natomiast jeżeli zmienna **ustawienie** miała wartość **true** będzie sprawdzał sąsiedztwo na osi OY. Algorytm będzie sprawdzał sąsiedztwo dopóki nie trafi na pole oznaczające koniec potencjalnego statku. Jeżeli po drodze natrafi na statek zwróci wartość **false**, a w innym przypadku zwróci wartość **true**.

Funkcja **zbij** zmienia **status** pól zajmowanych przez trafiony (status ma wartość **2**) statek na pola zbite (status przyjmuje wartość **3**). Algorytm będzie zmieniał status dopóki nie natrafi na pole oznaczające koniec statku. Zmieni wtedy status pola przez (lub za) statkiem na zбитy. Zapamiętywany jest koordynat, na którym przestano zmieniać status pól i następnie zmieniane są pola pod oraz nad statkiem, aby zaznaczyć miejsca, w których nie może być statku.

3.4 Obsługa plików

Program zajmuje się obsługą plików. Plikami, które obsługuje program są:

- **presety.txt** - plik, w którym zapisane są gotowe zestawy statków gotowe do wczytania przez komputer
- **logi.txt** - plik, w którym zapisywane są informacje o wszystkich ruchach wykonane przez graczy podczas ostatniej rozgrywki

Nazwy plików są stałe.

Jeżeli plik **logi.txt** nie istnieje zostanie utworzony, natomiast jeżeli istnieje, jego treść zostanie nadpisana nową. Do zapisania informacji do pliku **logi.txt** wykorzystywana jest funkcja **fprintf**.

Fragment przykładowego pliku **logi.txt**:


```
Tue Aug 11 18:17:18 2020 Gracz miedzy x=5 a x=5 ustawil statek na wysokosci y= 7.
Tue Aug 11 18:17:21 2020 Gracz miedzy x=2 a x=3 ustawil statek na wysokosci y= 2.
Tue Aug 11 18:17:24 2020 Komputer miedzy x=8 a x=5 ustawil statek na wysokosci y= 3.
Tue Aug 11 18:17:24 2020 Komputer miedzy x=3 a x=1 ustawil statek na wysokosci y= 9.
Tue Aug 11 18:17:24 2020 Komputer miedzy x=9 a x=7 ustawil statek na wysokosci y= 7.
Tue Aug 11 18:17:24 2020 Komputer miedzy y=9 a y=10 ustawil statek na szerokosci x= 8.
Tue Aug 11 18:17:24 2020 Komputer miedzy x=10 a x=9 ustawil statek na wysokosci y= 5.
Tue Aug 11 18:17:24 2020 Komputer miedzy y=7 a y=8 ustawil statek na szerokosci x= 5.
Tue Aug 11 18:17:24 2020 Komputer miedzy x=8 a x=8 ustawil statek na wysokosci y= 1.
Tue Aug 11 18:17:24 2020 Komputer miedzy x=3 a x=3 ustawil statek na wysokosci y= 3.
Tue Aug 11 18:17:24 2020 Komputer miedzy x=4 a x=4 ustawil statek na wysokosci y= 1.
Tue Aug 11 18:17:24 2020 Komputer miedzy x=2 a x=2 ustawil statek na wysokosci y= 7.
Tue Aug 11 18:17:28 2020 Gracz spudlowal strzelajac w pole x= 4 y= 4.
Tue Aug 11 18:17:34 2020 Komputer spudlowal strzelajac w pole x= 7 y= 1.
Tue Aug 11 18:17:51 2020 Gracz spudlowal strzelajac w pole x= 9 y= 8.
Tue Aug 11 18:17:57 2020 Komputer spudlowal strzelajac w pole x= 2 y= 1.
Tue Aug 11 18:18:04 2020 Gracz spudlowal strzelajac w pole x= 2 y= 6.
Tue Aug 11 18:18:10 2020 Komputer spudlowal strzelajac w pole x= 4 y= 8.
Tue Aug 11 18:18:18 2020 Gracz spudlowal strzelajac w pole x= 2 y= 2.
Tue Aug 11 18:18:24 2020 Komputer spudlowal strzelajac w pole x= 10 y= 5.
Tue Aug 11 18:18:34 2020 Gracz spudlowal strzelajac w pole x= 6 y= 7.
Tue Aug 11 18:18:40 2020 Komputer spudlowal strzelajac w pole x= 2 y= 10.
Tue Aug 11 18:18:58 2020 Gracz trafil w statek na polu x= 8 y= 7.
Tue Aug 11 18:19:06 2020 Gracz spudlowal strzelajac w pole x= 8 y= 6.
Tue Aug 11 18:19:12 2020 Komputer spudlowal strzelajac w pole x= 6 y= 8.
Tue Aug 11 18:19:18 2020 Gracz spudlowal strzelajac w pole x= 8 y= 8.
Tue Aug 11 18:19:25 2020 Komputer spudlowal strzelajac w pole x= 9 y= 8.
Tue Aug 11 18:19:41 2020 Gracz trafil w statek na polu x= 7 y= 7.
Tue Aug 11 18:19:45 2020 Gracz trafil w statek na polu x= 9 y= 7 - statek zatopiony.
Tue Aug 11 18:19:57 2020 Gracz trafil w statek na polu x= 7 y= 3.
Tue Aug 11 18:20:02 2020 Gracz trafil w statek na polu x= 8 y= 3.
Tue Aug 11 18:20:06 2020 Gracz spudlowal strzelajac w pole x= 9 y= 3.
Tue Aug 11 18:20:12 2020 Komputer spudlowal strzelajac w pole x= 3 y= 3.
Tue Aug 11 18:20:19 2020 Gracz trafil w statek na polu x= 6 y= 3.
Tue Aug 11 18:20:23 2020 Gracz trafil w statek na polu x= 5 y= 3 - statek zatopiony.
Tue Aug 11 18:20:33 2020 Gracz trafil w statek na polu x= 2 y= 9.
Tue Aug 11 18:20:39 2020 Gracz spudlowal strzelajac w pole x= 2 y= 8.
Tue Aug 11 18:20:45 2020 Komputer spudlowal strzelajac w pole x= 1 y= 3.
Tue Aug 11 18:20:55 2020 Gracz trafil w statek na polu x= 3 y= 9.
Tue Aug 11 18:21:00 2020 Gracz spudlowal strzelajac w pole x= 4 y= 9.
Tue Aug 11 18:21:06 2020 Komputer spudlowal strzelajac w pole x= 9 y= 3.
Tue Aug 11 18:21:12 2020 Gracz trafil w statek na polu x= 1 y= 9 - statek zatopiony.
Tue Aug 11 18:21:28 2020 Gracz spudlowal strzelajac w pole x= 2 y= 5.
Tue Aug 11 18:21:34 2020 Komputer spudlowal strzelajac w pole x= 1 y= 7.
Tue Aug 11 18:21:43 2020 Gracz spudlowal strzelajac w pole x= 4 y= 5.
Tue Aug 11 18:21:49 2020 Komputer trafil w statek na polu x= 10 y= 3.
Tue Aug 11 18:21:55 2020 Komputer trafil w statek na polu x= 10 y= 2.
Tue Aug 11 18:22:01 2020 Komputer trafil w statek na polu x= 10 y= 1 - statek zatopiony.
Tue Aug 11 18:22:07 2020 Komputer spudlowal strzelajac w pole x= 8 y= 6.
Tue Aug 11 18:22:15 2020 Gracz trafil w statek na polu x= 3 y= 3 - statek zatopiony.
Tue Aug 11 18:22:22 2020 Gracz spudlowal strzelajac w pole x= 7 y= 10.
Tue Aug 11 18:22:28 2020 Komputer spudlowal strzelajac w pole x= 7 y= 7.
Tue Aug 11 18:22:37 2020 Gracz trafil w statek na polu x= 10 y= 5.
```

Do odczytu danych pliku zostały wykorzystane funkcje **fscanf** oraz **fgets**. Jeżeli plik **presety.txt** nie istnieje lub jest wadliwy komputer samodzielnie ustawi statki na planszy przy pomocy funkcji **komputer_dostaw_statek**. Przykład prawidłowo zapisanego zestawu w pliku **presety.txt**:


SET 1:
4.1:
USTAWIENIE:TRUE
1:7
2:3
3:6
3.1:
USTAWIENIE:TRUE
1:5
2:7
3:9
3.2:
USTAWIENIE:TRUE
1:3
2:10
3:8
2.1:
USTAWIENIE:TRUE
1:4
2:3
3:2
2.2:
USTAWIENIE:TRUE
1:2
2:6
3:5
2.3:
USTAWIENIE:TRUE
1:1
2:10
3:9
1.1:
USTAWIENIE:FALSE
1:3
2:1
3:1
1.2:
USTAWIENIE:FALSE
1:6
2:9
3:9
1.3:
USTAWIENIE:FALSE
1:10
2:8
3:8
1.4:
USTAWIENIE:FALSE
1:2
2:10
3:10

Tak samo jak w przypadku innych funkcji sprawdzających ustawienie jeżeli ustawienie przyjmuje wartość **false** oznacza to, że statek jest ustawiony horyzontalnie, a w przypadku wartości **true** wertykalnie. Pierwsza wartość oznacza koordynat niezmienny (czyli dla ustawienia horyzontalnego **koordynat y**, a w przypadku wertykalnego **koordynat x**), a pozostałe dwie wartości oznaczają koordynaty, między którymi ustawiany jest statek.

4. Specyfikacja zewnętrzna

Program po uruchomieniu wyświetla swoje menu. Program komunikuje się z użytkownikiem i wyświetla informacje o wszystkich próbach złamania zasad gry przez użytkownika, o niepoprawnych strzałach (np. gdy kolejny raz chce strzelać w te samo pole), o rezultacie każdego oddanego strzału oraz o tym, kto wygrał grę. Dodatkowo program wyświetla planszę do gry przy pomocy funkcji **wyświetl_tablice**, gdy wyświetlana jest tablica gracza ustawiającego statki oraz funkcji **wyświetl_tablice_przeciwnika** gdy wyświetlana jest tablica przeciwnika, gdzie nie są rozróżniane pola puste oraz pola zajmowane przez statki.

Menu:



```
C:\Users\matad\Desktop\Statki\Debug\Statki.exe
Witam w grze statki autorstwa Mateusza Adamczyka.
MENU:
1. Gracz na Gracza
2. Gracz na Komputer
3. Zasady gry

Jeżeli chcesz wyłączyć grę naciśnij ESC, ale pieskom będzie wtedy bardzo smutno :(((
_
```

Okno z zasadami gry:

Zasady gry:
1. Gra odbywa się na planszy 10 x 10
2. Gra jest możliwa w trybie Gracz na Gracza oraz Gracz na Komputer.
3. Każdy gracz na starcie gry ustawia zestaw składający się z:
- Jednego statku zajmującego cztery pola
- Dwoch statków zajmujących trzy pola
- Trzech statków zajmujących dwa pola
- Czterech statków zajmujących jedno pole
4. Statki nie mogą stykać się ze sobą.
5. Gracz strzela w planszę przeciwnika, dopóki nie spudluje.
6. Plansza wyświetlana jest przy użyciu następujących znaków:
- . - oznacza puste pole
- O - oznacza pole zajęte przez statek
- X - oznacza trafiony statek
- x - oznacza zatopiony statek oraz pola dookoła niego
- X - oznacza pudło
7. Pierwsza osoba, która zatopi wszystkie statki przeciwnika - zdobędzie 20 punktów - wygrywa.
8. Logi z ostatnio rozegranej gry są dostępne w pliku logi.txt.

Nacisnij dowolny przycisk by wrócić do menu.

Po wybraniu trybu gry użytkownik staje przed wyborem samodzielnego ustawienia statków, lub wyręczenia się komputerem:

Gracz ustawia statki
Czy chcesz sam ustawić swoje statki?
1. Tak
2. Nie - komputer ustawi statki za mnie
-

Samodzielne ustawianie statków przez użytkownika:

```
C:\Users\matad\Desktop\Statki\Debug\Statki.exe
Gracz ustawia statki

10 0 . . . . . . . . . .
 9 . . . . 0 . . 0 0 .
 8 . . . . . . . . . .
 7 . 0 . . . . . . . .
 6 . 0 . . . . . . 0 .
 5 . 0 . . . . . . . .
 4 . 0 . . . . . . . .
 3 . . . . . . . . . .
 2 . . . . 0 0 . . . .
 1 . . . . . . . . . .
 0 1 2 3 4 5 6 7 8 9 10

Ilosc Jedynek: 1
Ilosc Dwojek: 1
Ilosc Trojek: 2
Ilosc Czwoerek: 0
Podaj rodzaj statku: 3
Podaj początkowe koordynaty statku(x y): 4 4
Podaj kierunek ustawienia statku(<, >, v, ^): _
```

Screenshot ze środka rozgrywki (gdy strzelamy na plansze przeciwnika):

```
C:\Users\matad\Desktop\Statki\Debug\Statki.exe
Gracz strzela
Aktualne punkty: 12
10 X X X X . . X . . .
 9 X X X X . . . . .
 8 X X X X . X X X X X
 7 . . . . . X X X X X
 6 . X . . . X X X X X
 5 . X . X . . . . . X
 4 . X X X X X X X X .
 3 . X X X X X X X X .
 2 . X X X X X X X X .
 1 . . . . . . . . .
 0 1 2 3 4 5 6 7 8 9 10

Podaj koordynaty strzału(x y):
```

Screenshot ze środka rozgrywki (gdy komputer strzela na planszę użytkownika):

```
C:\Users\matad\Desktop\Statki\Debug\Statki.exe
Komputer strzela
Aktualne punkty: 7
10 0 X . . . . X X X X
9 X . X . 0 . X X X X
8 X X . X . X X X X X
7 X X . X 0 . X . . X
6 X X . . . X . X 0 .
5 . 0 . . . . X . . X
4 . 0 . 0 0 0 . . X X
3 X X X . . . X . X X
2 X 0 0 . 0 0 . . X X
1 . X . . . . X X X X
0 1 2 3 4 5 6 7 8 9 10

Komputer spudlował!
```

Screenshot z końca gry:

```
C:\Users\matad\Desktop\Statki\Debug\Statki.exe
Gracz pokonał Komputer!
Tablica gracza
10 0 X . . X . X X X X
9 X X X . 0 . X X X X
8 X X X X X X X X X X
7 X X X X 0 . X . . X
6 X X X X . X . X 0 X
5 X X X . . X X . . X
4 X X X 0 0 0 X . X X
3 X X X X X X X . X X
2 X 0 0 X X X X . X X
1 X X X X X X X X X X
0 1 2 3 4 5 6 7 8 9 10

Tablica Komputera:
10 X X X X X X X X X X
9 X X X X X X X X X X
8 X X X X X X X X X X
7 X X X X X X X X X X
6 X X X X X X X X X X
5 X X X X X X X X X X
4 X X X X X X X X X X
3 X X X X X X X X X X
2 X X X X X X X X X X
1 X X X X X X X X X X
0 1 2 3 4 5 6 7 8 9 10

Nacisnij dowolny przycisk by wroci do menu
```

5. Testowanie

Program został przetestowany pod względem błędnego pliku lub jego braku oraz w przypadku błędnych (niezgodnych z regułami gry) danych wejściowych podawanych przez użytkownika. W przypadku sytuacji błędnych danych podanych przez użytkownika program informuje użytkownika o tym użytkownika i ignoruje te dane, by nie zakłóciły jego działania. W przypadku błędnego pliku wejściowego lub jego braku program samodzielnie ustawi statki przy pomocy funkcji **komputer_dostaw_statek**.

6. Wnioski

Gra konsolowa w statki wymagała ode mnie przypomnienia oraz rozwinięcia moich umiejętności programowania zgodnego z paradygmatem strukturalnym, nauczyła mnie również podstaw algorytmu backtrackingu oraz nauczyła mnie programowania w języku C. W projekcie należało również samodzielnie zarządzać pamięcią, przez co należało zwrócić uwagę na zwalnianie zaalokowanej pamięci, w celu uniknięcia wycieków. Istotna była też komunikacja z użytkownikiem oraz przedstawienie planszy do gry w czytelny sposób.

Statki

Generated by Doxygen 1.8.17

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 Komputer Struct Reference	5
3.1.1 Detailed Description	5
3.2 Komputer_strzelanie Struct Reference	6
3.2.1 Detailed Description	6
3.3 Lista_kierunkow Struct Reference	6
3.3.1 Detailed Description	6
3.4 Lista_ustawien Struct Reference	7
3.4.1 Detailed Description	7
3.5 Uzytkownik Struct Reference	7
3.5.1 Detailed Description	8
3.6 Wspolrzedne_x Struct Reference	8
3.6.1 Detailed Description	8
3.7 Wspolrzedne_y Struct Reference	9
3.7.1 Detailed Description	9
4 File Documentation	11
4.1 funkcje.h File Reference	11
4.1.1 Function Documentation	12
4.1.1.1 Gracz_na_Gracza()	12
4.1.1.2 Gracz_na_Komputer()	12
4.1.1.3 komputer_dalszy_strzal()	13
4.1.1.4 komputer_dodaj_kierunek()	13
4.1.1.5 komputer_dodaj_strzal()	13
4.1.1.6 komputer_dostaw_statek()	14
4.1.1.7 komputer_drugi_strzal()	14
4.1.1.8 komputer_pierwszy_strzal()	15
4.1.1.9 komputer_pobierz_statki()	15
4.1.1.10 komputer_standardowy_zestaw()	15
4.1.1.11 komputer_strzal()	16
4.1.1.12 komputer_usun_kierunki()	16
4.1.1.13 komputer_usun_strzaly()	16
4.1.1.14 komputer_validacja_kierunku()	17
4.1.1.15 komputer_validacja_ruchu()	17
4.1.1.16 menu()	17
4.1.1.17 nowe_ustawienie()	17
4.1.1.18 nowy_X()	18

4.1.1.19 nowy_Y()	18
4.1.1.20 pobierz_statki_dla_uzytkownika()	18
4.1.1.21 Standardowy_zestaw()	19
4.1.1.22 statek_na_plansze()	19
4.1.1.23 strzal()	19
4.1.1.24 ustaw_statek()	20
4.1.1.25 usun_gracza()	20
4.1.1.26 usun_komputer()	20
4.1.1.27 usun_listy()	21
4.1.1.28 usun_ustawienia()	21
4.1.1.29 usun_X()	21
4.1.1.30 walidacja_ilosci()	22
4.1.1.31 walidacja_koordinatow()	22
4.1.1.32 walidacja_rodzaju()	22
4.1.1.33 walidacja_statku()	22
4.1.1.34 walidacja_zbicia()	23
4.1.1.35 wyswietl_tablice()	23
4.1.1.36 wyswietl_tablice_przeciwnika()	23
4.1.1.37 zapisz_strzal_do_pliku()	25
4.1.1.38 zapisz_ustawienie_do_pliku()	25
4.1.1.39 zbij()	26
4.2 struktury.h File Reference	26
Index	27

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Komputer	5
Komputer_strzelanie	6
Lista_kierunkow	6
Lista_ustawien	7
Uzytkownik	7
Wspolzedne_x	8
Wspolzedne_y	9

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

funkcje.h	11
struktury.h	26

Chapter 3

Data Structure Documentation

3.1 Komputer Struct Reference

```
#include <struktury.h>
```

Data Fields

- struct [Wspolrzedne_y](#) * **Tablica**
- int **punkty**
- int **ilosc_czworek**
- int **ilosc_trojek**
- int **ilosc_dwojek**
- int **ilosc_jedynek**
- struct [Komputer_strzelanie](#) * **Strzaly**
- struct [Lista_kierunkow](#) * **Kierunki**
- bool **status_strzalu**
- bool **status_drugiego_strzalu**

3.1.1 Detailed Description

Struktura, która zawiera w sobie informacje o komputerze

Parameters

<i>Tablica</i>	glowa listy typu Wspolrzedne_y , która pełni rolę planszy do gry
<i>punkty</i>	ilość punktów, które aktualnie posiada komputer
<i>ilosc_czworek</i>	informacja ile statków typu czwartego komputer może jeszcze położyć na planszy @Param ilosc_trojek informacja ile statków typu trzeciego komputer może jeszcze położyć na planszy
<i>ilosc_dwojek</i>	informacja ile statków typu drugiego komputer może jeszcze położyć na planszy @param ilosc_jedynek informacja ile statków typu pierwszego komputer może jeszcze położyć na planszy
<i>Strzaly</i>	glowa na liście typu Komputer_strzelanie
<i>Kierunki</i>	glowa na liście typu Lista_kierunkow @bool status_strzalu informacja o tym, czy komputer może przejść do drugiego strzału @bool status_drugiego_strzalu informacja o tym, czy komputer może przejść do dalszego strzału

The documentation for this struct was generated from the following file:

- [struktury.h](#)

3.2 Komputer_strzelanie Struct Reference

```
#include <struktury.h>
```

Data Fields

- int **x**
- int **y**
- struct [Komputer_strzelanie](#) * **pNext**
- struct [Komputer_strzelanie](#) * **pPrev**
- int **kierunek**
- bool **status_strzalu**

3.2.1 Detailed Description

Lista dwukierunkowa, która przechowuje pola z planszy przeciwnika, w które strzelał już komputer

Parameters

<i>x</i>	koordynat x pola z planszy przeciwnika
<i>y</i>	koordynat y pola z planszy przeciwnika
<i>kierunek</i>	informacja w którym kierunku siedzi komputer strzelając w te pole
<i>status_strzalu</i>	informacja o tym czy komputer trafił czy spudłował

The documentation for this struct was generated from the following file:

- [struktury.h](#)

3.3 Lista_kierunkow Struct Reference

```
#include <struktury.h>
```

Data Fields

- int **kierunek**
- struct [Lista_kierunkow](#) * **pNext**

3.3.1 Detailed Description

Lista jednokierunkowa przechowująca kierunki w które z danego pola siedzi komputer

Parameters

<i>kierunek</i>	informacja w kierunku w ktorym probowal isc komputer
-----------------	--

The documentation for this struct was generated from the following file:

- [strukтуры.h](#)

3.4 Lista_ustawien Struct Reference

```
#include <strukтуры.h>
```

Data Fields

- int **xp**
- int **xk**
- int **yp**
- int **yk**
- struct [Lista_ustawien](#) * **pNext**

3.4.1 Detailed Description

Lista jednokierunkowa ktora zawiera koordynaty statkow wykorzystywane w ustawianiu losowo statkow przez komputer

Parameters

<i>xp</i>	poczatkowy koordynat statku na osi x
<i>xk</i>	koncowy koordynat statku na osi x
<i>yp</i>	poczatkowy koordynat statku na osi y
<i>yk</i>	koncowy koordynat statku na osi y
<i>pNext</i>	wskaznik na nastepny element listy

The documentation for this struct was generated from the following file:

- [strukтуры.h](#)

3.5 Uzytkownik Struct Reference

```
#include <strukтуры.h>
```

Data Fields

- struct [Wspolrzedne_y](#) * **Tablica**
- int **punkty**
- int **ilosc_czworek**
- int **ilosc_trojek**
- int **ilosc_dwojek**
- int **ilosc_jedynek**

3.5.1 Detailed Description

Struktura, która zawiera w sobie informacje o graczu

Parameters

<i>Tablica</i>	glowa listy typu Wspolrzedne_y , która pełni rolę planszy do gry
<i>punkty</i>	ilość punktów, które aktualnie posiada gracz
<i>ilosc_czworek</i>	informacja ile statków typu czwartego gracz może jeszcze położyć na planszy @Param ilosc_trojek informacja ile statków typu trzeciego gracz może jeszcze położyć na planszy
<i>ilosc_dwojek</i>	informacja ile statków typu drugiego gracz może jeszcze położyć na planszy @param ilosc_jedynek informacja ile statków typu pierwszego gracz może jeszcze położyć na planszy

The documentation for this struct was generated from the following file:

- [struktury.h](#)

3.6 Wspolrzedne_x Struct Reference

```
#include <struktury.h>
```

Data Fields

- int **x**
- int **status**
- struct [Wspolrzedne_x](#) * **pNext**
- struct [Wspolrzedne_x](#) * **pPrev**
- bool **ustawienie**

3.6.1 Detailed Description

Podwieszana lista dwukierunkowa, której elementami są:

Parameters

<i>x</i>	koordynat x
<i>status</i>	ustala status danego pola. 0 - oznacza puste pole. 1 - oznacza pole zajęte przez statek 2 - oznacza trafiony statek 3 - oznacza zбиты statek i pole dookoła niego 4 - oznacza pudło
<i>ustawienie</i>	przyjmuje wartość false, jeżeli obecny na nim statek jest ustawiony horyzontalnie lub true, gdy ustawiony na nim statek jest ustawiony wertykalnie

The documentation for this struct was generated from the following file:

- [strukтуры.h](#)

3.7 Wspolrzedne_y Struct Reference

```
#include <strukтуры.h>
```

Data Fields

- int **y**
- struct [Wspolrzedne_x](#) * **pHeadx**
- struct [Wspolrzedne_y](#) * **pUp**
- struct [Wspolrzedne_y](#) * **pDown**

3.7.1 Detailed Description

Lista dwukierunkowa, ktorej elementami sa:

Parameters

<i>y</i>	koordynat y
<i>pHeadx</i>	glowa listy podwieszanej typu Wspolrzedne_x

The documentation for this struct was generated from the following file:

- [strukтуры.h](#)

Chapter 4

File Documentation

4.1 funkcje.h File Reference

```
#include "struktury.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <windows.h>
#include <conio.h>
#include <string.h>
#include <time.h>
```

Functions

- void [nowy_X](#) (struct [Wspolrzedne_x](#) **pHead_ref, int X)
- void [nowy_Y](#) (struct [Wspolrzedne_y](#) **pHead_ref, int Y)
- void [Standardowy_zestaw](#) (struct [Uzytkownik](#) **uzytkownik_ref)
- void [wyswietl_tablice](#) (struct [Wspolrzedne_y](#) *tablica)
- bool [walidacja_statku](#) (struct [Wspolrzedne_y](#) **pHead, int yPocatkowe, int xPocatkowe, int yKoncowe, int xKoncowe)
- bool [walidacja_ilosci](#) (struct [Uzytkownik](#) *uzytkownik, int rodzaj)
- bool [walidacja_rodzaju](#) (int rodzaj, struct [Uzytkownik](#) *uzytkownik)
- bool [walidacja_koordinatow](#) (int x, int y)
- void [ustaw_statek](#) (struct [Uzytkownik](#) **uzytkownik_ref, char *nazwa)
- void [statek_na_plansze](#) (struct [Wspolrzedne_y](#) **pHead_ref, int Ypocatkowe, int Xpocatkowe, int Ykoncowe, int Xkoncowe)
- void [zapisz_ustawienie_do_pliku](#) (int xp, int yp, int xk, int yk, char *nazwa)
- void [wyswietl_tablice_przeciwnika](#) (struct [Wspolrzedne_y](#) *tablica)
- int [strzal](#) (struct [Wspolrzedne_y](#) **pHead_ref, char *nazwa)
- void [zapisz_strzal_do_pliku](#) (int x, int y, int status, char *nazwa)
- bool [walidacja_zbicia](#) (struct [Wspolrzedne_y](#) *pHead, struct [Wspolrzedne_x](#) *pHeadx)
- void [zbij](#) (struct [Wspolrzedne_y](#) **pHead_ref, struct [Wspolrzedne_x](#) **pHeadx_ref)
- void [Gracz_na_Gracza](#) (struct [Uzytkownik](#) **Gracz1_ref, struct [Uzytkownik](#) **Gracz2_ref)
- void [usun_X](#) (struct [Wspolrzedne_x](#) **pHead_ref)
- void [usun_listy](#) (struct [Wspolrzedne_y](#) **pHead_ref)
- void [usun_gracza](#) (struct [Uzytkownik](#) **Gracz_ref)
- void [komputer_standardowy_zestaw](#) (struct [Komputer](#) **Komputer_ref)

- void `komputer_dodaj_kierunek` (struct `Lista_kierunkow` **pHead_ref, int kierunek)
- bool `komputer_validacja_kierunku` (struct `Lista_kierunkow` *pHead, int kierunek)
- void `komputer_dostaw_statek` (struct `Wspolrzedne_y` **pHead_ref, int rodzaj, char *nazwa)
- void `komputer_pobierz_statki` (struct `Komputer` **Komputer_ref, int set)
- int `pobierz_statki_dla_uzytkownika` (struct `Uzytkownik` **Uzytkownik_ref, char *nazwa, int set)
- void `nowe_ustawienie` (struct `Lista_ustawien` **pHead_ref, int xp, int xk, int yp, int yk)
- void `usun_ustawienia` (struct `Lista_ustawien` **pHead_ref)
- void `komputer_usun_kierunki` (struct `Lista_kierunkow` **pHead_ref)
- void `komputer_dodaj_strzal` (struct `Komputer_strzelanie` **pHead_ref, int x, int y, int kierunek, bool trafienie)
- int `komputer_validacja_ruchu` (struct `Wspolrzedne_y` *pHead, struct `Komputer_strzelanie` *Strzaly, int x, int y)
- int `komputer_strzal` (struct `Wspolrzedne_y` **pHead_ref, int x, int y)
- int `komputer_pierwszy_strzal` (struct `Komputer` **Komputer_ref, struct `Wspolrzedne_y` **Tablica_przeciwnika_ref)
- int `komputer_drugi_strzal` (struct `Komputer` **Komputer_ref, struct `Wspolrzedne_y` **Tablica_przeciwnika_ref)
- int `komputer_dalszy_strzal` (struct `Komputer` **Komputer_ref, struct `Wspolrzedne_y` **Tablica_przeciwnika_ref)
- void `Gracz_na_Komputer` (struct `Uzytkownik` **Gracz_ref, struct `Komputer` **Komputer_ref)
- void `komputer_usun_strzaly` (struct `Komputer_strzelanie` **pHead_ref)
- void `usun_komputer` (struct `Komputer` **Komputer_ref)
- void `menu` ()

4.1.1 Function Documentation

4.1.1.1 Gracz_na_Gracza()

```
void Gracz_na_Gracza (
    struct Uzytkownik ** Gracz1_ref,
    struct Uzytkownik ** Gracz2_ref )
```

funkcja w której odbywa się rozgrywka typu Gracz na Gracza

Parameters

<i>Gracz1_ref</i>	adres pierwszego gracza
<i>Gracz2_ref</i>	adres drugiego gracza

4.1.1.2 Gracz_na_Komputer()

```
void Gracz_na_Komputer (
    struct Uzytkownik ** Gracz_ref,
    struct Komputer ** Komputer_ref )
```

funkcja w której odbywa się rozgrywka typu Gracz na `Komputer`

Parameters

<i>Gracz_ref</i>	adres gracza
<i>Komputer_ref</i>	adres komputera

4.1.1.3 komputer_dalszy_strzal()

```
int komputer_dalszy_strzal (
    struct Komputer ** Komputer_ref,
    struct Wspolrzedne_y ** Tablica_przeciwnika_ref )
```

funkcja która po znalezieniu osi, w której leży statek stara się go zatopić, jeżeli status ostatniego ruchu będzie negatywny funkcja wróci się do pierwszego strzału w aktualnie ostrzeliwanym statku i zacznie strzelać w przeciwnym kierunku

Parameters

<i>Komputer_ref</i>	adres komputera
<i>Tablica_przeciwnika_ref</i>	adres głowy listy będącej planszą przeciwnika
<i>plik</i>	adres pliku, w którym zostanie zapisana informacja o strzale funkcja zwraca 0 jeżeli strzał jest nieprawidłowy, 2 jeżeli komputer trafi w statek, 3 jeżeli zatopi statek -wtedy też Komputer->status_strzału jak i Komputer->status_drugiego_strzału zostanie zmieniony na false, i 4 jeżeli spudluje

4.1.1.4 komputer_dodaj_kierunek()

```
void komputer_dodaj_kierunek (
    struct Lista_kierunkow ** pHead_ref,
    int kierunek )
```

funkcja która dodaje nowy element do listy kierunków

Parameters

<i>pHead_ref</i>	adres głowy listy
<i>kierunek</i>	dodawany kierunek

4.1.1.5 komputer_dodaj_strzal()

```
void komputer_dodaj_strzal (
    struct Komputer_strzelanie ** pHead_ref,
```

```

int x,
int y,
int kierunek,
bool trafienie )

```

funkcja, która dodaje do listy strzałów nowy element

Parameters

<i>pHead_ref</i>	adres głowy listy
<i>x</i>	koordynat x strzału
<i>y</i>	koordynat y strzału
<i>kierunek</i>	kierunek, w którym siedzi komputer wykonując strzał
<i>trafienie</i>	informacja o tym, czy komputer trafił w statek, czy nie

4.1.1.6 komputer_dostaw_statek()

```

void komputer_dostaw_statek (
    struct Wspolrzedne_y ** pHead_ref,
    int rodzaj,
    char * nazwa )

```

funkcja, która ustawia statek wybranego rodzaju na planszy komputera

Parameters

<i>pHead_ref</i>	adres głowy listy
<i>rodzaj</i>	rodzaj statku
<i>plik</i>	adres pliku, w którym zostanie zapisana informacja o ustawieniu statku na plansze

4.1.1.7 komputer_drugi_strzal()

```

int komputer_drugi_strzal (
    struct Komputer ** Komputer_ref,
    struct Wspolrzedne_y ** Tablica_przeciwnika_ref )

```

funkcja, która po trafieniu w statek próbuje trafić w statek losując kierunek strzału

Parameters

<i>Komputer_ref</i>	adres komputera
<i>Tablica</i>	przeciwnika_ref adres głowy listy będąca planszą przeciwnika
<i>plik</i>	adres pliku, w którym zostanie zapisana informacja o strzale funkcja zwraca 0 jeżeli strzał jest nieprawidłowy, 2 jeżeli komputer trafi w statek - wtedy również Komputer->status_drugiego_strzału zmieniany jest na true, 3 jeżeli zatopi statek - wtedy też Komputer->status_strzału zostanie zmieniony na false, i 4 jeżeli spudłuje

4.1.1.8 komputer_pierwszy_strzal()

```
int komputer_pierwszy_strzal (
    struct Komputer ** Komputer_ref,
    struct Wspolrzedne_y ** Tablica_przeciwnika_ref )
```

funkcja, która losuje miejsce, w które strzeli komputer

Parameters

<i>Komputer_ref</i>	adres komputera
<i>Tablica_przeciwnika_ref</i>	adres głowy listy będąca planszą przeciwnika
<i>plik</i>	adres pliku, w którym zostanie zapisana informacja o strzale funkcja zwraca 0 jeżeli strzał jest nieprawidłowy, 2 jeżeli komputer trafi w statek - wtedy również Komputer->status_strzalu zmieniany jest na true, 3 jeżeli zatopi statek i 4 jeżeli spudluje

4.1.1.9 komputer_pobierz_statki()

```
void komputer_pobierz_statki (
    struct Komputer ** Komputer_ref,
    int set )
```

funkcja, która pobiera z pliku presets.txt losowy gotowy zestaw ustawień statków, które wykorzysta komputer

Parameters

<i>Komputer_ref</i>	adres Komputera
<i>plik_wyjsciowy</i>	wskaznik na plik, w którym są zapisywane logi gry
<i>set</i>	informacja o tym, który set statków nie może być pobrany do użycia

4.1.1.10 komputer_standardowy_zestaw()

```
void komputer_standardowy_zestaw (
    struct Komputer ** Komputer_ref )
```

funkcja, która przypisuje komputerowi jego standardowy zestaw do gry

Parameters

<i>Komputer_ref</i>	adres komputera
---------------------	-----------------

4.1.1.11 komputer_strzal()

```
int komputer_strzal (
    struct Wspolrzedne_y ** pHead_ref,
    int x,
    int y )
```

funkcja, która strzela w wybrane przez komputer miejsce

Parameters

<i>pHead_ref</i>	adres listy bedacej plansza przeciwnika
<i>x</i>	koordynat x strzalu
<i>y</i>	koordynat y strzalu
<i>plik</i>	adres pliku, w ktorym zostanie zapisana informacja o ruchu funkcja zwraca 2 jezeli komputer trafi w statek, 3 jezeli komputer zatopi statek lub 4 jezeli spudluje

4.1.1.12 komputer_usun_kierunki()

```
void komputer_usun_kierunki (
    struct Lista_kierunkow ** pHead_ref )
```

funkcja, która usuwa elementy z listy kierunkow

Parameters

<i>pHead_ref</i>	adres glowy listy
------------------	-------------------

4.1.1.13 komputer_usun_strzaly()

```
void komputer_usun_strzaly (
    struct Komputer_strzelanie ** pHead_ref )
```

funkcja która usuwa liste strzalow komputera

Parameters

<i>pHead_ref</i>	adres glowy listy
------------------	-------------------

4.1.1.14 komputer_walidacja_kierunku()

```
bool komputer_walidacja_kierunku (
    struct Lista_kierunkow * pHead,
    int kierunek )
```

funkcja ktora sprawdza, czy komputer nie strzelal juz w wybranym kierunku

Parameters

<i>pHead</i>	adres glowy
<i>kierunek</i>	wybrany kierunek

4.1.1.15 komputer_walidacja_ruchu()

```
int komputer_walidacja_ruchu (
    struct Wspolrzedne_y * pHead,
    struct Komputer_strzelanie * Strzaly,
    int x,
    int y )
```

funkcja, ktora sprawdza czy proponowany przez komputer strzal jest poprawny

Parameters

<i>pHead</i>	adres glowy listy bedacej plansza przeciwnika
<i>Strzaly</i>	adres glowy listy strzalow komputera
<i>x</i>	koordynat x strzalu
<i>y</i>	koordynat y strzalu funkcja zwraca 0 jezeli komputer strzelal juz w te miejsce, 1 jezeli ruch jest niepoprawny ze wzgledu na status pola lub 2 jezeli ruch jest poprawny

4.1.1.16 menu()

```
void menu ( )
```

funckcja ktora wyswietla menu projektu

4.1.1.17 nowe_ustawienie()

```
void nowe_ustawienie (
    struct Lista_ustawien ** pHead_ref,
    int xp,
    int xk,
    int yp,
    int yk )
```

funckja, ktora dodaje ustawienie statkow do listy poprawnych ustaiwnien, by potem moc wylosowac statek

Parameters

<i>pHead_ref</i>	adres glowy listy
<i>xp</i>	początkowy koordynat statku na osi x
<i>xk</i>	koncowy koordynat statku na osi x
<i>yp</i>	początkowy koordynat statku na osi y
<i>yk</i>	koncowy koordynat statku na osi y

4.1.1.18 nowy_X()

```
void nowy_X (
    struct Wspolrzedne_x ** pHead_ref,
    int X )
```

funkcja tworząca nowy element listy dwukierunkowej typu [Wspolrzedne_x](#)

Parameters

<i>pHead_ref</i>	adres pierwszego elementu listy
<i>X</i>	informacja jaka wspolrzedna przypisac elementowi

4.1.1.19 nowy_Y()

```
void nowy_Y (
    struct Wspolrzedne_y ** pHead_ref,
    int Y )
```

funkcja tworząca nowy element listy dwukierunkowej typu [Wspolrzedne_y](#)

Parameters

<i>pHead_ref</i>	adres pierwszego elementu listy
<i>Y</i>	informacja jaka wspolrzedna przypisac elementowi

4.1.1.20 pobierz_statki_dla_uzytkownika()

```
int pobierz_statki_dla_uzytkownika (
    struct Uzytkownik ** Uzytkownik_ref,
    char * nazwa,
    int set )
```

funkcja, która pobiera z pliku presety.txt losowy gotowy zestaw ustawień statków, które zostaną ustawione na plan-szy użytkownika

Parameters

<i>uzytkownik_ref</i>	adres uzytkownika
<i>nazwa</i>	nazwa uzytkownika
<i>set</i>	informacja o tym, ktory set nie moze byc pobrany do uzycia

4.1.1.21 Standardowy_zestaw()

```
void Standardowy_zestaw (
    struct Uzytkownik ** uzytkownik_ref )
```

funkcja ktora przypisuje graczowi standardowy zestaw do gry

Parameters

<i>uzytkownik_ref</i>	adres gracza
-----------------------	--------------

4.1.1.22 statek_na_plansze()

```
void statek_na_plansze (
    struct Wspolrzedne_y ** pHead_ref,
    int Ypocatkowe,
    int Xpocatkowe,
    int Ykoncowe,
    int Xkoncowe )
```

funkcja ktora zmienia status odpowiednich pol na planszy na statek

Parameters

<i>pHead_ref</i>	adres glowy listy
<i>Ypocatkowe</i>	początkowy koordynat na osi y
<i>Xpocatkowe</i>	początkowy koordynat na osi x
<i>Ykoncowe</i>	koncowy koordynat na osi y
<i>Xkoncowe</i>	koncowy koordynat na osi x

4.1.1.23 strzal()

```
int strzal (
    struct Wspolrzedne_y ** pHead_ref,
    char * nazwa )
```

funkcja w ktorej uzytkownik strzela w plansze przeciwnika

Parameters

<i>pHead_ref</i>	adres glowy listy
<i>nazwa</i>	nazwa gracza, ktorzy strzela
<i>plik</i>	adres pliku, w ktorym zapisana bedzie informacja o oddanym strzale funkcja zwraca 0 jezeli strzal jest nieprawidlowy, 2 jezeli gracz trafi w statek, 3 jezeli uzytkownik zatopi statek, 4 jezeli spudluje i 5 jezeli ruch bedzie wykraczal poza tablice

4.1.1.24 ustaw_statek()

```
void ustaw_statek (
    struct Uzytkownik ** uzytkownik_ref,
    char * nazwa )
```

funckja w ktorej gracz ustawia statki na planszy

Parameters

<i>uzytkownik_ref</i>	adres gracza
<i>plik</i>	adres pliku, w ktorym zostanie zapisana informacja o ustawieniu statku
<i>nazwa</i>	nazwa gracza, ktory ustawia statek

4.1.1.25 usun_gracza()

```
void usun_gracza (
    struct Uzytkownik ** Gracz_ref )
```

funkcja ktora usuwa strukture [Uzytkownik](#)

Parameters

<i>Gracz_ref</i>	adres gracza
------------------	--------------

4.1.1.26 usun_komputer()

```
void usun_komputer (
    struct Komputer ** Komputer_ref )
```

funkcja, ktora usuwa strukture komputera wywoluje funkcje komputer_usun_strzaly oraz komputer_usun_kierunki

Parameters

<i>Komputer_ref</i>	adres komputera
---------------------	-----------------

4.1.1.27 `usun_listy()`

```
void usun_listy (
    struct Wspolrzedne\_y ** pHead_ref )
```

funkcja, która usuwa liste typu `Wsolrzedne_y` i wywołuje funkcję `usun_X`

Parameters

<i>pHead_ref</i>	adres glowy listy
------------------	-------------------

4.1.1.28 `usun_ustawienia()`

```
void usun_ustawienia (
    struct Lista\_ustawien ** pHead_ref )
```

funkcja, która usuwa elementy z listy ustawien

Parameters

<i>pHead_ref</i>	adres glowy listy
------------------	-------------------

4.1.1.29 `usun_X()`

```
void usun_X (
    struct Wspolrzedne\_x ** pHead_ref )
```

funkcja, która usuwa liste typu `Wspolrzedne_x`

Parameters

<i>pHead_ref</i>	adres glowy listy
------------------	-------------------

4.1.1.30 walidacja_ilosci()

```
bool walidacja_ilosci (
    struct Uzytkownik * uzytkownik,
    int rodzaj )
```

funkcja sprawdzająca czy gracz nie wyczerpał już statków rodzaju, który chce ustawić

Parameters

<code>uzytkownik</code>	adres użytkownika
<code>rodzaj</code>	sprawdzany rodzaj statku

4.1.1.31 walidacja_koordinatow()

```
bool walidacja_koordinatow (
    int x,
    int y )
```

funkcja która sprawdza czy koordynaty nie wykraczają poza planszę do gry

Parameters

<code>x</code>	sprawdzany koordynat x
<code>y</code>	sprawdzany koordynat y

4.1.1.32 walidacja_rodzaju()

```
bool walidacja_rodzaju (
    int rodzaj,
    struct Uzytkownik * uzytkownik )
```

funkcja która sprawdza czy użytkownik podał prawidłowy rodzaj statku

Parameters

<code>uzytkownik</code>	adres użytkownika
<code>rodzaj</code>	sprawdzany rodzaj statku

4.1.1.33 walidacja_statku()

```
bool walidacja_statku (
    struct Wspolrzedne_y ** pHead,
```



```

int yPocatkowe,
int xPocatkowe,
int yKoncowe,
int xKoncowe )

```

funkcja ktora sprawdza czy proponowane ustawienie statku spelnia zasady gry

Parameters

<i>pHead</i>	adres pierwszego elementu listy
<i>yPocatkowe</i>	pocatkowy koordynat statku na osi y
<i>xPocatkowe,pocatkowy</i>	koordynat statku na osi x
<i>yKoncowe</i>	koncowy koordynat statku na osi y
<i>xKoncowe</i>	koncowy koordynat statku na osi x

4.1.1.34 walidacja_zbicia()

```

bool walidacja_zbicia (
    struct Wspolrzedne_y * pHead,
    struct Wspolrzedne_x * pHeadx )

```

funkcja sprawdzajaca czy uzytkownik zatopil statek @Param pHead adres glowy listy glownej

Parameters

<i>pHeadx</i>	adres elementu listy w ktory trafil gracz
---------------	---

4.1.1.35 wyswietl_tablice()

```

void wyswietl_tablice (
    struct Wspolrzedne_y * tablica )

```

funckcja, ktora wyswietla tablice gracza

Parameters

<i>tablica</i>	adres pierwszego elementu listy
----------------	---------------------------------

4.1.1.36 wyswietl_tablice_przeciwnika()

```

void wyswietl_tablice_przeciwnika (
    struct Wspolrzedne_y * tablica )

```

funkcja blizniacza do funkcji "wyswietl_tablice, ktora nie rozroznia pol pustych i pol zajetych przez statek

Parameters

<i>tablica</i>	adres glowy listy
----------------	-------------------

4.1.1.37 zapisz_strzal_do_pliku()

```
void zapisz_strzal_do_pliku (
    int x,
    int y,
    int status,
    char * nazwa )
```

funkcja, ktora zapisuje do pliku informacje o wykonanym strzale

Parameters

<i>plik</i>	adres pliku
<i>x</i>	koordynat x strzalu
<i>y</i>	koordynat y strzalu @paran status rezultat strzalu
<i>nazwa</i>	nazwa gracza, ktory strzela

4.1.1.38 zapisz_ustawienie_do_pliku()

```
void zapisz_ustawienie_do_pliku (
    int xp,
    int yp,
    int xk,
    int yk,
    char * nazwa )
```

funkcja ktora zapisuje do pliku informacje o ustawieniu statku na danych koordynatach

Parameters

<i>plik</i>	adres pliku, w ktorym zapisywana jest informacja
<i>xp</i>	poczatkowy koordynat na osi x
<i>yp</i>	poczatkowy koorynat na osi y
<i>xk</i>	koncowy koordynat na osi x
<i>yk</i>	koncowy koordynat na osi y
<i>nazwa</i>	nazwa gracza ustawiajacego statki

4.1.1.39 zbij()

```
void zbij (
    struct Wspolrzedne_y ** pHead_ref,
    struct Wspolrzedne_x ** pHeadx_ref )
```

funkcja ktora zmienia status pol zajetych przez statek jak i pola do okola niego na zbity

Parameters

<i>pHead_ref</i>	adres glowy glownej listy
<i>pHeadx_ref</i>	adres elementu listy, w ktory trafil gracz

4.2 struktury.h File Reference

```
#include <stdbool.h>
```

Data Structures

- struct [Wspolrzedne_x](#)
- struct [Wspolrzedne_y](#)
- struct [Uzytkownik](#)
- struct [Komputer_strzelanie](#)
- struct [Lista_kierunkow](#)
- struct [Komputer](#)
- struct [Lista_ustawien](#)

Index

funkcje.h, [11](#)

Gracz_na_Gracza, [12](#)

Gracz_na_Komputer, [12](#)

komputer_dalszy_strzal, [13](#)

komputer_dodaj_kierunek, [13](#)

komputer_dodaj_strzal, [13](#)

komputer_dostaw_statek, [14](#)

komputer_drugi_strzal, [14](#)

komputer_pierwszy_strzal, [15](#)

komputer_pobierz_statki, [15](#)

komputer_standardowy_zestaw, [15](#)

komputer_strzal, [16](#)

komputer_usun_kierunki, [16](#)

komputer_usun_strzaly, [16](#)

komputer_walidacja_kierunku, [16](#)

komputer_walidacja_ruchu, [17](#)

menu, [17](#)

nowe_ustawienie, [17](#)

nowy_X, [18](#)

nowy_Y, [18](#)

pobierz_statki_dla_uzytkownika, [18](#)

Standardowy_zestaw, [19](#)

statek_na_plansze, [19](#)

strzal, [19](#)

ustaw_statek, [20](#)

usun_gracza, [20](#)

usun_komputer, [20](#)

usun_listy, [21](#)

usun_ustawienia, [21](#)

usun_X, [21](#)

walidacja_ilosci, [21](#)

walidacja_koordinatow, [22](#)

walidacja_rodzaju, [22](#)

walidacja_statku, [22](#)

walidacja_zbicia, [23](#)

wyswietl_tablice, [23](#)

wyswietl_tablice_przeciwnika, [23](#)

zapisz_strzal_do_pliku, [25](#)

zapisz_ustawienie_do_pliku, [25](#)

zbij, [25](#)

Gracz_na_Gracza

funkcje.h, [12](#)

Gracz_na_Komputer

funkcje.h, [12](#)

Komputer, [5](#)

komputer_dalszy_strzal

funkcje.h, [13](#)

komputer_dodaj_kierunek

funkcje.h, [13](#)

komputer_dodaj_strzal

funkcje.h, [13](#)

komputer_dostaw_statek

funkcje.h, [14](#)

komputer_drugi_strzal

funkcje.h, [14](#)

komputer_pierwszy_strzal

funkcje.h, [15](#)

komputer_pobierz_statki

funkcje.h, [15](#)

komputer_standardowy_zestaw

funkcje.h, [15](#)

komputer_strzal

funkcje.h, [16](#)

Komputer_strzelanie, [6](#)

komputer_usun_kierunki

funkcje.h, [16](#)

komputer_usun_strzaly

funkcje.h, [16](#)

komputer_walidacja_kierunku

funkcje.h, [16](#)

komputer_walidacja_ruchu

funkcje.h, [17](#)

Lista_kierunkow, [6](#)

Lista_ustawien, [7](#)

menu

funkcje.h, [17](#)

nowe_ustawienie

funkcje.h, [17](#)

nowy_X

funkcje.h, [18](#)

nowy_Y

funkcje.h, [18](#)

pobierz_statki_dla_uzytkownika

funkcje.h, [18](#)

Standardowy_zestaw

funkcje.h, [19](#)

statek_na_plansze

funkcje.h, [19](#)

struktury.h, [26](#)

strzal

funkcje.h, [19](#)

ustaw_statek

funkcje.h, [20](#)

- usun_gracza
 - funkcje.h, [20](#)
- usun_komputer
 - funkcje.h, [20](#)
- usun_listy
 - funkcje.h, [21](#)
- usun_ustawienia
 - funkcje.h, [21](#)
- usun_X
 - funkcje.h, [21](#)
- Uzytkownik, [7](#)

- walidacja_ilosci
 - funkcje.h, [21](#)
- walidacja_koordinatow
 - funkcje.h, [22](#)
- walidacja_rodzaju
 - funkcje.h, [22](#)
- walidacja_statku
 - funkcje.h, [22](#)
- walidacja_zbicia
 - funkcje.h, [23](#)
- Wspolrzedne_x, [8](#)
- Wspolrzedne_y, [9](#)
- wyswietl_tablice
 - funkcje.h, [23](#)
- wyswietl_tablice_przeciwnika
 - funkcje.h, [23](#)

- zapisz_strzal_do_pliku
 - funkcje.h, [25](#)
- zapisz_ustawienie_do_pliku
 - funkcje.h, [25](#)
- zbij
 - funkcje.h, [25](#)