Programopwanie komputerów 2- Gra w statki

Wygenerowano przez Doxygen 1.8.14

Spis treści

1 Indeks struktur danych

1.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

2 Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

3 Dokumentacja struktur danych

3.1 Dokumentacja struktury Gracz

```
#include <Funkcje.h>
```

Pola danych

- int ** **pole**
- int **statki**

3.1.1 Opis szczegółowy

struktura przechowująca dane gracza.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/Ja/source/repos/Projekt-PK2/Project1/Project1/Funkcje.h

3.2 Dokumentacja struktury Konfiguracja

Pola danych

- int **trybGry**
- int **automat**
- int **wczytaj**
- int **rozmiarPola_x**
- int **rozmiarPola_y**
- int **trudnosc**
- int **no_system_new_line_number**
- char * **plikZapisu**
- int **zad_brak**
- int **zad_wczytaj**
- int **zad_zapisz**

- int **zad_cofnij**
- int **zad_flota**
- int **b_koniec**
- int **b_niekoniec**
- int **b_wczytaj**
- int **st_blad**
- int **st_pudlo**
- int **st_cel**
- int **st_zatop**

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/Ja/source/repos/Projekt-PK2/Project1/Project1/Funkcje.h

3.3 Dokumentacja struktury Lista

```
#include <Funkcje.h>
```

Pola danych

- Zadanie **zadanie**
- int **argument**
- int **rodzaj**
- struct Lista * **pPoprzednia**

3.3.1 Opis szczegółowy

Lista dynamicza, będąca historią ruchów, które zostały wykonane.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/Ja/source/repos/Projekt-PK2/Project1/Project1/Funkcje.h

3.4 Dokumentacja struktury Rozgrywka

Pola danych

- Gracz **gracz1**
- Gracz **gracz2**
- mxml_node_t * **xml**
- mxml_node_t * **tura_n**
- mxml_node_t * **czyja_n**
- Historia * **ruchy**
- Wybor **AI**

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/Ja/source/repos/Projekt-PK2/Project1/Project1/Funkcje.h

3.5 Dokumentacja struktury Wybor

```
#include <Funkcje.h>
```

Pola danych

- int **stanPoprzedni**
- int(∗ **stan** [6])(Konfiguracja ∗konfiguracja, int poprzedniePole)
- int **aktualnePole**

3.5.1 Opis szczegółowy

struktura obsługująca sztuczną inteligencję.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/Ja/source/repos/Projekt-PK2/Project1/Project1/Funkcje.h

4 Dokumentacja plików

4.1 Dokumentacja pliku C:/Users/Ja/source/repos/Projekt-PK2/Project1/Project1/Funkcje.h

```
#include "mxml-3.0/mxml.h"
```

Struktury danych

- struct Gracz
- struct Konfiguracja
- struct Wybor
- struct Lista
- struct Rozgrywka

Definicje typów

- typedef enum Dzialania Zadanie
- typedef enum _kolor **Kolor**
- typedef struct Lista Historia

Wyliczenia

- enum Dzialania { **start**, **ustaw**, **strzal** }
- enum **_kolor** { **zielony** =10, **niebieski**, **czerwony**, **bialy** =15 }

Funkcje

- int PobierzParametry (int ileArg, char *arg[ ], Konfiguracja *konfiguracja)
- int Skonfiguruj (Konfiguracja *konfiguracja)
- void UstawParametry (Konfiguracja *konfiguracja)
- int Iniciuj (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka)
- void GameLoop (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka)
- void GameLoopAI (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka)
- void ZamianaGraczy (Rozgrywka *rozgrywka)
- void InicializujAI (Wybor *AI, Konfiguracja *konfiguracja)
- int PobierzKodowanie (Konfiguracja *konfiguracja)
- void DodajdoFloty (Gracz *gracz, int pozycja, int dlugosc)
- void ZakonczGre (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka, int wynik)
- void ZapiszGracza (mxml_node_t *parent, Konfiguracja *konfiguracja, Gracz gracz)
- void WczytajGracza (mxml_node_t *parent, Konfiguracja *konfiguracja, Gracz *gracz)
- int SprawdzZgodnosc (mxml_node_t *xml, Konfiguracja *konfiguracja)
- void WyczyscBufor ()
- void RysujPlansze (Konfiguracja *konfiguracja, Gracz gracz, int dyskrecja)
- void Rozmieszczenie (Konfiguracja *konfiguracja, Gracz *gracz)
- void AutoRozmieszczenie (Konfiguracja *konfiguracja, Gracz *gracz)
- int WypelnijTablice (Konfiguracja *konfiguracja, Gracz *gracz1, Gracz *gracz2)
- int UstawStatek (Konfiguracja *konfiguracja, Gracz *gracz, int dlugosc, int pole, int kierunek, int rodzajStatku)
- int UsunStatek (Konfiguracja *konfiguracja, Historia **historia, Gracz *gracz)
- void UsunTablice (Konfiguracja *konfiguracja, Gracz *gracz1, Gracz *gracz2)
- int Strzal (Konfiguracja *konfiguracja, Gracz *atakowanyGracz, int pole)
- int Bitwa (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka)
- void Oczysc (Konfiguracja *konfiguracja)
- void ZmienKolor (Kolor typ)
- Historia * DodajdoListy (Historia **lista, Zadanie zadanie, int argument, int rodzaj)
- int WprowadzZadanie (int liczbaArgumentow)
- void UsunListe (Historia **lista)
- void WypiszFlote (Gracz *przeciwnik)
- void WypiszRuchy (Historia *ruchy)
- int Losuj (Konfiguracja *konfiguracja, int poprzedniePole)
- int IdzN (Konfiguracja *konfiguracja, int poprzedniePole)
- int IdzS (Konfiguracja *konfiguracja, int poprzedniePole)
- int IdzE (Konfiguracja *konfiguracja, int poprzedniePole)
- int IdzW (Konfiguracja *konfiguracja, int poprzedniePole)
- int IdzSkos (Konfiguracja *konfiguracja, int poprzedniePole)
- int BitwaAI (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka, Wybor *AI)
- void PobierzKoordynaty (Konfiguracja *konfiguracja, int dlugosc, Gracz *gracz, Historia **historia, int rodzaj)
- int UtworzZapis (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka)
- int Zapisz (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka)
- int Wczytaj (Konfiguracja *konfiguracja, Rozgrywka *rozgrywka)

4.1.1 Dokumentacja definicji typów

4.1.1.1 Historia

```
typedef struct Lista Historia
```

Lista dynamicza, będąca historią ruchów, które zostały wykonane.

4.1.1.2 Zadanie

```
typedef enum Dzialania Zadanie
```

typ wyliczeniowy określający jaka czynność została wykonana.

4.1.2 Dokumentacja typów wyliczanych

4.1.2.1 Dzialania

```
enum Dzialania
```

typ wyliczeniowy określający jaka czynność została wykonana.

4.1.3 Dokumentacja funkcji

4.1.3.1 AutoRozmieszczenie()

```
void AutoRozmieszczenie (
            Konfiguracja * konfiguracja,
            Gracz * gracz )
```

Funkcja rozpoczyna przygotowanie do gry losowo umieszczając wszystkie statki.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
gracz	Gracz, którego pole gry będzie ustalane poprzez losowe rozmieszczenie statków

4.1.3.2 Bitwa()

```
int Bitwa (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka )
```

Komunikacja z użytkownikiem po zakończeniu przygotowań. Pojedyncze wywołanie jest pojedynczą turą jednego gracza

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki

Zwraca

0 (B_KONIEC) Informacja o tym, że jeden z graczy stracił wszystkie statki. Wartość ta kończy pętle gry
1 (B_NIEKONIEC) Informacja o tym, że każdy gracz ma jeszcze statki
2 (B_WCZYTAJ) Wartość zwracana w przypadku udanego wcytania gry

4.1.3.3 BitwaAI()

```
int BitwaAI (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka,
            Wybor * AI )
```

Funkcja obsługująca turę dla gry z komputerem.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki
AI	Struktura posiadająca zainicjowaną tablice wskaźników na funckje celujące oraz indeks następnej funkcji do wykonania

Zwraca

0 (B_KONIEC) Informacja o tym, że jeden z graczy stracił wszystkie statki. Wartość ta kończy pętle gry
1 (B_NIEKONIEC) Informacja o tym, że każdy gracz ma jeszcze statki.

4.1.3.4 DodajdoFloty()

```
void DodajdoFloty (
            Gracz * gracz,
            int pozycja,
            int dlugosc )
```

Funkcja dodaje odpowiednią ilość punktów życia do statków gracza, na odpowiedniej pozycji.

Parametry

gracz	Gracz do którego dodane zostaną punkty życia
pozycja	Pozycja w systemie usemkowym na którą zostanie dodana liczba punktów życia
dlugosc	Dodawana liczba punktów życia, zależna od statku

4.1.3.5 DodajdoListy()

```
Historia* DodajdoListy (
            Historia ** lista,
```

```
            Zadanie zadanie,
            int argument,
            int rodzaj )
```

Funkcja do dająca element do listy na jej początek.

Parametry

lista	Wskaźnik na listę do której należy dodać element
zadanie	Wartość określa jaki typ ruchy został wykonany.
argument	Informacja do zapsiania na liście. Może być dowolna, ale została przygotowana na koordynaty
rodzaj	informacja do zapsiania na liście. Może być dowolna, ale została przygotowana na kierunek ustawienia statku lub wynik strzału

Zwraca

Wskaznik na liste
NULL, gdy nie udało się zaalokować pamięci

4.1.3.6 GameLoop()

```
void GameLoop (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka )
```

Funckja stanowi główną pętle gry. Przekazuje graczom odpowiednie informacje porządkujące grę. Tylko dla gry w trybie hotseat

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki

4.1.3.7 GameLoopAI()

```
void GameLoopAI (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka )
```

Funckja stanowi główną pętle gry. Przekazuje graczowi odpowiednie informacje porządkujące grę. Tylko dla gry z komputerem.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki

4.1.3.8 IdzE()

```
int IdzE (
            Konfiguracja * konfiguracja,
            int poprzedniePole )
```

Funckja przenosząca celowanie o jedno pole w prawo.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
poprzedniePole	Poprzednie pole w które celowano

Zwraca

Nowe pole, w które należy wycelować

4.1.3.9 IdzN()

```
int IdzN (
            Konfiguracja * konfiguracja,
            int poprzedniePole )
```

Funkcja przenosząca celowanie o jedno pole do góry.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
poprzedniePole	Poprzednie pole w które celowano

Zwraca

Nowe pole, w które należy wycelować

4.1.3.10 IdzS()

```
int IdzS (
            Konfiguracja * konfiguracja,
            int poprzedniePole )
```

Funkcja przenosząca celowanie o jedno pole w dół.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
poprzedniePole	Poprzednie pole w które celowano

Zwraca

Nowe pole, w które należy wycelować

4.1.3.11 IdzSkos()

```
int IdzSkos (
            Konfiguracja * konfiguracja,
            int poprzedniePole )
```

Funkcja przenosząca celowanie o jedno pole w prawo i jedno w dół.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
poprzedniePole	Poprzednie pole w które celowano

Zwraca

Nowe pole, w które należy wycelować

4.1.3.12 IdzW()

```
int IdzW (
            Konfiguracja * konfiguracja,
            int poprzedniePole )
```

Funckja przenosząca celowanie o jedno pole w lewo.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
poprzedniePole	Poprzednie pole w które celowano

Zwraca

Nowe pole, w które należy wycelować

4.1.3.13 InicializujAI()

```
void InicializujAI (
            Wybor * AI,
            Konfiguracja * konfiguracja )
```

Funkcja ustawia sztuczną inteligencję zgdonie z konfiguracją.

Parametry

AI,struktura	przechowująca dane sztucznej inteligencji
konfiguracja	struktura przechowująca ustawienia gry

4.1.3.14 Iniciuj()

```
int Iniciuj (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka )
```

Funkcja przeprowadza całą początkową fazę gry.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki

Zwraca

0, gdy nie udało się poprawnie rozpocząć gry
1, gdy cały proces przebiegł pomyślnie

4.1.3.15 Losuj()

```
int Losuj (
            Konfiguracja * konfiguracja,
            int poprzedniePole )
```

Funckcja losująca koordynaty dla algorytmu celowania komputera.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
poprzedniePole	Poprzednie pole w które celowano

Zwraca

Nowe pole, w które należy wycelować

4.1.3.16 Oczysc()

```
void Oczysc (
            Konfiguracja * konfiguracja )
```

Funkcja czyszcząca konsolę. Zabezpieczenie przed podglądaniem pól przeciwnika. W przyadku nieobsługiwanego systemu operacyjnego funkcja wypisze dostateczną ilość nowych linii, aby gracze nie widzieli w swojej turze statków przeciwnika.

Parametry

konfiguracja	struktura przechowująca ustawienia gry

4.1.3.17 PobierzKodowanie()

```
int PobierzKodowanie (
            Konfiguracja * konfiguracja )
```

Funkcja pobiera kodowanie z pliku kod.txt.

Parametry

konfiguracja	struktura przechowująca ustawienia gry

Zwraca

0, gdy nie znaleziono pliku z kodowaniem
1, gdy odczytano kodowanie z pliku. Nie gwarantuje to poprawności kodowania w przypadku zmiany kodu źródłowego programu.

4.1.3.18 PobierzKoordynaty()

```
void PobierzKoordynaty (
            Konfiguracja * konfiguracja,
            int dlugosc,
            Gracz * gracz,
            Historia ** historia,
            int rodzaj )
```

Funkcja pobiera od użytkownika informacje gdzie powinien zostać umieszczony pojedyczy statek.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
dlugosc	Długość statku, który należy umieścić
gracz	Gracz, który aktualnie ustawia statki
historia	Lista ruchów gracza. Pozwala na cofanie statków
rodzaj	identyfikator statku. Ta wartość zostanie wpisana na pole w momencie ustawiania statku

4.1.3.19 PobierzParametry()

```
int PobierzParametry (
            int ileArg,
            char * arg[],
            Konfiguracja * konfiguracja )
```

Funkcja pobiera argumenty wiersza poleceń i ustawia według nich konfiguracje.

Parametry

ileArg	ilość argrumentów wiersza poleceń
arg	argumenty wiersza poleceń
konfiguracja	struktura przechowująca ustawienia gry

Zwraca

0, gdy pobranie parametrów było nieudane
1, gdy pobranie prametrów powiodło się.

4.1.3.20 Rozmieszczenie()

```
void Rozmieszczenie (
            Konfiguracja * konfiguracja,
            Gracz * gracz )
```

Funkcja rozpoczyna przygotowanie do gry w przypadku, gdy nie rozpoczęto od wczytania zapisu. Funkcja komunikuje się z użytkwonikiem przez konsole.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
gracz	Gracz, którego pole gry będzie ustalane na podstawie informacji otrzymywanych od użytkownika

4.1.3.21 RysujPlansze()

```
void RysujPlansze (
            Konfiguracja * konfiguracja,
            Gracz gracz,
            int dyskrecja )
```

Funkcja wyświetlająca graczowi planszę jego lub przeciwnika.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
gracz	Gracz, którego planszę należy wyświetlić
dyskrecja	Parametr określający czy należy wyświetlić statki. Dla zmiennej równej 0, statki są wyświetlane, w przeciwnym wypadku są pomijane.

4.1.3.22 Skonfiguruj()

```
int Skonfiguruj (
            Konfiguracja * konfiguracja )
```

Funkcja pobiera konfiguracje z pliku config.ini

Parametry

konfiguracja	struktura przechowująca ustawienia gry 0, gdy znaleziono plik i odczytano konfiguracje

Zwraca

1, gdy nie znaleziono pliku lub zakres pola gry był niedozwolony

4.1.3.23 SprawdzZgodnosc()

```
int SprawdzZgodnosc (
            mxml_node_t * xml,
            Konfiguracja * konfiguracja )
```

Funkcja sprawdza zgodność pliku zapisu z obecną konfiguracją gry.

Parametry

xml	Drzewo xml odczytane z pliku zapisu
konfiguracja	struktura przechowująca ustawienia gry

Zwraca

0, gdy dane nie są zgodne
1, gdy dane są zgodne

4.1.3.24 Strzal()

```
int Strzal (
            Konfiguracja * konfiguracja,
            Gracz * atakowanyGracz,
            int pole )
```

Funckja sprawdzająca czy strzał jest możliwy do wykonanania oraz zwracająca informacja o tym jaki był wynik strzału.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
atakowanyGracz	Gracz, w którego pole zostanie wykonany strzał
pole	Pole, w które należy strzelić. Koordynaty są podane jako liczba dwucyfrowa. Pierwsza cyfra określa pole na osi X, a druga na osi Y

Zwraca

0 (ST_BLAD) Informacja o tym, że strzał nie może sotać wykonany w podane pole
1 (ST_PUDLO) Informacja o tym, że strzał został wykonany poprawnie w pole na którym nie było statku
2 (ST_CEL) Informacja o tym, że strzał został wykonany poprawnie w pole na którym znajdował się statek
3 (ST_ZATOP) Informacja o tym, że strzał został wykonany poprawnie oraz statek, który trafiono został zatopiony

4.1.3.25 UstawParametry()

```
void UstawParametry (
            Konfiguracja * konfiguracja )
```

Funckcja ustawia brakujące parametry, przekazywane zwykle przez wiersz poleceń, komunikując się z użytkownikiem.

Parametry

konfiguracja	struktura przechowująca ustawienia gry

4.1.3.26 UstawStatek()

```
int UstawStatek (
            Konfiguracja * konfiguracja,
            Gracz * gracz,
            int dlugosc,
            int pole,
            int kierunek,
            int rodzajStatku )
```

Funkcja ustawiająca statki na planszy. Sprawdza czy statek może zostać ustawiony, a następnie go ustawia zmieniając wartości znajdujące się na tablicy dwu wymiarowej na identyfikator statku.

Parametry

gracz	Gracz, który ustawia statek
dlugosc	Długość ustawianego statku
pole	Pole, w którym zaczyna się statek. Jest to liczba dwucyfrowa, której pierwsza cyfra to pole na osi X, a druga to pole na osi Y
kierunek	Informacja, w którym kierunku powinien być ustawiony statek. 1- wertykalnie. 0- horyzontalnie
rodzajStatku	Identyfikator statku. Ta liczba zostanie wpisana na pole podczas ustawiania. Powinna być mniejsza od -3 oraz unikalna (wyjątkiem są statki zajmujące jedno pole)

Zwraca

0, gdy na podanych koordynatach nie można ustawić statku w danym kierunku
1, gdy statek został ustawiony poprawnie

4.1.3.27 UsunListe()

```
void UsunListe (
            Historia ** lista )
```

Usuwanie dynamicznie zaalokowanej listy.

Parametry

lista	Wskaźnik na listę, której pamięc należy usunąć

4.1.3.28 UsunStatek()

```
int UsunStatek (
            Konfiguracja * konfiguracja,
            Historia ** historia,
            Gracz * gracz )
```

Funkcja usuwająca statek z planszy. Używana tylko w trakcie przygotowania do gry na żądanie gracza. Następnie prosi o podanie nowych koordynatów tak jakby poprzedni ruch nie został wykonany .Do poprawnego działania lista historia powinna być poprawnie zainicjowana, a jej najstarszy element powinien być utworzony z typem wyliczeniowym start.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
historia	Wskaźnik na liste ruchów
gracz	Gracz, który aktualnie wykonuje ruch

Zwraca

1, gdy usunięto i ponownie ustawiono statek
0, gdy nie ma już ruchów do cofnięcia na liście
-1, gdy lista została źle utworzona (jej pierwszy element nie ma określonego odpowiedniego typu wyliczeniowego)

4.1.3.29 UsunTablice()

```
void UsunTablice (
            Konfiguracja * konfiguracja,
            Gracz * gracz1,
            Gracz * gracz2 )
```

Procedura usuwania dynamicznie zaalokownaych tablic.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
gracz1	Gracz, którego tablicę należy wypełnić
gracz2	Gracz, którego tablicę należy wypełnić. Powinien być rózny od gracza pierwszego

4.1.3.30 UtworzZapis()

```
int UtworzZapis (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka )
```

Funkcja tworzy zapis na podstawie ustawień i zakończonego, przez obu graczy, przygotowania do gry. Tworzy drzewo xml, które pozwala na łatwe zapisywanie w późniejszej częsci gry.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki

Zwraca

1, gdy poprawnie wykonano zapis do pliku
0, gdy nie można było zapisać do pliku. W tym przypadku drzewo zostało utworzone tylko w pamięci aplikacji

4.1.3.31 Wczytaj()

```
int Wczytaj (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka )
```

Funkcja wczytuje dane z pliku z zapisem nadpisując obecne informacje. Nadpisane zostaną dane graczy, drzewo xml oraz lista wykonanych ruchów.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki

Zwraca

0, gdy nie można było otworzyć pliku lub załadowanie drzewa z tego pliku było niemożliwe
1, gdy procedura odczytywania przebiegła pomyślnie

4.1.3.32 WczytajGracza()

```
void WczytajGracza (
            mxml_node_t * parent,
            Konfiguracja * konfiguracja,
            Gracz * gracz )
```

Funkcja odczytuje dane gracza z drzewa xml.

Parametry

parent	Część drzewa pod które zapisane zostały dane gracza
konfiguracja	struktura przechowująca ustawienia gry
gracz	Struktura z danymi gracza czytanymi z drzewa

4.1.3.33 WprowadzZadanie()

```
int WprowadzZadanie (
            int liczbaArgumentow )
```

Funkcja do komunikacji z użytkownikiem, pozwalająca na używanie komend w dowolnej chwili. Po wprowadzeniu komendy wypisuje informacje, które działanie próbowano wykonać.

Parametry

liczbaArgumentow	Parametr określający jak dużo danych chcemy otrzymać. Dla 0 wymaga naciśnięcia klawisza "Enter", dla 1 wymaga jednej wartości, dla 2 dwóch wartości. Inne wartości nie są obsługiwane

Zwraca

-2 (ZAD_FLOTA) Zwracana, gdy użyto komendy "flota"
-3 (ZAD_COFNIJ) Zwracana, gdy użyto komendy "cofnij"
-4 (ZAD_ZAPISZ) Zwracana, gdy użyto komendy "zapisz"
-5 (ZAD_WCZYTAJ) Zwracana, gdy użyto komendy "wczytaj"
0 (ZAD_BRAK) Zwracana, gdy liczbaArgumentow wynosi 0 i nie podano komedy
Liczba całkowita, której cyfra jedności ma wartość drugiego argumentu a pozostała część liczby wartość pierwszego. Zwracana tylko gdy nie użyto żadnej komendy

4.1.3.34 WyczyscBufor()

```
void WyczyscBufor ( )
```

Funkcja czyszcząca pozostałości z bufora.

4.1.3.35 WypelnijTablice()

```
int WypelnijTablice (
            Konfiguracja * konfiguracja,
            Gracz * gracz1,
            Gracz * gracz2 )
```

Procedura alokowania i wypełnienia tablic będących planszami dla graczy. Tablice alokowane są dynamicznie korzystając z rozmiaru określonego w makro ROZMIAR_POLA. Gdy alokacja pamięci będzie niemożliwa, funkcja zwolni zaalokowaną przez nią pamięć.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
gracz1	Gracz, którego tablicę należy wypełnić
gracz2	Gracz, którego tablicę należy wypełnić. Powinien być rózny od gracza pierwszego

Zwraca

1, gdy alokacja zakończyła się sukcesem
0, gdy wystąpiły błędy alokacji. Pamięć zaalokowana do błędu zostanie w funkcji zwolniona

4.1.3.36 WypiszFlote()

```
void WypiszFlote (
            Gracz * przeciwnik )
```

Funkcja wypisująca statki przeciwnika, które nie zostały zatopione.

Parametry

przeciwnik	Gracz, którego niezatopione statki należy wypisać

4.1.3.37 WypiszRuchy()

```
void WypiszRuchy (
            Historia * ruchy )
```

Funkcja wypisująca ruchy poprzedniego gracza. Nie modyfikuje listy. Wypisuje do pierwszego sppotkanego elementu z typem wyliczeniowym określonym jako "start".

Parametry

ruchy	Lista wykonanych ruchów.

4.1.3.38 ZakonczGre()

```
void ZakonczGre (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka,
            int wynik )
```

Funkcja kończy grę ogłaszając gracza i usuwając dynamicznie zaalokowaną pamięć.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki
wynik	Wynik na podstawie którego zaostaje wytypowany wygrywający gracz

4.1.3.39 ZamianaGraczy()

```
void ZamianaGraczy (
            Rozgrywka * rozgrywka )
```

Fukcja zamienia pozycjami graczy by gracz drugi mógł wykonać swoją turę.

Parametry

rozgrywka	struktura przechowująca dane obecnej rozgrywki

4.1.3.40 Zapisz()

```
int Zapisz (
            Konfiguracja * konfiguracja,
            Rozgrywka * rozgrywka )
```

Funkcja wykonująca zapis w tracie rozgrywki. Początkowo funkcja aktualizuje drzewo danymi, których ciągły zapis do drzewa byłby nieoptymalny. Dane aktualizowane znajdują sie w strukturach Gracz.

Parametry

konfiguracja	struktura przechowująca ustawienia gry
rozgrywka	struktura przechowująca dane obecnej rozgrywki

Zwraca

1, gdy poprawnie wykonano zapis do pliku
0, gdy nie można było zapisać do pliku. W tym przypadku drzewo zostało utworzone tylko w pamięci aplikacji

4.1.3.41 ZapiszGracza()

```
void ZapiszGracza (
            mxml_node_t * parent,
            Konfiguracja * konfiguracja,
            Gracz gracz )
```

Funkcja zapisuje dane gracza do drzewa xml.

Parametry

parent	Część drzewa pod które zapisane zostaną dane gracza
konfiguracja	struktura przechowująca ustawienia gry
gracz	Struktura z danymi gracza zapisywanymi do drzewa

4.1.3.42 ZmienKolor()

```
void ZmienKolor (
            Kolor typ )
```

Funkcja zmieniająca kolor tekstu. Ma zastosowanie estetyczne. Dla nieobsługiwanych systemów nie wykona żadnej instrukcji.

Parametry

typ	Kod koloru na który zostanie zamieniony kolor tekstu w konsoli

Skorowidz