Katowice 22.05.2019r.

Laboratorium Programowania Komputerów

Temat:

Gra w statki

Autor: Michał Góral
Informatyka, sem., grupa II, sekcja II
Prowadzący: Mgr inż. Wojciech Dudzik

1. Temat

Napisać grę w statki, z możliwością zapisu stanu plansz i statystyk do pliku

2. Analiza, projektowanie

2.1. Struktury danych, ograniczenia specyfikacji

Struktury danych: dane planszy do gry przechowuję w strukturze statycznej, o ograniczonym z góry rozmiarze. W tym przypadku zastosowałem ją, ponieważ rozmiar planszy nie może się zmieniać i umożliwia to szybki dostęp do dowolnego miejsca na planszy. Do zapisu statystyk i informacji o statkach użyłem struktury dynamicznej w postaci listy jednokierunkowej, ponieważ uznałem to za najbardziej stosowne

Ograniczenia specyfikacji: W programie nie możemy korzystać z myszy ze względu na to, że program uruchamiany jest w konsoli. Możemy jedynie rozmieścić 5 statków o stałej długości i niezamienialnej nazwie, ponieważ takie zasady gry ustaliłem. Nie możemy również grać w więcej niż dwie osoby.

W Programie korzystam z biblioteki `windows.h`, ponieważ postanowiłem, że niektóre przejścia do kolejnych etapów rozgrywki nie powinny być za każdym razem inicjowane kliknięciem entera. Wprowadziłem do programu pewien system zabezpieczeń który działa, gdy np. zamiast ‘9’ wprowadzimy ‘a’, jednak, gdy wprowadzimy wiele liter i cyfr, gdy program oczekuje od nas jednego znaku może on działać nie poprawnie. Niestety nie byłem w stanie znaleźć rozwiązania tego problemu.

3. Specyfikacja zewnętrzna

3.1. Obsługa programu

Wszystkie zasady gry pojawiają się na ekranie podczas uruchomienia rozgrywki. Zalecam ustawienie domyślnej wysokości okna konsoli na wartość 36 a szerokości na 120.

Klawisze – w menu musimy wybrać kilka opcji, głównie wprowadzając jedną cyfrę lub literę odpowiadającą danemu wyborowi. Przy wprowadzaniu współrzędnych do strzału wprowadzamy najpierw indeks wiersza, a następnie po spacji indeks kolumny. Gdy wpiszemy podczas tej akcji `’11 11’` kończymy rozgrywkę.

Dane wejściowe – wymagany format pliku wejściowego to `'.txt'`.

Parametry main'a – Program możemy uruchomić z parametrami lub nie. Pierwszy i drugi parametr to nazwa pliku do którego będziemy chcieli zapisać dane planszy kolejno pierwszego i drugiego gracza. Trzeci parametr to nazwa pliku do którego będziemy chcieli zapisać statystyki gry. Czwarty i piąty parametr to nazwa pliku z którego będziemy chcieli sczytać dane planszy kolejno pierwszego i drugiego gracza. Jeżeli program uruchomimy bez parametrów to nazwy plików będą wyglądały następująco: `player1_board.txt, player2_board.txt, stats.txt,` a plansza zainicjalizuje się bazowymi znakami.

Zabezpieczenia - Wprowadziłem do programu pewien system zabezpieczeń który działa, gdy np. zamiast '9' wprowadzimy `'a'` lub jakąś inną liczbę, jednak, gdy wprowadzimy wiele cyfr, gdy program oczekuje od nas jednego znaku może on działać nie poprawnie. Niestety nie byłem w stanie znaleźć rozwiązania tego problemu.

3.2. Format danych wejściowych

Plik wejściowy ten musi zawierać 10 wierszy i 10 kolumn wypełnionych odpowiednimi znakami: bazowy znak – '~', znak trafienia – 'S', znak pudła – 'm', znak statku Carrier – 'c', znak statku Battleship – 'b', znak statku Cruiser – 'r', znak statku Submarine – 's', znak statku Destroyer – 'd'. Gdy plik będzie miał inną zawartość niż podana wyżej to wprowadzi nieprawidłowe dane do plansz i rozgrywka będzie niemożliwa.

3.3. Komunikaty

Wszystkie komunikaty w programie są jednoznaczne i wydaje mi się, że niemożliwym jest ich niezrozumienie, dlatego pominę ten podrozdział.

4. Specyfikacja wewnętrzna

Na samym początku program inicjalizuje plansze dla obu graczy sam lub pobiera dane z plików. Następnie inicjalizuje listy które będą w dalszej części używane. Po tym działaniu przechodzi do funkcji `menu()` w której pobiera od użytkownika wszystkie niezbędne dane. Potem przechodzi do funkcji `play()` która koordynuje całą rozgrywką. W poziomie trudności easy komputer strzela w losowe miejsca, nawet te które teoretycznie nie mogą mieć w sobie statku. W poziomie trudności medium strzela losowo w miejsca w których może się znajdować statek i dodatkowo jeśli poprzedni strzał był trafieniem to następny będzie obok tego trafienia. W poziomie trudności hard zasady są tak jak w medium oraz komputer za zapisany pierwszy strzał trafiony i jeśli np. nastąpi sekwencja: traf, traf, pudło to w następnym strzale sprawdzi on w którym kierunku statek jest ustawiony i strzeli w następną

komórkę po ostatnim trafieniu. Dodatkowo jeśli graczowi zostały dwa trafienia do zakończenia rozgrywki to strzały komputera będą wyłącznie trafione. Lista ze statystykami jest na bieżąco aktualizowana, a lista ze statkami jest używana do rozmieszczenia ich na planszy. Zastosowałem odpowiednie funkcje do tego, aby gracz przeciwny widział jedynie swoje trafienia, bądź pudła. Miejsca statków gracza przeciwnego nie są widoczne. Bardzo przydatne okazały się w tym rozwiązaniu stałe preprocesora.

4.1. Funkcje i struktury

Szczegółowy opis funkcji i struktur zawarty jest w załączniku.

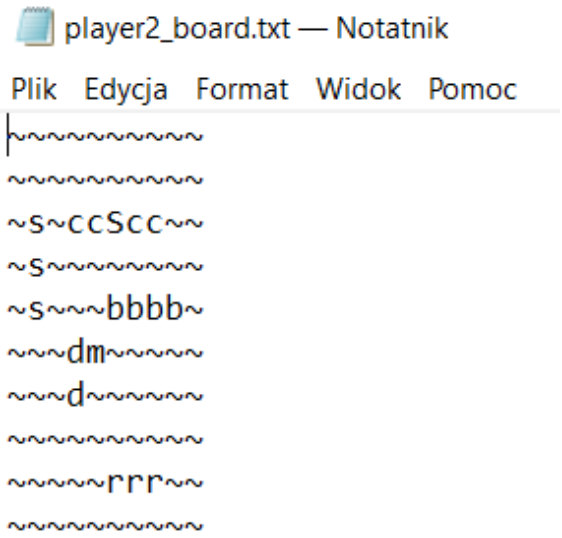
5. Testowanie

5.1. Dane testowe – uzasadnienie

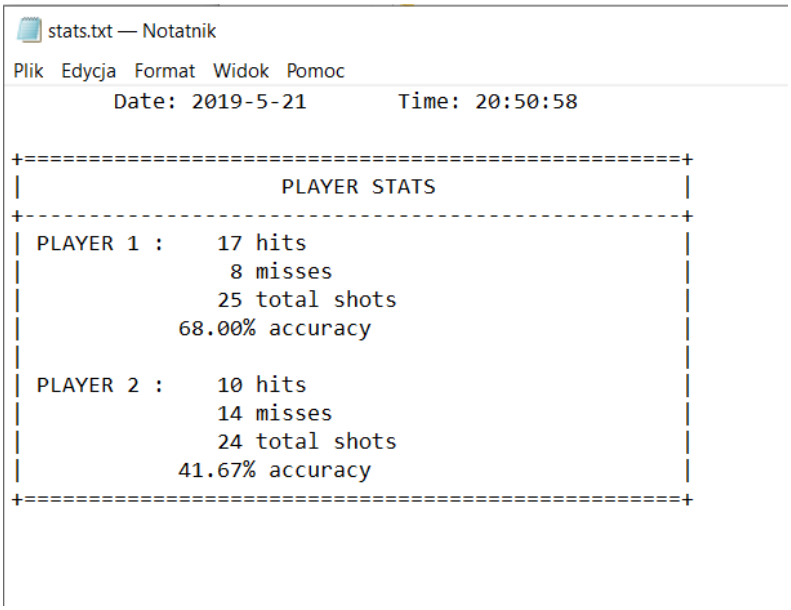
Program testowałem wiele razy z danymi właściwymi, jak i niewłaściwymi. Podczas wprowadzania współrzędnych musimy wprowadzać wartości z zakresu <0;9> i gdy wprowadzimy liczbę spoza niego, program nie zawiesi się, tylko poprosi o ponowne ich wprowadzenie. Czytając z pliku po podanej w jednym z parametrów, jeśli zawiera on właściwy format wszystko działa jak należy. Niewłaściwego formatu nie sprawdzałem, ponieważ jest z góry założone, że format ma być właściwy.

5.2. Wyniki

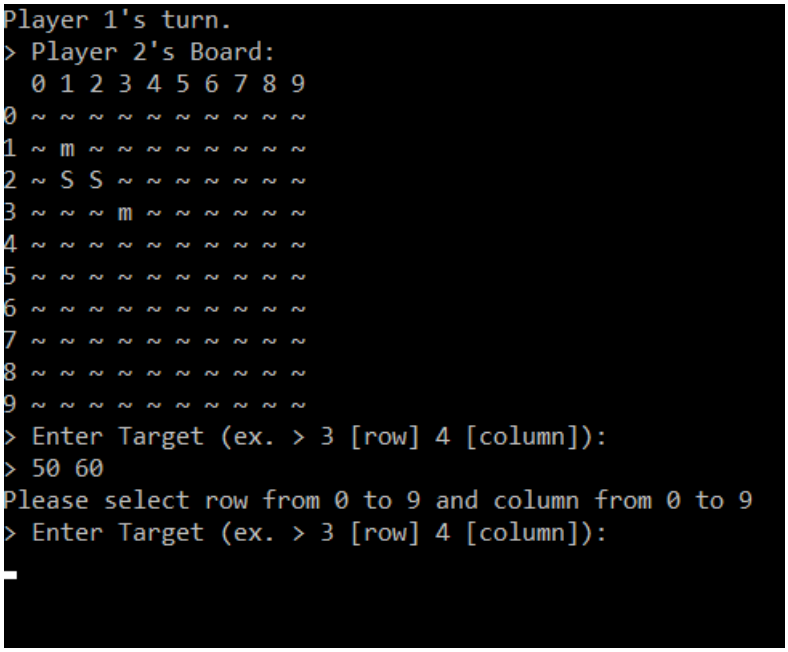
- Przykładowy plik wynikowy po wyjściu w trakcie rozgrywki:



- Przykładowy plik wynikowy statystyk po zakończeniu rozgrywki:



- Przykładowa plansza w trakcie rozgrywki:



6. Wnioski

Napisanie tego programu wiele mnie nauczyło. W moim odczuciu w porównaniu do języka C++, język C nie ma tyle gotowych rozwiązań i wiele trzeba zrobić samemu. Obsługa wskaźników jest trudniejsza. Przy używaniu niektórych funkcji np. `scanf()` lub `getchar()` Kompilator dawał mi informacje zwrotne, że funkcje te są już przestarzałe. Uważam, że w tym momencie opanowałem ten język w stopniu który mnie na ten moment satysfakcjonuje. Najwięcej czasu przy pisaniu poświęciłem na usprawnienie trudnego poziomu trudności. To on sprawił mi najwięcej problemów. Również kilka problemów miałem przy działaniu na wskaźnikach, jednak poradziłem sobie z nimi względnie szybko. Na następnych stronach zawarłem załącznik w którym znajdują się szczegółowe informacje o funkcjach i strukturach programu.

Battleship

Michał Góral
V1.0
Wt, 22 maj 2019

Spis treści

Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

Indeks plików

Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Dokumentacja struktur danych

Dokumentacja struktury _ship

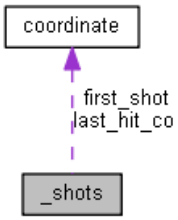
`#include <ships.h>`
Diagram współpracy dla _ship:

Pola danych

- char **symbol**
- int **length**
- char * **name**
- struct **_ship** * **pNext**

Dokumentacja struktury _shots

`#include <game.h>`
Diagram współpracy dla _shots:



Pola danych

- **coordinate last_hit_co**
- **coordinate first_shot**
- **booleann has_ship_sunk**
- **booleann if_last_shot**

Dokumentacja struktury _stats

`#include <game.h>`

Diagram współpracy dla _stats:

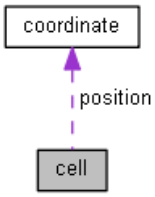
Pola danych

- int **player**
- int **num_of_hits**
- int **num_of_misses**
- int **total_shots**
- double **accuracy**
- struct **_stats** * **pNext**

Dokumentacja struktury cell

`#include <board.h>`

Diagram współpracy dla cell:



Pola danych

- char **symbol**
- **coordinate position**

Dokumentacja struktury coordinate

`#include <board.h>`

Pola danych

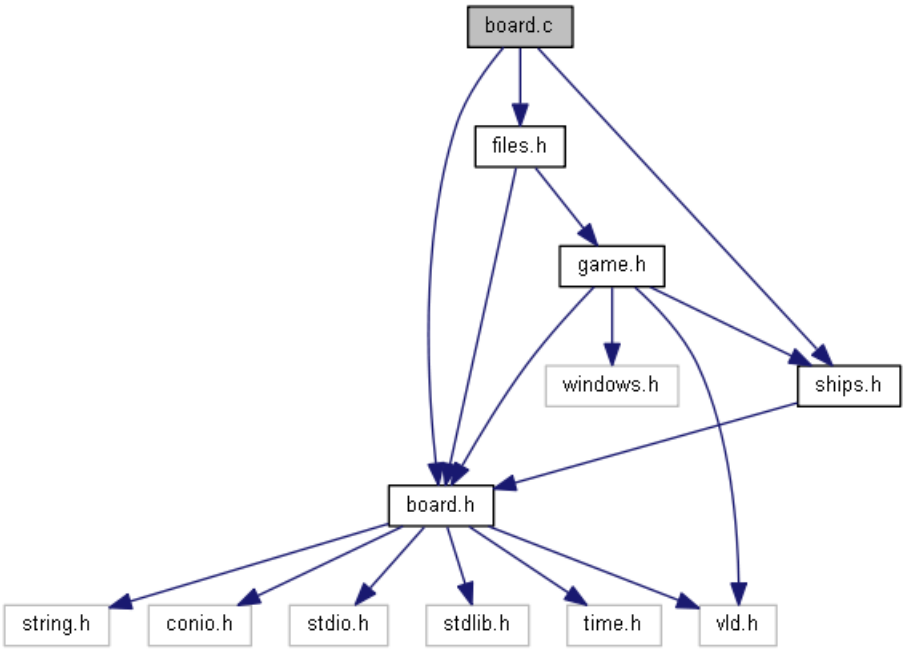
- int **row**
- int **column**
- int **shot**

Dokumentacja plików

Dokumentacja pliku board.c

```
#include "board.h"
#include "ships.h"
#include "files.h"
```

Wykres zależności załączania dla board.c:



Funkcje

- void **initialize_game_board** (**cell** player1_board[][**COLS**], **cell** player2_board[][**COLS**], int argc, char **argv)
- void **print_board** (**cell** game_board[][**COLS**], **booleann** show)
- void **update_game_board** (**cell** game_board[][**COLS**], **coordinate** target)

Dokumentacja funkcji

void initialize_game_board (cell *player1_board*[][COLS], cell *player2_board*[][COLS], int *argc*, char ** *argv*)

Funkcja inicjalizująca planszę znakami '~' lub czytająca z pliku jeśli wywołamy maina z parametrami

Parametry:

player1_board	dwuwymiarowa tablica przechowująca informacje o planszy pierwszego gracza
player2_board	dwuwymiarowa tablica przechowująca informacje o planszy drugiego gracza lub komputera
argc	jeden z parametrów wywołania programu
argv	jeden z parametrów wywołania programu

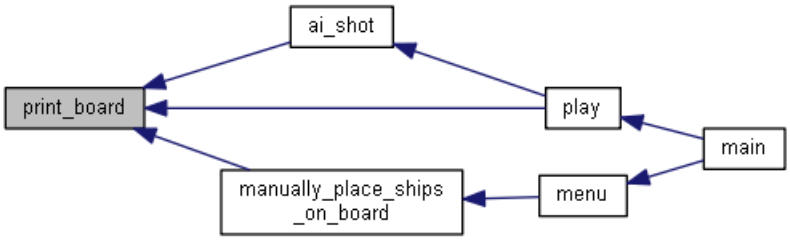
void print_board (cell *game_board*[][COLS], booleann *show*)

Funkcja wypisująca aktualną planszę

Parametry:

game_board	dwuwymiarowa tablica przechowująca informacje o planszy drugiego gracza
show	parametr pozwalający wyświetlić odpowiednią dla gracza wersje planszy

Oto graf wywoływań tej funkcji:



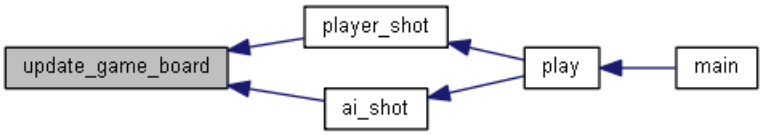
void update_game_board (cell *game_board*[][COLS], coordinate *target*)

Funkcja aktualizująca pola na planszy po wykonaniu strzału

Parametry:

game_board	dwuwymiarowa tablica przechowująca informacje o planszy drugiego gracza
target	współrzędne wykonanego strzału

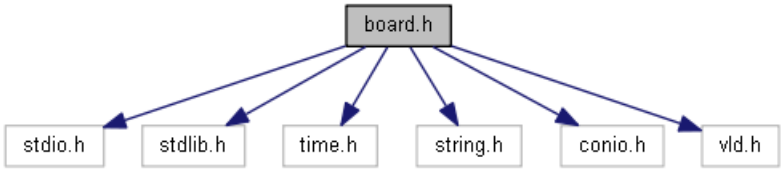
Oto graf wywoływań tej funkcji:



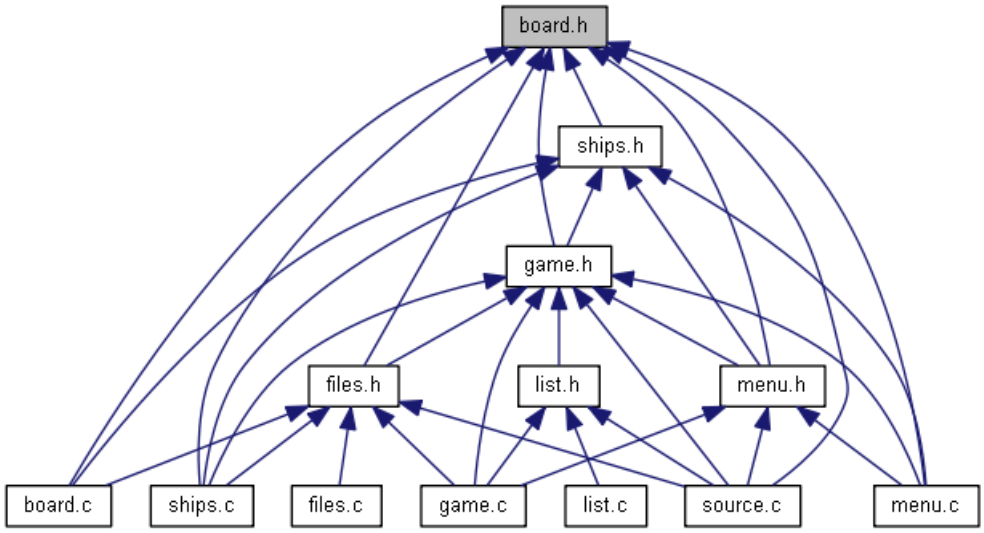
Dokumentacja pliku board.h

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <conio.h>
#include <vld.h>
```

Wykres zależności załączania dla board.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Struktury danych

- struct **coordinate**
- struct **cell**

Definicje

- #define **ROWS** 10
- #define **COLS** 10
- #define **WATER** '~'
- #define **HIT** 'S'
- #define **MISS** 'm'

Wyliczenia

- enum **booleann** { **FALSE**, **TRUE** }

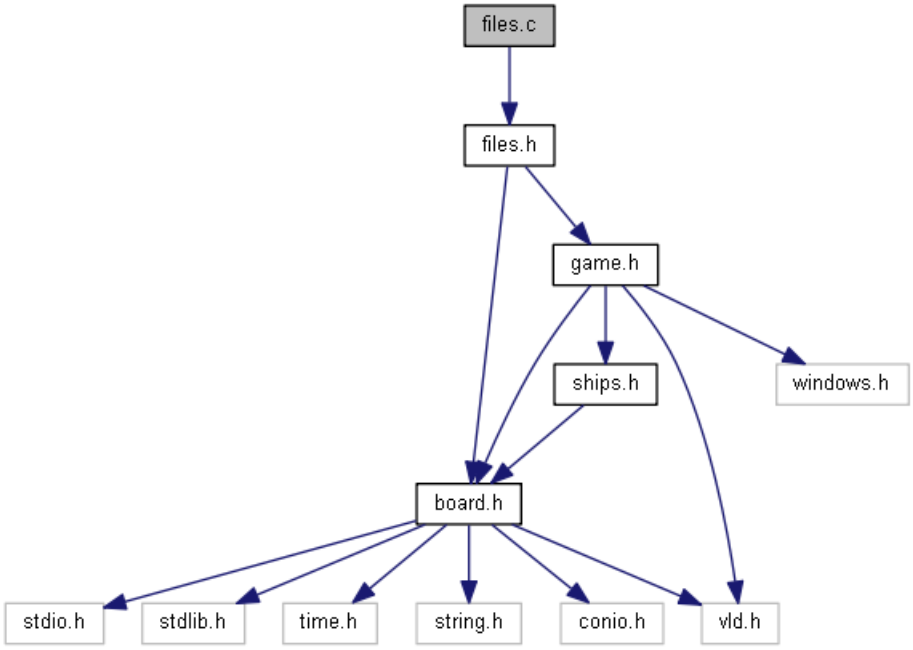
Funkcje

- void **print_board** (**cell** game_board[][**COLS**], **booleann** show)
- void **initialize_game_board** (**cell** player1_board[][**COLS**], **cell** player2_board[][**COLS**], int argc, char **argv)
- void **update_game_board** (**cell** game_board[][**COLS**], **coordinate** target)

Dokumentacja pliku files.c

```
#include "files.h"
```

Wykres zależności załączania dla files.c:



Funkcje

- void **write_stats_in_file** (char *name, **stats** *pHead)
- void **read_board_from_file** (char *name, **cell** game_board[][**COLS**])
- void **save_board_to_file** (char *name, **cell** game_board[][**COLS**])

Dokumentacja funkcji

void read_board_from_file (char * *name*, cell *game_board*[][COLS])

Funkcja czytająca planszę z pliku i zapisująca ją w tablicy dwuwymiarowej

Parametry:

name	nazwa pliku w którego zczytuje informacje
game_board	dwuwymiarowa tablica przechowująca informacje o planszy

void save_board_to_file (char * *name*, cell *game_board*[][COLS])

Funkcja zapisująca aktualny stan planszy do pliku o podanej nazwie

Parametry:

name	nazwa pliku w którego zczytuje informacje
game_board	dwuwymiarowa tablica przechowująca informacje o planszy

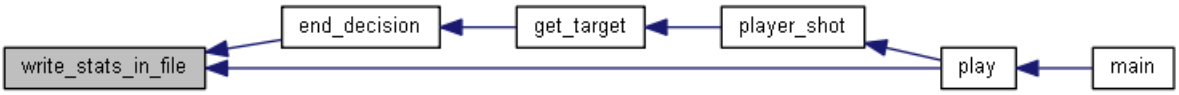
void write_stats_in_file (char * *name*, stats * *pHead*)

Funkcja zapisująca statystyki w pliku o podanej nazwie

Parametry:

name	nazwa pliku w którym zapisuje informacje
pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach

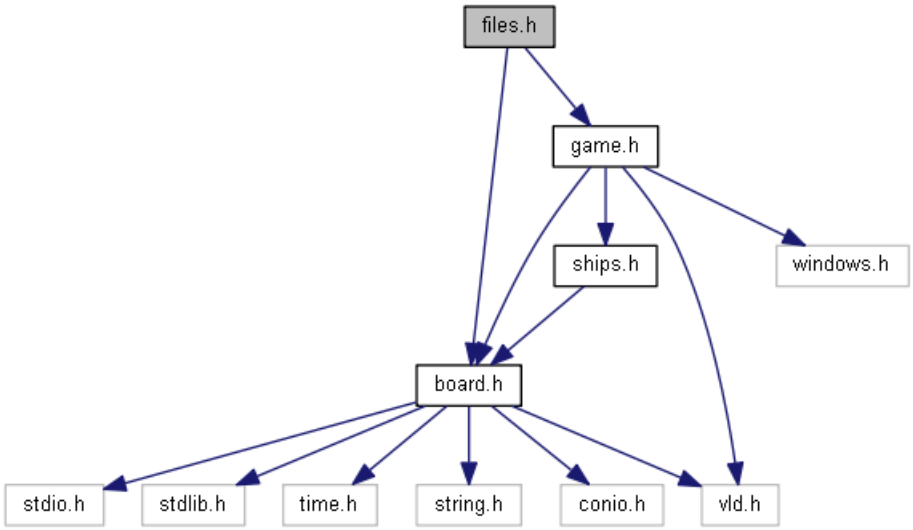
Oto graf wywoływań tej funkcji:



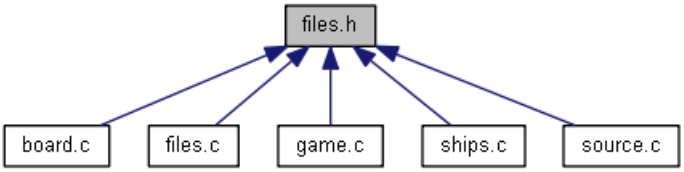
Dokumentacja pliku files.h

```
#include "board.h"
#include "game.h"
```

Wykres zależności załączania dla files.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Definicje typów

- typedef FILE * **file**

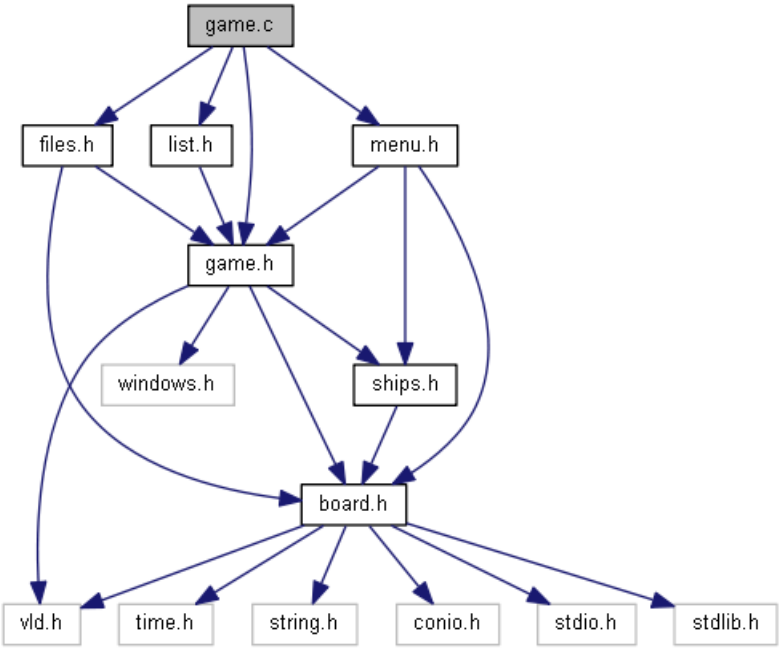
Funkcje

- void **write_stats_in_file** (char *name, **stats** *pHead)
- void **read_board_from_file** (char *name, **cell** game_board[][**COLS**])
- void **save_board_to_file** (char *name, **cell** game_board[][**COLS**])

Dokumentacja pliku game.c

```
#include "game.h"
#include "menu.h"
#include "files.h"
#include "list.h"
```

Wykres zależności załączania dla game.c:



Funkcje

- **booleann is_winner** (**stats** **pHead, int Player)
- void **end_decision** (**cell** player1_board[][**COLS**], **cell** player2_board[][**COLS**], **stats** *st_pHead, **ship** *sh_pHead, int argc, char **argv)
- **coordinate get_target** (**cell** player1_board[][**COLS**], **cell** player2_board[][**COLS**], **stats** *st_pHead, **ship** *sh_pHead, int argc, char **argv)
- void **player_shot** (**cell** player1_board[][**COLS**], **cell** player2_board[][**COLS**], **stats** *st_pHead, int player, int sunk_ship[2][**NUMBER_OF_SHIPS**], **ship** *sh_pHead, int argc, char **argv)
- **coordinate near_shot** (**cell** player_board[][**COLS**], **shots** *shots_info)
- **coordinate easy** (**cell** player_board[][**COLS**], int less, int more)
- **coordinate medium** (**cell** player_board[][**COLS**], **shots** *shots_info)
- **coordinate hard** (**cell** player_board[][**COLS**], **shots** *shots_info, **stats** *pHead)
- **coordinate make_hit** (**cell** player_board[][**COLS**])
- void **ai_shot** (**cell** player_board[][**COLS**], int difficulty, int sunk_ship[2][**NUMBER_OF_SHIPS**], int player, **stats** **pHead, **shots** *shots_info)
- int **check_shot** (**cell** board[][**COLS**], int row, int column)
- int **who_first** (void)
- void **play** (**cell** player1_game_board[][**COLS**], **cell** player2_game_board[][**COLS**], **booleann** with_computer, **stats** **pHead, int difficulty, **ship** *sh_pHead, int argc, char **argv)

Dokumentacja funkcji

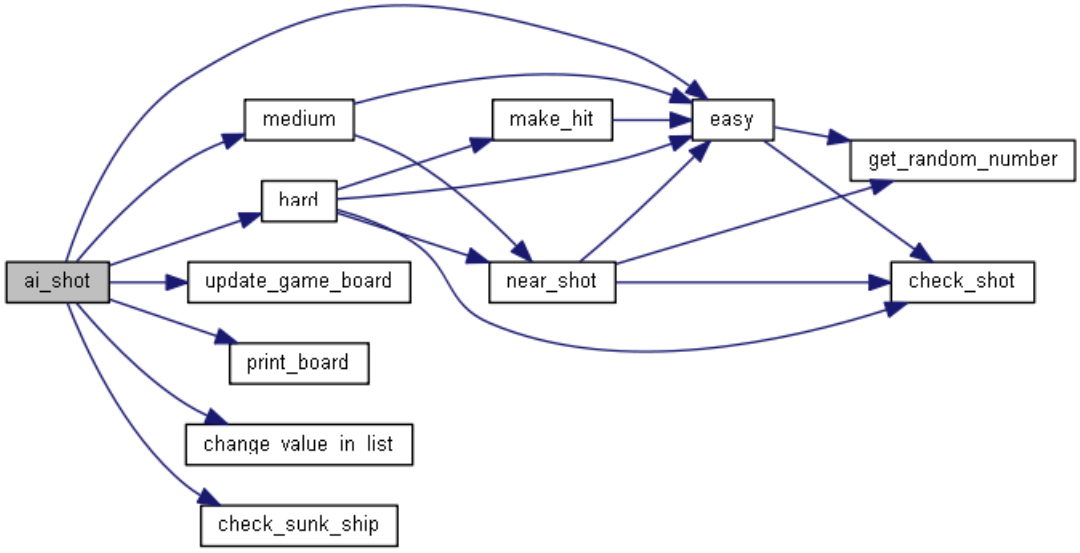
void ai_shot (cell *player_board*[][COLS], int *difficulty*, int *sunk_ship*[2][NUMBER_OF_SHIPS], int *player*, stats ** *pHead*, shots * *shots_info*)

Główna funkcja odpowiedzialna za strzał komputera

Parametry:

player_board	dwuwymiarowa tablica przechowująca informacje o planszy
difficulty	wybrana przez użytkownika trudność
sunk_ship	dwuwymiarowa tablica przechowująca informacje o tym ile każdy statek danego gracza ma nietraionych pól
player	numer gracza strzelającego
pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
shots_info	struktura przechowująca informacje o strzałach komputera

Oto graf wywołań dla tej funkcji:



int check_shot (cell *board*[][COLS], int *row*, int *column*)

Funkcja sprawdzająca w co gracz trafił

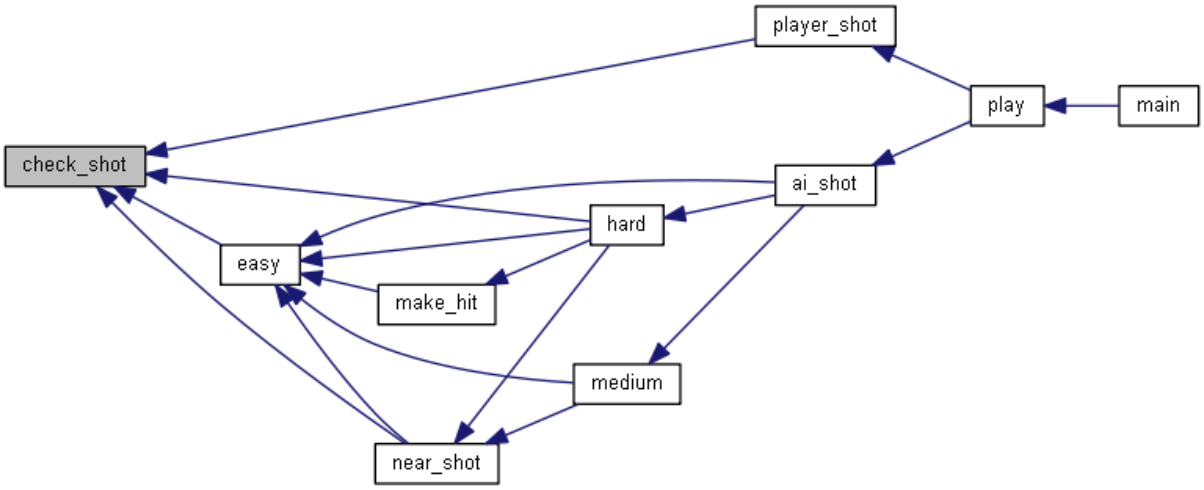
Parametry:

board	dwuwymiarowa tablica przechowująca informacje o planszy
row	wiersz wykonanego strzału
column	kolumna wykonanego strzału

Zwraca:

0 gdy było to pudło, 1 gdy trafiono w statek, -1 jeśli trafiono we wcześniej trafione miejsce

Oto graf wywoływań tej funkcji:



coordinate easy (cell *player_board*[][COLS], int *less*, int *more*)

Funkcja odpowiedzialna za strzał komputera na poziomie trudności - łatwy

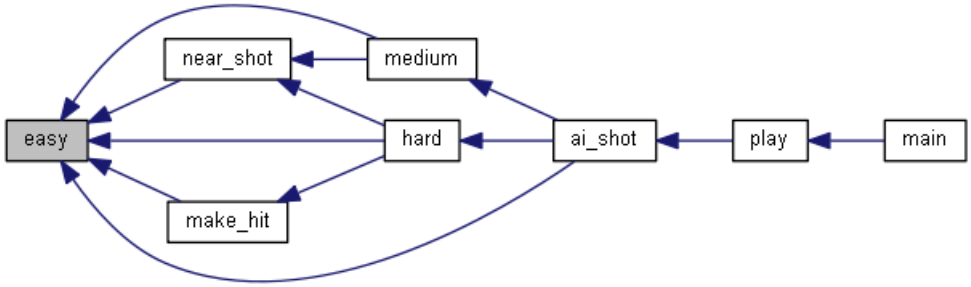
Parametry:

player_board	dwuwymiarowa tablica przechowująca informacje o planszy
less	zakres liczb do losowania
more	zakres liczb do losowania

Zwraca:

współrzędne strzału

Oto graf wywoływań tej funkcji:



void end_decision (cell *player1_board*[][COLS], cell *player2_board*[][COLS], stats * *st_pHead*, ship * *sh_pHead*, int *argc*, char ** *argv*)

Funkcja wywoływana, gdy gracz postanowił zakończyć rozgrywkę. Zapisuje stan planszy i statystyk jeśli gracz sobie tego zażyczy i usuwa listy

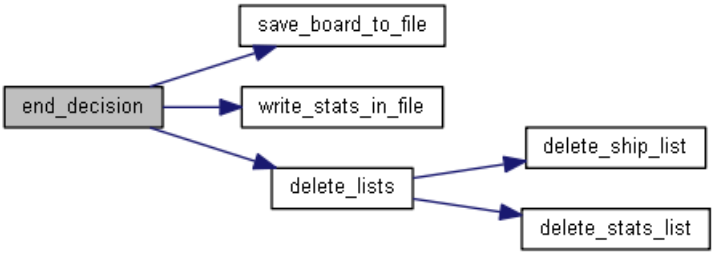
Parametry:

player1_board	dwuwymiarowa tablica przechowująca informacje o planszy pierwszego gracza
player2_board	dwuwymiarowa tablica przechowująca informacje o planszy drugiego gracza lub komputera
st_pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
sh_pHead	głowa listy jednokierunkowej przechowującej informacje o statkach
argc	jeden z parametrów wywołania programu
argv	jeden z parametrów wywołania programu

Zwraca:

współrzędne strzału

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

coordinate get_target (cell *player1_board*[][COLS], cell *player2_board*[][COLS], stats * *st_pHead*, ship * *sh_pHead*, int *argc*, char ** *argv*)

Funkcja pobierająca od gracza współrzędne do strzału

Parametry:

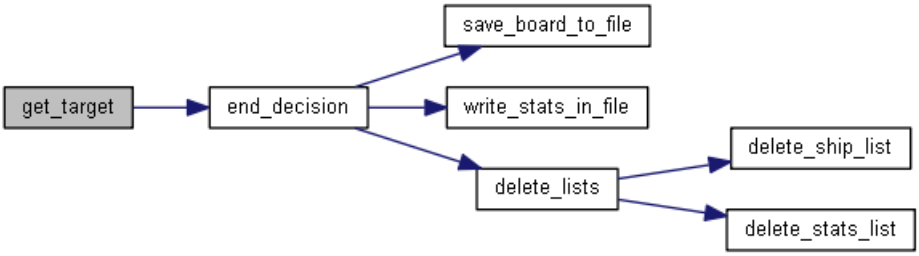
player1_board	dwuwymiarowa tablica przechowująca informacje o planszy pierwszego gracza

player2_board	dwuwymiarowa tablica przechowująca informacje o planszy drugiego gracza lub komputera
st_pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
sh_pHead	głowa listy jednokierunkowej przechowującej informacje o statkach
argc	jeden z parametrów wywołania programu
argv	jeden z parametrów wywołania programu

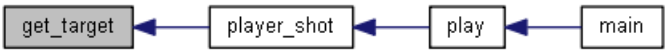
Zwraca:

współrzędne strzału

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



coordinate hard (cell *player_board*[][COLS], shots * *shots_info*, stats * *pHead*)

Funkcja odpowiedzialna za strzał komputera na poziomie trudności - trudny

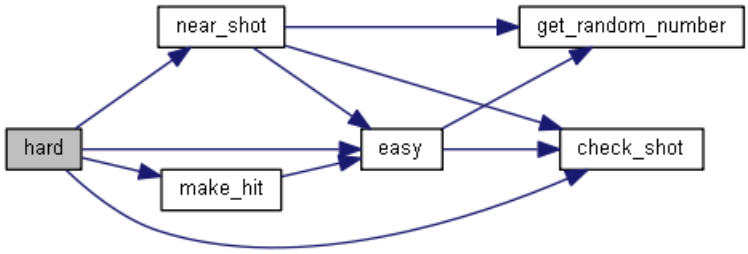
Parametry:

player_board	dwuwymiarowa tablica przechowująca informacje o planszy
shots_info	struktura przechowująca informacje o strzałach komputera
pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach

Zwraca:

współrzędne strzału

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

booleann is_winner (stats ** *pHead*, int *Player*)

Funkcja sprawdzająca czy dany gracz wystrzelał już wszystkie statki

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
Player	numer sprawdzanego gracza

Zwraca:

wartość TRUE jeśli dany gracz zwyciężył, FALSE jeśli nie

coordinate make_hit (cell *player_board*[][COLS])

Funkcja odpowiedzialna za strzał komputera w statek gracza. Wywoływana na poziomie hard, gdy graczowi zostały 3 trafy do zwycięstwa

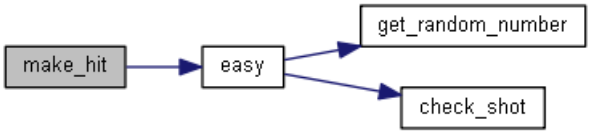
Parametry:

player_board	dwuwymiarowa tablica przechowująca informacje o planszy

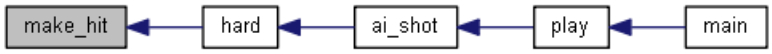
Zwraca:

współrzędne strzału

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



coordinate medium (cell *player_board*[][COLS], shots * *shots_info*)

Funkcja odpowiedzialna za strzał komputera na poziomie trudności - średni

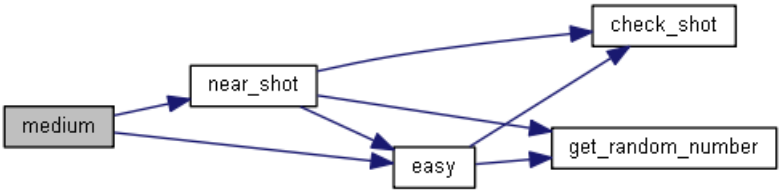
Parametry:

player_board	dwuwymiarowa tablica przechowująca informacje o planszy
shots_info	struktura przechowująca informacje o strzałach komputera

Zwraca:

współrzędne strzału

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

coordinate near_shot (cell *player_board*[][COLS], shots * *shots_info*)

Funkcja odpowiedzialna na wybranie komórki do strzału obok ostatniego strzału przez komputer

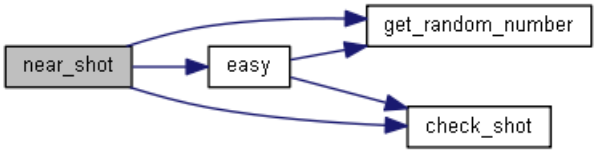
Parametry:

player_board	dwuwymiarowa tablica przechowująca informacje o planszy
shots_info	struktura przechowująca informacje o strzałach komputera

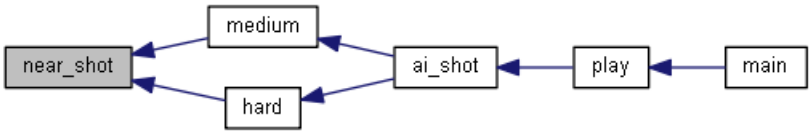
Zwraca:

współrzędne strzału

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



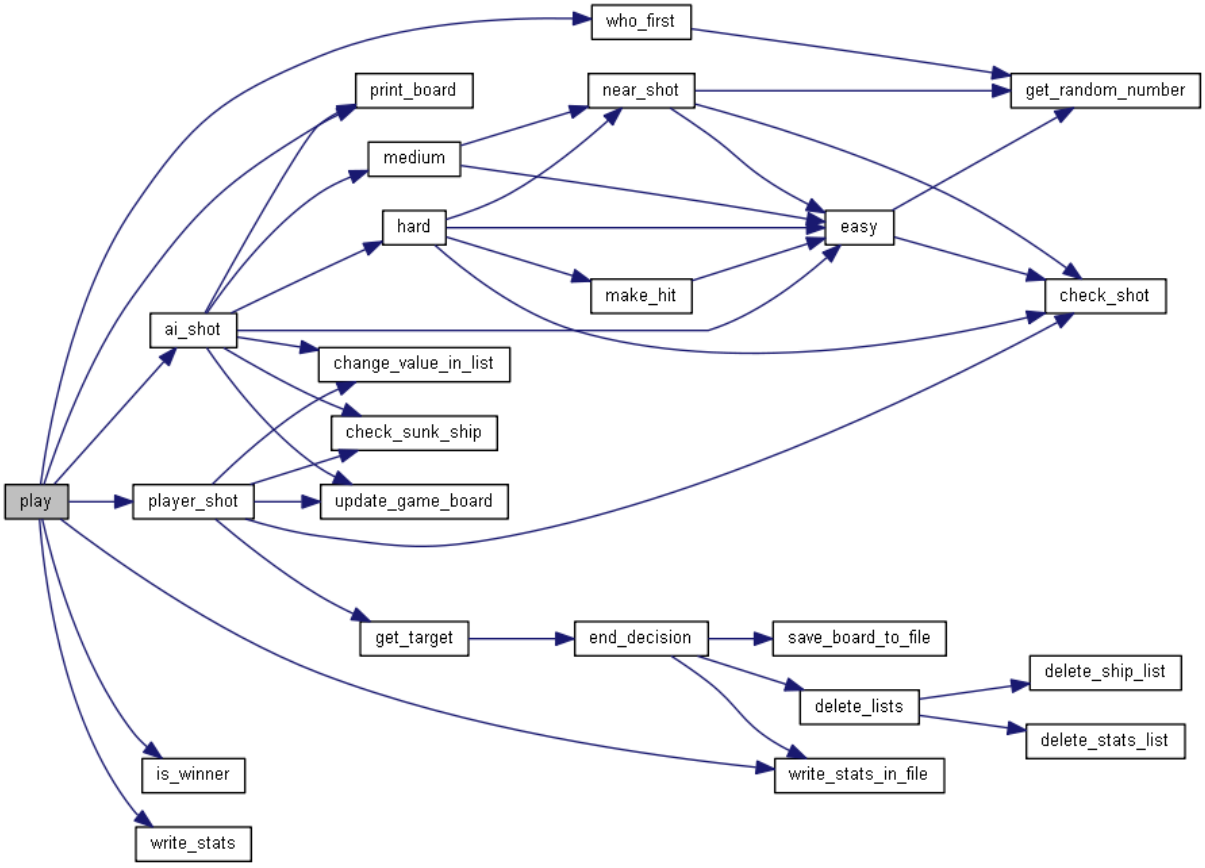
void play (cell *player1_game_board*[][COLS], cell *player2_game_board*[][COLS], booleann *with_computer*, stats ** *pHead*, int *difficulty*, ship * *sh_pHead*, int *argc*, char ** *argv*)

Główna funkcja odpowiedzialna za całą rozgrywkę

Parametry:

player1_game_board	dwuwymiarowa tablica przechowująca informacje o planszy gracza o numerze 1
player2_game_board	dwuwymiarowa tablica przechowująca informacje o planszy gracza o numerze 2
with_computer	informacja, czy gramy z komputerem czy z innym graczem (TRUE - gramy z komputerem
pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
difficulty	wybrana przez użytkownika trudność
sh_pHead	głowa listy jednokierunkowej przechowującej informacje o statkach
argc	jeden z parametrów wywołania programu
argv	jeden z parametrów wywołania programu

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

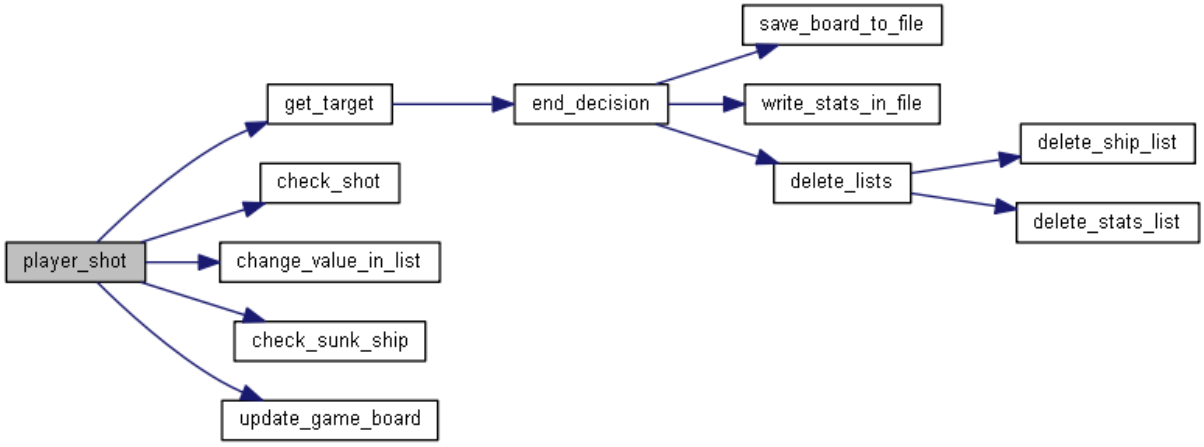
void player_shot (cell *player1_board*[][COLS], cell *player2_board*[][COLS], stats * *st_pHead*, int *player*, int *sunk_ship*[2][NUMBER_OF_SHIPS], ship * *sh_pHead*, int *argc*, char ** *argv*)

Funkcja odpowiedzialna na wybranie komórki do strzału przez gracza

Parametry:

player1_board	dwuwymiarowa tablica przechowująca informacje o planszy pierwszego gracza
player2_board	dwuwymiarowa tablica przechowująca informacje o planszy drugiego gracza lub komputera
st_pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
player	numer gracza strzelającego
sunk_ship	dwuwymiarowa tablica przechowująca informacje o tym ile każdy statek danego gracza ma nietraionych pól
sh_pHead	głowa listy jednokierunkowej przechowującej informacje o statkach
argc	jeden z parametrów wywołania programu
argv	jeden z parametrów wywołania programu

Oto graf wywołań dla tej funkcji:



int who_first (void)

Funkcja losująca cyfrę 0 lub 1 w celu wybrania gracza rozpoczynającego rozgrywkę

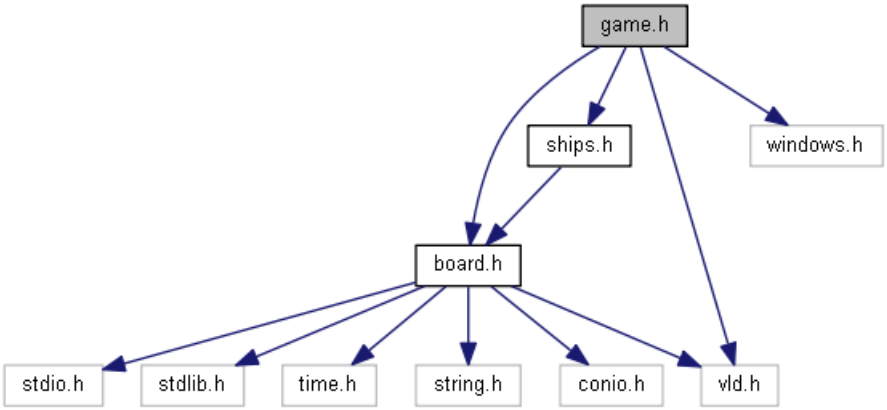
Zwraca:

numer gracza rozpoczynającego rozgrywkę

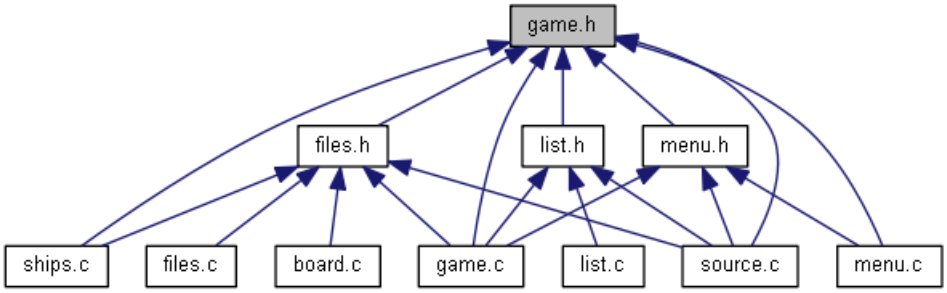
Dokumentacja pliku game.h

```
#include "board.h"
#include "ships.h"
#include <windows.h>
#include <vld.h>
```

Wykres zależności załączania dla game.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Struktury danych

- struct **_stats**
- struct **_shots**

Definicje

- #define **PLAYER_ONE** 0
- #define **PLAYER_TWO** 1
- #define **WAIT** 1400

Definicje typów

- typedef struct **_stats stats**
- typedef struct **_shots shots**

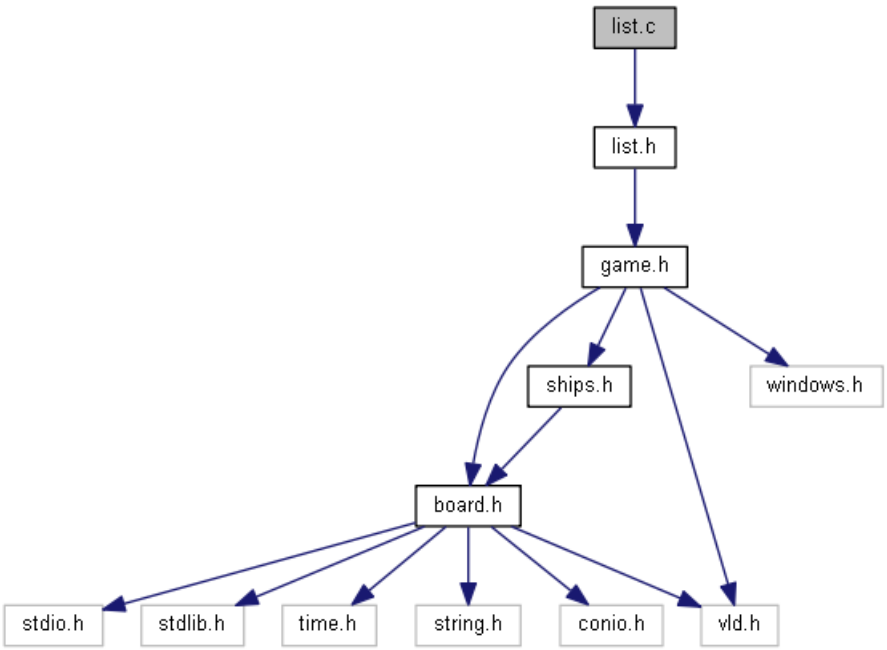
Funkcje

- int **who_first** (void)
- **booleann is_winner** (**stats** **pHead, int Player)
- void **end_decision** (**cell** player1_board[][**COLS**], **cell** player2_board[][**COLS**], **stats** *st_pHead, **ship** *sh_pHead, int argc, char **argv)
- **coordinate get_target** (**cell** player1_board[][**COLS**], **cell** player2_board[][**COLS**], **stats** *st_pHead, **ship** *sh_pHead, int argc, char **argv)
- int **check_shot** (**cell** board[][**COLS**], int row, int column)
- void **player_shot** (**cell** player1_board[][**COLS**], **cell** player2_board[][**COLS**], **stats** *st_pHead, int player, int sunk_ship[2][**NUMBER_OF_SHIPS**], **ship** *sh_pHead, int argc, char **argv)
- **coordinate near_shot** (**cell** player_board[][**COLS**], **shots** *shots_info)
- **coordinate easy** (**cell** player_board[][**COLS**], int less, int more)
- **coordinate medium** (**cell** player_board[][**COLS**], **shots** *shots_info)
- **coordinate hard** (**cell** player_board[][**COLS**], **shots** *shots_info, **stats** *pHead)
- **coordinate make_hit** (**cell** player_board[][**COLS**])
- void **ai_shot** (**cell** player_board[][**COLS**], int difficulty, int sunk_ship[2][**NUMBER_OF_SHIPS**], int player, **stats** **pHead, **shots** *shots_info)
- void **play** (**cell** player1_game_board[][**COLS**], **cell** player2_game_board[][**COLS**], **booleann** with_computer, **stats** **pHead, int difficulty, **ship** *sh_pHead, int argc, char **argv)

Dokumentacja pliku list.c

```
#include "list.h"
```

Wykres zależności załączania dla list.c:



Funkcje

- void **initialize_lists** (**stats** **stats_pHead, **ship** **ship_pHead)
- void **initialize_stats_list** (**stats** **pHead)
- void **initialize_ship_list** (**ship** **pHead)
- void **delete_stats_list** (**stats** **pHead)
- void **delete_ship_list** (**ship** **pHead)
- void **delete_lists** (**stats** **stats_pHead, **ship** **ship_pHead)
- void **push_stats_front** (**stats** **pHead, int Player)
- void **push_stats_back** (**stats** *pHead, int Player)
- void **push_ship_front** (**ship** **pHead, char Symbol, int Length, char *Name)
- void **clear_stats** (**stats** **pHead)
- void **change_value_in_list** (**stats** **pHead, int Player, char *what_to_add)

Dokumentacja funkcji

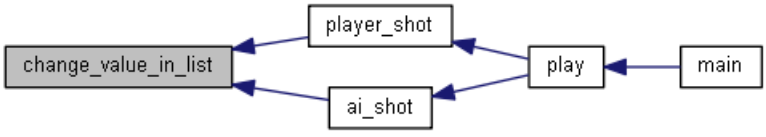
void change_value_in_list (stats ** *pHead*, int *Player*, char * *what_to_add*)

Funkcja zmieniająca wartość w jednym, wybranym polu

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
Player	numer gracza u którego zmieniamy wartość
what_to_add	wybieramy którą wartość chcemy zmienić

Oto graf wywoływań tej funkcji:



void clear_stats (stats ** *pHead*)

Funkcja zerująca statystyki obu graczy

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach

Oto graf wywoływań tej funkcji:

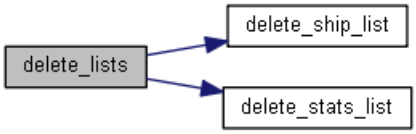
void delete_lists (stats ** *stats_pHead*, ship ** *ship_pHead*)

Funkcja usuwająca listy jednokierunkowe statystyk i statków

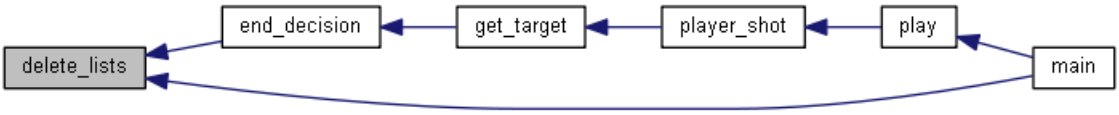
Parametry:

stats_pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
ship_pHead	głowa listy jednokierunkowej przechowującej informacje o statkach

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



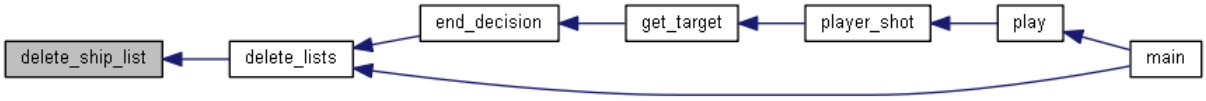
void delete_ship_list (ship ** *pHead*)

Funkcja usuwająca listę jednokierunkową statków

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statkach

Oto graf wywoływań tej funkcji:



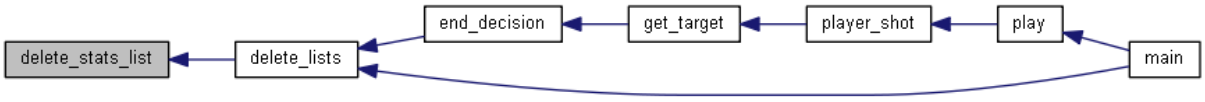
void delete_stats_list (stats ** *pHead*)

Funkcja usuwająca listę jednokierunkową statystyk

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach

Oto graf wywoływań tej funkcji:



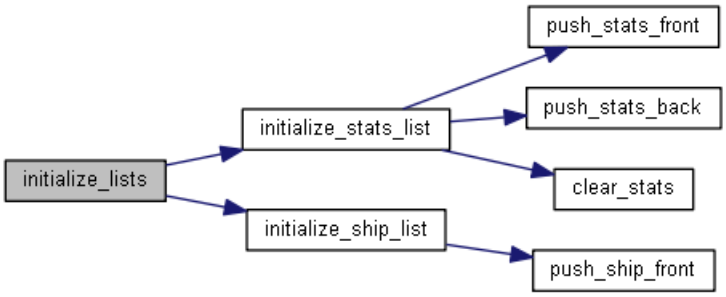
void initialize_lists (stats ** *stats_pHead*, ship ** *ship_pHead*)

Funkcja inicjalizująca listy jednokierunkowe statków i statystyk

Parametry:

stats_pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
ship_pHead	głowa listy jednokierunkowej przechowującej informacje o statkach

Oto graf wywołań dla tej funkcji:



void initialize_ship_list (ship ** *pHead*)

Funkcja inicjalizująca listę jednokierunkową statków

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statkach

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

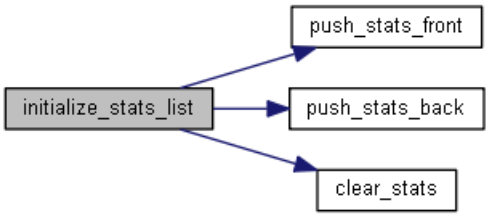
void initialize_stats_list (stats ** *pHead*)

Funkcja inicjalizująca listę jednokierunkową statystyk

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

void push_ship_front (ship ** *pHead*, char *Symbol*, int *Length*, char * *Name*)

Funkcja dodająca na początek listy jednokierunkowej statków element

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statkach
Symbol	symbol danego statku
Length	długość danego statku
Name	nazwa danego statku

Oto graf wywoływań tej funkcji:

void push_stats_back (stats * *pHead*, int *Player*)

Funkcja dodająca na koniec listy jednokierunkowej statystyk element

Parametry:

pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
Player	numer gracza którego dodajemy do listy

Oto graf wywoływań tej funkcji:

void push_stats_front (stats ** *pHead*, int *Player*)

Funkcja dodająca na początek listy jednokierunkowej statystyk element

Parametry:

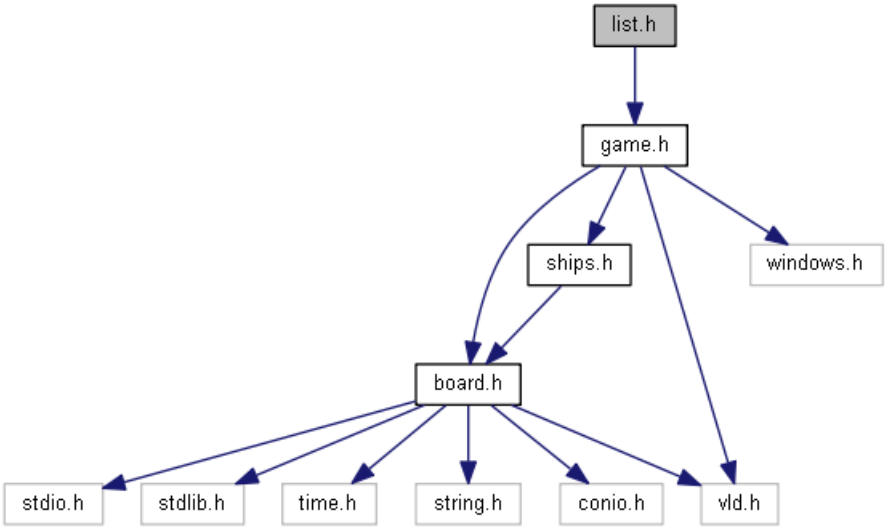
pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach
Player	numer gracza którego dodajemy do listy

Oto graf wywoływań tej funkcji:

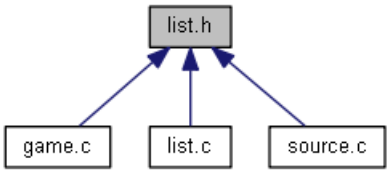
Dokumentacja pliku list.h

```
#include "game.h"
```

Wykres zależności załączania dla list.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



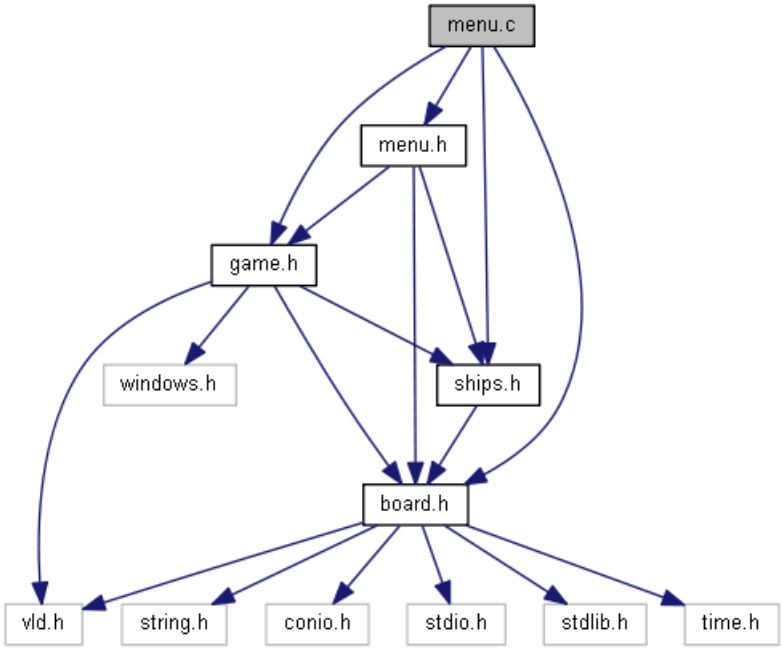
Funkcje

- void **initialize_lists** (**stats** **stats_pHead, **ship** **ship_pHead)
- void **initialize_ship_list** (**ship** **pHead)
- void **initialize_stats_list** (**stats** **pHead)
- void **delete_stats_list** (**stats** **pHead)
- void **delete_ship_list** (**ship** **pHead)
- void **delete_lists** (**stats** **stats_pHead, **ship** **ship_pHead)
- void **push_stats_front** (**stats** **pHead, int Player)
- void **push_stats_back** (**stats** *pHead, int Player)
- void **push_ship_front** (**ship** **pHead, char Symbol, int Length, char *Name)
- void **clear_stats** (**stats** **pHead)
- void **change_value_in_list** (**stats** **pHead, int Player, char *what_to_add)

Dokumentacja pliku menu.c

```
#include "menu.h"
#include "ships.h"
#include "board.h"
#include "game.h"
```

Wykres zależności załączania dla menu.c:



Funkcje

- void **welcome_screen** (void)
- **booleann with_who** (void)
- int **what_difficulty** (void)
- **booleann way_of_placing** (int player)
- void **menu** (**cell** player1_game_board[][**COLS**], **cell** player2_game_board[][**COLS**], **ship** *pHead, **booleann** *option_with_who, int *difficulty, int argc)
- void **write_stats** (**stats** *pHead)

Dokumentacja funkcji

void menu (cell *player1_game_board*[][COLS], cell *player2_game_board*[][COLS], ship * *pHead*, booleann * *option_with_who*, int * *difficulty*, int *argc*)

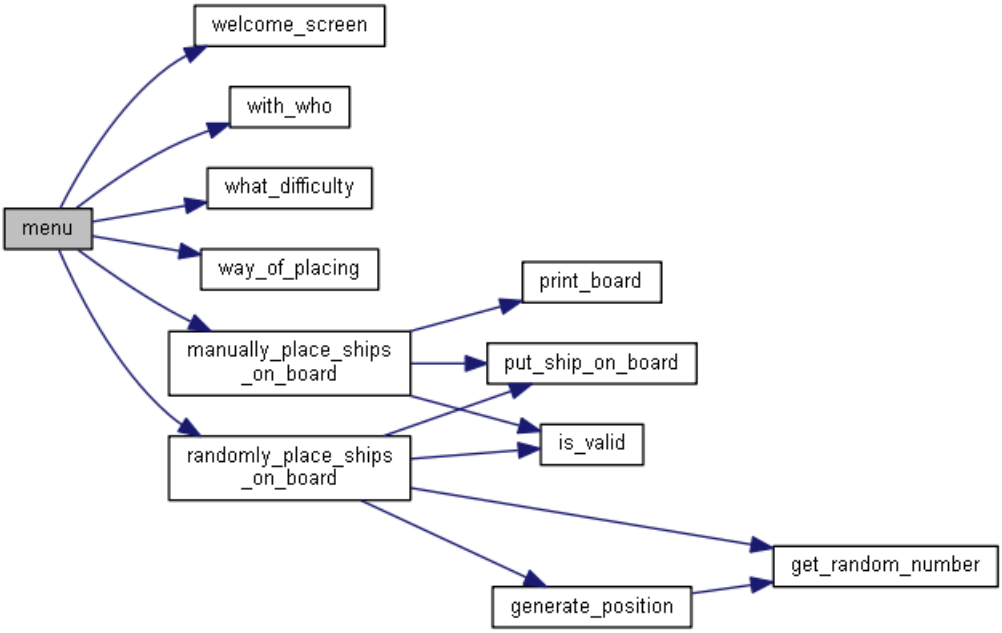
Główna funkcja odpowiedzialna za początkową fazę gry - wybieramy tu poziom trudności, sposób rozmieszczenia statków itp.

Parametry:

player1_game_board	dwuwymiarowa tablica przechowująca informacje o planszy gracza o numerze 1
player2_game_board	dwuwymiarowa tablica przechowująca informacje o planszy gracza o numerze 2
pHead	głowa listy jednokierunkowej przechowującej informacje o statkach
option_with_who	informacja, czy gramy z komputerem czy z innym graczem (TRUE - gramy z komputerem)
difficulty	wybrana przez użytkownika trudność

argc	jeden z parametrów wywołania programu

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

booleann way_of_placing (int *player*)

Funkcja w której wybieramy jak chcemy rozmieścić statki na planszy

Parametry:

player	numer gracza wybierającego sposób rozmieszczenia statków

Zwraca:

TRUE - ręcznie, FALSE - losowo

Oto graf wywoływań tej funkcji:

void welcome_screen (void)

Funkcja wypisująca ekran powitalny

Oto graf wywoływań tej funkcji:

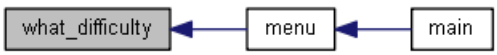
int what_difficulty (void)

Funkcja w której wybieramy na jakim poziomie trudności chcemy grać

Zwraca:

wybrany przez nas poziom trudnoćci

Oto graf wywoływań tej funkcji:



booleann with_who (void)

Funkcja w której wybieramy z kim chcemy zagrać - z komputerem czy innym graczem

Zwraca:

TRUE - z komputerem, FALSE - z innym graczem

Oto graf wywoływań tej funkcji:

void write_stats (stats * *pHead*)

Funkcja wypisująca statystyki na ekran

Parametry:

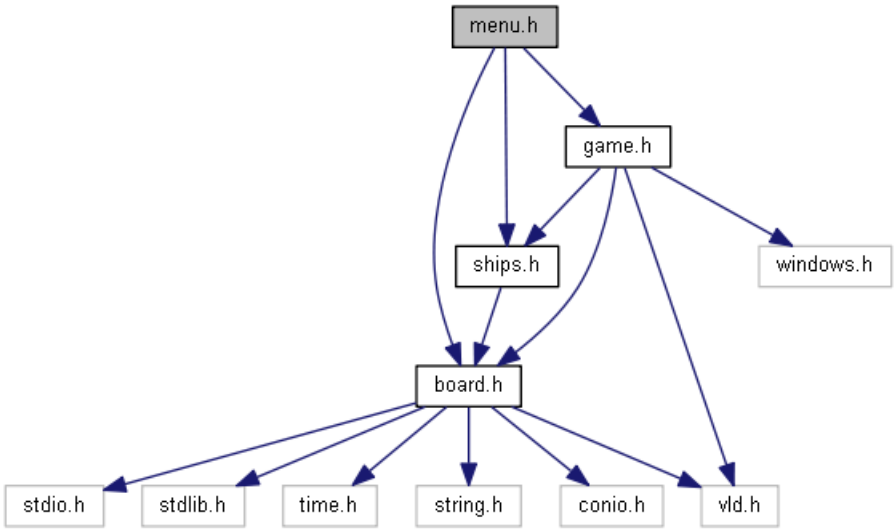
pHead	głowa listy jednokierunkowej przechowującej informacje o statystykach

Oto graf wywoływań tej funkcji:

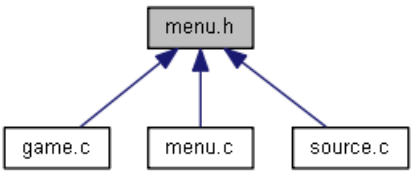
Dokumentacja pliku menu.h

```
#include "board.h"
#include "ships.h"
#include "game.h"
```

Wykres zależności załączania dla menu.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



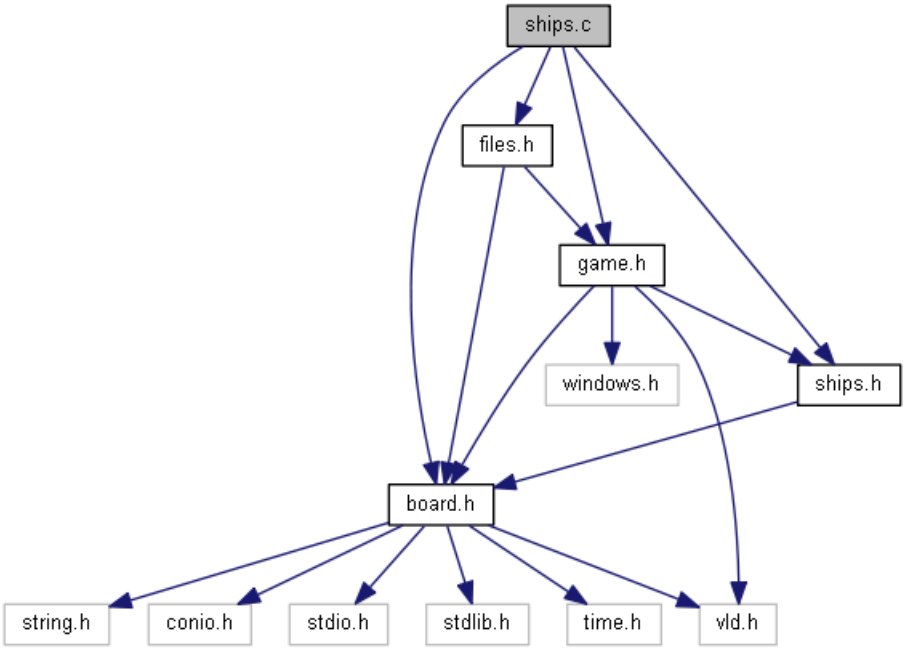
Funkcje

- void **welcome_screen** (void)
- void **menu** (**cell** player1_game_board[][**COLS**], **cell** player2_game_board[][**COLS**], **ship** *pHead, **booleann** *option_with_who, int *difficulty, int argc)
- **booleann with_who** (void)
- **booleann way_of_placing** (int player)
- int **what_difficulty** (void)
- void **write_stats** (**stats** *pHead)

Dokumentacja pliku ships.c

```
#include "ships.h"
#include "board.h"
#include "game.h"
#include "files.h"
```

Wykres zależności załączania dla ships.c:



Funkcje

- void **put_ship_on_board** (**cell** game_board[][**COLS**], char Symbol, **ship** *pHead, **coordinate** position, int direction)
- void **randomly_place_ships_on_board** (**cell** game_board[][**COLS**], **ship** *pHead)
- void **manually_place_ships_on_board** (**cell** game_board[][**COLS**], **ship** *pHead)
- **coordinate generate_position** (int direction, int length)
- **booleann is_valid** (**cell** game_board[][**COLS**], **coordinate** position, int direction, int length)
- **booleann check_sunk_ship** (int sunk_ship[][**NUMBER_OF_SHIPS**], int player, char ship_symbol, **cell** game_board[][**COLS**])
- int **get_random_number** (int less, int more)

Dokumentacja funkcji

booleann check_sunk_ship (int *sunkShip*[][NUMBER_OF_SHIPS], int *player*, char *ship_symbol*, cell *game_board*[][COLS])

Funkcja sprawdzająca czy dany statek zatonął

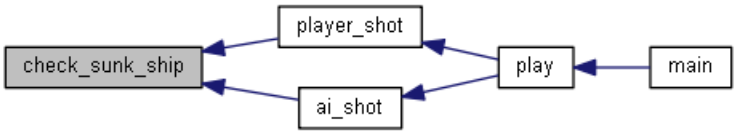
Parametry:

sunkShip	dwuwymiarowa tablica przechowująca informacje o tym ile każdy statek danego gracza ma nietraionych pól
player	numer gracza strzelającego
ship_symbol	symbol statku w postaci literki
game_board	dwuwymiarowa tablica przechowująca informacje o planszy

Zwraca:

informacja czy dany statek zatonął

Oto graf wywoływań tej funkcji:



coordinate generate_position (int *direction*, int *length*)

Funkcja generująca pozycje do ustawienia statku

Parametry:

direction	kierunek w którym zostanie ustawiony statek (poziomo lub pionowo)
length	długość statku dla którego zostanie wygenerowana pozycja

Zwraca:

współrzędne wygenerowanej pozycji

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

int get_random_number (int *less*, int *more*)

Funkcja losująca jedną liczbę z danego zakresu

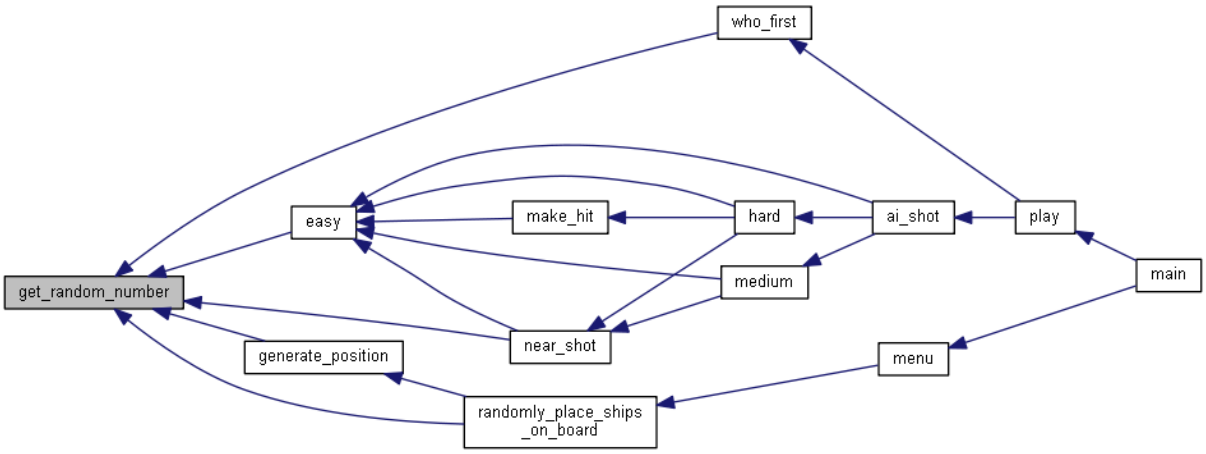
Parametry:

less	zakres liczb do losowania
more	zakres liczb do losowania

Zwraca:

losowa liczba

Oto graf wywoływań tej funkcji:



booleann is_valid (cell *board*[][COLS], coordinate *position*, int *direction*, int *length*)

Funkcja sprawdzająca czy dana pozycja jest dostępna

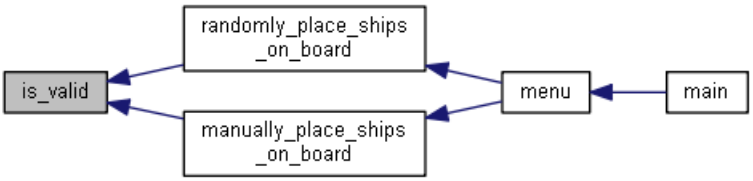
Parametry:

board	dwuwymiarowa tablica przechowująca informacje o planszy
position	współrzędne na których ma być ustawiony statek
direction	kierunek w którym zostanie ustawiony statek (poziomo lub pionowo)
length	długość statku dla którego zostanie wygenerowana pozycja

Zwraca:

informacja czy dana pozycja jest dostępna

Oto graf wywoływań tej funkcji:



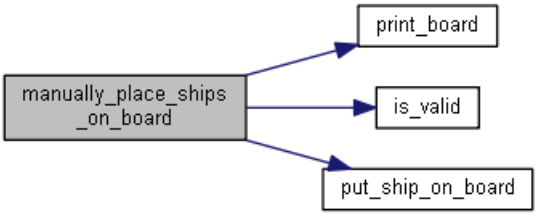
void manually_place_ships_on_board (cell *game_board*[][COLS], ship * *pHead*)

Funkcja ustawiająca manualnie statki na planszy

Parametry:

game_board	dwuwymiarowa tablica przechowująca informacje o planszy
pHead	głowa listy jednokierunkowej przechowującej informacje o statkach

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

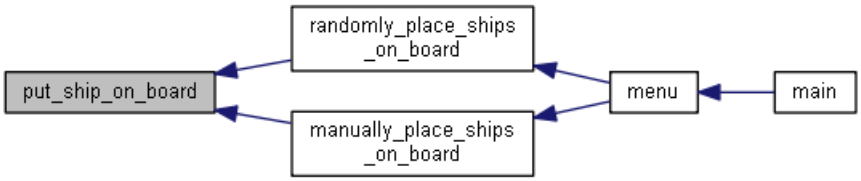
void put_ship_on_board (cell *game_board*[][COLS], char *Symbol*, ship * *pHead*, coordinate *position*, int *direction*)

Funkcja ustawiająca statek na planszy

Parametry:

game_board	dwuwymiarowa tablica przechowująca informacje o planszy
Symbol	symbol danego statku
pHead	głowa listy jednokierunkowej przechowującej informacje o statkach
position	współrzędne na których ustawiony zostanie statek
direction	kierunek w którym zostanie ustawiony statek (poziomo lub pionowo)

Oto graf wywoływań tej funkcji:



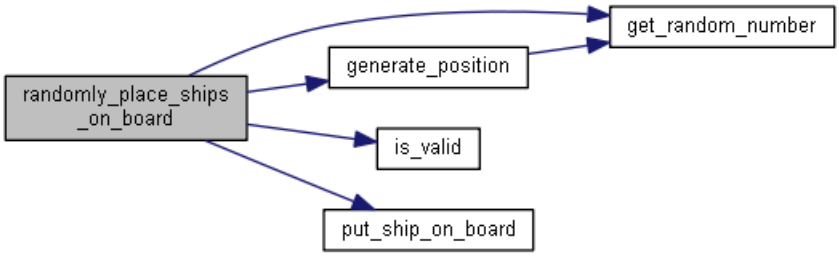
void randomly_place_ships_on_board (cell *game_board*[][COLS], ship * *pHead*)

Funkcja ustawiająca losowo statki na planszy

Parametry:

game_board	dwuwymiarowa tablica przechowująca informacje o planszy
pHead	głowa listy jednokierunkowej przechowującej informacje o statkach

Oto graf wywołań dla tej funkcji:

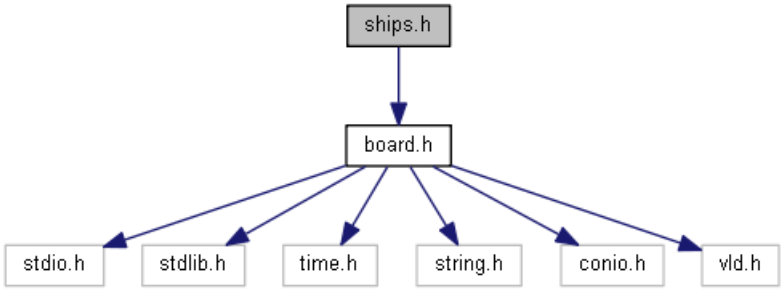


Oto graf wywoływań tej funkcji:

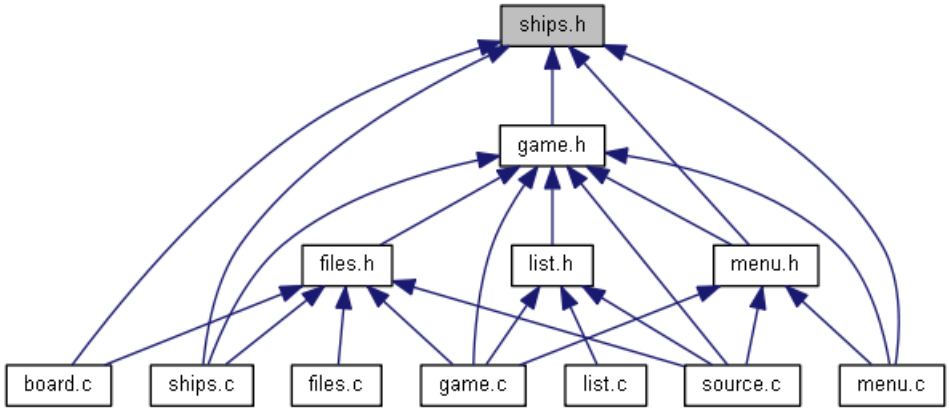
Dokumentacja pliku ships.h

```
#include "board.h"
```

Wykres zależności załączania dla ships.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Struktury danych

- struct **_ship**

Definicje

- #define **HORIZONTAL** 0
- #define **VERTICAL** 1
- #define **NUMBER_OF_SHIPS** 5
- #define **CARRIER** 'c'
- #define **BATTLESHIP** 'b'
- #define **CRUISER** 'r'
- #define **SUBMARINE** 's'
- #define **DESTROYER** 'd'

Definicje typów

- typedef struct **_ship ship**

Funkcje

- void **put_ship_on_board** (**cell** game_board[][**COLS**], char Symbol, **ship** *pHead, **coordinate** position, int direction)
- void **randomly_place_ships_on_board** (**cell** game_board[][**COLS**], **ship** *pHead)
- **coordinate generate_position** (int direction, int length)
- **booleann check_sunk_ship** (int sunkShip[][**NUMBER_OF_SHIPS**], int player, char ship_symbol, **cell** game_board[][**COLS**])
- **booleann is_valid** (**cell** board[][**COLS**], **coordinate** position, int direction, int length)

- void **manually_place_ships_on_board** (**cell** game_board[][**COLS**], **ship** *pHead)
- int **get_random_number** (int less, int more)

Dokumentacja definicji

#define BATTLESHIP 'b'

#define CARRIER 'c'

#define CRUISER 'r'

#define DESTROYER 'd'

#define HORIZONTAL 0

#define NUMBER_OF_SHIPS 5

#define SUBMARINE 's'

#define VERTICAL 1

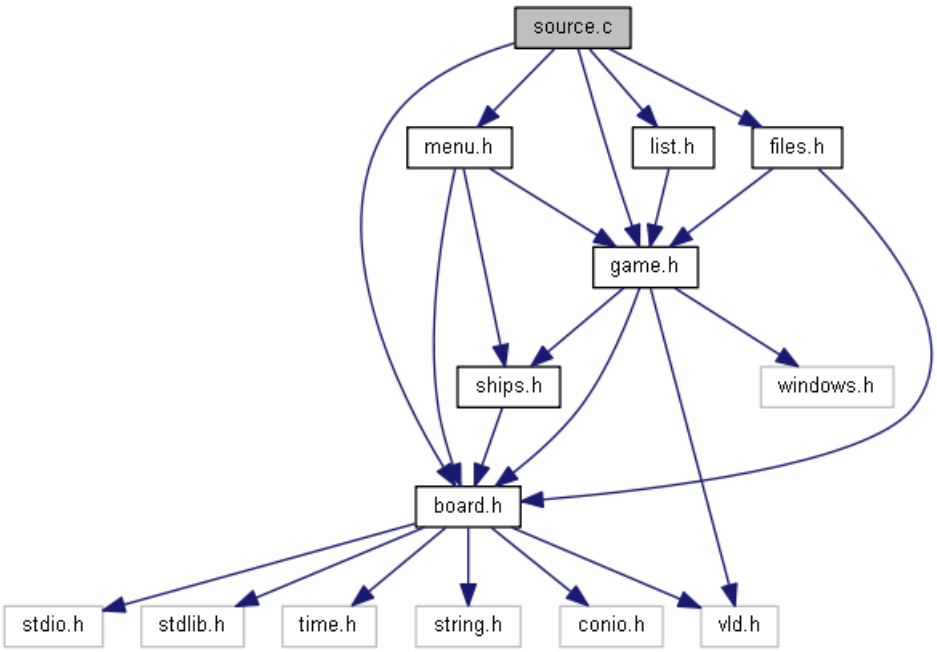
Dokumentacja definicji typów

typedef struct _ship ship

Dokumentacja pliku source.c

```
#include "board.h"
#include "menu.h"
#include "game.h"
#include "files.h"
#include "list.h"
```

Wykres zależności załączania dla source.c:



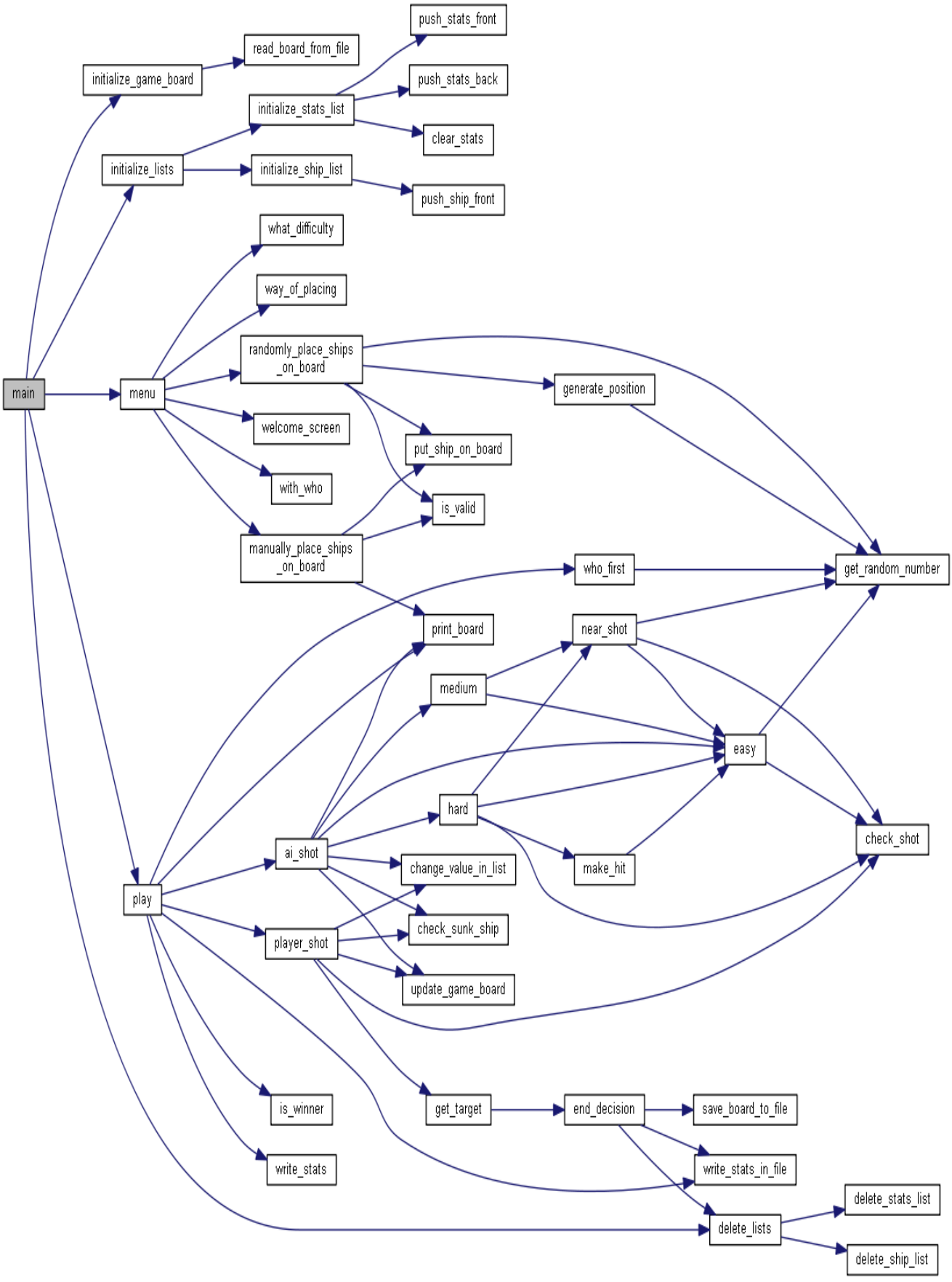
Funkcje

- int **main** (int argc, char **argv)

Dokumentacja funkcji

int main (int *argc*, char ** *argv*)

Oto graf wywołań dla tej funkcji:


main
menu
initialize_game_board
read_board_from_file
initialize_lists
initialize_stats_list
push_stats_front
push_stats_back
clear_stats
initialize_ship_list
push_ship_front
what_difficulty
way_of_placing
randomly_place_ships_on_board
generate_position
welcome_screen
with_who
put_ship_on_board
is_valid
manually_place_ships_on_board
who_first
get_random_number
print_board
near_shot
medium
easy
hard
check_shot
make_hit
play
ai_shot
change_value_in_list
player_shot
check_sunk_ship
update_game_board
is_winner
write_stats
get_target
end_decision
save_board_to_file
write_stats_in_file
delete_lists
delete_stats_list
delete_ship_list