

Projekt zaliczeniowy z PK3 Statki

Wygenerowano przez Doxygen 1.8.13



# Spis treści

<b>1</b>	<b>Indeks hierarchiczny</b>	<b>1</b>
1.1	Hierarchia klas . . . . .	1
<b>2</b>	<b>Indeks klas</b>	<b>3</b>
2.1	Lista klas . . . . .	3
<b>3</b>	<b>Indeks plików</b>	<b>5</b>
3.1	Lista plików . . . . .	5
<b>4</b>	<b>Dokumentacja klas</b>	<b>7</b>
4.1	Dokumentacja struktury el_historia . . . . .	7
4.1.1	Dokumentacja konstruktora i destruktora . . . . .	8
4.1.1.1	el_historia() . . . . .	8
4.1.2	Dokumentacja atrybutów składowych . . . . .	8
4.1.2.1	nazwa . . . . .	8
4.1.2.2	next . . . . .	8
4.1.2.3	trafienie . . . . .	8
4.1.2.4	x . . . . .	8
4.1.2.5	y . . . . .	9
4.2	Dokumentacja struktury el_maszt . . . . .	9
4.2.1	Dokumentacja konstruktora i destruktora . . . . .	9
4.2.1.1	el_maszt() . . . . .	9
4.2.2	Dokumentacja atrybutów składowych . . . . .	10
4.2.2.1	next . . . . .	10
4.2.2.2	x . . . . .	10

4.2.2.3	y	10
4.3	Dokumentacja struktury el_ranking	10
4.3.1	Dokumentacja konstruktora i destruktora	11
4.3.1.1	el_ranking()	11
4.3.2	Dokumentacja atrybutów składowych	11
4.3.2.1	nazwa	11
4.3.2.2	next	11
4.3.2.3	punkty	11
4.4	Dokumentacja struktury el_statek	12
4.4.1	Dokumentacja konstruktora i destruktora	13
4.4.1.1	el_statek()	14
4.4.2	Dokumentacja atrybutów składowych	14
4.4.2.1	ile_m	14
4.4.2.2	next	14
4.5	Dokumentacja klasy gracz	14
4.5.1	Dokumentacja konstruktora i destruktora	17
4.5.1.1	gracz()	17
4.5.2	Dokumentacja funkcji składowych	17
4.5.2.1	czytaj()	17
4.5.2.2	getNazwa()	17
4.5.2.3	getWrog()	18
4.5.2.4	ruch()	18
4.5.2.5	setNazwa()	18
4.5.2.6	setWrog()	18
4.5.2.7	wypelnij()	18
4.5.3	Dokumentacja atrybutów składowych	18
4.5.3.1	nazwa	18
4.5.3.2	wrog	19
4.6	Dokumentacja klasy historia	19
4.6.1	Dokumentacja konstruktora i destruktora	20

4.6.1.1	historia()	20
4.6.1.2	~historia()	20
4.6.2	Dokumentacja funkcji składowych	20
4.6.2.1	wypisz()	20
4.6.2.2	zapisz()	20
4.6.3	Dokumentacja atrybutów składowych	21
4.6.3.1	head	21
4.7	Dokumentacja klasy konfiguracja	21
4.7.1	Dokumentacja konstruktora i destruktora	22
4.7.1.1	konfiguracja()	22
4.7.1.2	~konfiguracja()	22
4.7.2	Dokumentacja funkcji składowych	22
4.7.2.1	getElement()	22
4.7.2.2	getIle()	23
4.7.2.3	getKolumny()	23
4.7.2.4	getWiersze()	23
4.7.2.5	setKolumny()	23
4.7.2.6	setWiersze()	23
4.7.2.7	utworzListe()	23
4.7.2.8	wczytaj()	23
4.7.3	Dokumentacja atrybutów składowych	24
4.7.3.1	ileStatkow	24
4.7.3.2	kolumny	24
4.7.3.3	tab	24
4.7.3.4	wiersze	24
4.8	Dokumentacja klasy maszt	25
4.8.1	Dokumentacja konstruktora i destruktora	26
4.8.1.1	maszt()	27
4.8.1.2	~maszt()	27
4.8.2	Dokumentacja funkcji składowych	27

4.8.2.1	operator"!()	27
4.8.2.2	operator+=()	27
4.8.2.3	porownaj()	28
4.8.2.4	przepisz()	28
4.8.2.5	szukaj()	28
4.8.2.6	trafienie()	29
4.8.2.7	zapisz()	29
4.8.3	Dokumentacja atrybutów składowych	29
4.8.3.1	head	29
4.9	Dokumentacja klasy plansza	30
4.9.1	Dokumentacja konstruktora i destruktor	31
4.9.1.1	plansza()	31
4.9.1.2	~plansza()	31
4.9.2	Dokumentacja funkcji składowych	31
4.9.2.1	strzal()	31
4.9.2.2	wyswietl()	32
4.9.3	Dokumentacja atrybutów składowych	32
4.9.3.1	tGracz	32
4.9.3.2	tWrog	32
4.10	Dokumentacja klasy punkty	32
4.10.1	Dokumentacja konstruktora i destruktor	34
4.10.1.1	punkty()	34
4.10.2	Dokumentacja funkcji składowych	35
4.10.2.1	getCelnosc()	35
4.10.2.2	getPunkty()	35
4.10.2.3	getRuchy()	35
4.10.2.4	getTrafione()	35
4.10.2.5	operator++()	35
4.10.2.6	operator--()	35
4.10.2.7	setRuchy()	35

4.10.2.8	setTrafione()	36
4.10.3	Dokumentacja atrybutów składowych	36
4.10.3.1	ruchy	36
4.10.3.2	trafione	36
4.10.3.3	trudnosc	36
4.11	Dokumentacja klasy ranking	37
4.11.1	Dokumentacja konstruktora i destruktora	37
4.11.1.1	ranking()	38
4.11.1.2	~ranking()	38
4.11.2	Dokumentacja funkcji składowych	38
4.11.2.1	operator+=()	38
4.11.2.2	operator<<()	38
4.11.2.3	wczytaj()	39
4.11.2.4	zapisz()	39
4.11.3	Dokumentacja atrybutów składowych	39
4.11.3.1	head	39
4.12	Dokumentacja klasy statek	40
4.12.1	Dokumentacja konstruktora i destruktora	42
4.12.1.1	statek()	42
4.12.1.2	~statek()	42
4.12.2	Dokumentacja funkcji składowych	42
4.12.2.1	operator"!()	42
4.12.2.2	operator+=()	42
4.12.2.3	szukaj()	43
4.12.2.4	trafienie()	43
4.12.2.5	wczytaj()	44
4.12.2.6	wprowadz()	44
4.12.2.7	zapisz()	44
4.12.3	Dokumentacja atrybutów składowych	44
4.12.3.1	head	44

<b>5 Dokumentacja plików</b>	<b>45</b>
5.1 Dokumentacja pliku config.cpp	45
5.2 Dokumentacja pliku config.h	45
5.2.1 Dokumentacja zmiennych	47
5.2.1.1 Hist	47
5.2.1.2 ustawienia	47
5.3 Dokumentacja pliku funkcje.h	47
5.3.1 Dokumentacja funkcji	48
5.3.1.1 gra()	48
5.3.1.2 kontynuuj()	48
5.3.1.3 menu()	48
5.3.1.4 nowa()	48
5.3.1.5 poprawnosc()	49
5.3.1.6 rozmiarPlanszy()	49
5.3.1.7 wielkosc()	49
5.3.1.8 wpiszWsp()	50
5.3.1.9 wspolrzedne()	50
5.3.1.10 zasady()	50
5.4 Dokumentacja pliku gra.cpp	50
5.4.1 Dokumentacja funkcji	51
5.4.1.1 gra()	51
5.4.1.2 kontynuuj()	51
5.4.1.3 menu()	51
5.4.1.4 nowa()	52
5.4.1.5 poprawnosc()	52
5.4.1.6 rozmiarPlanszy()	52
5.4.1.7 wielkosc()	52
5.4.1.8 wpiszWsp()	53
5.4.1.9 wspolrzedne()	53
5.4.1.10 zasady()	53
5.4.2 Dokumentacja zmiennych	53
5.4.2.1 Hist	53
5.4.2.2 ustawienia	54
5.5 Dokumentacja pliku main.cpp	54
5.5.1 Dokumentacja funkcji	54
5.5.1.1 main()	54
5.6 Dokumentacja pliku statki.cpp	55
5.7 Dokumentacja pliku statki.h	55
<b>Indeks</b>	<b>57</b>



# Rozdział 1

## Indeks hierarchiczny

### 1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

el_historia . . . . .	7
el_maszt . . . . .	9
el_ranking . . . . .	10
historia . . . . .	19
konfiguracja . . . . .	21
maszt . . . . .	25
el_statek . . . . .	12
plansza . . . . .	30
gracz . . . . .	14
punkty . . . . .	32
gracz . . . . .	14
ranking . . . . .	37
statek . . . . .	40
gracz . . . . .	14



## Rozdział 2

# Indeks klas

### 2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

el_historia	7
el_maszt	9
el_ranking	10
el_statek	12
gracz	14
historia	19
konfiguracja	21
maszt	25
plansza	30
punkty	32
ranking	37
statek	40



## Rozdział 3

# Indeks plików

### 3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

<a href="#">config.cpp</a>	45
<a href="#">config.h</a>	45
<a href="#">funkcje.h</a>	47
<a href="#">gra.cpp</a>	50
<a href="#">main.cpp</a>	54
<a href="#">statki.cpp</a>	55
<a href="#">statki.h</a>	55



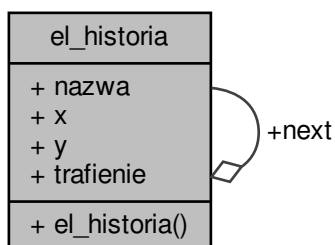
## Rozdział 4

# Dokumentacja klas

### 4.1 Dokumentacja struktury el\_historia

```
#include <config.h>
```

Diagram współpracy dla el\_historia:



#### Metody publiczne

- `el_historia` (`std::string _n`, `int _x`, `int _y`, `bool _t`, `el_historia *n`)

#### Atrybuty publiczne

- `std::string nazwa`
- `int x`
- `int y`
- `bool trafienie`
- `el_historia * next`

### 4.1.1 Dokumentacja konstruktora i destruktora

#### 4.1.1.1 el\_historia()

```
el_historia::el_historia (
    std::string _n,
    int _x,
    int _y,
    bool _t,
    el_historia * n ) [inline]
```

Konstruktor wieloargumentowy struktury elementu historii

### 4.1.2 Dokumentacja atrybutów składowych

#### 4.1.2.1 nazwa

```
std::string el_historia::nazwa
```

nazwa gracza, który wykonał ruch

#### 4.1.2.2 next

```
el_historia* el_historia::next
```

wskaźnik na kolejny element listy historii

#### 4.1.2.3 trafienie

```
bool el_historia::trafienie
```

informacja, czy gracz trafił w tym ruchu

#### 4.1.2.4 x

```
int el_historia::x
```

współrzędne wprowadzone przez gracza



## 4.1.2.5 y

```
int el_historia::y
```

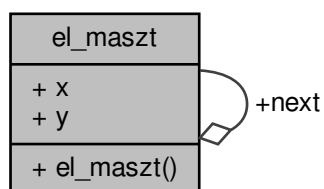
Dokumentacja dla tej struktury została wygenerowana z pliku:

- [config.h](#)

## 4.2 Dokumentacja struktury el\_maszt

```
#include <statki.h>
```

Diagram współpracy dla el\_maszt:



### Metody publiczne

- [el\\_maszt](#) (int \_x, int \_y, [el\\_maszt](#) \*n)

### Atrybuty publiczne

- int [x](#)
- int [y](#)
- [el\\_maszt](#) \* [next](#)

### 4.2.1 Dokumentacja konstruktora i destruktora

#### 4.2.1.1 el\_maszt()

```
el_maszt::el_maszt (
    int _x,
    int _y,
    el_maszt * n ) [inline]
```

Konstruktor wieloargumentowy struktury elementu masztu

## 4.2.2 Dokumentacja atrybutów składowych

### 4.2.2.1 next

```
el_maszt* el_maszt::next
```

wskaźnik na kolejny element listy masztów

### 4.2.2.2 x

```
int el_maszt::x
```

współrzędne masztu

### 4.2.2.3 y

```
int el_maszt::y
```

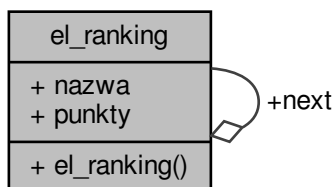
Dokumentacja dla tej struktury została wygenerowana z pliku:

- [statki.h](#)

## 4.3 Dokumentacja struktury el\_ranking

```
#include <config.h>
```

Diagram współpracy dla el\_ranking:



### Metody publiczne

- `el_ranking` (`std::string n`, `int p`, `el_ranking *ptr`)

### Atrybuty publiczne

- `std::string nazwa`
- `int punkty`
- `el_ranking * next`

## 4.3.1 Dokumentacja konstruktora i destruktora

### 4.3.1.1 `el_ranking()`

```
el_ranking::el_ranking (
    std::string n,
    int p,
    el_ranking * ptr ) [inline]
```

Konstruktor wieloargumentowy struktury elementu rankingu

## 4.3.2 Dokumentacja atrybutów składowych

### 4.3.2.1 `nazwa`

```
std::string el_ranking::nazwa
```

nazwa gracza

### 4.3.2.2 `next`

```
el_ranking* el_ranking::next
```

wskaźnik na kolejny element listy rankingu

### 4.3.2.3 `punkty`

```
int el_ranking::punkty
```

ilość punktów gracza

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [config.h](#)

## 4.4 Dokumentacja struktury el\_statek

```
#include <statki.h>
```

Diagram dziedziczenia dla el\_statek

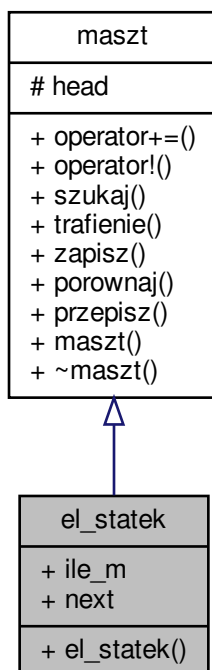
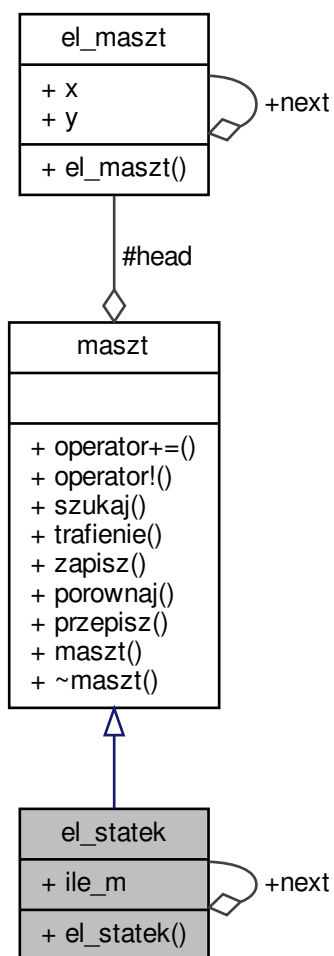


Diagram współpracy dla el\_statek:



### Metody publiczne

- `el_statek` (int i, `el_statek` \*n)

### Atrybuty publiczne

- int `ile_m`
- `el_statek` \* `next`

### Dodatkowe Dziedziczone Składowe

#### 4.4.1 Dokumentacja konstruktora i destruktora

#### 4.4.1.1 `el_statek()`

```
el_statek::el_statek (
    int i,
    el_statek * n ) [inline]
```

Konstruktor wieloargumentowy struktury elementu statku

### 4.4.2 Dokumentacja atrybutów składowych

#### 4.4.2.1 `ile_m`

```
int el_statek::ile_m
```

ilość masztów danego statku

#### 4.4.2.2 `next`

```
el_statek* el_statek::next
```

wskaźnik na kolejny element listy statków

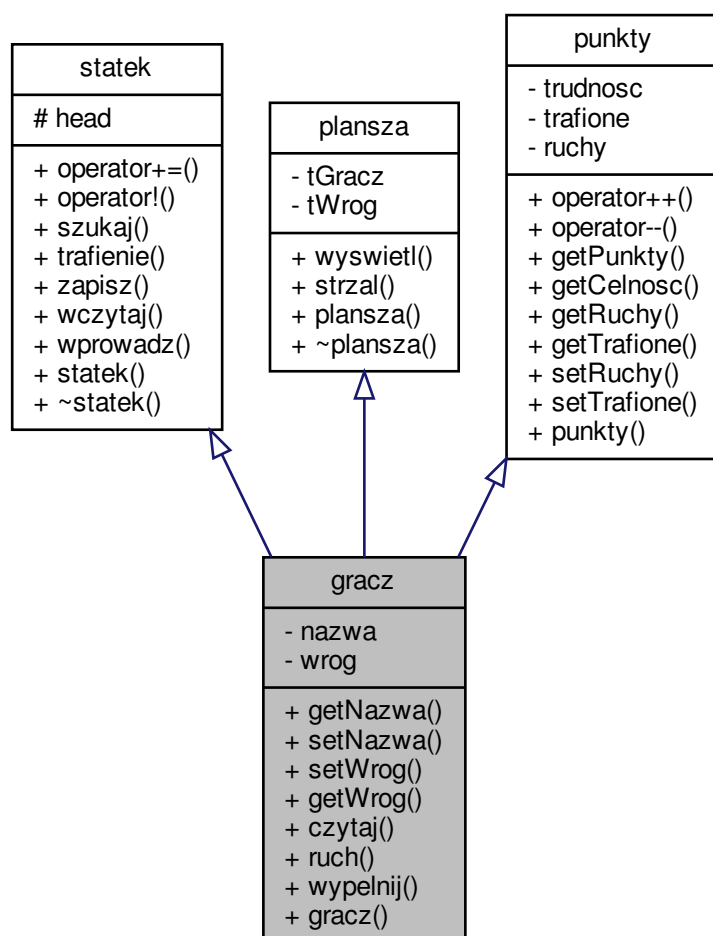
Dokumentacja dla tej struktury została wygenerowana z pliku:

- [statki.h](#)

## 4.5 Dokumentacja klasy `gracz`

```
#include <statki.h>
```

Diagram dziedziczenia dla gracz







- bool `ruch` ()
- void `wypelnij` ()
- `gracz` ()

### Atrybuty prywatne

- std::string `nazwa`
- `gracz` \* `wrog`

## Dodatkowe Dziedziczone Składowe

### 4.5.1 Dokumentacja konstruktora i destruktora

#### 4.5.1.1 `gracz()`

```
gracz::gracz ( ) [inline]
```

Konstruktor bezargumentowy klasy gracz

### 4.5.2 Dokumentacja funkcji składowych

#### 4.5.2.1 `czytaj()`

```
void gracz::czytaj (
    std::ifstream & plik )
```

Metoda wczytuje z pliku informacje o graczu oraz ustawieniu statków

#### Parametry

<i>plik</i>	strumień, z którego czytamy
-------------	-----------------------------

#### 4.5.2.2 `getNazwa()`

```
std::string gracz::getNazwa ( ) [inline]
```

Metoda zwraca nazwę gracza

#### 4.5.2.3 getWrog()

```
gracz* gracz::getWrog ( ) [inline]
```

Metoda zwraca wskaźnik na przeciwnika

#### 4.5.2.4 ruch()

```
bool gracz::ruch ( )
```

Metoda obsługuje ruch w grze - pobiera współrzędne, sprawdza czy było trafienie, zaznacza na planszy strzał, zlicza punkty oraz sprawdza czy rozgrywka się zakończyła

**Zwraca**

true jeżeli gramy dalej; false jeżeli któryś z graczy nie ma już statków

#### 4.5.2.5 setNazwa()

```
void gracz::setNazwa (
    std::string n ) [inline]
```

Metoda ustawia nazwę gracza

#### 4.5.2.6 setWrog()

```
void gracz::setWrog (
    gracz *& g ) [inline]
```

Metoda ustawia wskaźnik na przeciwnika

#### 4.5.2.7 wypelnij()

```
void gracz::wypelnij ( )
```

Metoda obsługuje wypełnianie planszy statkami

### 4.5.3 Dokumentacja atrybutów składowych

#### 4.5.3.1 nazwa

```
std::string gracz::nazwa [private]
```

**nazwa** gracza

## 4.5.3.2 wrog

```
gracz* gracz::wrog [private]
```

wskaźnik na przeciwnika

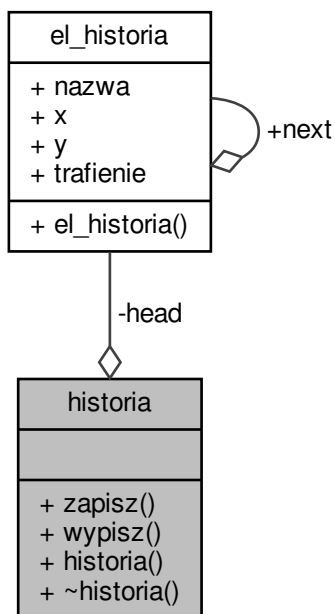
Dokumentacja dla tej klasy została wygenerowana z plików:

- [statki.h](#)
- [statki.cpp](#)

## 4.6 Dokumentacja klasy historia

```
#include <config.h>
```

Diagram współpracy dla historia:



## Metody publiczne

- void `zapisz` (std::string nazwa, std::pair< int, int > wsp, bool traf)
- void `wypisz` (int i)
- `historia` ()
- `~historia` ()

## Atrybuty prywatne

- `el_historia` \* `head`

## 4.6.1 Dokumentacja konstruktora i destruktora

### 4.6.1.1 historia()

```
historia::historia ( ) [inline]
```

Konstruktor bezargumentowy klasy historia

### 4.6.1.2 ~historia()

```
historia::~~historia ( )
```

Destruktor klasy historia

## 4.6.2 Dokumentacja funkcji składowych

### 4.6.2.1 wypisz()

```
void historia::wypisz (
    int i )
```

Metoda wypisuje uprzednie ruchy graczy

#### Parametry

<i>i</i>	ilość ruchów do wyświetlenia
----------	------------------------------

### 4.6.2.2 zapisz()

```
void historia::zapisz (
    std::string nazwa,
    std::pair< int, int > wsp,
    bool traf )
```

Metoda dodaje element na początek listy historii

## Parametry

<i>nazwa</i>	nazwa gracza, który wykonał ruch
<i>wsp</i>	współrzędne ruchu wykonanego przez gracza
<i>traf</i>	informacja czy gracz trafił w maszt podczas wykonywania ruchu

## 4.6.3 Dokumentacja atrybutów składowych

## 4.6.3.1 head

```
el_historia* historia::head [private]
```

wskaźnik na głowę listy historii

Dokumentacja dla tej klasy została wygenerowana z plików:

- [config.h](#)
- [config.cpp](#)

## 4.7 Dokumentacja klasy konfiguracja

```
#include <config.h>
```

Diagram współpracy dla konfiguracja:

konfiguracja
- wiersze - kolumny - tab - ileStatkow
+ getWiersze() + getKolumny() + setWiersze() + setKolumny() + getElement() + getIle() + wczytaj() + utworzListe() + konfiguracja() + ~konfiguracja()

## Metody publiczne

- `int getWiersze ()`
- `int getKolumny ()`
- `void setWiersze (int n)`
- `void setKolumny (int n)`
- `int getElement (int i)`
- `int getIle ()`
- `void wczytaj (std::ifstream &plik)`
- `void utworzListe ()`
- `konfiguracja ()`
- `~konfiguracja ()`

## Atrybuty prywatne

- `int wiersze = 10`
- `int kolumny = 10`
- `int * tab`
- `int ileStatkow = 6`

### 4.7.1 Dokumentacja konstruktora i destruktora

#### 4.7.1.1 konfiguracja()

```
konfiguracja::konfiguracja ( )
```

Konstruktor bezargumentowy klasy konfiguracja \*

#### 4.7.1.2 ~konfiguracja()

```
konfiguracja::~~konfiguracja ( )
```

Destruktor klasy konfiguracja \*

### 4.7.2 Dokumentacja funkcji składowych

#### 4.7.2.1 getElement()

```
int konfiguracja::getElement (
    int i ) [inline]
```

Metoda zwraca wartość elementu spod danego indeksu tablicy

#### 4.7.2.2 `getIle()`

```
int konfiguracja::getIle ( ) [inline]
```

Metoda zwraca informację o ilości statków

#### 4.7.2.3 `getKolumny()`

```
int konfiguracja::getKolumny ( ) [inline]
```

Metoda zwraca ilość kolumn

#### 4.7.2.4 `getWiersze()`

```
int konfiguracja::getWiersze ( ) [inline]
```

Metoda zwraca ilość wierszy

#### 4.7.2.5 `setKolumny()`

```
void konfiguracja::setKolumny (
    int n ) [inline]
```

Metoda pozwala na ustawienie liczby kolumn

#### 4.7.2.6 `setWiersze()`

```
void konfiguracja::setWiersze (
    int n ) [inline]
```

Metoda pozwala na ustawienie liczby wierszy

#### 4.7.2.7 `utworzListe()`

```
void konfiguracja::utworzListe ( )
```

Metoda tworzy jednowymiarową tablicę przechowującą ilości masztów kolejnych statków \*

#### 4.7.2.8 `wczytaj()`

```
void konfiguracja::wczytaj (
    std::ifstream & plik )
```

Metoda wczytuje z pliku informacje o konfiguracji

## Parametry

<i>plik</i>	strumień, z którego czytamy
-------------	-----------------------------

### 4.7.3 Dokumentacja atrybutów składowych

#### 4.7.3.1 ileStatkow

```
int konfiguracja::ileStatkow = 6 [private]
```

ilość statków w rozgrywce

#### 4.7.3.2 kolumny

```
int konfiguracja::kolumny = 10 [private]
```

ilość kolumn planszy

#### 4.7.3.3 tab

```
int* konfiguracja::tab [private]
```

tabela z ilościami masztów kolejnych statków

#### 4.7.3.4 wiersze

```
int konfiguracja::wiersze = 10 [private]
```

ilość wierszy planszy

Dokumentacja dla tej klasy została wygenerowana z plików:

- [config.h](#)
- [config.cpp](#)



## 4.8 Dokumentacja klasy maszt

```
#include <statki.h>
```

Diagram dziedziczenia dla maszt

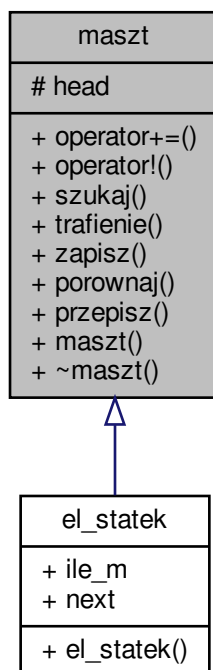
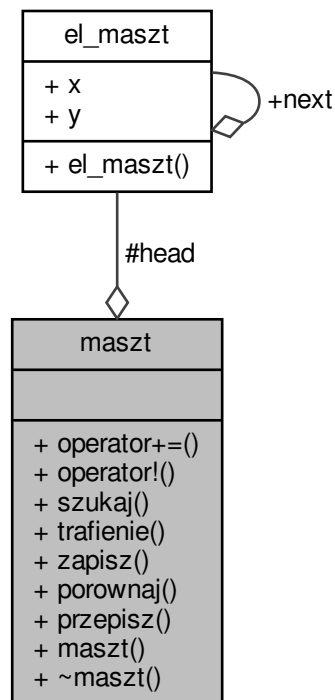


Diagram współpracy dla maszt:



## Metody publiczne

- `maszt & operator+= (std::pair< int, int > wsp)`
- `bool operator! ()`
- `bool szukaj (std::pair< int, int > wsp)`
- `bool trafienie (std::pair< int, int > wsp)`
- `void zapisz (std::ofstream &plik)`
- `bool porownaj (el_statek *&istniejace)`
- `void przepisz (el_statek *&temp)`
- `maszt ()`
- `~maszt ()`

## Atrybuty chronione

- `el_maszt * head`

### 4.8.1 Dokumentacja konstruktora i destruktor

#### 4.8.1.1 maszt()

```
maszt::maszt ( ) [inline]
```

Konstruktor bezargumentowy klasy maszt

#### 4.8.1.2 ~maszt()

```
maszt::~~maszt ( )
```

Destruktor klasy maszt

### 4.8.2 Dokumentacja funkcji składowych

#### 4.8.2.1 operator!()

```
bool maszt::operator! ( )
```

Metoda sprawdza czy istnieje lista masztów

**Zwraca**

true jeśli istnieje wskaźnik na głowę listy masztów; false jeśli ten wskaźnik to nullptr

#### 4.8.2.2 operator+=( )

```
maszt & maszt::operator+= (
    std::pair< int, int > wsp )
```

Metoda dodaje nowy maszt na początek listy masztów

**Parametry**

wsp	współrządne masztu
-----	--------------------

**Zwraca**

referencja do obiektu maszt

#### 4.8.2.3 porównaj()

```
bool maszt::porównaj (
    el_statek *& istnieje )
```

Metoda porównuje tymczasową listę masztów z istniejącymi masztami

##### Parametry

<i>istnieje</i>	lista istniejących już statków
-----------------	--------------------------------

##### Zwraca

true jeżeli znaleziono powtarzające się maszty; false jeśli maszty się nie powtarzają

#### 4.8.2.4 przepisz()

```
void maszt::przepisz (
    el_statek *& temp )
```

Metoda przepisuje tymczasową listę masztów do listy istniejącego statku

##### Parametry

<i>temp</i>	wskaźnik na tymczasową listę masztów
-------------	--------------------------------------

#### 4.8.2.5 szukaj()

```
bool maszt::szukaj (
    std::pair< int, int > wsp )
```

Metoda przeszukuje listę statków w poszukiwaniu masztu o podanych współrzędnych

##### Parametry

<i>wsp</i>	współrzędne szukanego masztu
------------	------------------------------

##### Zwraca

true jeżeli znaleziono maszt o podanych współrzędnych; false jeżeli nie znaleziono takiego masztu

#### 4.8.2.6 trafienie()

```
bool maszt::trafienie (
    std::pair< int, int > wsp )
```

Metoda przeszukuje listę masztów w poszukiwaniu masztu o podanych współrzędnych i usuwa go

##### Parametry

<i>wsp</i>	współrzędne szukanego masztu
------------	------------------------------

##### Zwraca

true jeżeli znaleziono maszt o podanych współrzędnych i go usunięto; false jeżeli nie znaleziono takiego masztu

#### 4.8.2.7 zapisz()

```
void maszt::zapisz (
    std::ofstream & plik )
```

Metoda zapisuje maszty z listy do pliku

##### Parametry

<i>plik</i>	strumień, do którego zapisujemy
-------------	---------------------------------

### 4.8.3 Dokumentacja atrybutów składowych

#### 4.8.3.1 head

```
el_maszt* maszt::head [protected]
```

wskaźnik na głowe listy masztów

Dokumentacja dla tej klasy została wygenerowana z plików:

- [statki.h](#)
- [statki.cpp](#)

## 4.9 Dokumentacja klasy plansza

```
#include <statki.h>
```

Diagram dziedziczenia dla plansza

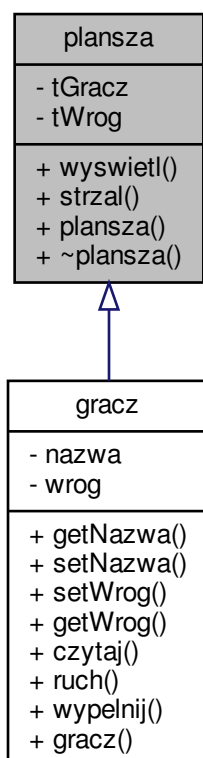
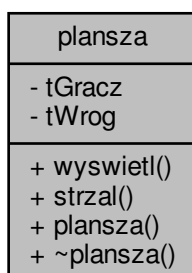


Diagram współpracy dla plansza:



## Metody publiczne

- void `wyswietl` ()
- void `strzal` (std::pair< int, int > *wsp*, char *c*, char *kto*)
- `plansza` ()
- `~plansza` ()

## Atrybuty prywatne

- char \*\* `tGracz`
- char \*\* `tWrog`

### 4.9.1 Dokumentacja konstruktora i destruktora

#### 4.9.1.1 plansza()

```
plansza::plansza ( )
```

Konstruktor bezargumentowy klasy plansza

#### 4.9.1.2 ~plansza()

```
plansza::~~plansza ( )
```

Destruktor klasy plansza

### 4.9.2 Dokumentacja funkcji składowych

#### 4.9.2.1 strzal()

```
void plansza::strzal (
    std::pair< int, int > wsp,
    char c,
    char kto )
```

Metoda zaznacza na planszy strzał gracza

#### Parametry

<i>wsp</i>	współrzedne strzału
<i>c</i>	symbol wstawiany do planszy
<i>kto</i>	symbol wskazujący na planszę, do której wstawiamy strzał

#### 4.9.2.2 wyswietl()

```
void plansza::wyswietl ( )
```

Metoda wyświetla plansze gracza i jego przeciwnika

### 4.9.3 Dokumentacja atrybutów składowych

#### 4.9.3.1 tGracz

```
char** plansza::tGracz [private]
```

plansza gracza

#### 4.9.3.2 tWrog

```
char** plansza::tWrog [private]
```

pierwotnie pusta plansza wypełnianiana strzałami gracza

Dokumentacja dla tej klasy została wygenerowana z plików:

- [statki.h](#)
- [statki.cpp](#)

### 4.10 Dokumentacja klasy punkty

```
#include <statki.h>
```



Diagram dziedziczenia dla punkty

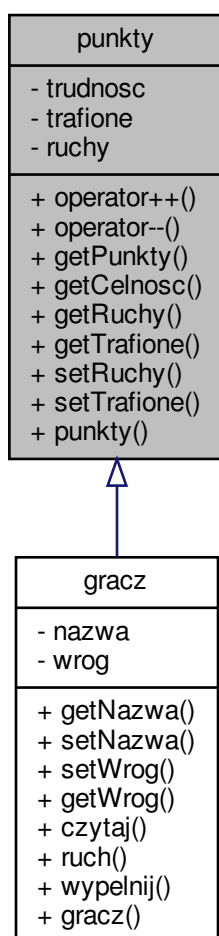


Diagram współpracy dla punkty:

punkty
- trudnosc - trafione - ruchy
+ operator++() + operator--() + getPunkty() + getCelnosc() + getRuchy() + getTrafione() + setRuchy() + setTrafione() + punkty()

### Metody publiczne

- `punkty & operator++ ()`
- `punkty & operator-- ()`
- `double getPunkty ()`
- `double getCelnosc ()`
- `int getRuchy ()`
- `int getTrafione ()`
- `void setRuchy (int n)`
- `void setTrafione (int n)`
- `punkty ()`

### Atrybuty prywatne

- `double trudnosc`
- `int trafione`
- `int ruchy`

## 4.10.1 Dokumentacja konstruktora i destruktor

### 4.10.1.1 punkty()

```
punkty::punkty ( )
```

Konstruktor bezargumentowy klasy punkty

## 4.10.2 Dokumentacja funkcji składowych

### 4.10.2.1 getCelnosc()

```
double punkty::getCelnosc ( )
```

Metoda oblicza i zwraca współczynnik celności

### 4.10.2.2 getPunkty()

```
double punkty::getPunkty ( ) [inline]
```

Metoda oblicza i zwraca punkty

### 4.10.2.3 getRuchy()

```
int punkty::getRuchy ( ) [inline]
```

Metoda zwraca liczbę ruchów gracza

### 4.10.2.4 getTrafione()

```
int punkty::getTrafione ( ) [inline]
```

Metoda zwraca liczbę trafień gracza

### 4.10.2.5 operator++()

```
punkty & punkty::operator++ ( )
```

Metoda dodaje jeden do licznika trafionych statków (ponieważ każde trafienie zwiększa współczynnik celności)

### 4.10.2.6 operator--()

```
punkty & punkty::operator-- ( )
```

Metoda dodaje jeden do licznika wykonanych ruchów (ponieważ każdy ruch zmniejsza współczynnik celności)

### 4.10.2.7 setRuchy()

```
void punkty::setRuchy (
    int n ) [inline]
```

Metoda ustawia ilość ruchów gracza

#### 4.10.2.8 setTrafione()

```
void punkty::setTrafione (
    int n ) [inline]
```

Metoda ustawia ilość trafień gracza

### 4.10.3 Dokumentacja atrybutów składowych

#### 4.10.3.1 ruchy

```
int punkty::ruchy [private]
```

ilość wykonanych ruchów

#### 4.10.3.2 trafione

```
int punkty::trafione [private]
```

ilość trafień gracza

#### 4.10.3.3 trudnosc

```
double punkty::trudnosc [private]
```

współczynnik trudności gry obliczany z rownania:  $100 * ( \text{ilość\_masztów} / \text{powierzchnia\_planszy} )$

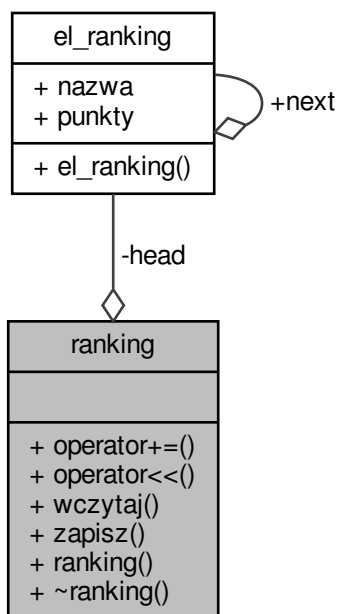
Dokumentacja dla tej klasy została wygenerowana z plików:

- [statki.h](#)
- [statki.cpp](#)

## 4.11 Dokumentacja klasy ranking

```
#include <config.h>
```

Diagram współpracy dla ranking:



### Metody publiczne

- `ranking` & `operator+=` (`std::pair< std::string, int > wpis`)
- `ranking` & `operator<<` (`std::pair< std::string, int > wpis`)
- `void wczytaj ()`
- `void zapisz ()`
- `ranking ()`
- `~ranking ()`

### Atrybuty prywatne

- `el_ranking * head`

#### 4.11.1 Dokumentacja konstruktora i destruktor

#### 4.11.1.1 ranking()

```
ranking::ranking ( ) [inline]
```

Konstruktor bezargumentowy klasy ranking

#### 4.11.1.2 ~ranking()

```
ranking::~~ranking ( )
```

Destruktor klasy ranking

### 4.11.2 Dokumentacja funkcji składowych

#### 4.11.2.1 operator+=( )

```
ranking & ranking::operator+= (
    std::pair< std::string, int > wpis )
```

Metoda dodaje wpis rankingu na koniec jednokierunkowej listy rankingowej

##### Parametry

<i>wpis</i>	nazwa gracza oraz ilość jego punktów
-------------	--------------------------------------

##### Zwraca

referencja do obiektu ranking

#### 4.11.2.2 operator<<()

```
ranking & ranking::operator<< (
    std::pair< std::string, int > wpis )
```

Metoda dodaje wpis rankingu w posortowanej kolejności

##### Parametry

<i>wpis</i>	nazwa gracza oraz ilość jego punktów
-------------	--------------------------------------

Zwraca

referencja do obiektu ranking

#### 4.11.2.3 wczytaj()

```
void ranking::wczytaj ( )
```

Metoda wyświetla ranking, odczytując go z pliku

#### 4.11.2.4 zapisz()

```
void ranking::zapisz ( )
```

Metoda zapisuje ranking do pliku

### 4.11.3 Dokumentacja atrybutów składowych

#### 4.11.3.1 head

```
el_ranking* ranking::head [private]
```

wkaźnik na głowę listy rankingowej

Dokumentacja dla tej klasy została wygenerowana z plików:

- [config.h](#)
- [config.cpp](#)

## 4.12 Dokumentacja klasy statek

```
#include <statki.h>
```

Diagram dziedziczenia dla statek

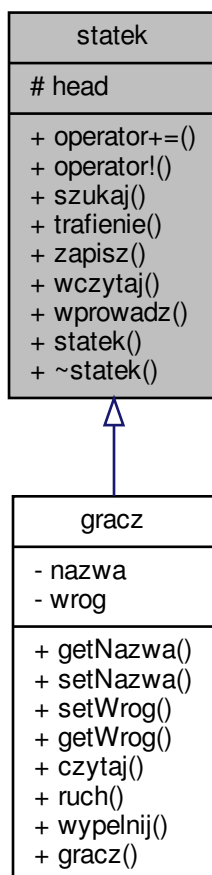
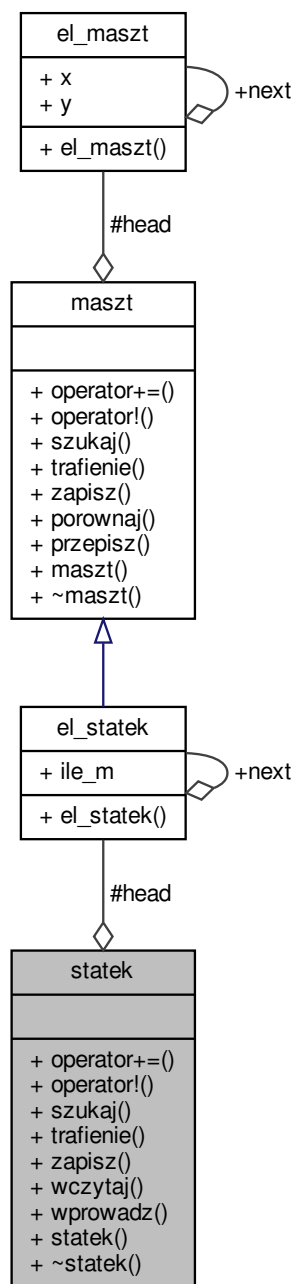




Diagram współpracy dla statek:



### Metody publiczne

- `statek & operator+= (int ile)`
- `bool operator! ()`
- `bool szukaj (std::pair< int, int > wsp)`
- `bool trafienie (std::pair< int, int > wsp)`
- `void zapisz (std::ofstream &plik)`

- void `wczytaj` (std::pair< int, int > wsp)
- void `wprowadz` ()
- `statek` ()
- `~statek` ()

### Atrybuty chronione

- `el_statek` \* `head`

## 4.12.1 Dokumentacja konstruktora i destruktora

### 4.12.1.1 `statek()`

```
statek::statek ( ) [inline]
```

Konstruktor bezargumentowy klasy `statek`

### 4.12.1.2 `~statek()`

```
statek::~~statek ( )
```

Destruktor klasy `statek`

## 4.12.2 Dokumentacja funkcji składowych

### 4.12.2.1 `operator"!()`

```
bool statek::operator! ( )
```

Metoda sprawdza czy istnieje lista statków

Zwraca

true jeśli istnieje wskaźnik na głowę listy statków; false jeśli ten wskaźnik to nullptr

### 4.12.2.2 `operator+=(`

```
statek & statek::operator+= (
    int ile )
```

Metoda dodaje nowy statek na początek listy statków

**Parametry**

<i>ile</i>	ilość masztów statku
------------	----------------------

**Zwraca**

referencja do obiektu statek

**4.12.2.3 szukaj()**

```
bool statek::szukaj (
    std::pair< int, int > wsp )
```

Metoda przeszukuje listę statków w poszukiwaniu statku z masztem o podanych współrzędnych

**Parametry**

<i>wsp</i>	współrzędne szukanego masztu
------------	------------------------------

**Zwraca**

true jeżeli znaleziono maszt o podanych współrzędnych; false jeżeli nie znaleziono takiego masztu

**4.12.2.4 trafienie()**

```
bool statek::trafienie (
    std::pair< int, int > wsp )
```

Metoda przeszukuje listę statków w poszukiwaniu masztu o podanych współrzędnych i usuwa statek, w którym znaleziono ten maszt, jeśli był on ostatnim masztem

**Parametry**

<i>wsp</i>	współrzędne szukanego masztu
------------	------------------------------

**Zwraca**

true jeżeli znaleziono statek o podanych współrzędnych; false jeżeli nie znaleziono takiego masztu

#### 4.12.2.5 wczytaj()

```
void statek::wczytaj (
    std::pair< int, int > wsp )
```

Metoda wczytuje maszty z pliku do listy

Parametry

<i>wsp</i>	współrzędne masztu
------------	--------------------

#### 4.12.2.6 wprowadz()

```
void statek::wprowadz ( )
```

Metoda obsługuje wprowadzanie i kontrolę poprawności wprowadzania statków

#### 4.12.2.7 zapisz()

```
void statek::zapisz (
    std::ofstream & plik )
```

Metoda zapisuje statki z listy do pliku

Parametry

<i>plik</i>	strumień, do którego zapisujemy
-------------	---------------------------------

### 4.12.3 Dokumentacja atrybutów składowych

#### 4.12.3.1 head

```
el_statek* statek::head [protected]
```

wskaźnik na głowę listy statków

Dokumentacja dla tej klasy została wygenerowana z plików:

- [statki.h](#)
- [statki.cpp](#)

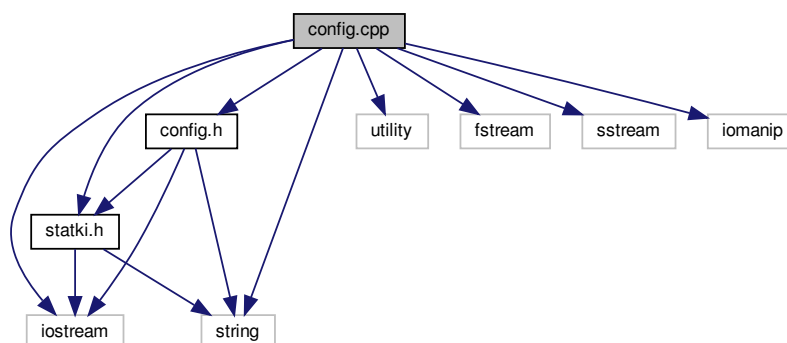
## Rozdział 5

# Dokumentacja plików

### 5.1 Dokumentacja pliku config.cpp

```
#include <iostream>
#include <utility>
#include <fstream>
#include <string>
#include <sstream>
#include <iomanip>
#include "config.h"
#include "statki.h"
```

Wykres zależności załączania dla config.cpp:

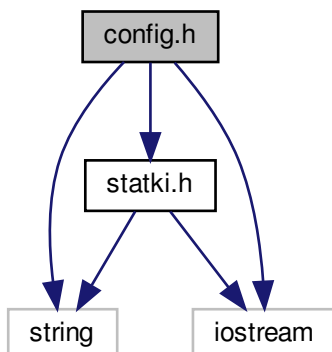


### 5.2 Dokumentacja pliku config.h

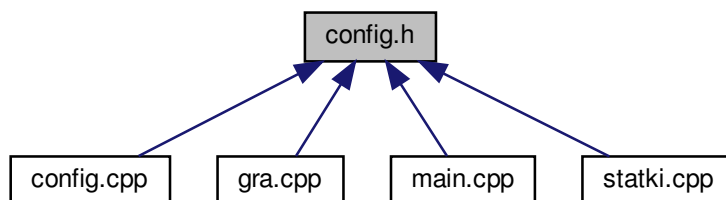
```
#include <string>
#include <iostream>
```

```
#include "statki.h"
```

Wykres zależności załączania dla config.h:



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



## Komponenty

- struct [el\\_ranking](#)
- class [ranking](#)
- struct [el\\_historia](#)
- class [historia](#)
- class [konfiguracja](#)

## Zmienne

- [konfiguracja ustawienia](#)
- [historia Hist](#)

### 5.2.1 Dokumentacja zmiennych

#### 5.2.1.1 Hist

`historia` Hist

Globalna zmienna przechowująca ustawienia

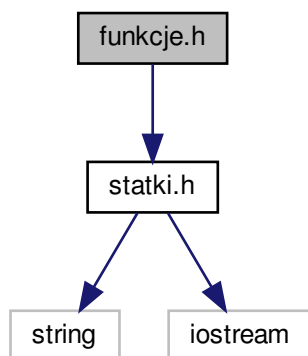
#### 5.2.1.2 ustawienia

`konfiguracja` ustawienia

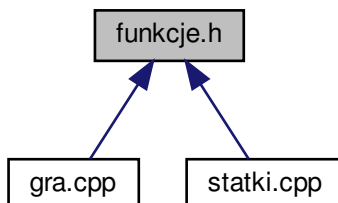
## 5.3 Dokumentacja pliku funkcje.h

```
#include "statki.h"
```

Wykres zależności załączania dla funkcje.h:



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



## Funkcje

- void `menu` ()
- void `rozmiarPlanszy` ()
- void `nowa` ()
- void `kontynuuj` ()
- void `gra` (`gracz` \*&g)
- std::pair< int, int > `wpiszWsp` ()
- std::pair< int, int > `wspolrzedne` ()
- bool `wielkosc` (int a, int b, int ile)
- bool `poprawnosc` (std::pair< int, int > w1, std::pair< int, int > w2, int ile)
- void `zasady` ()

### 5.3.1 Dokumentacja funkcji

#### 5.3.1.1 `gra()`

```
void gra (  
    gracz *& g )
```

Funkcja obsługuje rozgrywkę

##### Parametry

<code>g</code>	wskaźnik na gracza nr 1 (umożliwia dostęp do obiektu gracza 1 i gracza 2)
----------------	---

#### 5.3.1.2 `kontynuuj()`

```
void kontynuuj ( )
```

Funkcja odczytuje z pliku informacje o graczach i ich statkach, pochodzące z uprzednio rozpoczętej, nieskończonej i zapisanej rozgrywki

#### 5.3.1.3 `menu()`

```
void menu ( )
```

Funkcja wyświetla i obsługuje menu główne

#### 5.3.1.4 `nowa()`

```
void nowa ( )
```

Funkcja uruchamia nową grę - umożliwia stworzenie graczy i ustawienie statków na planszach



#### 5.3.1.5 poprawnosc()

```
bool poprawnosc (
    std::pair< int, int > w1,
    std::pair< int, int > w2,
    int ile )
```

Funkcja sprawdza czy wprowadzone maszty dzioba i rufy znajdują się w jednej linii oraz czy długość statku jest odpowiednia

##### Parametry

<i>w1</i>	współrzędne dzioba
<i>w2</i>	współrzędne rufy
<i>ile</i>	liczba masztów danego statku

##### Zwraca

true jeżeli wartości są poprawne; false jeżeli są niepoprawne

#### 5.3.1.6 rozmiarPlanszy()

```
void rozmiarPlanszy ( )
```

Funkcja pozwala użytkownikowi ustawić rozmiar planszy oraz ilość i wymiary statków

#### 5.3.1.7 wielkosc()

```
bool wielkosc (
    int a,
    int b,
    int ile )
```

Funkcja sprawdza czy odległość między współrzędnymi jest odpowiednia

##### Parametry

<i>a</i>	pierwsza współrzędna (x)
<i>b</i>	druga współrzędna (y)
<i>ile</i>	ilość masztów danego statku

##### Zwraca

true jeżeli współrzędne zapewniają odpowiednią ilość masztów; false jeżeli jej nie zapewniają

#### 5.3.1.8 wpiszWsp()

```
std::pair<int, int> wpiszWsp ( )
```

Funkcja obsługuje wpisywanie współrzędnych

**Zwraca**

typ para składający się z dwóch zmiennych typu int, symbolizujących współrzędną x oraz współrzędną y

#### 5.3.1.9 wspolrzedne()

```
std::pair<int, int> wspolrzedne ( )
```

Funkcja wczytuje współrzędne i kontroluje ich poprawność - czy nie wychodzą poza planszę

**Zwraca**

typ para składający się z dwóch zmiennych typu int, symbolizujących współrzędną x oraz współrzędną y

#### 5.3.1.10 zasady()

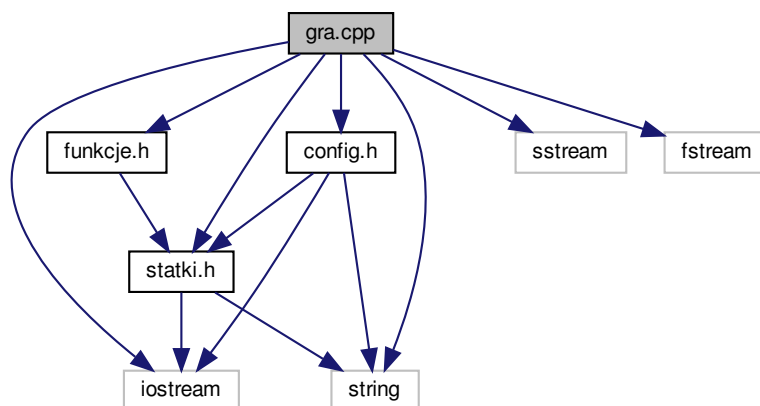
```
void zasady ( )
```

Funkcja wyświetla zasady i instrukcje gry

## 5.4 Dokumentacja pliku gra.cpp

```
#include <iostream>
#include <string>
#include <sstream>
#include <fstream>
#include "statki.h"
#include "config.h"
#include "funkcje.h"
```

Wykres zależności załączania dla gra.cpp:



## Funkcje

- void `menu` ()
- void `rozmiarPlanszy` ()
- std::pair< int, int > `wspolzedne` ()
- std::pair< int, int > `wpiszWsp` ()
- bool `wielkosc` (int a, int b, int ile)
- bool `poprawnosc` (std::pair< int, int > w1, std::pair< int, int > w2, int ile)
- void `gra` (`gracz` \*&g)
- void `nowa` ()
- void `kontynuuj` ()
- void `zasady` ()

## Zmienne

- konfiguracja ustawienia
- historia Hist

### 5.4.1 Dokumentacja funkcji

#### 5.4.1.1 `gra()`

```
void gra (
    gracz *& g )
```

Funkcja obsługuje rozgrywkę

##### Parametry

<code>g</code>	wskaźnik na gracza nr 1 (umożliwia dostęp do obiektu gracza 1 i gracza 2)
----------------	---

#### 5.4.1.2 `kontynuuj()`

```
void kontynuuj ( )
```

Funkcja odczytuje z pliku informacje o graczach i ich statkach, pochodzące z uprzednio rozpoczętej, nieskończonej i zapisanej rozgrywki

#### 5.4.1.3 `menu()`

```
void menu ( )
```

Funkcja wyświetla i obsługuje menu główne

#### 5.4.1.4 nowa()

```
void nowa ( )
```

Funkcja uruchamia nową grę - umożliwia stworzenie graczy i ustawienie statków na planszach

#### 5.4.1.5 poprawnosc()

```
bool poprawnosc (
    std::pair< int, int > w1,
    std::pair< int, int > w2,
    int ile )
```

Funkcja sprawdza czy wprowadzone maszty dzioba i rufy znajdują się w jednej linii oraz czy długość statku jest odpowiednia

##### Parametry

<i>w1</i>	współrzędne dzioba
<i>w2</i>	współrzędne rufy
<i>ile</i>	liczba masztów danego statku

##### Zwraca

true jeżeli wartości są poprawne; false jeżeli są niepoprawne

#### 5.4.1.6 rozmiarPlanszy()

```
void rozmiarPlanszy ( )
```

Funkcja pozwala użytkownikowi ustawić rozmiar planszy oraz ilość i wymiary statków

#### 5.4.1.7 wielkosc()

```
bool wielkosc (
    int a,
    int b,
    int ile )
```

Funkcja sprawdza czy odległość między współrzędnymi jest odpowiednia

##### Parametry

<i>a</i>	pierwsza współrzędna (x)
<i>b</i>	druga współrzędna (y)
<i>ile</i>	ilość masztów danego statku

**Zwraca**

true jeżeli współrzędne zapewniają odpowiednią ilość masztów; false jeżeli jej nie zapewniają

**5.4.1.8 wpiszWsp()**

```
std::pair<int, int> wpiszWsp ( )
```

Funkcja obsługuje wpisywanie współrzędnych

**Zwraca**

typ para składający się z dwóch zmiennych typu int, symbolizujących współrzędną x oraz współrzędną y

**5.4.1.9 wspolzedne()**

```
std::pair<int, int> wspolzedne ( )
```

Funkcja wczytuje współrzędne i kontroluje ich poprawność - czy nie wychodzą poza planszę

**Zwraca**

typ para składający się z dwóch zmiennych typu int, symbolizujących współrzędną x oraz współrzędną y

**5.4.1.10 zasady()**

```
void zasady ( )
```

Funkcja wyświetla zasady i instrukcje gry

**5.4.2 Dokumentacja zmiennych****5.4.2.1 Hist**

```
historia Hist
```

Globalna zmienna przechowująca ustawienia

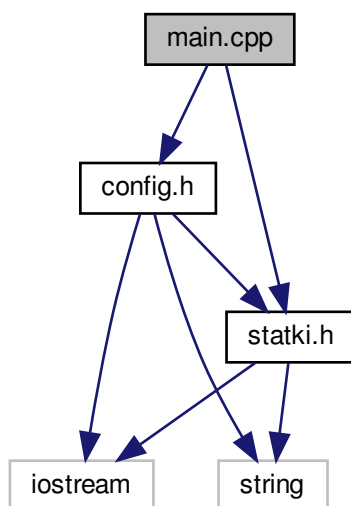
#### 5.4.2.2 ustawienia

`konfiguracja` ustawienia

## 5.5 Dokumentacja pliku main.cpp

```
#include "statki.h"  
#include "config.h"
```

Wykres zależności załączania dla main.cpp:



### Funkcje

- int `main` ()

#### 5.5.1 Dokumentacja funkcji

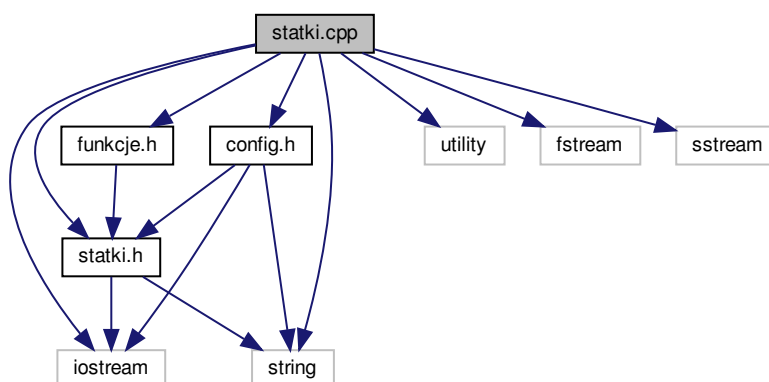
##### 5.5.1.1 main()

```
int main ( )
```

## 5.6 Dokumentacja pliku statki.cpp

```
#include <iostream>
#include <string>
#include <utility>
#include <fstream>
#include <sstream>
#include "statki.h"
#include "config.h"
#include "funkcje.h"
```

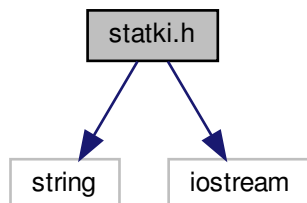
Wykres zależności załączania dla statki.cpp:



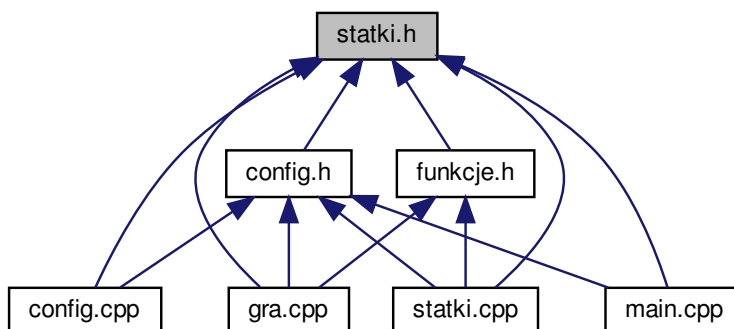
## 5.7 Dokumentacja pliku statki.h

```
#include <string>
#include <iostream>
```

Wykres zależności załączania dla statki.h:



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



## Komponenty

- struct [el\\_maszt](#)
- class [maszt](#)
- struct [el\\_statek](#)
- class [statek](#)
- class [plansza](#)
- class [punkty](#)
- class [gracz](#)



# Skorowidz

- ~historia
  - historia, [20](#)
- ~konfiguracja
  - konfiguracja, [22](#)
- ~maszt
  - maszt, [27](#)
- ~plansza
  - plansza, [31](#)
- ~ranking
  - ranking, [38](#)
- ~statek
  - statek, [42](#)
- config.cpp, [45](#)
- config.h, [45](#)
  - Hist, [47](#)
  - ustawienia, [47](#)
- czytaj
  - gracz, [17](#)
- el\_historia, [7](#)
  - el\_historia, [8](#)
  - nazwa, [8](#)
  - next, [8](#)
  - trafienie, [8](#)
  - x, [8](#)
  - y, [8](#)
- el\_maszt, [9](#)
  - el\_maszt, [9](#)
  - next, [10](#)
  - x, [10](#)
  - y, [10](#)
- el\_ranking, [10](#)
  - el\_ranking, [11](#)
  - nazwa, [11](#)
  - next, [11](#)
  - punkty, [11](#)
- el\_statek, [12](#)
  - el\_statek, [13](#)
  - ile\_m, [14](#)
  - next, [14](#)
- funkcje.h, [47](#)
  - gra, [48](#)
  - kontynuuj, [48](#)
  - menu, [48](#)
  - nowa, [48](#)
  - poprawnosc, [48](#)
  - rozmiarPlanszy, [49](#)
  - wielkosc, [49](#)
  - wpiszWsp, [49](#)
  - wspolrzedne, [50](#)
  - zasady, [50](#)
- getCelnosc
  - punkty, [35](#)
- getElement
  - konfiguracja, [22](#)
- getIle
  - konfiguracja, [22](#)
- getKolumny
  - konfiguracja, [23](#)
- getNazwa
  - gracz, [17](#)
- getPunkty
  - punkty, [35](#)
- getRuchy
  - punkty, [35](#)
- getTrafione
  - punkty, [35](#)
- getWiersze
  - konfiguracja, [23](#)
- getWrog
  - gracz, [17](#)
- gra
  - funkcje.h, [48](#)
  - gra.cpp, [51](#)
- gra.cpp, [50](#)
  - gra, [51](#)
  - Hist, [53](#)
  - kontynuuj, [51](#)
  - menu, [51](#)
  - nowa, [51](#)
  - poprawnosc, [52](#)
  - rozmiarPlanszy, [52](#)
  - ustawienia, [53](#)
  - wielkosc, [52](#)
  - wpiszWsp, [53](#)
  - wspolrzedne, [53](#)
  - zasady, [53](#)
- gracz, [14](#)
  - czytaj, [17](#)
  - getNazwa, [17](#)
  - getWrog, [17](#)
  - gracz, [17](#)
  - nazwa, [18](#)
  - ruch, [18](#)
  - setNazwa, [18](#)
  - setWrog, [18](#)
  - wrog, [18](#)

- wypelnij, 18
- head
  - historia, 21
  - maszt, 29
  - ranking, 39
  - statek, 44
- Hist
  - config.h, 47
  - gra.cpp, 53
- historia, 19
  - ~historia, 20
  - head, 21
  - historia, 20
  - wypisz, 20
  - zapisz, 20
- ile\_m
  - el\_statek, 14
- ileStatkow
  - konfiguracja, 24
- kolumny
  - konfiguracja, 24
- konfiguracja, 21
  - ~konfiguracja, 22
  - getElement, 22
  - getIle, 22
  - getKolumny, 23
  - getWiersze, 23
  - ileStatkow, 24
  - kolumny, 24
  - konfiguracja, 22
  - setKolumny, 23
  - setWiersze, 23
  - tab, 24
  - utworzListe, 23
  - wczytaj, 23
  - wiersze, 24
- kontynuuj
  - funkcje.h, 48
  - gra.cpp, 51
- main
  - main.cpp, 54
- main.cpp, 54
  - main, 54
- maszt, 25
  - ~maszt, 27
  - head, 29
  - maszt, 26
  - operator!, 27
  - operator+=", 27
  - porownaj, 27
  - przepisz, 28
  - szukaj, 28
  - trafienie, 28
  - zapisz, 29
- menu
  - funkcje.h, 48
  - gra.cpp, 51
- nazwa
  - el\_historia, 8
  - el\_ranking, 11
  - gracz, 18
- next
  - el\_historia, 8
  - el\_maszt, 10
  - el\_ranking, 11
  - el\_statek, 14
- nowa
  - funkcje.h, 48
  - gra.cpp, 51
- operator!
  - maszt, 27
  - statek, 42
- operator<<
  - ranking, 38
- operator++
  - punkty, 35
- operator+=
  - maszt, 27
  - ranking, 38
  - statek, 42
- operator--
  - punkty, 35
- plansza, 30
  - ~plansza, 31
  - plansza, 31
  - strzal, 31
  - tGracz, 32
  - tWrog, 32
  - wyswietl, 32
- poprawnosc
  - funkcje.h, 48
  - gra.cpp, 52
- porownaj
  - maszt, 27
- przepisz
  - maszt, 28
- punkty, 32
  - el\_ranking, 11
  - getCelnosc, 35
  - getPunkty, 35
  - getRuchy, 35
  - getTrafione, 35
  - operator++, 35
  - operator--, 35
  - punkty, 34
  - ruchy, 36
  - setRuchy, 35
  - setTrafione, 35
  - trafione, 36
  - trudnosc, 36

ranking, 37  
    ~ranking, 38  
    head, 39  
    operator<<, 38  
    operator+&, 38  
    ranking, 37  
    wczytaj, 39  
    zapisz, 39

rozmiarPlanszy  
    funkcje.h, 49  
    gra.cpp, 52

ruch  
    gracz, 18

ruchy  
    punkty, 36

setKolumny  
    konfiguracja, 23

setNazwa  
    gracz, 18

setRuchy  
    punkty, 35

setTrafione  
    punkty, 35

setWiersze  
    konfiguracja, 23

setWrog  
    gracz, 18

statek, 40  
    ~statek, 42  
    head, 44  
    operator!, 42  
    operator+&, 42  
    statek, 42  
    szukaj, 43  
    trafienie, 43  
    wczytaj, 43  
    wprowadz, 44  
    zapisz, 44

statki.cpp, 55

statki.h, 55

strzal  
    plansza, 31

szukaj  
    maszt, 28  
    statek, 43

tGracz  
    plansza, 32

tWrog  
    plansza, 32

tab  
    konfiguracja, 24

trafienie  
    el\_historia, 8  
    maszt, 28  
    statek, 43

trafione  
    punkty, 36

trudnosc  
    punkty, 36

ustawienia  
    config.h, 47  
    gra.cpp, 53

utworzListe  
    konfiguracja, 23

wczytaj  
    konfiguracja, 23  
    ranking, 39  
    statek, 43

wielkosc  
    funkcje.h, 49  
    gra.cpp, 52

wiersze  
    konfiguracja, 24

wpiszWsp  
    funkcje.h, 49  
    gra.cpp, 53

wprowadz  
    statek, 44

wrog  
    gracz, 18

wspolrzedne  
    funkcje.h, 50  
    gra.cpp, 53

wypelnij  
    gracz, 18

wypisz  
    historia, 20

wyswietl  
    plansza, 32

x  
    el\_historia, 8  
    el\_maszt, 10

y  
    el\_historia, 8  
    el\_maszt, 10

zapisz  
    historia, 20  
    maszt, 29  
    ranking, 39  
    statek, 44

zasady  
    funkcje.h, 50  
    gra.cpp, 53