# Projekt zaliczeniowy z PK3 Statki

Wygenerowano przez Doxygen 1.8.13

# Spis treści

# Rozdział 1

# Indeks hierarchiczny

## 1.1 Hierarchia klas

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

# Rozdział 2

# Indeks klas

## 2.1 Lista klas

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

# Rozdział 3

# Indeks plików

## 3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

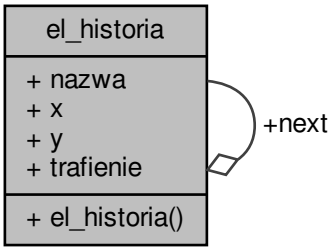
# Rozdział 4

# 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
```

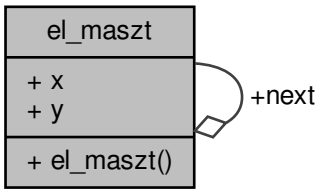
Dokumentacja dla tej struktury została wygenerowana z pliku:

- 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
```

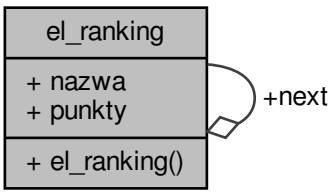
Dokumentacja dla tej struktury została wygenerowana z pliku:

- 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

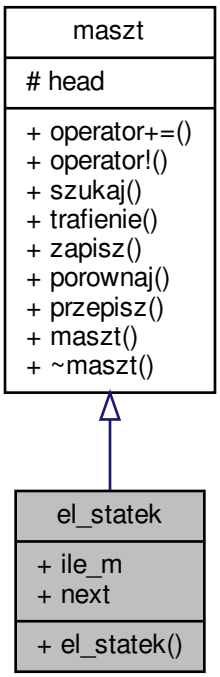
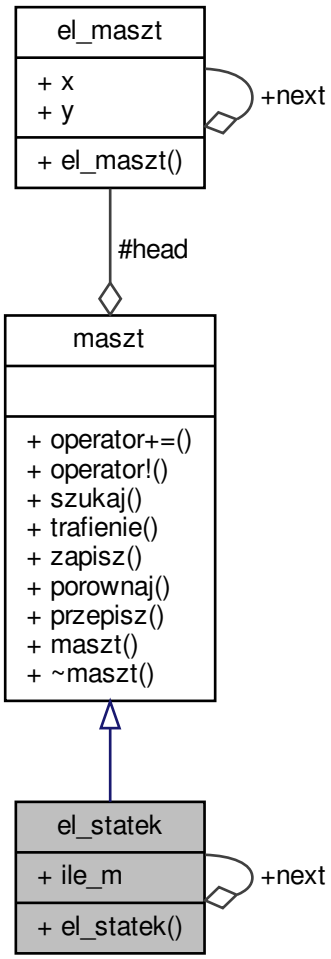


Diagram współpracy dla el_statek:




### 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

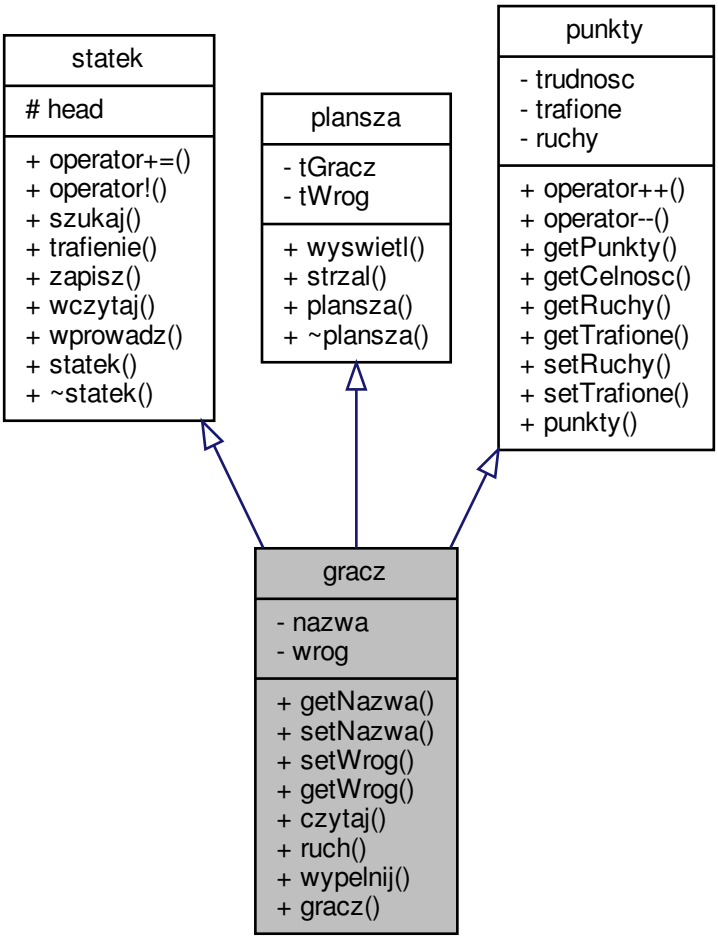
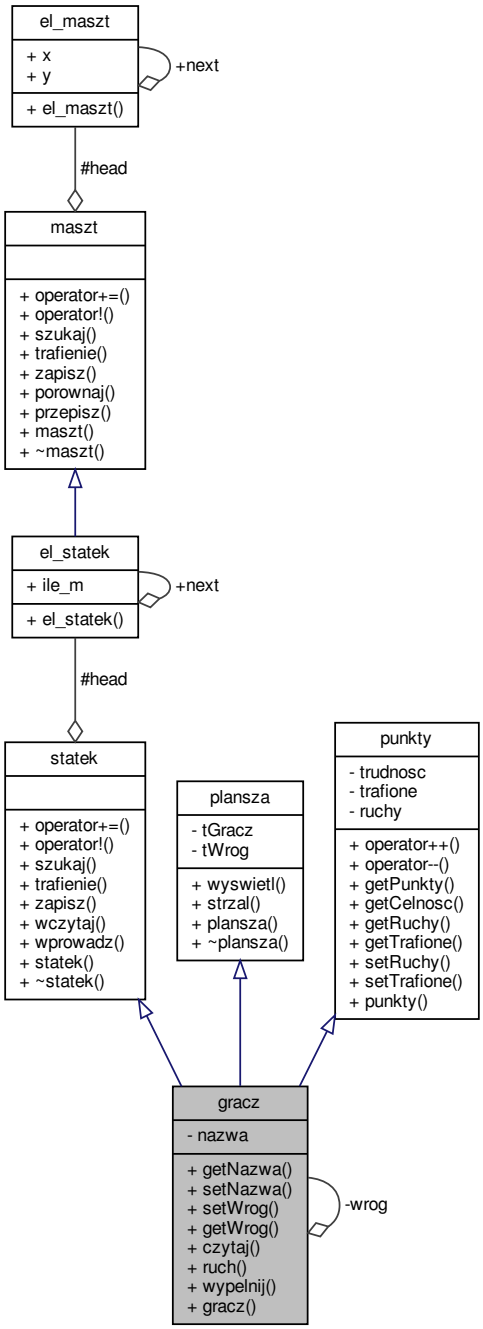


Diagram współpracy dla gracz:




## Metody publiczne

- std::string getNazwa ()
- void setNazwa (std::string n)
- void setWrog (gracz ∗&g)
- gracz ∗ getWrog ()
- void czytaj (std::ifstream &plik)

- 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**

| | |
|---|---|
| *plik* | 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

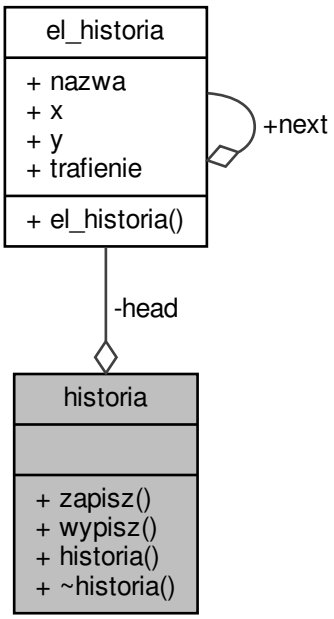
Dokumentacja dla tej klasy została wygenerowana z plików:

- 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* | 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**

| | |
|---|---|
| *nazwa* | nazwa gracza, który wykonał ruch |
| *wsp* | współrzędne ruchu wykonanego przez gracza |
| *traf* | 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:

### 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 infomacje o konfiguracji

**Parametry**

| | |
|---|---|
| *plik* | 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

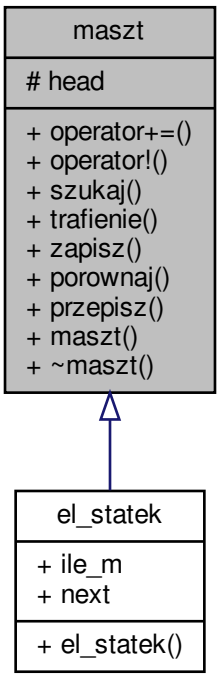
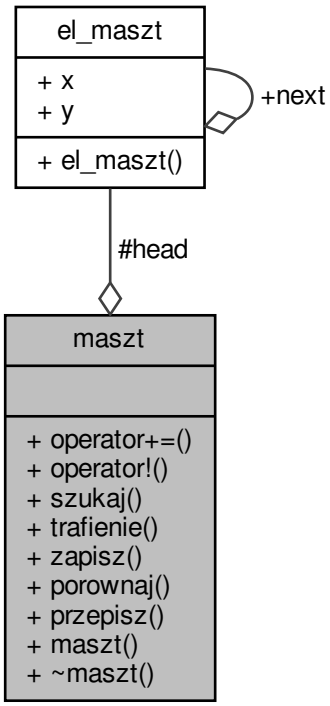


Diagram współpracy dla maszt:




### 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 destruktora

#### 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* | współrzędne masztu |

**Zwraca**

referencja do obiektu maszt

#### 4.8.2.3 porownaj()

```
bool maszt::porownaj (
            el_statek *& istniejace )
```

Metoda porównuje tymczasową listę masztów z istniejącymi masztami

**Parametry**

| | |
|---|---|
| *istniejace* | 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**

| | |
|---|---|
| *temp* | 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**

| | |
|---|---|
| *wsp* | współrzędne szukanego masztu |

**Zwraca**

true jeżeli znaleziono maszt o podanych współrzędnych; false jeżli 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**

| | |
|---|---|
| *wsp* | współrzędne szukanego masztu |

**Zwraca**

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

#### 4.8.2.7 zapisz()

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

Metoda zapisuje maszty z listy do pliku

**Parametry**

| | |
|---|---|
| *plik* | strumień, do którego zapisujemy |

### 4.8.3 Dokumentacja atrybutów składowych

#### 4.8.3.1 head

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

wskaźnik na glowe 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

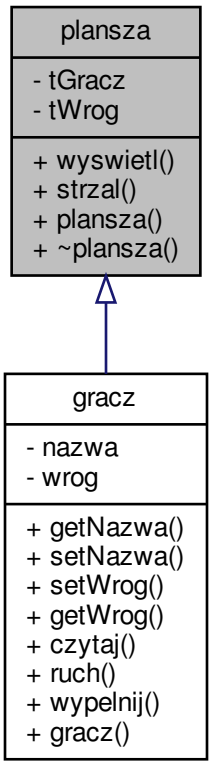
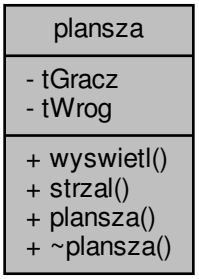



Diagram współpracy dla plansza:



### 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**

| | |
|---|---|
| *wsp* | współrzędne strzału |
| *c* | symbol wstawiany do planszy |
| *kto* | 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

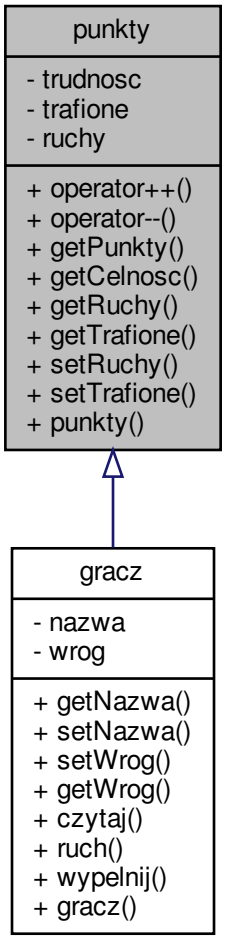
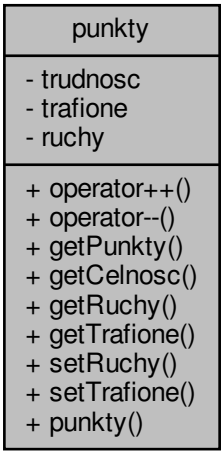


Diagram współpracy dla punkty:




### 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 destruktora

#### 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 ∗ ( ilość_masztów / powierzchnia_planszy )

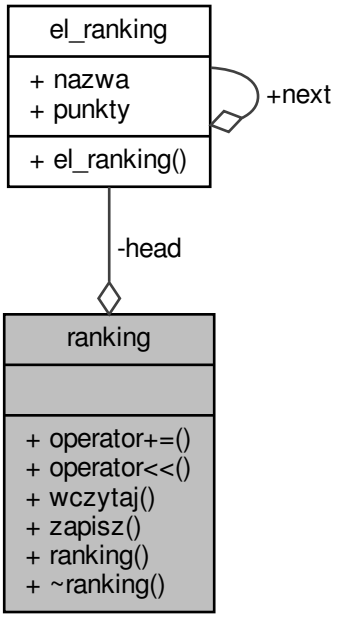
Dokumentacja dla tej klasy została wygenerowana z plików:

- 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 destruktora

#### 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**

| | |
|---|---|
| *wpis* | 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**

| | |
|---|---|
| *wpis* | 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

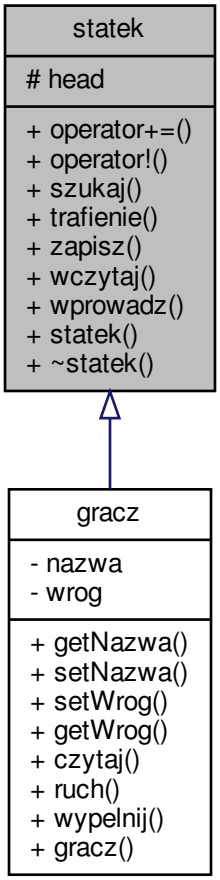
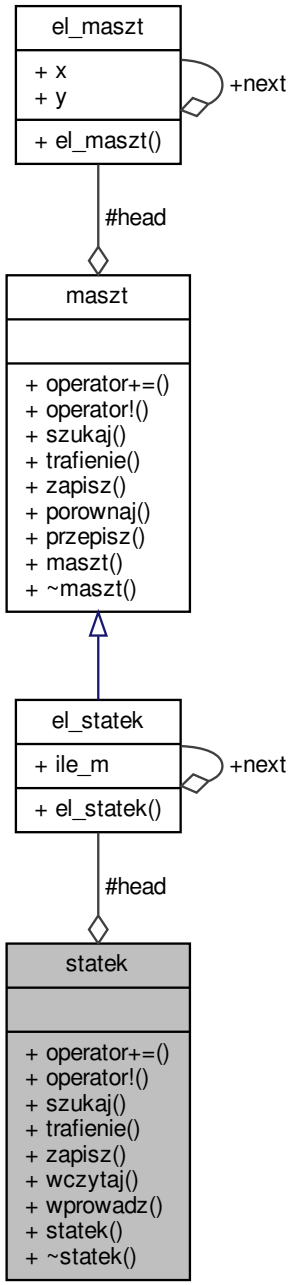


Diagram współpracy dla statek:




## 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 statkóœ; 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**

| | |
|---|---|
| *ile* | 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**

| | |
|---|---|
| *wsp* | współrzędne szukanego masztu |

**Zwraca**

true jeżeli znaleziono maszt o podanych współrzędnych; false jeżli 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**

| | |
|---|---|
| *wsp* | współrzędne szukanego masztu |

**Zwraca**

true jeżeli znaleziono statek o podanych współrzędnych; false jeżli 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**

| | |
|---|---|
| *wsp* | 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**

| | |
|---|---|
| *plik* | 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

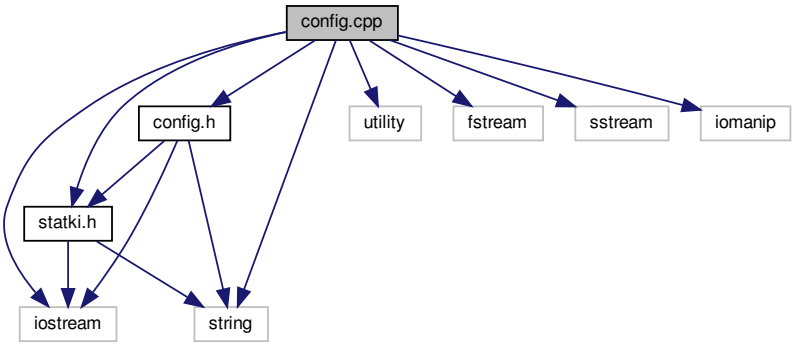
# Rozdział 5

# 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:

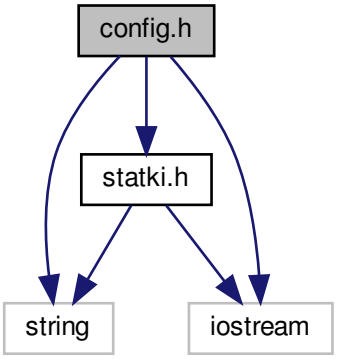



## 5.2 Dokumentacja pliku config.h

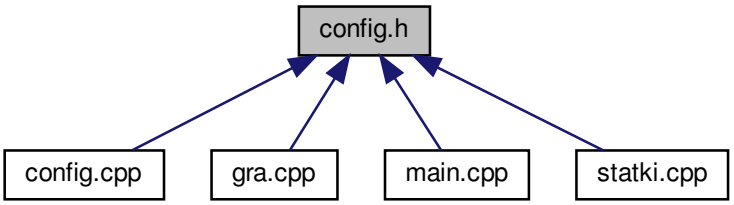
```
#include <string>
#include <iostream>
```

```
#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

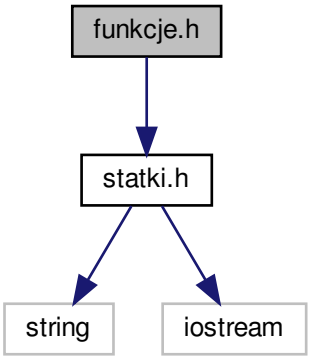
#### 5.2.1.2 ustawienia

```
konfiguracja ustawienia
```

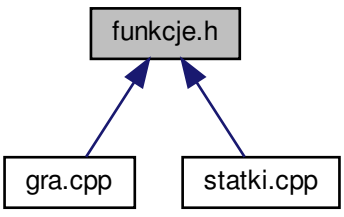
## 5.3 Dokumentacja pliku funkcje.h

```
#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**

| | |
|---|---|
| *g* | 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**

| | |
|---|---|
| *w1* | współrzędne dzioba |
| *w2* | współrzędne rufy |
| *ile* | 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 odpowienia

**Parametry**

| | |
|---|---|
| *a* | pierwsza współrzędna (x) |
| *b* | druga współrzędna (y) |
| *ile* | ilość masztów danego statku |

**Zwraca**

true jeżeli wspłrzędne 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()

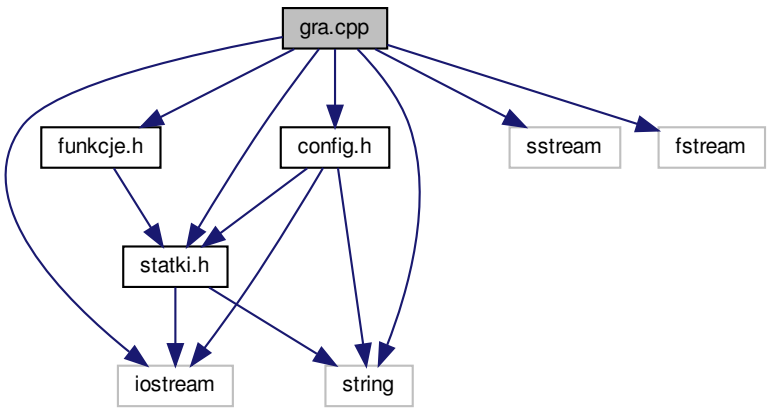
```
void zasady ( )
```

Funkcja wyświetla zasady i instrukje 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 > wspolrzedne ()
- 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**

| | |
|---|---|
| *g* | 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**

| | |
|---|---|
| *w1* | współrzędne dzioba |
| *w2* | współrzędne rufy |
| *ile* | 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 odpowienia

**Parametry**

| | |
|---|---|
| *a* | pierwsza współrzędna (x) |
| *b* | druga współrzędna (y) |
| *ile* | 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 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

#### 5.4.1.10 zasady()

```
void zasady ( )
```

Funkcja wyświetla zasady i instrukje gry

### 5.4.2 Dokumentacja zmiennych

#### 5.4.2.1 Hist

```
historia Hist
```

Globalna zmienna przechowująca ustawienia

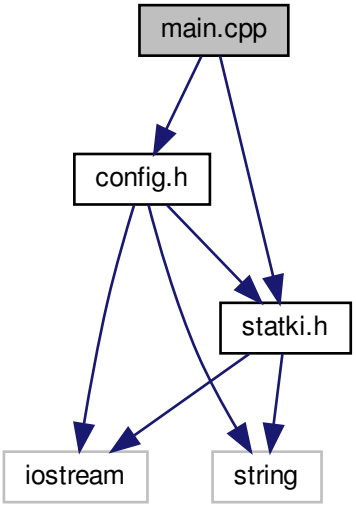
#### 5.4.2.2 ustawienia

```
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

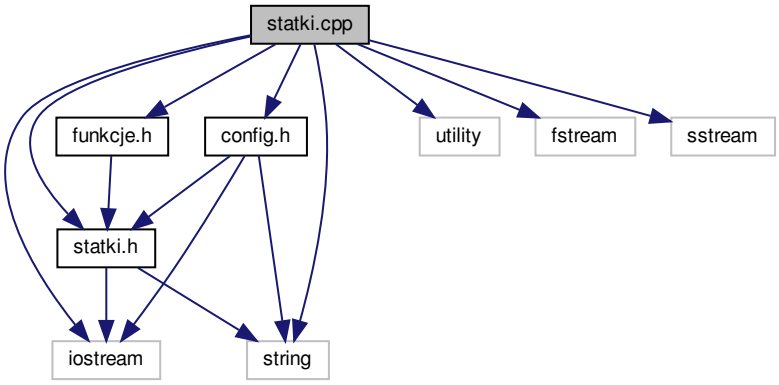
#### 5.5.1.1 main()

```
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"
```

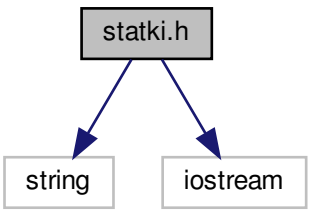
Wykres zależności załączania dla statki.cpp:



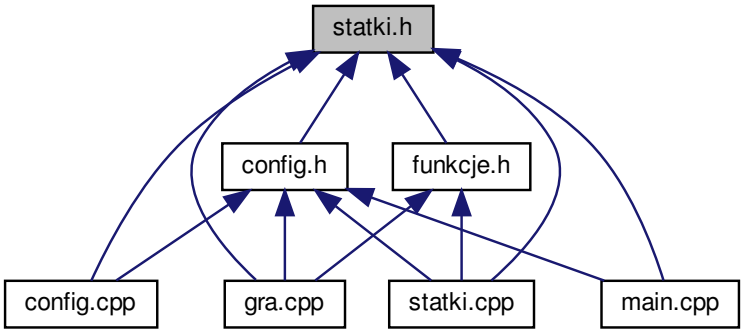

## 5.7 Dokumentacja pliku statki.h

```
#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