Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Laboratorium Programowania Komputerów

Gra Labirynt

autor	Dariusz Kluczewski
prowadzący	mgr inż. Maciej Długosz
rok akademicki	2018/2019
kierunek	Informatyka
rodzaj studiów	SSI
semestr	2
termin laboratorium	wtorek, 12:00 – 13:30
sekcja	17
termin oddania sprawozdania	2019-06-14

1 Treść zadania

Napisać grę, w której zadaniem użytkownika będzie chodzenie po labiryncie, zbieranie przedmiotów oraz unikanie ruchomych przeciwników. Jednym z elementów mapy powinno być wyjście, które skieruje użytkownika do kolejnego poziomu. Poziomy powinny być zapisywane w osobnych plikach; program powinien obsługiwać możliwość dodawania nowych poziomów. Program musi zapamiętywać oraz zapisywać w pliku binarnym wszystkie wyniki uzyskane przez graczy.

2 Analiza, projektowanie

2.1 Struktury danych

W programie zostały wykorzystane listy i tablice alokowane dynamicznie. Listy zostały wykorzystane do zapisania potworów, które poruszają się po mapie, oraz ekwipunku postaci i tablicy wyników. Dzięki zastosowaniu takiej struktury danych możliwe jest szybsze usunięcie elementu. Do stworzenia mapy została użyta dynamicznie alokowana tablica, co pozwoliło oszczędzić pamięć, ponieważ program zajmuje tylko tyle miejsca, ile jest mu potrzebne do wczytania mapy.

2.2 Algorytmy

Szukanie przedmiotów w ekwipunku lub przeciwników odbywa się poprzez liniowe przeszukanie listy.

3 Specyfikacja zewnętrzna

3.1 Obsługa programu

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwę gracza ograniczoną do 30 znaków, np.

```
program anonim
program  12345fddfgdad
```

3.2 Format danych wejściowych

Program pobiera dwa rodzaje danych z plików. Pierwszą z nich jest tablcia wyników. Jest ograniczona tylko do pięciu wyników. Dana ma postać

nazwagracza wynik. Następnymi danymi są mapy. Są one zapisane w postaci plików binarnych. Dana ma postać:

```
wysokość szerokość (od 0 do 200)
wypisane wierszami znaki jaki są na mapie
znak gracza (może go nie być)
pozycja gracza (dwie liczby)
znak przeciwnika (może go nie być)
pozycja przeciwników(tyle par liczb ile jest przeciwnikow)
```

Jeśli dane będą błędne program poinformuje gracza o tym stosownym komunikatem.

3.3 Komunikaty

Komunikaty błędów:
"Blad odczytu" - podany plik posiada błędne dane, należy podać inny plik;
" *Nie znaleziono pliku. Bledna nazwa."*- program nie znalazł pliku, należy podać inny plik;
"Bledna wartosc! Podaj jeszcze raz ilosc kolumn (max 200) : " - podano błędną wartość, należy podać liczbę jescze raz;
"Bledna wartosc! Podaj jeszcze raz ilosc wierszy (max 200) : " - podano błędną wartość, należy podać liczbę jescze raz;
" *Na tej mapie nie ma bohatera! Nacisnij dowolny klawisz, wrocic do menu."* - gracz podał mape do grania, na której nie ma postaci gracza, należy podać inną mapę lub dodać postać gracza w edytorze map

4 Specyfikacja wewnętrzna

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany na różnego rodzaju plikach. Pliki niepoprawne powodują ponowne poproszenie o podanie poprawnego pliku. Jeśli na mapie nie będzie postaci gracza to program wyświetli stosowny komunikat. Program zaczyna wykonywać pętle rekurencyjną, gdy przeciwnicy zostaną ustawieni w kółko, gdzie każdy czeka na to, aż ruszy się przeciwnik przed

nim. Jak na rysunku poniżej.

$$\rightarrow \downarrow$$
$$\uparrow \leftarrow$$

6 Wnioski

Zadanie zostało zrealizowane. Najwięcej problemów sprawiło napisanie algorytmu dla poruszających się przeciwników. W ciągu dalszych prac nad programrm należało by usunąć błąd opisany w sekscji testowania. W ramach rozwoju programu można dodać nowe elementy otoczenia np. teleporty. oraz nowe typy przeciwników.

Dodatek Szczegółowy opis typów i funkcji

My Project

Wygenerowano przez Doxygen 1.8.15

Rozdział 1

Indeks struktur danych

1.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

Rozdział 2

Indeks plików

2.1 Lista plików

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

Rozdział 3

Dokumentacja struktur danych

3.1 Dokumentacja struktury Ekwipunekstr

Pola danych

- struct Ekwipunekstr * pNext

  wskaznik na kolejny element listy
- unsigned char przedmiot

  rodzaj przedmiotu

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Header.h

3.2 Dokumentacja struktury kordy

Pola danych

- int cordx

  pozycja w pionie
- int cordy

  pozycja w poziomie

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Header.h

3.3 Dokumentacja struktury pole

Pola danych

- unsigned char obraz
  jaki obrazek ma sie wyswietlac na tym polu
- int wejscie
  czy gracz lub potwor moze wejsc na to pole

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Header.h

3.4 Dokumentacja struktury tabwyn

Pola danych

- struct tabwyn * pNext
  wskaznik na kolejny element tablicy
- char nazwa [30]
  dlugosc nazwy gracza
- int wynik
  ilosc punktow, jaka zdobyl gracz

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Header.h

3.5 Dokumentacja struktury wrog

Pola danych

- kordy pozycja
  pozycja wroga na mapie
- int kierunek
  kierunek, w ktorym sie porusza
- unsigned char naczymstoi
  co znajduje sie na polu na ktorym stoi
- struct wrog * pNext
  wskaznik na kolejnego przeciwnika w liscie
- bool ruch
  czy przeciwnik sie ruszyl w tym przebiegu petli

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Header.h

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku edytormap.h

```
#include "Header.h"
```

Funkcje

- void edytormap ()
- void nowamapa ()
- void staramapa ()
- void wpisz (int znak, kordy pozycja, wskpole mapa)
- int wktorastrone ()
- void przemieszczenie (kordy ∗pozycja, int wiersze, int kolumny)
- void edycja (wskpole mapa, int wiersze, int kolumny, wrog ∗∗przeciwnicy, kordy ∗∗postac)

4.1.1 Dokumentacja funkcji

4.1.1.1 edycja()

```
void edycja (
             wskpole mapa,
             int wiersze,
             int kolumny,
             wrog ** przeciwnicy,
             kordy ** postac )
```

Funkcja pozwala edytowac mape oraz ja zapisac

Parametry

mapa	wskaznik na mape
wiersze	ilosc wierszy mapy
kolumny	ilosc kolumn mapy
przeciwnicy	wskaznik na liste przeciwnikow
pozycja	postaci na mapie

4.1.1.2 edytormap()

```
void edytormap ( )
```

Funkcja obsluguje uruchomienie edytora map

4.1.1.3 nowamapa()

```
void nowamapa ( )
```

Funkcja tworzy nowa mape

4.1.1.4 przemieszczenie()

```
void przemieszczenie (
            kordy * pozycja,
            int wiersze,
            int kolumny )
```

Funkcja obsluguje porusznie sie po mapie podczas edycji

Parametry

pozycja	aktualna pozycja znacznika
wiersze	ilosc wierszy mapy
kolumny	ilosc kolumn mapy

4.1.1.5 staramapa()

```
void staramapa ( )
```

Funkcja wczytuje istniejaca mape

4.1.1.6 wktorastrone()

```
int wktorastrone ( )
```

Funkcja zczytuje od gracza, w ktora strone ma poruszac sie przeciwnik

Zwraca

kierunek poruszania sie

4.1.1.7 wpisz()

```
void wpisz (
            int znak,
            kordy pozycja,
            wskpole mapa )
```

Funkcja zapisuje znak do tablicy

Parametry

znak	jak iznak ma byc wpisany
pozycja	w jakim miejscu ma sie znajdowac ten znak
mapa	wskaznik na mape

4.2 Dokumentacja pliku gra.h

```
#include "Header.h"
```

Funkcje

- void gra (tabwyn ∗∗pHead, char ∗nazwa)
- bool klawiatura (kordy ∗ja, pole ∗∗∗tablica, Ekwipunekstr ∗∗pGlowa, int wiersze, int kolumny, wrog ∗∗przeciw)
- bool poruszaniesie (kordy ∗ja, kordy ∗kolja, pole ∗∗∗tablica, Ekwipunekstr ∗∗pGlowa, int wiersze, int kolumny, unsigned char ∗tymczasowy, bool ∗koniec)
- void dodlista (Ekwipunekstr ∗∗pGlowa, unsigned char ∗znak)
- void znaklu (int wysokosc, Ekwipunekstr ∗∗pGlowa, unsigned char ∗znak)
- Ekwipunekstr ∗ znajdzwekw (Ekwipunekstr ∗pGlowa, unsigned char znak)
- bool otwdrzwi (int wysokosc, Ekwipunekstr ∗∗pGlowa, unsigned char ∗znak)
- bool ruch (kordy ∗ja, kordy ∗kolja)

4.2.1 Dokumentacja funkcji

4.2.1.1 dodlista()

```
void dodlista (
            Ekwipunekstr ** pGlowa,
            unsigned char * znak )
```

Funkcja dodaje nowy klucz do listy ekwipunku

Parametry

pGlowa	wskaznik na liste kluczy ktore posiada gracz
znak	jaki klucz jest wstawiany

4.2.1.2 gra()

```
void gra (
            tabwyn ** pHead,
            char * nazwa )
```

Funkcja obsluguje uruchomienie gry oraz jej zakonczenie

Parametry

pHead	wskaznik na tablice wynikow
nazwa	imie uzytkownika

4.2.1.3 klawiatura()

```
bool klawiatura (
            kordy * ja,
            pole *** tablica,
            Ekwipunekstr ** pGlowa,
            int wiersze,
            int kolumny,
            wrog ** przeciw )
```

Funkcja czysci konslole

Parametry

ja	wskaznik na pozycje gracza
tablica	wskaznik na tablice zwierajaca mape
pGlowa	wskaznik na tablice wynikow
wiersze	ilosc wierszy mapy
kolumny	ilosc kolumn mapy
przeciw	wspaznik na liste przeciwnikow

Zwraca

czy gracz wygral (true - wygral, false - przegral)

4.2.1.4 otwdrzwi()

```
bool otwdrzwi (
            int wysokosc,
```

```
            Ekwipunekstr ** pGlowa,
            unsigned char * znak )
```

Funkcja usuwa klucz z ekwipunku i otwiera drzwi

Parametry

wysokosc	ilosc wierszy mapy
pGlowa	wskaznik na liste kluczy ktore posiada gracz
jakie	drzwi gracz probuje otworzyc

Zwraca

czy udalo sie otworzyc drzwi

4.2.1.5 poruszaniesie()

```
bool poruszaniesie (
            kordy * ja,
            kordy * kolja,
            pole *** tablica,
            Ekwipunekstr ** pGlowa,
            int wiersze,
            int kolumny,
            unsigned char * tymczasowy,
            bool * koniec )
```

Funkcja obsluguje porusznie sie gracza

Parametry

ja	wskaznik na pozycje gracza
kolja	wskaznik na nastepna pozycje gracza
tablica	wskaznik na tablice zwierajaca mape
pGlowa	wskaznik na liste kluczy ktore posiada gracz
wiersze	ilosc wierszy mapy
kolumny	ilosc kolumn mapy
tymczasowy	wskaznik na znak jaki miało pole przed wejsciem gracza
koniec	czy gracz doszedl do wyjscia z mapy

Zwraca

wartość sygnału wyjsciowego

4.2.1.6 ruch()

```
bool ruch (
            kordy * ja,
            kordy * kolja )
```

Funkcja zczytuje wcisniety klawisz oraz przekazuje kolejna pozycje gracza

Parametry

ja	wskaznik na pozycje gracza
kolja	wskaznik na nastepna pozycje gracza

Zwraca

czy gracz chce wyjsc z rozgrywki

4.2.1.7 znajdzwekw()

```
Ekwipunekstr* znajdzwekw (
            Ekwipunekstr * pGlowa,
            unsigned char znak )
```

Funkcja znajduje klucz w ekwipunku

Parametry

pGlowa	wskaznik na liste kluczy ktore posiada gracz
znak	jaki klucz ma byc zaleziony w ekwipunku

Zwraca

wskaznik na znaleziony klucz

4.2.1.8 znaklu()

```
void znaklu (
            int wysokosc,
            Ekwipunekstr ** pGlowa,
            unsigned char * znak )
```

Funkcja dodaje znaleziony klucz do ekwipunku oraz wypisuje ponownie ekwipunek

Parametry

wysokosc	ilosc wierszy mapy
pGlowa	wskaznik na liste kluczy ktore posiada gracz
znak	jaki klucz zostal znaleziony

4.3 Dokumentacja pliku Header.h

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

Struktury danych

- struct pole
- struct kordy
- struct Ekwipunekstr
- struct wrog
- struct tabwyn

Definicje typów

- typedef pole **** **wskpole**
- typedef struct Ekwipunekstr **Ekwipunekstr**
- typedef struct wrog **wrog**
- typedef struct tabwyn **tabwyn**

Wyliczenia

- enum **rodzaj** {
  glr = 201, **gpr** = 187, **dlr** = 200, **dpr** = 188,
  poz = 205, **pio** = 186, **lsc** = 204, **psc** = 185,
  gsc = 203, **dsc** = 202, **sro** = 204, **kro** = 248,
  pus = 32, **klu1** ='a', **klu2** ='b', **klu3** ='c',
  klu4, **klu5**, **klu6**, **zam1** ='A',
  zam2 ='B', **zam3** ='C', **zam4**, **zam5**,
  zam6, **wyjscie** ='X' }
- enum **czmwej** { **tak** = 0, **nie** = 1, **drz** = 2, **kon** = 3 }
- enum **sterowanie** {
  zkps =224, **zkps0** =0, **lewo** =75, **prawo** =77,
  gora =72, **dol** =80 }
- enum **postacie** { **gracz** =208, **potwor** = 245 }
- enum **elementydlagracza** {
  chodzenie =1, **eglr**, **egpr**, **edlr**,
  edpr, **epoz**, **epio**, **elsc**,
  epsc, **egsc**, **edsc**, **esro**,
  ekro, **epus**, **eklu1**, **eklu2**,
  eklu3, **eklu4**, **eklu5**, **eklu6**,
  ezam1, **ezam2**, **ezam3**, **ezam4**,
  ezam5, **ezam6**, **ewyjscie**, **epotwor**,
  egracz, **wyjsciezzapisem**, **wyjsciebezzapisu** }

Funkcje

- void gotoxy (int x, int y)
- void czyszczenie ()
- void czyszczeniebufora ()
- void podmien (int wiersz, int kolumna, int znak, int wysokosc)
- void dowolnyprzycisk ()

4.3.1 Dokumentacja funkcji

4.3.1.1 czyszczenie()

```
void czyszczenie ( )
```

Funkcja czysci konslole

4.3.1.2 czyszczeniebufora()

```
void czyszczeniebufora ( )
```

Funkcja usuwa wszystkie elementy z bufora stdin

4.3.1.3 dowolnyprzycisk()

```
void dowolnyprzycisk ( )
```

Funkcja czeka, az gracz wcisnie dowolny klawisz

4.3.1.4 gotoxy()

```
void gotoxy (
             int x,
             int y )
```

Funkcja przemieszcza kursor wpisywania na podana pozycje

Parametry

x	pozycja w pionie
y	pozycja w poziomie

4.3.1.5 podmien()

```
void podmien (
             int wiersz,
             int kolumna,
             int znak,
             int wysokosc )
```

Funkcja wpisuje znak w podanym miejscu

Parametry

wiersz	w ktorym wierszu ma byc wstawiony znak
kolumna	w ktorej kolumnie ma byc wstawiony znak
znak	jaki znak ma byc wstawiony
wysokosc	mapy

4.4 Dokumentacja pliku obslugapliku.h

```
#include "Header.h"
```

Funkcje

- void zczytajtablice (tabwyn **pHead)
- bool zczytaj (wskpole tablica, wrog **przeciwnicy, int *wysokosc, int *szerokosc, kordy **postac, char *nazwa)
- bool czytaniemapy (wskpole tab, wrog **przeciwnicy, int *wysokosc, int *szerokosc, kordy **postac)
- void zapiszmape (wskpole mapa, wrog **przeciwnicy, int wiersze, int kolumny, kordy *postac)

4.4.1 Dokumentacja funkcji

4.4.1.1 czytaniemapy()

```
bool czytaniemapy (
            wskpole tab,
            wrog ** przeciwnicy,
            int * wysokosc,
            int * szerokosc,
            kordy ** postac )
```

Funkcja obsluguje menu wpisywania nazwy mapy

Parametry

tablica	wskaznik na mape
przeciwnicy	wskaznik na liste przeciwnikow
wysokosc	ilosc wierszy mapy
szerokosc	ilosc kolumn mapy
postac	wskaznik na pozycje postaci

Zwraca

czy gracz chce przerwac wpisywanie mapy i wrocic do menu glownego

4.4.1.2 zapiszmape()

```
void zapiszmape (
            wskpole mapa,
            wrog ** przeciwnicy,
            int wiersze,
            int kolumny,
            kordy * postac )
```

Funkcja zapisuje do pliku mape

Parametry

tablica	wskaznik na mape
przeciwnicy	wskaznik na liste przeciwnikow
wiersze	ilosc wierszy mapy
kolumny	ilosc kolumn mapy
postac	wskaznik na pozycje postaci

4.4.1.3 zczytaj()

```
bool zczytaj (
            wskpole tablica,
            wrog ** przeciwnicy,
            int * wysokosc,
            int * szerokosc,
            kordy ** postac,
            char * nazwa )
```

Funkcja zczytuje plik zawierajacy mape

Parametry

tablica	wskaznik na mape
przeciwnicy	wskaznik na liste przeciwnikow
wysokosc	ilosc wierszy mapy
szerokosc	ilosc kolumn mapy
postac	wskaznik na pozycje postaci
nazwa	nazwa mapy

Zwraca

czy otwarcie pliku sie udalo i czy dane sa poptawne

4.4.1.4 zczytajtablice()

```
void zczytajtablice (
            tabwyn ** pHead )
```

Funkcja zczytuje z pliku tablice wynikow

Parametry

pHead	wskaznik na tablice wynikow

4.5 Dokumentacja pliku przeciwnicy.h

```
#include "Header.h"
```

Funkcje

- void dodprzec (wrog **przeciwnicy, kordy pozycja, int kierunek, int naczym)
- void zmianapozycjiprze (kordy *ja, kordy *kolja, pole ***tablica, int wiersze, unsigned char *tymczasowy)
- wrog * znajdzprzeciwnika (kordy gdzie, wrog *przec)
- void ruchnormalny (wrog *przeciw, kordy *kolejne)
- void ruchodwrotny (wrog *przeciw, kordy *kolejne)
- bool ruchnormalnyzw (wrog *przeciw, pole ***tablica, int wysokosc, int szerokosc, bool *koniec, kordy *kolejne, wrog *wszyscy)
- bool ruchodwrotnyzw (wrog *przeciw, pole ***tablica, int wysokosc, int szerokosc, bool *koniec, kordy *kolejne, wrog *wszyscy)
- bool potworprzed (wrog *przeciw, pole ***tablica, int wysokosc, int szerokosc, bool *koniec, kordy odkogo, bool sciana, wrog *wszyscy)
- void gdzieiscprze (wrog *przeciw, pole ***tablica, int wysokosc, int szerokosc, bool *koniec, wrog *wszyscy)
- void wyszczscruch (wrog **przeciw)
- bool poruszprzeciwnikami (wrog **przeciw, pole ***tablica, int wiersze, int kolumny)
- void usunprzeciw (wrog **przeciwnicy, kordy pozycja)

4.5.1 Dokumentacja funkcji

4.5.1.1 dodprzec()

```
void dodprzec (
            wrog ** przeciwnicy,
            kordy pozycja,
            int kierunek,
            int naczym )
```

Funkcja dodaje przeciwnika do listy przeciwnikow

Parametry

przeciwnicy	lista przeciwnikow
pozycja	miejsce w ktorym znajduje sie przeciwnik
kierunek	kierunek, w ktorym porusz sie przeciwnik
naczym	na jakim polu stoi przeciwnik

4.5.1.2 gdzieiscprze()

```
void gdzieiscprze (
            wrog * przeciw,
            pole *** tablica,
            int wysokosc,
            int szerokosc,
            bool * koniec,
            wrog * wszyscy )
```

Funkcja wykonuje ruch jednego przeciwnika

Parametry

przeciw	aktualnie ruszajacy sie przeciwnik
tablica	wkaznik na mape
wysokosc	ilosc wierszy mapy
szerokosc	ilosc kolumn mapy
koniec	czy gracz zostal zabity
wszyscy	wskaznika na liste przeciwnikow

4.5.1.3 poruszprzeciwnikami()

```
bool poruszprzeciwnikami (
            wrog ** przeciw,
            pole *** tablica,
            int wiersze,
            int kolumny )
```

Funkcja wykonuje ruchy wszystkich przeciwnikow

Parametry

przeciw	wskaznik na liste przeciwnikow
tablica	wskaznik na mape
wiersze	ilosc wierszy mapy
kolumny	ilosc kolumn mapy

Zwraca

czy gracz zostal zabity

4.5.1.4 potworprzed()

```
bool potworprzed (
            wrog * przeciw,
            pole *** tablica,
            int wysokosc,
            int szerokosc,
            bool * koniec,
            kordy odkogo,
            bool sciana,
            wrog * wszyscy )
```

Funkcja obsluguje sytuacje gdy na drodze przeciwnika pojawi sie drugi przeciwnik

Parametry

przeciw	aktualnie ruszajacy sie przeciwnik
tablica	wkaznik na mape
wysokosc	ilosc wierszy mapy
szerokosc	ilosc kolumn mapy
koniec	czy gracz zostal zabity
odkogo	ktory przeciwnik kazal ruszyc sie aktualnemu przeciwnikowi
sciana	czy przeciwnik wykonuje ruch odwrotny
wszyscy	wskaznika na liste przeciwnikow

Zwraca

czy udalo sie wykonac ruch

4.5.1.5 ruchnormalny()

```
void ruchnormalny (
            wrog * przeciw,
            kordy * kolejne )
```

Funkcja podaje nastepna pozycje przeciwnika

Parametry

przeciw	aktualnie ruszjacy sie przeciwnik
kolejne	kolejna pozycja przeciwnika

4.5.1.6 ruchnormalnyzw()

```
bool ruchnormalnyzw (
            wrog * przeciw,
            pole *** tablica,
            int wysokosc,
            int szerokosc,
            bool * koniec,
            kordy * kolejne,
            wrog * wszyscy )
```

Funkcja obsluguje ruch, gdy potwor idzie zgodnie z aktualnym kierunkiem ruchu

Parametry

przeciw	aktualnie ruszajacy sie przeciwnik
tablica	wkaznik na mape
wysokosc	ilosc wierszy mapy
szerokosc	ilosc kolumn mapy
koniec	czy gracz zostal zabity
kolejne	kolejna pozycja przeciwnika
wszyscy	wskaznika na liste przeciwnikow

Zwraca

czy udalo sie wykonac ruch

4.5.1.7 ruchodwrotny()

```
void ruchodwrotny (
            wrog * przeciw,
            kordy * kolejne )
```

Funkcja podaje pozycje pozycji przeciwnika pod zmianie kierunku oraz zmienia kierunek poruszania sie przeciwnika

Parametry

przeciw	aktualnie ruszjacy sie przeciwnik
kolejne	kolejna pozycja przeciwnika

4.5.1.8 ruchodwrotnyzw()

```
bool ruchodwrotnyzw (
```

```
            wrog * przeciw,
            pole *** tablica,
            int wysokosc,
            int szerokosc,
            bool * koniec,
            kordy * kolejne,
            wrog * wszyscy )
```

Funkcja obsluguje ruch, gdy potwor nie moze isc zgodnie z wczesniejszym kierunkiem ruchu

Parametry

przeciw	aktualnie ruszajacy sie przeciwnik
tablica	wkaznik na mape
wysokosc	ilosc wierszy mapy
szerokosc	ilosc kolumn mapy
koniec	czy gracz zostal zabity
kolejne	kolejna pozycja przeciwnika
wszyscy	wskaznika na liste przeciwnikow

Zwraca

czy udalo sie wykonac ruch

4.5.1.9 usunprzeciw()

```
void usunprzeciw (
            wrog ** przeciwnicy,
            kordy pozycja )
```

Funkcja usuwa przeciwnika znajdujacego sie na danej pozycji na mapie z listy przeciwnikow

Parametry

przeciwnicy	wskaznik na liste przeciwnikow
pozycja	pozycja usuwanego przeciwnika

4.5.1.10 wyszczscruch()

```
void wyszczscruch (
            wrog ** przeciw )
```

Funkcja czysci znacznik wykoniania ruchu przez przeciwnikow w danej turze

Parametry

przeciw	wskaznik na liste przeciwnikow

4.5.1.11 zmianapozycjiprze()

```
void zmianapozycjiprze (
            kordy * ja,
            kordy * kolja,
            pole *** tablica,
            int wiersze,
            unsigned char * tymczasowy )
```

Funkcja zmienia pozycje przeciwnika na mapie

Parametry

ja	aktualna pozycja przeciwnika
kolja	nastepna pozycja przeciwnika
tablica	wskaznik na mape
wiersze	ilosc wierszy mapy
tymczasowy	na jakim polu stoi przeciwnik

4.5.1.12 znajdzprzeciwnika()

```
wrog* znajdzprzeciwnika (
            kordy gdzie,
            wrog * przec )
```

Funkcja szuka przeciwnika znajdujacego sie na danej pozcji w liscie przeciwnikow

Parametry

gdzie	pozycja przeciwnika
przec	lista przeciwnikow

Zwraca

wskaznik na znalezionego przeciwnika

4.6 Dokumentacja pliku tablicawynikow.h

```
#include "Header.h"
```

Funkcje

- void dodajwynik (tabwyn **pHead, tabwyn *nowy)
- void tablicawynikow (tabwyn *pHead)
- void wipiszdotablicy (tabwyn **pHead, char nazwa[30], int ptk)
- void zapisztablice (tabwyn *pHead)

4.6.1 Dokumentacja funkcji

4.6.1.1 dodajwynik()

```
void dodajwynik (
            tabwyn ** pHead,
            tabwyn * nowy )
```

Funkcja dodaje nowy wynik do listy wynikow

Parametry

pHead	wskaznik na tablice wynikow
nowy	wstawiany element tablicy

4.6.1.2 tablicawynikow()

```
void tablicawynikow (
            tabwyn * pHead )
```

Funkcja wypisuje tablice wynikow

Parametry

pHead	wskaznik na tablice wynikow

4.6.1.3 wipiszdotablicy()

```
void wipiszdotablicy (
            tabwyn ** pHead,
            char nazwa[30],
            int ptk )
```

Funkcja wpisuje do tablicy uzyskany przez gracza wynik

Parametry

pHead	wskaznik na tablice wynikow
nazwa	imie gracza
ptk	ilosc punktow jakie uzyskal gracz

4.6.1.4 zapisztablice()

```
void zapisztablice (
            tabwyn * pHead )
```

Funkcja zapisuje tablice wynikow do pliku

Parametry

pHead	wskaznik na tablice wynikow

4.7 Dokumentacja pliku usuwanie.h

```
#include "Header.h"
```

Funkcje

- void usuntablice (tabwyn **lista)
- void usunmape (wskpole mapa, int wiersze, int kolumny)
- void usunliste (wrog **przeciwnik)
- void usuncalamape (wskpole mapa, int wiersze, int kolumny, wrog **przeciwnik, kordy **postac)
- void usunekwipunek (Ekwipunekstr **pHeadekw)
- void usunnazwe (char **nazwa)

4.7.1 Dokumentacja funkcji

4.7.1.1 usuncalamape()

```
void usuncalamape (
            wskpole mapa,
            int wiersze,
            int kolumny,
            wrog ** przeciwnik,
            kordy ** postac )
```

Funkcja zwalnia pamiec zajeta prze mape oraz liste przeciwnikow i postac

Parametry

mapa	wskaznik na mape
wiersze	ilosc wierszy mapy
kolumny	ilisc kolumn mapy
przeciwnik	wskaznik na liste przeciwnikow
postac	wskaznik na miejsce, w ktorym znajduje sie postac

4.7.1.2 usunekwipunek()

```
void usunekwipunek (
            Ekwipunekstr ** pHeadekw )
```

Funkcja zwalnia pamiec zajeta przez liste przedmiotow

Parametry

pHeadekw	wskaznik na liste przedmiotow

4.7.1.3 usunliste()

```
void usunliste (
            wrog ** przeciwnik )
```

Funkcja zwalnia pamiec zajenta przez liste przeciwnikow

Parametry

przeciwnik	wskaznik na liste przeciwnikow

4.7.1.4 usunmape()

```
void usunmape (
            wskpole mapa,
            int wiersze,
            int kolumny )
```

Funkcja zwalnia pamiec zajeta przez mape

Parametry

mapa	wskaznik na mape
wiersze	ilosc wierszy mapy
kolumny	ilisc kolumn mapy

4.7.1.5 usunnazwe()

```
void usunnazwe (
            char ** nazwa )
```

Funkcja zwalnia pamiec zajenta przez nazwe gracza

Parametry

nazwa	imie uzytkownika

4.7.1.6 usuntablice()

```
void usuntablice (
            tabwyn ** lista )
```

Funkcja zwalnia pamiec zajenta przez liste wynikow

Parametry

lista	wskaznik na liste wynikow

4.8 Dokumentacja pliku wypisywanie.h

```
#include "Header.h"
```

Funkcje

- void wypiszekw (Ekwipunekstr ∗pGlowa)
- void wypiszmapiekw (pole ∗∗∗tablica, int wiersze, int kolumny, Ekwipunekstr ∗ekw)
- void wypiszmapa (pole ∗∗∗tablica, int wiersze, int kolumny)
- void wypiszpomoc (int szerokosc)
- void wypiszpomchodzenie (int kolumny)

4.8.1 Dokumentacja funkcji

4.8.1.1 wypiszekw()

```
void wypiszekw (
            Ekwipunekstr * pGlowa )
```

Funkcja wypisuje w konsoli ekwipunek

Parametry

pGlowa	wskaznik na liste przedmiotow

4.8.1.2 wypiszmapa()

```
void wypiszmapa (
            pole *** tablica,
            int wiersze,
            int kolumny )
```

Funkcja wypisuje w konsoli mape

Parametry

tablica	wskaznik na mape
wiersze	ilosc wierczy mapy
kolumny	ilosc kolumn mapy

4.8.1.3 wypiszmapiekw()

```
void wypiszmapiekw (
            pole *** tablica,
            int wiersze,
            int kolumny,
            Ekwipunekstr * ekw )
```

Funkcja wypisuje w konsoli mape i ekwipunek

Parametry

tablica	wskaznik na mape
wiersze	ilosc wierczy mapy
kolumny	ilosc kolumn mapy
ekw	wskaznik na liste przedmiotow

4.8.1.4 wypiszpomchodzenie()

```
void wypiszpomchodzenie (
            int kolumny )
```

Funkcja wypisuje pomoc w trybie chodzenia w edytorze map

Parametry

kolumny	ilosc kolumn mapy

4.8.1.5 wypiszpomoc()

```
void wypiszpomoc (
            int szerokosc )
```

Funkcja wypisuje pomoc w edytorze map

Parametry

szerokosc	ilosc kolumn mapy

Indeks