Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania 2

Gra w statki

Autor	Krzysztof Małek
Prowadzący	dr inż. Bożena Wieczorek
Rok akademicki	2019/2020
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	2
Sekcja	72

1 Treść zadania

Napisać program będący grą „Statki” dla dwóch osób. Działanie programu polega na wygenerowaniu 4 tablic będących listą list gdzie zapisywane są stany poszczególnych pól w trakcie trwania rozgrywki.

2 Analiza zadania

Zagadnienie przedstawia problem odpowiedniego zapisania stanu aktualnej gry w celu płynnego korzystania z programu.

2.1 Struktury danych

W programie wykorzystano listy list do przechowywania, jako wiersze i kolumny, plansz do gry dla graczy, gdzie zapisywane są aktualne stany pól (puste, pełne, trafione, bądź pudło) i zmieniane w zależności od poczynań graczy.

Ta struktura umożliwia łatwe zobrazowanie plansz, łatwy dostęp, zmienianie stanów i wyszukiwanie odpowiednich pól do gry.

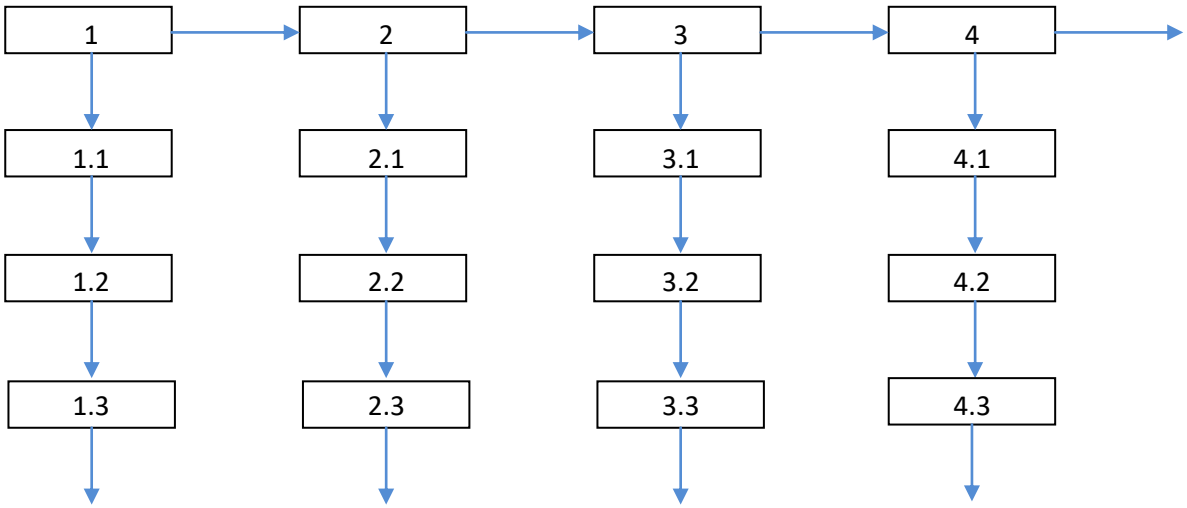
2.2 Algorytmy

Utworzenie listy list rozpoczyna się przez zajęcie przez program części pamięci i utworzeniu głowy każdej z 4 list (plansz). Następnie do każdej z głów iteracyjnie „dołączane” są kolejne elementy list pod które tworzone są listy list wraz z tworzeniem się kolejnych elementów listy, przypisując każdemu elementowi stan „pusty”.

3 Specyfikacja wewnętrzna

Program uruchomiony może być z linii poleceń bez dodatkowych przełączników.

Rysunek 1: Fragment listy list (planszy)



4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs od logiki aplikacji.

4.1 Ogólna struktura programu

W funkcji głównej program tworzy głowy plansz a następnie wywołuje funkcje **make_board,** która tworzy całe plansze. Później wyświetla kolejność rund w zależności od tego, którego gracza następuje tura i wyświetla planszę gracza wywołaną funkcją **print_board**, zaczyna się etap rozstawiania statków na planszy. Program pozwala na rozstawienie 10 statków: 1 statek o rozmiarze 4 pól, 2 statki o rozmiarze 3 pól, 3 statki o rozmiarze 2 pól i 4 statki po 1 pole. Program pyta teraz o rozmiar statku, który chcemy postawić, sprawdza czy nie wykorzystaliśmy puli statków z tego rozmiaru, jeśli tak, wyświetla komunikat „This size was used", bądź jeśli wybraliśmy zły rozmiar „Wrong size". Jeśli odpowiednio wybraliśmy rozmiar statku program wywołuje funkcje **choosing_place**, która pyta o koordynaty gdzie chcemy postawić statek, sprawdzając ich poprawność, a następnie sprawdzając poprzez funkcję **check_state** czy pole jest już zajęte jeśli nie funkcja sprawdza czy reszta statku może być postawiona po lewej stronie od wybranego miejsca (sprawdza czy nie kończy się plansza oraz czy nie znajduje się tam już jakiś inny statek), następnie zapisuje stan do 4 przełączników, (które oznaczają lewo, dół, prawo i górę) i wywołuje funkcje **choosing_the_rest_of_the_ship**, która wyświetla na ekranie możliwe do wyboru miejsca w które ma iść reszta statku, oraz wywołuje funkcje **change_one** aby po wybraniu miejsca zmienić stan pól z „puste" na „zajęte". Po postawieniu wszystkich statków dla pierwszego gracza to samo dzieje się dla gracza drugiego po poprzedzającym to komunikacie „Player 2 Turn:". Gdy gracze postawią już swoje statki na planszy następuje właściwa rozgrywka, program wywołuje funkcję **turn**. Teraz zamiast jednej planszy program wyświetla dwie (jedną od gracza i jedną od przeciwnika z zakrytymi miejscami gdzie znajdują się statki). Uruchamia się funkcja **choosing_place** tym razem jednak z odpowiednim przełącznikiem, dzięki czemu funkcja ta tym razem pyta o miejsce, w które gracz chce strzelić. Funkcja zamienia koordynaty na odpowiednie im liczby (np. przy wyborze B4 zamienia B na 1 zostawiając 4 bez zmian), program wybiera teraz wybrane pole i sprawdza jego status, w zależności od statusu program reaguje odpowiednio:

-pusty: Program wypisuje słowo „Miss!!!" po czym następuje runda gracza przeciwnego.
-pełny: Program wypisuje słowo „Hit!!!" po czym gracz ma kolejną próbę strzału.
-trafiony i pudło: Program wypisuje słowo „Choose another spot!!!" po czym prosi raz jeszcze o podanie koordynat.

Jeśli statek został trafiony funkcja zwraca liczbę całkowitą 1, która dodaje się do wyniku gracza, gdy gracz osiągnie wynik 20 gra kończy się z odpowiednim komunikatem mówiącym, który gracz wygrał.

Na sam koniec program zwalnia zajętą przez planszę pamięć wywołując funkcje **delete_memory**.

4.2

Szczegółowy opis typów i funkcji wraz z grafami zawarty jest w załączniku poniżej sprawozdania.

5 Testowanie

Program testowany był na podanie błędnych danych i możliwe nietypowe zachowania graczy.

Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Program gry w statki wymagał odpowiedniego dobrania struktury danych oraz zapisu danych i przedstawienia ich graczom. Program wymaga wprowadzenia wielu form sprawdzenia i weryfikacji tego, co wpisuje gracz. Najbardziej wymagające było wymyślenie sposobu sprawdzenia w jaki sposób można ułożyć resztę statku tak aby nie nachodziły one na siebie bądź nie wychodziły poza planszę.

Literatura

https://pl.wikibooks.org/
https://stackoverflow.com/
https://www.ascii-code.com/

Załącznik

Szczegółowy opis typów i funkcji

Gra w statki

Wygenerowano przez Doxygen 1.8.17

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:

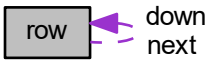
Rozdział 3

Dokumentacja struktur danych

3.1 Dokumentacja struktury row

`#include <struct.h>`

Diagram współpracy dla row:



Pola danych

- int data

  stan pola
- struct row * next

  adres kolejnej kolumny
- struct row * down

  adres kolejnego wiersza

3.1.1 Opis szczegółowy

Lista list przechwująca plansze do gry

Dokumentacja dla tej struktury została wygenerowana z pliku:

- struct.h

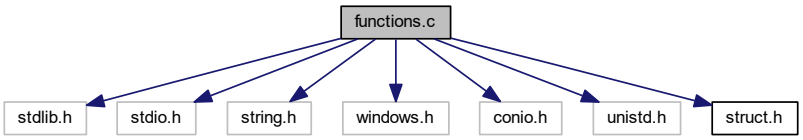
Rozdział 4

Dokumentacja plików

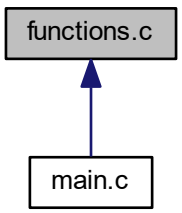
4.1 Dokumentacja pliku functions.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <conio.h>
#include <unistd.h>
#include "struct.h"
```

Wykres zależności załączania dla functions.c:



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



Funkcje

- void make_board (row_type ∗head)
- void print_board (row_type ∗head)
- int choosing_place (row_type ∗head, row_type ∗copy, int size)
- void **change_one** (row_type ∗head, int row, int column, int value)
- void chosing_the_rest_of_the_ship (row_type ∗head, int row, int column, int size, int f, int s, int t, int fo)
- int check_state (row_type ∗head, int row, int column, int state)
- void turn (row_type ∗P1_board, row_type ∗E1_board, row_type ∗P2_board, int ∗P)
- void delete_memory (row_type ∗current)

4.1.1 Dokumentacja funkcji

4.1.1.1 check_state()

```
int check_state (
            row_type * head,
            int row,
            int column,
            int state )
```

Funkcja sprawdza status pola na planszy

Parametry

head	głowa wybranej planszy
row	wiersz
column	kolumna
state	status w postaci liczby

Zwraca

int

4.1.1.2 choosing_place()

```
int choosing_place (
            row_type * head,
            row_type * copy,
            int size )
```

Funkcja pyta i weryfikuje miejsce wybrane przez gracza

Parametry

head	głowa wybranej planszy
copy	kopia planszy dla gracza pokazująca gdzie już strzelał
size	wielkość statku

Zwraca

int

4.1.1.3 chosing_the_rest_of_the_ship()

```
void chosing_the_rest_of_the_ship (
            row_type * head,
            int row,
            int column,
            int size,
            int f,
            int s,
            int t,
            int fo )
```

Funkcja sprawdza w którą stronę można położyć resztę statku

Parametry

head	głowa wybranej planszy
row	wiersz
column	kolumna
size	wielkość statku
f	ilość wolnego miejsa w lewą stronę
f	ilość wolnego miejsa w dół
f	ilość wolnego miejsa w prawą stronę
f	ilość wolnego miejsa w górę

Zwraca

void

4.1.1.4 delete_memory()

```
void delete_memory (
            row_type * current )
```

Funkcja usuwająca wybraną plansze - zwalnianie pamięci

Parametry

current	wybrana plansza

Zwraca

void

4.1.1.5 make_board()

```
void make_board (
            row_type * head )
```

Funkcja alokuje pamięć pod plansze

Parametry

head	głowa wybranej planszy

Zwraca

void

4.1.1.6 print_board()

```
void print_board (
            row_type * head )
```

Funkcja wyświetla planszę na ekranie

Parametry

head	głowa wybranej planszy

Zwraca

void

4.1.1.7 turn()

```
void turn (
            row_type * P1_board,
```

```
row_type * E1_board,
row_type * P2_board,
int * P )
```

Funkcja wywołująca turę gracza

Parametry

P1_board	głowa planszy gracza
E1_board	głowa planszy przeciwnika
P2_board	głowa planszy drugiego gracza
P	liczba trafień gracza

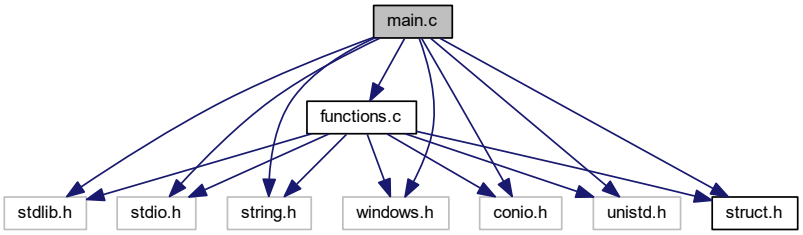
Zwraca

void

4.2 Dokumentacja pliku main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <conio.h>
#include <unistd.h>
#include "struct.h"
#include "functions.c"
```

Wykres zależności załączania dla main.c:

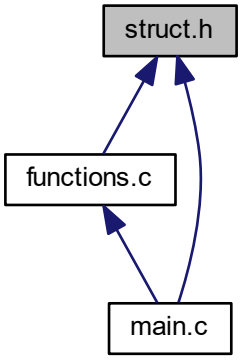


Funkcje

- int **main** ()

4.3 Dokumentacja pliku struct.h

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



Struktury danych

- struct row

Definicje typów

- typedef struct row row_type

Funkcje

- void make_board (row_type *head)
- void print_board (row_type *head)
- int choosing_place (row_type *head, row_type *copy, int size)
- void change_one (row_type *head, int row, int column)
- void chosing_the_rest_of_the_ship (row_type *head, int row, int column, int size, int f, int s, int t, int fo)
- int check_state (row_type *head, int row, int column, int state)
- void turn (row_type *P1_board, row_type *E1_board, row_type *P2_board, int *P)
- void delete_memory (row_type *current)

4.3.1 Dokumentacja definicji typów

4.3.1.1 row_type

```
typedef struct row row_type
```

Lista list przechwująca plansze do gry

4.3.2 Dokumentacja funkcji

4.3.2.1 change_one()

```
void change_one (
            row_type * head,
            int row,
            int column )
```

Funkcja zmienia status pola na planszy

Parametry

head	głowa wybranej planszy
row	wiersz
column	kolumna

Zwraca

void

4.3.2.2 check_state()

```
int check_state (
            row_type * head,
            int row,
            int column,
            int state )
```

Funkcja sprawdza status pola na planszy

Parametry

head	głowa wybranej planszy
row	wiersz
column	kolumna
state	status w postaci liczby

Zwraca

int

4.3.2.3 choosing_place()

```
int choosing_place (
            row_type * head,
            row_type * copy,
            int size )
```

Funkcja pyta i weryfikuje miejsce wybrane przez gracza

Parametry

head	głowa wybranej planszy
copy	kopia planszy dla gracza pokazująca gdzie już strzelał
size	wielkość statku

Zwraca

int

4.3.2.4 chosing_the_rest_of_the_ship()

```
void chosing_the_rest_of_the_ship (
            row_type * head,
            int row,
            int column,
            int size,
            int f,
            int s,
            int t,
            int fo )
```

Funkcja sprawdza w którą stronę można położyć resztę statku

Parametry

head	głowa wybranej planszy
row	wiersz
column	kolumna
size	wielkość statku
f	ilość wolnego miejsa w lewą stronę
f	ilość wolnego miejsa w dół
f	ilość wolnego miejsa w prawą stronę
f	ilość wolnego miejsa w górę

Zwraca

void

4.3.2.5 delete_memory()

```
void delete_memory (
            row_type * current )
```

Funkcja usuwająca wybraną plansze - zwalnianie pamięci

Parametry

current	wybrana plansza

Zwraca

void

4.3.2.6 make_board()

```
void make_board (
            row_type * head )
```

Funkcja alokuje pamięć pod plansze

Parametry

head	głowa wybranej planszy

Zwraca

void

4.3.2.7 print_board()

```
void print_board (
            row_type * head )
```

Funkcja wyświetla planszę na ekranie

Parametry

head	głowa wybranej planszy

Zwraca

void

4.3.2.8 turn()

```
void turn (
            row_type * P1_board,
            row_type * E1_board,
            row_type * P2_board,
            int * P )
```

Funkcja wywołująca turę gracza

Parametry

P1_board	głowa planszy gracza
E1_board	głowa planszy przeciwnika
P2_board	głowa planszy drugiego gracza
P	liczba trafień gracza

Zwraca

void

Indeks