Politechnika Warszawska
Wydział Elektryczny

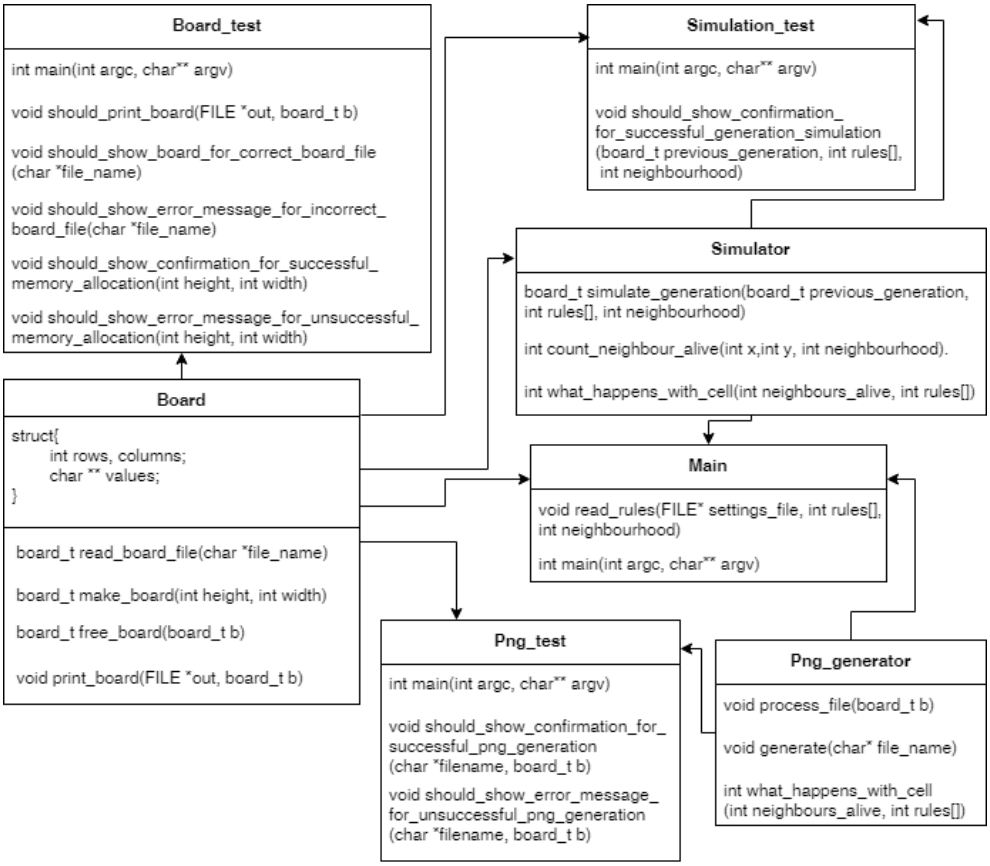
Specyfikacja implementacyjna symulatora automatu komórkowego *Life*

Autorzy:
J. Korczakowski, nr albumu 291079
B. Suchocki, nr albumu 291111
Grupa projektowa nr 11

14 marca 2018

Spis treści

1 Diagram programu



Rysunek 1: Diagram

Na diagramie naszego projektu przedstawiliśmy wszystkie moduły, które wchodzą w jego skład. Moduły korzystają z funkcji tych modułów, które mają strzałki skierowane w ich kierunku.

2 Opis modułów i ich funkcji

2.1 Board

Zawiera strukturę przechowywującą planszę i rozłożone na niej komórki oraz następujące funkcje:

1. `board_t make_board(int height, int width).`
   - Argumenty:
     - `height`, `width` - wysokość i szerokość planszy.

- Wartość zwracana:
  - Nowa plansza (obiekt struktury `board_t`)
- Działanie:
  - Funkcja alokuje pamięć na strukturę `board_t` i zwraca nowo utworzony obiekt.

2. `board_t read_board_file(char* file_name).`

   - Argumenty:
     - `filename` - nazwa pliku konfiguracyjnego wejciowego.
   - Wartość zwracana:
     - Plansza reprezentująca stan generacji opisanej w pliku konfiguracyjnym wejściowym(obiekt struktury `board_t`)
   - Działanie:
     - Funkcja wywołuje `make_board` i do nowo stworzonej struktury przepisuje stany kolejnych komórek ('0' w pliku konfiguracyjnym oznacza komórkę martwą, a '1' żywą).

3. `board_t free_board(board_t b).`

   - Argumenty:
     - `b` - plansza wskazująca na pamięć, którą program zwolni.
   - Wartość zwracana:
     - Null
   - Działanie:
     - Funkcja zwalnia pamięć wskazywaną przez argument wywołania.

4. `void print_board(FILE* out, board_t b).`

   - Argumenty:
     - `out` - strumień wyjściowy, na który program wypisze reprezentację planszy.
     - `b` - plansza, której reprezentację chcemy wypisać na podany strumień wyjściowy.
   - Wartość zwracana:
     - Void
   - Działanie:
     - Funkcja wypisuje reprezentację planszy na zadany strumień wyjściowy.

2.2 Simulator

Odpowiada za symulację kolejnych generacji komórek. Zawiera następujące funkcje:

1. `board_t simulate_generation(board_t previous_generation, int rules[],int neighbourhood).`
   - Argumenty:
     - `previous_generation` - aktualny stan planszy,
     - `rules` - zasady życia komórek,
     - `neighbourhood` - przyjęta zasada sąsiedztwa.
   - Wartość zwracana:
     - Plansza reprezentująca stan nowej generacji po przeprowadzeniu symulacji (obiekt struktury `board_t`)
   - Działanie:
     - Funkcja na podstawie podanych argumentów, symuluje przejcie komórek do następnej generacji. Symulacja wykonywana jest poprzez iteracyjne sprawdzanie ilości żywych sąsiadów (w podanej jako argument planszy reprezentującej stan generacji przed symulacją) i ustalanie (na podstawie danych zawartych w zasadach podanych jako argument wywołania funkcji) czy komórka przeżywa, umiera lub się ożywa. Nowy stan każdej komórki zapisywany jest do nowej planszy.
2. `int count_neighbour_alive(int x,int y, int neighbourhood).`
   - Argumenty:
     - `x,y` - współrzędne komórki, dla której zliczana jest iloć żywych sąsiadów,
     - neighbourhood - przyjęta zasada sąsiedztwa.
   - Wartość zwracana:
     - Ilość żywych sąsiadów komórki o współrzędnych podanych jako argumenty wywołania funkcji.
   - Działanie:
     - Funkcja zlicza ilość żywych sąsiadów komówki o podanym położeniu według zadanych zasad sąsiedztwa (Moore'a lub von Neumanna).
3. `int what_happens_with_cell(int neighbours_alive, int rules[]).`
   - Argumenty:

- `neighbours_alive` - ilość żywych sąsiadów komórki,
- `rules` - przyjęte zasady przżywania komórek.

- Wartość zwracana:
  - Liczba określająca czy komórka przeżywa (2), umiera (0) czy ożywa (1).
- Działanie:
  - Funkcja na podstawie podanej ilości żywych sąsiadów ustala (korzystając z przyjętych zasad przeżywania komórek) czy komórka przeżyje, umrze lub ożyje.

2.3 Png generator

Odpowiada za generowanie obrazów `.png` przedstawiających stan kolejnych generacji. Zawiera następujące funkcje:

1. `void process_file(board_t b).`

   - Argumenty:
     - `b` - plansza reprezentująca stan generacji, dla której wygenerowany zostanie obraz `.png`.
   - Wartość zwracana:
     - Void
   - Działanie:
     - Funkcja ustawia parametry obrazu wynikowego na podstawie podanej jako argument planszy (stanu generacji).

2. `void genarate(char* file_name).`

   - Argumenty:
     - `file_name` - nazwa pliku obrazu wyjściowego `.png`.
   - Wartość zwracana:
     - Void
   - Działanie:
     - Funkcja generuje obraz `.png` o podanej jako argument wywołania nazwie.

2.4 Main

Jest modułem sterującym programu. Zawiera następujące funkcje:

1. `void read_rules(FILE* settings_file, int rules[], int neighbourhood)`

- Argumenty:
  - `settings_file` - plik ustawień opisujący ustawienia gry w życie,
  - `rules` - tablica, w której przechowywane będą zasady przeżywania komórek,
  - `neighbourhood` - zmienna, w której przechowywana będzie przyjęta zasada sąsiedztwa.
- Wartość zwracana:
  - Void
- Działanie:
  - Funkcja czyta z pliku zasady sąsiedztwa i umieszcza je w podanej jako argument zmiennej `neighbourhood`. Oprócz tego czyta zasady przeżywania komórek i zapisuje je do podanej jako argument tablicy `rules`.

2. `int main(int argc, char** argv)`

- Argumenty:
  - `argc` - ilość podanych argumentów w wierszu poleceń ,
  - `argv` - argumenty podane przez użytkownika w wierszu poleceń.
- Wartość zwracana:
  - Liczba całkowita. W przypadku błędu: 1, przy poprawnym wywołaniu: 0.
- Działanie:
  - Funkcja steruje wykonywaniem całego programu. Wywołuje funkcje z innych modułów, które razem tworzą spójne działanie.

2.5 Board test

Jest modułem testującym działanie modułu `Board`. Zawiera następujące funkcje:

1. `void should_print_board(FILE* out, board_t b)`

- Argumenty:
  - `out` - strumień wyjściowy, na który powinna zostać wypisana plansza,
  - `b` - plansza, która powinna zostać wypisana.
- Wartość zwracana:

- Void.

- Działanie:
  - Funkcja powinna poprawnie wypisać podaną jako argument planszę na dany strumień wyjściowy za pomocą funkcji `print_board` z modułu `Board`.

2. `void should_show_board_for_correct_board_file(char *file_name)`

   - Argumenty:
     - `file_name` - nazwa pliku, z którego powinna zostać wczytana plansza.
   - Wartość zwracana:
     - Void.
   - Działanie:
     - Funkcja powinna poprawnie przeczytać planszę z podanego (poprawnego) pliku wejściowego, korzystając z funkcji `read_from_board_file` z modułu `Board` oraz wypisać ją na standardowe wyjście.

3. `void should_show_error_message_for_incorrect_board_file (char *file_name)`

   - Argumenty:
     - `file_name` - nazwa błędnego pliku konfiguracyjnego wejściowego.
   - Wartość zwracana:
     - Void.
   - Działanie:
     - Funkcja próbuje przeczytać planszę z niepoprawnego pliku wejściowego, używając funkcji `read_from_board_file` z modułu `Board`. Próba ta powinna zakończyć się niepowodzeniem. W takim przypadku zostanie wypisany komunikat o błędzie.

4. `void should_show_confirmation_for_succesful_memory_ allocation(int height, int width)`

   - Argumenty:
     - `height, width` - wysokość i szerokość planszy, dla której alokowana będzie pamięć.
   - Wartość zwracana:

- Void.

- Działanie:
  - Funkcja próbuje zaalokować pamięć na planszę korzystając z funkcji `make_matrix` z modułu `Board`. Jeżeli ta próba zakończy się powodzeniem, funkcja wypisze komunikat: "Udalo sie".

5. `void should_show_error_message_for_unsuccessful_memory_allocation(int height, int width)`

   - Argumenty:
     - `height, width` - wysokość i szerokość planszy, dla której alokowana będzie pamięć.
   - Wartość zwracana:
     - Void.
   - Działanie:
     - Funkcja próbuje zaalokować pamięć na planszę korzystając z funkcji `make_matrix` z modułu `Board`. Jeżeli ta próba zakończy się niepowodzeniem, funkcja wypisze komunikat o błędzie. Przy wywoływaniu tej funkcji, jako argumenty podawane będą bardzo duże liczby w celu przetestowania zachowania funkcji przy braku potrzebnej pamięci.

6. `int main(int argc, char** argv)`

   - Argumenty:
     - `argc` - ilość podanych argumentów w wierszu poleceń,
     - `argv` - argumenty podane przez użytkownika w wierszu poleceń.
   - Wartość zwracana:
     - Liczba '0'.
   - Działanie:
     - Funkcja steruje działaniem modułu `Board_test`. Wywołuje funkcje testowe.

2.6 Simulation test

Jest modułem testującym działanie modułu `Simulator`. Zawiera następujące funkcje:

1. `void should_show_generation_after_simulation(board_t prevoius_generation, int rules[], int neighbourhood)`

- Argumenty:
  - `previous_generation` - aktualny stan planszy,
  - `rules` - zasady życia komórek,
  - `neighbourhood` - przyjęta zasada sąsiedztwa.
- Wartość zwracana:
  - Void.
- Działanie:
  - Funkcja przeprowadza symulację jednej generacji komórek korzystając z funkcji `simulate_generation` z modułu `Simulator`. Wynikową generację komórek wypisuje na standardowe wyjście.

2. `int main(int argc, char** argv)`

- Argumenty:
  - `argc` - ilość podanych argumentów w wierszu poleceń ,
  - `argv` - argumenty podane przez użytkownika w wierszu poleceń.
- Wartość zwracana:
  - Liczba '0'.
- Działanie:
  - Funkcja steruje działaniem modułu `Simulation_test`. Wywołuje funkcję testową `should_show_generation_after_simulation`.

2.7 Png test

Jest modułem testującym działanie modułu `Png_generator`. Zawiera następujące funkcje:

1. `void should_show_confirmation_for_successful_png_generation (char *filename, board_t b)`

- Argumenty:
  - `filename` - nazwa pliku png,
  - `b` - dane w strukturze `board_t` zawierające planszę przeznaczoną do zapisania w png,
- Wartość zwracana:
  - Void.
- Działanie:

- Funkcja próbuje wygenerować plik png o nazwie podanej jako argument na podstawie podanej planszy za pomocą funkcji `process_file` oraz `generate` z modułu `Png_generator`.

2. `void should_show_error_message_for_unsuccessful_png_generation (char *filename, board_t b)`

   - Argumenty:
     - `filename` - nazwa pliku png,
     - `b` - dane w strukturze `board_t` zawierające planszę przeznaczoną do zapisania w png,
   - Wartość zwracana:
     - Void.
   - Działanie:
     - Funkcja próbuje wygenerować plik png o nazwie podanej jako argument na podstawie podanej planszy za pomocą funkcji `process_file` oraz `generate` z modułu `Png_generator`. Próba ta powinna zakończyć się niepowodzeniem i wypisaniem komunikatu o błędzie.

3. `int main(int argc, char** argv)`

   - Argumenty:
     - `argc` - ilość podanych argumentów w wierszu poleceń ,
     - `argv` - argumenty podane przez użytkownika w wierszu poleceń.
   - Wartość zwracana:
     - Liczba '0'.
   - Działanie:
     - Funkcja steruje działaniem modułu `Png_test`. Wywołuje funkcję testową `should_show_confirmation_for_successful_png_generation` oraz `void should_show_error_message_for_unsuccessful_png_generation`.

3 Testy

Opis funkcji testujących został opisany w poprzednim rozdziale podczas opisu modułów `Board_test`, `Simulation_test` oraz `Png_test`.

Podczas testów będziemy analizować dane, które zostaną zwrócone przez funkcje testujące i na ich podstawie ocenimy działanie naszego programu.

4 Strona techniczna projektu

Używana wersja C
W naszym projekcie zostanie użyty język C w wersji 89.

Używany system operacyjny
Projekt będzie tworzony w systemie Linux.

Użyte biblioteki
Nasz program będzie korzystał z biblioteki `libpng`.

Wersjonowanie
Podczas tworzenia projektu będziemy go wersjonować zaczynając od `0.1`, a gotowa wersja naszego programu będzie miała numer `1.0`. Podczas pracy z Gitem będziemy tworzyć nowy branch za każdym razem gdy będziemy chcieli wprowadzić nową funkcjonalność.