Specyfikacja Implementacyjna
Gra w życie

Michał Sut

1 Informacje ogólne

1.1 Nazwa programu

Automat komórkowy *Life*

1.2 Parametry uruchomienia programu

Program uruchamiany będzie z linii komend, a dane wejściowe oraz opcje wykonania programu wprowadzane są jako lista argumentów uruchomienia programu. Dostępne opcje:

-n liczba generacji - domyślnie: 10
-p przedrostek - domyślnie: "picture"
-q przyrostek - domyślnie: brak
-s sąsiedztwo -s Moore, -s Neumann - domyślnie: "Moore"
-c kolor żywej komórki - domyślnie: "black"

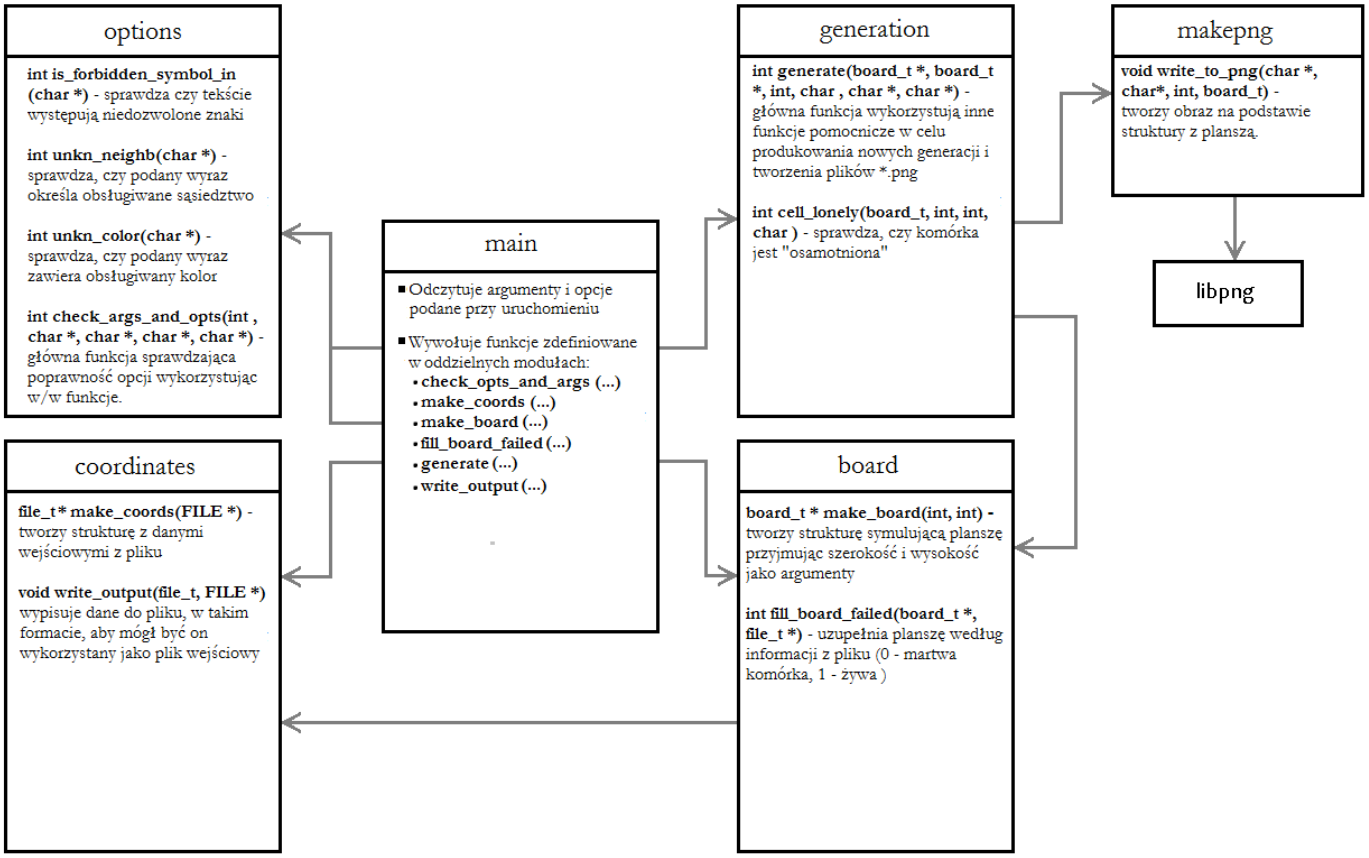
1.3 Wynik działania programu

Program nie prowadzi dialogu z użytkownikiem, a jego poprawne wykonanie skutkuje jedynie utworzeniem wymaganych plików, nie wypisując nic na ekran. Program zwraca komunikaty o błędach w przypadku nieprawidłowych danych lub braku możliwości zaalokowania pamięci. Próba uruchomienia programu z niepoprawnie wpisanymi argumentami i opcjami zakończy się wyświetleniem informacji o poprawnym uruchomieniu programu.

1.4 Wartości zwracane przez program

0 – Program zakończył się poprawnie.
1 – Błąd alokacji pamięci.
2 – Nieprawidłowy format danych w pliku wejściowym
3 – Nieprawidłowe opcje uruchomienia programu.
4 – Brak pliku z danymi o podanej nazwie

2 Powiązania między modułami



3 Opis modułów

3.1 main

Moduł główny. Odpowiada za pobranie argumentów i opcji, oraz obsługę zgłaszanych przez funkcje. Wywoływane są w nim zasadnicze funkcje programu, takie jak tworzenie struktur danych czy przeprowadzenie generacji. W tym celu **main** odwołuje się do modułów: **options**, **coordinates**, **board** oraz **generate**.

3.2 options

Zawiera funkcje sprawdzające poprawność podanych opcji uruchomienia. Wartość zwracana przez funkcję to typ wyliczeniowy. Możliwe zwracane wartości:
`ALL_CORRECT` - Wszystkie wprowadzone opcje poprawne.
`N_NEGATIVE` - Ujemna liczba generacji.
`UNKN_NEIGHB` - Nieznany typ sąsiedztwa.
`UNKN_COLOR` - Nieznany kolor.
`FORBIDDEN_SYMBOLS` - Przedrostek lub przyrostek zawierają nieprawidłowe znaki.

3.3 coordinates

Przechowuje strukturę zawierająca dane z pliku wejściowego oraz dane, które będą wypisane do pliku wyjściowego. Zawiera także funkcje operujące na tej strukturze: stworzenie struktury(z jednoczesnym pobraniem informacji z pliku) oraz wypisanie do pliku.

3.4 board

Moduł realizujący obsługę planszy na której toczy się gra. Zdefiniowane są w nim: struktura planszy, funkcja tworząca i wypełniająca tę strukturę. Wiąże się z: **coordinates**.

3.5 generate

Najważniejsza część programu. Realizuje **n** generacji oraz każdą z nich zapisuje do pliku ***.png**. Powiązana jest z modułem **board** z którego pobiera planszę, oraz **makepng**, z którego wykorzystuje funkcję zapisu do pliku graficznego.

3.6 makepng

Dodatkowy fragment programu stworzony dla modułu **generate** zawierający funkcję zapisującą strukturę planszy do pliku w formacie png.

4 Potrzebne biblioteki

4.1 Biblioteki standardowe

1. stdio.h - podstawowe operacje wejścia/wyjścia;
2. stdlib.h - alokowanie pamięci, NULL, EXIT_SUCCES, EXIT_FAILURE;
3. unistd.h - pobieranie opcji uruchomienia;
4. string.h - operacje na ciągach znaków.

4.2 Biblioteki zewnętrzne

1. libpng.h - tworzenie plików w formacie **png**.

5 Opis struktur danych

1. `file_t` - Struktura zdefiniowana w module **coordinates**. Służy do przechowywania danych wejściowych/wyjściowych.

```
typedef struct s {
    int height, width;      /*wysokość i szerokość planszy podane w dwóch pierwszych
                              liniach w pliku.*/
    int *x;                 //współrzędne X żywych komórek
    int *y;                 //współrzędne Y żywych komórek
    int n;                  //ilość wczytanych żywych komórek
} file_t;
```

2. `board_t` - Struktura zdefiniowana w module **board**. Symuluje planszę do gry za pomocą tablicy dwuwymiarowej liczb całkowitych, traktując 0 jako martwą komórkę, a 1 jak żywą. W pliku *.png wartości te zostaną zastąpione przez odpowiednio białe i (domyślnie) czarne piksele.

```
typedef struct e {
   int height, width;     //wysokość i szerokość planszy pobrane ze struktury file_t
   int **cells;           //dwuwymiarowa tablica
} board_t;
```

6 Testowanie

6.1 Testowanie ogólnego działania program

Pierwszy rodzaj testów ma na celu sprawdzenie czy program prawidłowo tworzy generacje oraz czy zapewniona podstawowa obsługa błędów. Można to zrobić używając następujących metod:

1. Porównywanie otrzymanej generacji z oczekiwaną (uzyskaną za pomocą innej aplikacji przeprowadzającej „Grę w życie")
2. Wykorzystanie specjalnych generacji początkowych, zwanych oscylatorami, które po pewnej liczbie etapów wracają do stanu początkowego.

6.2 Testy modułów

Osobne testy dla modułów sprawdzające ich funkcjonalność.

6.2.1 main

Testy polegające na wprowadzaniu niepoprawnych opcji i błędnych argumentów przy uruchamianiu programu z linii komend.

6.2.2 options

Sprawdzanie zachowania funkcji dla nieprawidłowych i nieznanych argumentów opcji.

6.2.3 coordinates

Używanie różnych plików wejściowych, również takich z niepoprawnym formatem danych i sprawdzanie czy w takich przypadkach zwracane są odpowiednie błędy.

6.2.4 board

Wypełnianie tablicy dwuwymiarowej poprawnie oraz błędnie wypełnioną strukturą `file_t` z modułu **coordinates**.

6.2.5 generate

Sprawdzanie zachowania funkcji dla błędnie stworzonej struktury z planszą.

6.2.6 makepng

Tworzenie plików o różnych wymiarach, nazwach, z poprawnych oraz błędnych planszy, a także o różnych kolorach żywych komórek.