Specyfikacja implementacyjna gry w życie - elohim.c

Michał Balas, Jan Dobrowolski

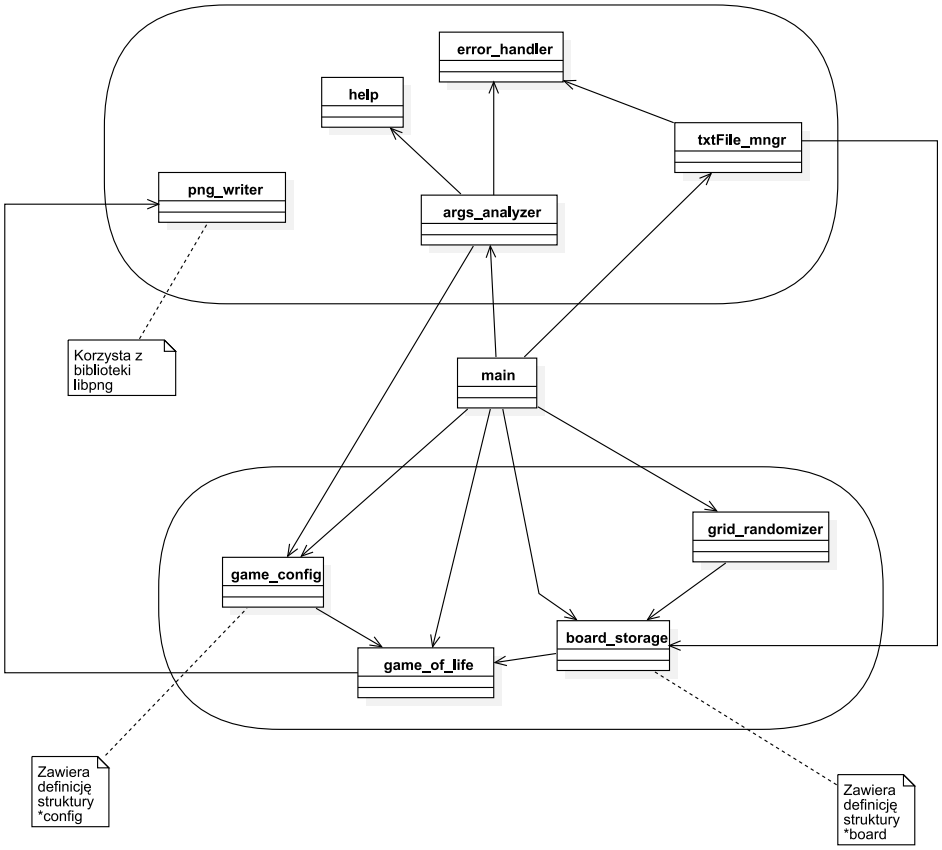
19 marca 2019

Spis treści

Wstęp

Celem działania programu jest symulacja „gry w życie" wg Johna Conwaya na planszy o wymiarach n x m (oraz jej możliwej początkowej konfiguracji). Program został napisany w języku C przy użyciu środowiska Code::Blocks IDE z opcjonalnym wykorzystaniem biblioteki *libpng.h* (przy wywołaniu programu z odpowiednim parametrem) oraz własnych bibliotek stworzonych na potrzeby programu: *util.h* i *game_ mngr.h*.

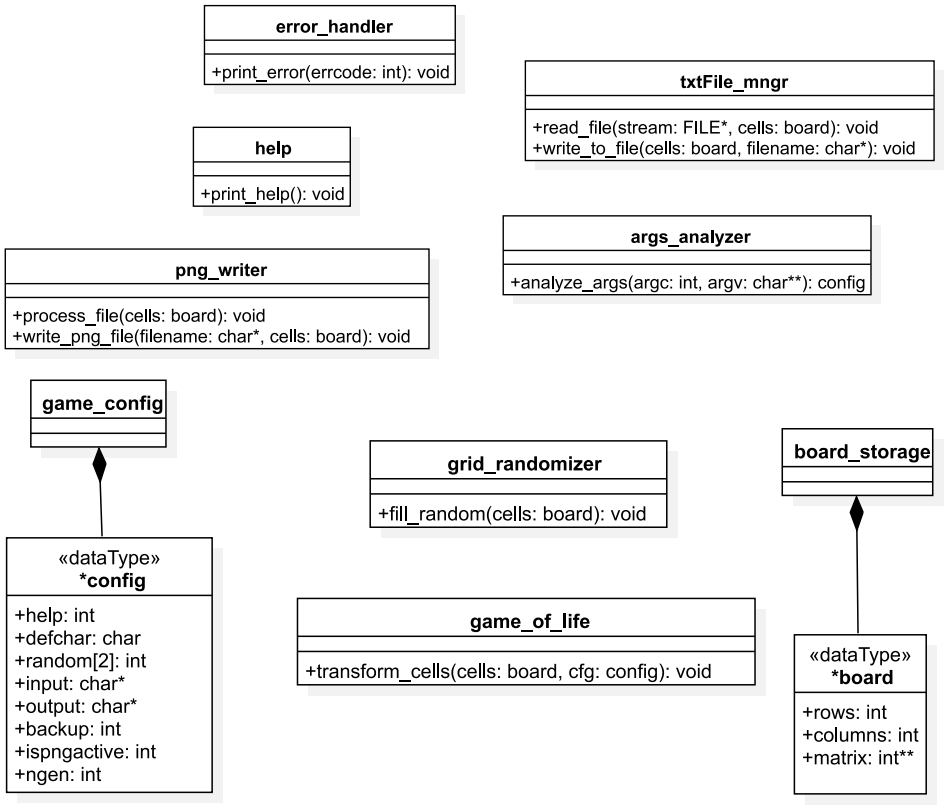
Diagram modułów



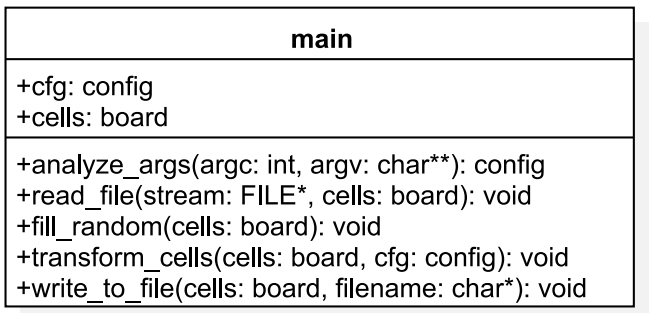
Rysunek 1: *Ogólny zarys modułów programu*

Quasi-prostokątne kształty oznaczają pogrupowanie funkcji na dwie bi-

blioteki: *util.h* - górna oraz *game_ mngr.h* - dolna.



Rysunek 2: *Sygnatury funkcji*



Rysunek 3: *Sygnatury funkcji w module main*

Opis poszczególnych modułów

1. *game_ config* – moduł przechowujący strukturę *config*.
2. *board_ storage* – moduł przechowujący strukturę *board*.
3. *help* – moduł pomocy, tu opisane jest działanie programu oraz jego poprawne użycie.
4. *args_ analyzer* – analizator argumentów. Flagi wywołania programu można podawać w dowolnej kolejności, więc moduł ten sprawdza istnienie danej flagi, poprawność argumentów przy fladze oraz zapisuje te wartości do odpowiednich zmiennych w strukturze *config*.
5. *error_ handler* – obsługa błędów przekazywanych z innych funkcji za pomocą numerycznego kodu. Błędy będą korygowane zgodnie z założeniami programu lub na ich podstawie praca programu będzie przerwana.
6. *txtFile_ mngr* – zawiera funkcje obsługujące strumień plików tekstowych będący możliwym wejściem programu (read_file(FILE*, board)) oraz możliwym wyjściem (write_to_file (char*, board)).
7. *png_ writer* – moduł zawiera funkcje pozwalające na stworzenie obrazu w formacie png na podstawie stanu automatu komórkowego.
8. *game_ of_ life* – moduł umożliwiający tworzenie kolejnych generacji (zawierający funkcję przejścia) oraz zawierający strukturę z parametrami konfiguracyjnymi.
9. *grid_ randomizer* – przechowuje funkcję tworzącą macierz o wymiarach n x m (podanych jako argumenty wywołania funkcji będące elementami struktury *config*) i wypełniającą ją losowo żywymi komórkami.
10. *main* – moduł główny, z niego wywoływane są wszystkie pozostałe funkcje i moduły.

Opis głównego algorytmu

Funkcja *transform_ cells(:board, :config)* – funkcja jako argument przyjmuje strukturę *board* zawierającą dwuwymiarową tablicę (macierz) typu int oraz strukturę *config* zawierającą m.in. liczbę generacji do przeprowadzenia.

1. Funkcja alokuje pamięć na tablicę tymczasową (tmp) o identycznych wymiarach, zaczerpniętych ze struktury *board*.

2. Nowo powstała tablica wypełniana jest zerami oraz znakami podanymi przy fladze *„–defchar”* (lub domyślnie jedynkami) na podstawie konfiguracji komórek tablicy będącej argumentem.

3. Za pomocą funkcji *memcpy* dane macierzy tmp są kopiowane do macierzy zawartej w strukturze *board*;

4. W przypadku zadeklarowania flagi „–png” wywoływany jest moduł *png_ writer*, który tworzy obraz *.png obecnej konfiguracji komórek.

5. Funkcja wywoływana jest z pętlą powtarzającą operacje 2. – 5. n razy (n - liczba generacji automatu). Dzięki temu alokacja pamięci odbywa się tylko raz.

Przepływ sterowania

1. Wczytanie argumentów wywołania programu i ich analiza (funkcja analyze_args) – zapisywanie wprowadzonych danych w strukturze *config* oraz *board*.

   (a) Jeśli wystąpiła flaga –help: wypisanie instrukcji na stdout (funkcja print_help) i przerwanie działania programu.

   (b) Zapisanie wartości podanych argumentów w strukturach *config* i *board*.

2. Czytanie pierwszej generacji z pliku (funkcja read_file) lub losowe generowanie planszy o zadanych wymiarach (fill_random). (Obydwie funkcje są typu void i operują na strukturze *board* i tam zapisują konfigurację będącą zerową generacją automatu).

3. Tworzenie kolejnych generacji (funkcja transform_cells).

   (a) Wywołanie funkcji z argumentami: struktura *board* zawierająca macierz z obecną generacją, struktura*config*.

   (b) Alokacja drugiej tablicy o identycznych wymiarach jak ta podana w argumencie

   (c) Wypełnienie nowo utworzonej tablicy na podstawie funkcji prześcia:

i. Sprawdzenie pozycji komórki (czy jest w narożniku, na krawędzi itp.).
ii. Zbadanie aktualnego stanu komórki (żywa/martwa).
iii. Zbadanie i zliczenie żywych komórek w sąsiedztwie Moore'a.
iv. Ożywienie lub uśmiercenie komórki w następnej generacji – zapis stanu komórki do utworzonej tablicy.
v. Iteracja po wszystkich komórkach w macierzy.

(d) Opcjonalnie (przy podaniu flagi „–png"): utworzenie pliku [numer generacji].png.

4. Jeśli podana została odpowiednia flaga: zapis ostatniej generacji do pliku tekstowego (funkcja write_to_file).

5. Zwalnianie pamięci. Zakończenie działania programu.

Plan testów

Przeprowadzone testy są testami jednostkowymi i dotyczą wyłącznie poszczególnych modułów programu.

a) Test czytania macierzy z pliku tekstowego i wypisywania jej do określonej ścieżki

Parametry wejściowe:

./test1 plik.txt "/outputtest/"

Odpowiednik flag programu:

–input plik_ in.txt –backup plik_ out.txt

Oczekiwany wynik:

Program wczyta macierz z pliku.txt (wypełnioną koniecznie zerami i jedynkami, ponieważ nie ma podanej flagi „–defchar") oraz wypisze ją do pliku txt o nazwie plik_out.txt w folderze, gdzie znajduje się plik wykonywalny.

b) Test funkcji przejścia

Parametry wejściowe

./test2 5 5 2

Odpowiednik flag programu:

–[input macierzy 5x5] –ngen 2

Oczekiwany wynik:

Program wczyta macierz 5x5 i przeprowadzi 2 kolejne generacje dla każdej z nich (modyfikacja - wypisanie na stdout kolejnych stanów automatu). Następnie należy porównać wygenerowane stany z ręcznymi przekształceniami macierzy.

c) Test generowania pliku *.png na podstawie konfiguracji komórek w aktualnym stanie automatu oraz użycia flagi „–defchar”

Parametry wejściowe:

./test3

Odpowiednik flag programu:

–png –defchar #

Oczekiwany wynik:

Program utworzy macierz i wypełni ją znakami # według konfiguracji zawartej w kodzie. Następnie na podstawie macierzy utworzy plik .png wypełniony czarnymi(żywa komórka) lub białymi (martwa) kwadratami.

d) Test tworzenia losowej planszy o zadanych wymiarach

Parametry wejściowe:

./test4 10 10

Odpowiednik flag programu:

–random 10 10

Oczekiwany wynik:

Program utworzy macierz 10x10 i wypełni ją losowo zerami i jedynkami. Następnie wypisze utworzoną tablicę do stdout.