Specyfikacja implementacyjna projektu "Gra w życie"

Krzysztof Kalata, Łukasz Laskowski

7 marca 2019

Spis treści

1 Informacje ogólne **2**
1.1 Język . . . 2
1.2 Konwencja . . . 2
1.3 Wykorzystane dodatkowe biblioteki . . . 2

2 Opis modułów **2**
2.1 GameOfLife . . . 2
2.1.1 Działanie . . . 2
2.1.2 Najważniejsze funkcje . . . 2
2.2 Reader . . . 3
2.2.1 Działanie . . . 3
2.2.2 Najważniejsze funkcje . . . 3
2.3 Simulation . . . 3
2.3.1 Działanie . . . 3
2.3.2 Najważniejsze funkcje . . . 3
2.4 Converter . . . 4
2.4.1 Działanie . . . 4
2.4.2 Najważniejsze funkcje . . . 4
2.5 Generator . . . 4
2.6 Diagram modułów . . . 4

3 Testowanie **4**
3.1 Konwencje . . . 4
3.2 Użyte narzędzia . . . 5

1 Informacje ogólne

1.1 Język

Program zostanie napisany w języku C i będzie przystosowany uruchomienia w standardzie znajdującym się na serwerze ssh: *ssh.jimp.iem.pw.edu.pl*.

1.2 Konwencja

Przyjmujemy, że:

- nazwy modułów zaczynamy wielką literą i następne słowa (bez odstępu) również rozpoczynane są od wielkich liter
- nazwy funkcji zaczynamy małą literą i następne słowa (bez odstępu) rozpoczynane są od wielkich liter
- nazwy zmiennych zaczynamy małą literą i następne słowa (bez odstępu) rozpoczynane są również małymi literami

1.3 Wykorzystane dodatkowe biblioteki

Poza standardowymi bibliotekami typu stdio.h czy stdlib.h, w programie wykorzystamy także biblioteki:

- time.h
- png.h

2 Opis modułów

2.1 GameOfLife

2.1.1 Działanie

Jest głównym modułem, którego zadaniami są:

- wywoływanie działania pozostałych modułów,
- interpretacja otrzymanych flag przy uruchamianiu programu,
- przekazywanie wydobytych z wejścia informacji do odpowiednich funkcji.

2.1.2 Najważniejsze funkcje

Jako że moduł ten "prowadzi" cały program, potrzebuje on tylko jednej funkcji:
`main()` – to w tej funkcji sprawdzane są flagi, otwierane są uchwyty do plików, sprawdzenie, czy zostało to przeprowadzone bez problemu oraz przekazanie uchwytu do pliku wejściowego do *Readera*.
Funkcja następnie wywołuje działanie głównego modułu, przeprowadzającego generacje, a po każdym wykonanym kroku (w przypadku flagi -sbs jedynie po wybranych) uruchamia funkcje konwertujące wyniki do odpowiednich rozszerzeń oraz plików.

2.2 Reader

2.2.1 Działanie

Moduł czytający dane, którego rolę można opisać następująco:

- analizuje plik wejściowy
- odczytuje dane z pliku wejściowego
- tworzy struktury potrzebne do przechowywania siatki w pamięci
- uzupełnia nowo stworzoną planszę informacjami z pliku wejściowego

2.2.2 Najważniejsze funkcje

Moduł ten również posiada dwie funkcje:
`readFile()`– po otrzymaniu ścieżki do pliku wejściowego, analizuje poprawność danych w pliku oraz wpisuje je do utworzonej struktury.
`checkFile()`– funkcja ta otrzymuje ścieżkę do pliku wejściowego jeszcze przed funkcją readFile(), aby sprawdzić poprawność danych. Sprawdzane jest przede wszystkim to, czy w pliku podane są wymiary planszy (jako dwie pierwsze liczby) oraz czy ciąg opisujący plaszę składa się wyłącznie z zer i jedynek.

2.3 Simulation

2.3.1 Działanie

Jest to serce *Gry w życie*. To w tym module następują takie czynności jak:

- pobranie od readera planszy wraz z informacją o liczbie iteracji do wykonania
- przygotowuje kopię planszy, niezbędną do prostego przeprowadzenia symulacji
- tworzy odpowiednią ilość generacji
- komunikuje się w modułem eksportującym wyniki każdej iteracji do plików png i txt

W module tym operujemy na samej planszy, zatem to też tu pojawi się definicja struktury spełniającej rolę swego rodzaju pojemnika, zawierającego wszystkie niezbędne informacje o aktualnej siatce:

```
struct b{
int height, width;
int **life, **copy;
} *board;
```

2.3.2 Najważniejsze funkcje

`generate()`– funkcja ta aktualizuje stany wszystkich komórek na planszy
`copy()`– funkcja przenosząca dane z wektora *life* na wektor *copy*

2.4 Converter

2.4.1 Działanie

Jest to moduł tworzący obrazy png oraz odpowiadające im pliki txt na podstawie planszy otrzymanej z przeprowadzonych symulacji. Pliki txt tworzone są po to, by móc uruchomić program od wybranej iteracji, bez konieczności tworzenia oddzielnego testu.

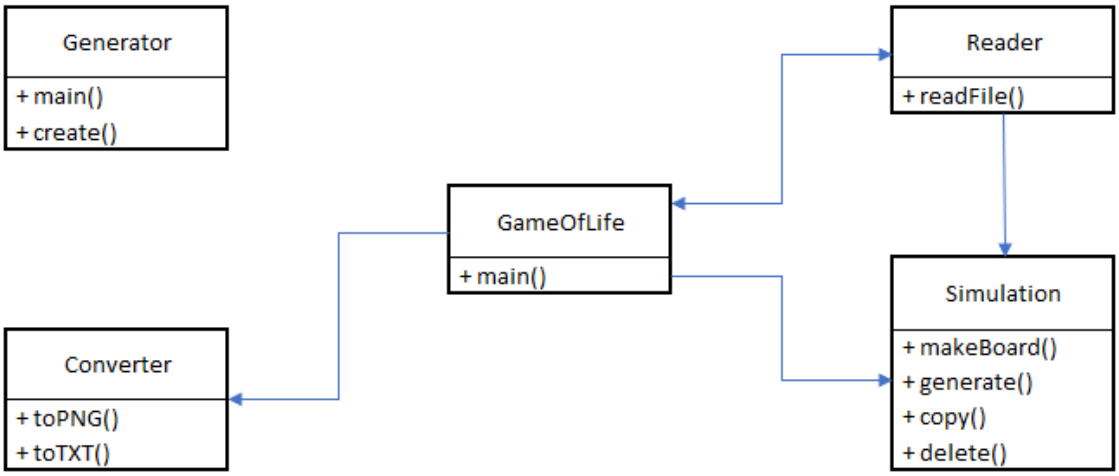
2.4.2 Najważniejsze funkcje

`toPNG()` – funkcja eksportująca planszę do obrazu formatu PNG.
`toTxt()` – funkcja eksportująca planszę do formatu txt.

2.5 Generator

Jest to oddzielny program, który w linii wywołania otrzymuje dwie wartości – wysokość i szerokość planszy. Na ich podstawie tworzy plik tekstowy w formacie odpowiadającym plikom wejściowym *Gry w życie*, uzupełniając go losowo symbolami '1' i '0'.

2.6 Diagram modułów



3 Testowanie

3.1 Konwencje

Testy będą przebiegały w kilku fazach:

- testowanie poprawnego współdziałania modułów, poprzez dostarczenie im wygenerowanych losowo w Generatorze plansz i obserwowanie w trybie *steb-by-step* wczytywania, poprawności zapisu, przekazywania struktur itp.
- testowanie ogólnego działania algorytmów modułu Simulation. W tej części testy będą odbywać się na odgórnie zdefiniowanych planszach z wbudowanymi strukturami typu *Niezmiennych* (klocek, łódź, bochenek), *Oscylatorów* (żabka, krokodyl), czy *Statków* (Glider, Dakota) i obserwowanie, czy zachowują się poprawnie.
- wypisywanie plansz na konsolę lub do plików tekstowych w celu sprawdzenia poprawności przeprowadzanych iteracji

3.2 Użyte narzędzia

Poza testowaniem poprawności działania algorytmów, przetestujemy program pod względem poprawnego zarządzania pamięcią. Użyjemy do tego dwóch narzędzi:

- program Valgrind na platformie Linux
- program Dr. Memory na platformie Windows