Specyfikacja Implementacyjna dla projektu pt. „Gra w życie”

...

Wykonali: Franciszek Wysocki, Bartosz Zdybel
Sprawdzający: mgr inż. Paweł Zawadzki
Data: 08.03.2019

.

Spis treści

1 Wstęp

Dokument ten przedstawia szczegóły dotyczące implementacji projektu pt. „Gra w Życie" oraz wytyczne dotyczące wersjonowania, środowiska, testów itd... (Więcej o działaniu opisane zostało w dokumencie „Specyfikacja funkcjonalna").

2 Informacje ogólne

Program będzie zawierał interfejs tekstowy. Użytkownik nie będzie prowadził z nim dialogu, a wszystkie parametry będzie musiał podać przy uruchamianiu. Wymagane będzie podanie nazwy pliku wejściowego jako pierwszego argumentu wywołania z ewenetualnymi flagami. Program będzie domyślnie rysował pierwszą i ostatnią iterację (w przypadku braku podania ilości iteracji będzie wykonana tylko jedna).

3 Środowisko deweloperskie

Program będzie tworzony równolegle:

- na komputerze Dell Inspiron 5584 z procesorem Intel(R) Core(TM) i7-8565U CPU @ 1.80Ghz 1.99Ghz z zainstalowaną pamięcią RAM 16 GB działającym na systemie Windows 10 Home (Wersja 10.0.18362.657) z subsystemem (Windows subsystem for Linux) Ubuntu 18.04 LTS, (na którym będzie odbywać się kompilacja).
- na komputerze Dell Vostro i7 z procesorem Intel(R) Core(TM) i7-8565U CPU @ 1.80Ghz 1.99Ghz z zainstalowaną pamięcia RAM 8 GB działającym na systemie Linux Ubuntu 18.04.

Kod będzie pisany w języku C (w standardzie ANSI).

4 Zasady wersjonowania

Wersjonowanie odbędzie się za pomocą gita (w formie dodanych tagów):

- „STABLE_n" - wersje stabilne, gdzie n = 1.0, 2.0, ...,
- „i" - dla kolejnych wersji na głównej gałęzi i = n.1, n.2, ..., gdzie n to ostatnia wersja stabila (1, 2, ...) lub 0 jeżeli taka jeszcze nie istnieje,
- „FINAL_RELEASE" - wesja finalna,

Wiadomości z „commitów" będą pisane w języku angielskim, a czasowniki będą występowały w formie gerund.

Np. Adding file operation (reading width, height, ...).

Praca na modułach będzie odbywać sie równolegle na gałęziach. Do scalania wykorzystywane będzie polecenie git merge.

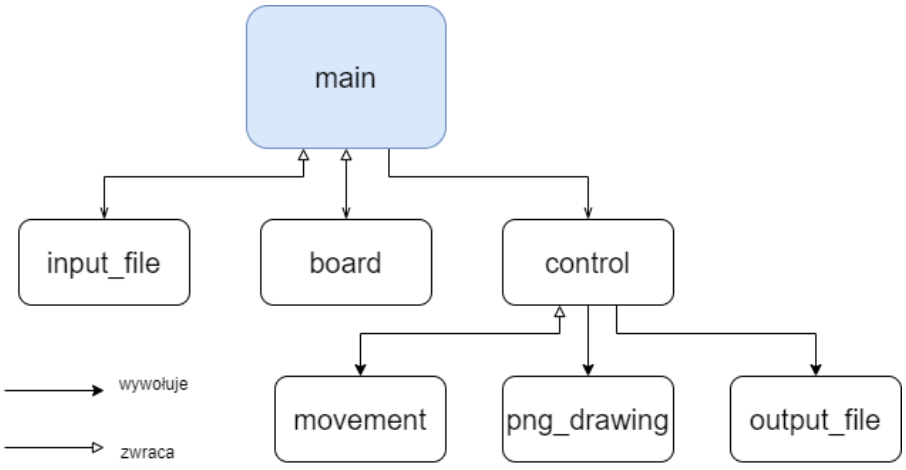
5 Moduły

5.1 Opis modułów

W programie zostanie wyróżnionych 7 poniższych modułów:

- main - będzie odpowiadał za uruchomienie całej procedury (sterowanie)
- control - będzie odpowiadał, za odczytanie flag i ich obsługę
- input_file - będzie odpowiadał za odczytanie danych z pliku wejściowego i sprawdzał jego poprawność
- output_file - będzie odpowiadał za zapisanie danych do pliku wyjściowego
- board - będzie odpowiadał za utworzenie tablicy (siatki) na stercie i jej wypełnienie + będzie znajdować sie tam funkcja rysująca siatkę
- png_drawing - będzie odpowiadał za wygenerowanie obrazu o formacie png
- movement - moduł ten będzie podzielony na dwa mniejsze: moor i neumann, a dany z nich będzie uruchamiany w zależności od rodzaju sąsiedztwa, które użytkownik będzie chciał zastosować

5.2 Diagram modułów



Rysunek 1

6 Argumenty wywołania

Podczas uruchamiania będzie można korzystać tylko z poniższych flag w dowolnej kolejności:

- **-SBS** - Zostaną narysowane (na wyjściu standardowym) generacje w każdej iteracji,
- **-N** - Zostanie wykonane n iteracji i wygenerowanie n +1 obrazów o formacie png (jeden z generacją inicjacyjną). Konieczny będzie kolejny argument (liczba n).
- **-S** - Zostanie nadpisany plik, do którego wpisany zostanie stan ostatniej generacji, tak aby plik ten mógł służyć jako plik wejściowy przy kolejnym uruchomieniu. Konieczny będzie kolejny argument (nazwa pliku).

Każdy inny argument (oprócz pierwszego, którym będzie nazwa pliku wejściowego) spowoduje zakończenie działania programu i wyświetlenie stosownego komunikatu.

7 Testowanie

Do każdego stworzonego modułu zostanie dodana jednostka testująca test_nazwamodułu.c. Każdy z tych testów będzie miał możliwość kompilacji i uruchomienia bezpośrednio z programu make, wywołując „make test_nazwamodułu.out". Testy będą niezależne od innych modułów.

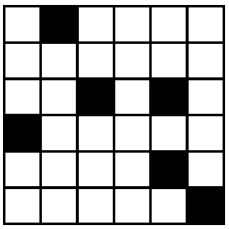
8 Struktury danych

Wykorzystanie zostanie przede wszystkim tablica dwuwymiarowa na stercie, która będzie odwzorowaniem siatki ze stanem generacji.

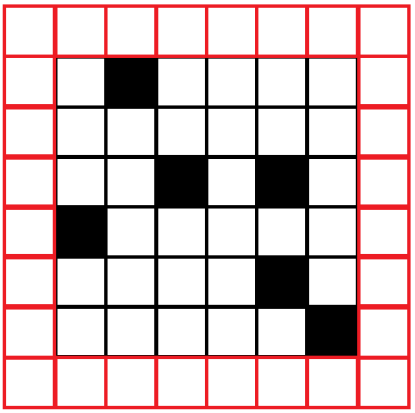
9 Algorytmy

Wykorzystany zostanie wymyślony przez autorów algorytm stosowania sąsiedztwa na krawędziach i rogach. Istniejąca tablica ze sterty (Rysunek 2) będzie przepisywana do tymczasowej (tworzonej na nowo podczas każdej iteracji) tablicy poszerzonej o krawędź z każdej strony (Rysunek 3).

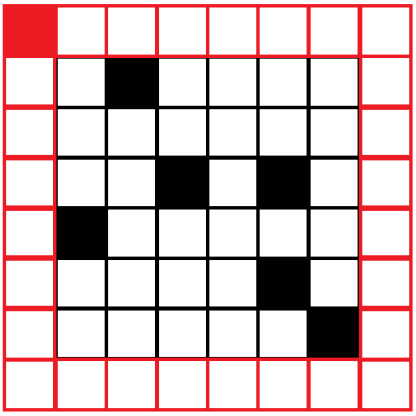
Każdy róg nowej siatki będzie odpowiadał naprzeciwległemu rogowi z istniejącej tablicy (Rysunek 4).



Rysunek 2

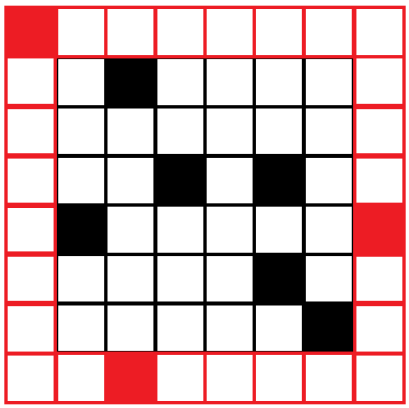


Rysunek 3



Rysunek 4

Każda krawędź nowej siatki zostanie przepisana z naprzeciwległej krawędzi z istniejącej tablicy (Rysunek 5). Funkcja odpowiadająca za ruch będzie zliczała ilość „żywych sąsiadów” na tablicy tymczasowej (dla komórek nieznajdujących



Rysunek 5

się w rogach i na krawędziach) i odrazu będzie modyfikowała tablicę na stercie zgodnie z zasadami gry i przyjętym rodzajem sąsiedztwa.