Sprawozdanie końcowe z projektu automatu komórkowego "Game of life"

Krzysztof Kalata, Łukasz Laskowski

28 marca 2019

Spis treści

1 Opis ogólny

1.1 Nazwa programu

Program nazywa się "gameOfLife".

1.2 Poruszany problem

Celem projektu było napisanie programu pozwalającego zasymulować *Grę w życie*.
Jest to przykład automatu komórkowego symulującego na swój sposób życie komórek. Każda komórka znajdować się może w jednym z dwóch stanów – może być żywa lub martwa. Jako że umieszczone są w prostokątnym układzie współrzędnych, każda komórka niepołożona przy brzegu posiada dokładnie 8 sąsiadów. Komórka żywa pozostaje, taką wyłącznie posiadając 2 lub 3 żywe komórki. W każdym innym przypadku komórka umiera z samotności lub zatłoczenia. Jeśli komórka była martwa, ale posiadała dokładnie trzech żywych sąsiadów w jednej iteracji, w kolejnej jej miejsce zajmuje komórka nowo narodzona. Ważną kwestią jest także to, że sąsiedztwo można definiować na dwa sposoby – wliczając wszystkie 8 komórek (Moore'a) lub licząc tylko te bezpośrednio stykające się krawędziami z obserwowaną komórką (von Neumanna).

1.3 Przykłady uruchomienia

Poniżej podane są przykładowe uruchomienia programu, a następnie opis tego, co nastąpi po ich wywołaniu.

- ./gameOfLife in.txt /first -n 100 -s 1
  Program odczyta plik wejściowy in.txt z katalogu z grą, wykona 100 generacji w trybie step-by-step z których wybrane przez użytkownika zapisze w katalogu o nazwie *first* (katalog musi istnieć) sto par plików .txt i .png ponumerowanych kolejno (0001, 0002, 0003, 0004,0005...), chyba że wcześniej program osiągnie stan ustalony, wtedy przerwie generowanie kolejnych.
- ./gameOfLife in.txt /first -n 100
  Program odczyta plik wejściowy *in.txt* z katalogu z grą, wykona 100 generacji, które zapisze w katalogu o nazwie *first* (katalog musi istnieć) sto par plików .txt i .png ponumerowanych kolejno (0001, 0002, 0003, 0004, 0005...) chyba że wcześniej program osiągnie stan ustalony, wtedy przerwie generowanie kolejnych.
- ./gameOfLife in.txt /first
  Program odczyta plik wejściowy *in.txt* z katalogu z grą, wykona 50 (wartość podstawowa) generacji, które zapisze w katalogu o nazwie *first* (katalog musi istnieć) pięćdziesiąt par plików .txt i .png ponumerowanych kolejno (0001,0002,0003,0004,0005...), chyba że wcześniej program osiągnie stan ustalony, wtedy przerwie generowanie kolejnych.
- ./gameOfLife in.txt
  Symulacja się nie wykona, a program wyświetli jedynie instrukcję uruchamiania.
- ./gameOfLife /first
  Symulacja się nie wykona, a program wyświetli jedynie instrukcję uruchamiania.
- ./gameOfLife in
  Symulacja się nie wykona, a program wyświetli jedynie instrukcję uruchamiania.

2 Osiągnięte założenia i zmiany

2.1 GameOfLife

2.1.1 Założenia

Moduł ten jest trzonem programu, kieruje on jego działaniem od początku, aż do samego końca. Na początku analizuje otrzymane flagi, zwraca odpowiednie komunikaty błędu lub kolejno uruchamia wczytywanie pliku, przeprowadzenie symulacji i na końcu zwalnia zajętą pamięć.

2.1.2 Zmiany względem specyfikacji

Teraz to nie ten moduł uruchamia funkcje konwertujące wyniki do plików wyjściowych w formatach .png i .txt. W końcowej wersji działanie to zostało przesunięte do modułu *Simulation*.

2.2 Reader

2.2.1 Założenia

Ta część programu po otrzymaniu ścieżki pliku wejściowego sprawdza poprawność danych, rezerwuje pamięć pod strukturę, jak i dwie tablice (life i copy) o zadanych wymiarach. Następnie wypełnia je danymi pobranymi z pliku.

2.2.2 Zmiany względem specyfikacji

Brak zmian modułu w stosunku do specyfikacji.

2.3 Simulation

2.3.1 Założenia

To tu wykonywana jest największa część pracy programu. Moduł ten wykonuje operacje na planszy gry, analizuje jej stan, na jego podstawie tworzy kolejne generacje na kopii planszy, powtarza to zadaną liczbę razy, w międzyczasie sprawdzając, czy nie nastąpił stan ustalony. Dodatkowo obsługuje moduł step-by-step (sbs) oraz wywołuje funkcje konwersji do plików wyjściowych o rozszerzeniach .txt i .png.

2.3.2 Zmiany względem specyfikacji

Względem specyfikacji działanie modułu zostało rozszerzone o obsługę modułu step-by-step oraz o wywoływanie funkcji konwersji do plików wyjściowych.

2.4 Converter

2.4.1 Założenia

Moduł ten składa się z dwóch funkcji, które wywołuje *Simulation*. Jedna z nich zajmuje się konwersją do pliku .txt. Tworzy jego nazwę oraz wypełnia go danymi, a następnie zapisuje go w określonym miejscu. Druga natomiast analogicznie tworzy i zapisuje plik .png.

2.4.2 Zmiany względem specyfikacji

Brak zmian w stosunku do specyfikacji.

2.5 Generator

2.5.1 Założenia

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'.

2.5.2 Zmiany względem specyfikacji

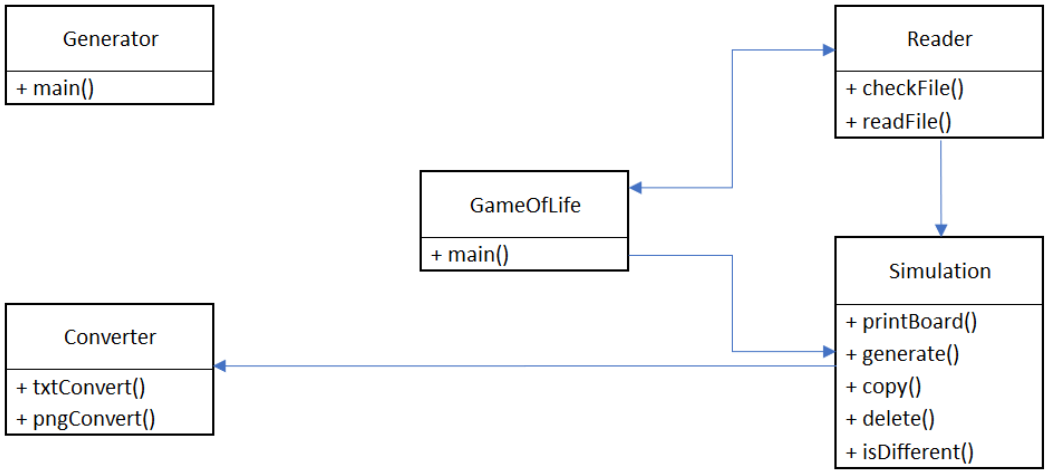
Brak zmian w stosunku do specyfikacji.

2.6 Pozostałe zmiany

W programie musieliśmy dodać także biblioteki, których nie przewidzieliśmy podczas pisania specyfikacji. Są to biblioteki:

- string.h – do podstawowych działań na "stringach" w C (strcpy, strcat, strlen)
- getopt.h – do obsługi załączanych flag (getopt)

Nie musieliśmy natomiast używać biblioteki time.h, ponieważ zmieniliśmy metodę tworzenia nazw plików wyjściowych na zwykłe numerowanie (0001, 0002...), zamiast godziny i daty, ponieważ problem stanowiłaby sytuacja, gdy kilka plików wygenerowałoby się w jednej sekundzie.
Zmianom uległ również lekko diagram modułów, a aktualny prezentuje się następująco:



3 Opis załączonych plików

3.1 Moduły programów

Wszystkie pliki składające się na modułu programu zz wyjątkiem Generatora znajdują się w katalogu głównym w dwóch wersjach (.h oraz .c). Generator znajduję się w podkatalogu */in* jedynie w wersji .c.

3.2 Dokumentacja

Dokumentacja funkcjonalna, jak i implementacyjna znajdują się w podkatalogu */Dokumentacja*, każda w dwóch wersjach (.pdf oraz .tex).

3.3 Testy

Przykładowe testowe plansze znajdują się w podkatalogu *./in/testy struktur*, gdzie przechowywane są w formacie .txt. Natomiast testy poszczególnych modułów znajdują się w katalogu głównym i ich nazwy zaczynają się od *test_*.