Specyfikacja funkcjonalna projektu *Gra w Życie* w języku C

Chabik Jan (291060), Łuczak Mateusz (291088)

8 marca 2018

Spis treści

1 Opis ogólny

1.1 Nazwa programu

Program nazwany został `Life`. Nazwa nawiązuje do rozwiązywanego problemu, którym jest stworzenie emulatora *Gry w Życie* Johna Conwaya.

1.2 Poruszany problem

Poruszanym problemem będzie stworzenie emulatora *Gry w Życie* Johna Conwaya w języku C.

Gra w życie jest jednym z pierwszych i najbardziej znanych automatów komórkowych. Pozwala ona zrozumieć koncepty matematyczne takie jak układy sąsiadów i reguły zmian, a rozgrywa się na planszy, na której istnieje skończona ilość komórek. Każda z nich może się znajdować w stanie "żywym" lub "martwym". W jednostce czasowej, każdej komórce przypisywany jest stan na podstawie stanów jej sąsiadów. Rozwiązaniem będzie N wygenerowanych plików opisujących kolejne generacje automatu komórkowego dla zadanego przez użytkownika wejścia.

1.3 Cel projektu

Celem projektu jest zapoznanie się przez studentów z problemem *Gry w Życie* Johna Conwaya oraz stworzenie implementacji w języku C. Zadanie przewidziane zostało dla grup dwuosobowych. Ze względu na to, ważna jest umiejętność pracy w zespole.

2 Opis funkcjonalności

2.1 Korzystanie z programu

Program po skompilowaniu powinien zostać uruchomiony w środowisku tekstowym wraz z koniecznymi argumentami opisującymi przykładowo plik wejściowy, plik docelowy, wybór sąsiedztwa czy ilość generacji automatu komórkowego.

2.2 Uruchomienie programu

Przy kompilacji programu za pomocą programu `Make` zostanie utworzony plik wynikowy `gameoflife`.

`./gameoflife <input_file.txt> <katalog_wyjsciowy> [-s 1/2] [-n ilosc_generacji] [-g numer_pierwszej] [-d] [-h --help]`

- `<input_file.txt>` - Plik wejściowy w formacie .txt zawierający dane dotyczące mapy automatu komórkowego. Zawiera on wymiary macierzy oraz informacje o występowaniu żywych komórek.
- `<katalog_wyjsciowy>` - Katalog, w którym zostaną zapisane dane wynikowe programu w formacie `.txt` oraz `.png`.
- `[-s sasiedztwo]` - Wybór pomiędzy wykorzystaniem sąsiedztwa Moore'a (1) i sąsiedztwa von Neumanna (2). W przypadku, jeżeli nie został użyty ten argument wywołania, zostanie użyte sąsiedztwo domyślne zapisane w kodzie źródłowym programu.
- `[-n ilosc_generacji]` - Ilość generacji do stworzenia. W przypadku, jeżeli nie został użyty ten argument wywołania, zostanie użyta domyślna wartość zapisana w kodzie źródłowym programu.
- `[-g numer_pierwszej]` - Numer pierwszej generacji (z pominięciem wprowadzanej przez dane wejściowe; ona zostanie uznana jako generacja 0). W przypadku, jeżeli nie został użyty ten argument wywołania, zostanie użyta domyślna wartość zapisana w kodzie źródłowym programu.
- `[-d]` - Ustawienie dialogu z użytkownikiem. Przy użyciu tego argumentu, po każdej stworzonej generacji, program będzie pytał użytkownika, czy ma zostać zapisana do pliku `.txt` oraz `.png`. Domyślnie, bez użycia argumentu, opcja jest wyłączona.
- `[-h]` lub `[--help]` - Wyświetlenie składni programu. Nie może być uruchomiony z innymi argumentami.

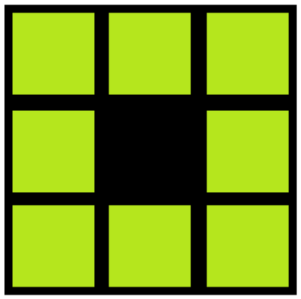
2.3 Możliwości programu

- Emulacja *Gry w Życie* Johna Conwaya;
- Odczyt danych z pliku tekstowego zawierającego informacje dotyczące mapy automatu komórkowego;
- Generowanie wybranej przez użytkownika ilości generacji automatu komórkowego;
- Symulacja automatu dla wybranego przez użytkownika sąsiedztwa (wybór spośród sąsiedztwa Moore'a i von Neumanna);
- Zapis danych wyjściowych do pliku tekstowego oraz pliku `.png`;
- Wyświetlanie planszy do terminala oraz możliwość zapisu do `.png` na życzenie (gdy wybierzemy opcję dialogu z użytkownikiem).

3 Format danych i struktura plików

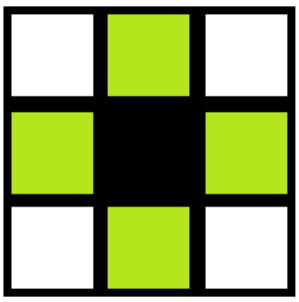
3.1 Słownik pojęć

- **Plasza** - jest to prostokąt podzielony na pola (komórki), z których każda ma przypisaną wartość - 0 albo 1. Gdy komórka znajduje się stanie '0' mówimy, że jest martwa, w przeciwnym wypadku jest żywa;
- **Sąsiedztwo Moore'a** - typ sąsiedztwa polegający na tym, iż rozpatrywane są wszystkie 8 komórek sąsiadujących z daną komórką (komórka na północ, południe, wschód, zachód, północny-wschód, północny-zachód, południowy-zachód, południowy-wschód);



Rysunek 1: Sąsiedztwo Moore'a. Po środku znajduje się komórka (czarna), dla której rozpatrujemy sąsiadów. Komórki zielone opisują, które z nich są sąsiadami w opisywanym sąsiedztwie.

- **Sąsiedztwo Von Neumanna** - typ sąsiedztwa polegający na tym, iż rozpatrywane są 4 komórki sąsiadujące (komórka na północ, południe, wschód i zachód);



Rysunek 2: Sąsiedztwo von Neumanna. Po środku znajduje się komórka (czarna), dla której rozpatrujemy sąsiadów. Komórki zielone opisują, które z nich są sąsiadami w opisywanym sąsiedztwie.

3.2 Struktura katalogów

```
.
|
+- /C/docs - Katalog z dokumentami projektowymi
|
+- /C/bin - Katalog z danymi wyjściowymi kompilacji programu
|
+- /C/src - Katalog z kodami źródłowymi programu
|
+- /C/tests - Katalog z testami dla programu
|
+- /C/mod_tests - Katalog z testami modułów i funkcji programu
```

3.3 Przechowywanie danych w programie

- Dane na temat planszy przechowujemy w tablicy typu char, celem oszczędności pamięci.
- Program przechowuje wskaźnik do pliku, z którego "czyta" dane oraz wskaźnik do katalogu, do którego dodaje pliki `.png` i `.txt`;

3.4 Dane wejściowe

Jako dane wejściowe służy plik w formacie `.txt`, w którym zapisana jest plansza.

Plik rozpoczyna się wymiarami planszy zapisanej za pomocą liczb naturalnych większych od 0. Następnie podawane są cyfry '0' i '1' oznaczające stan komórki (martwa lub żywa). Ilość cyfr powinna odpowiadać wielkości planszy. Przykładowo, jeżeli wymiary macierzy to `3x4`, to powinno być podanych 12 cyfr oznaczających stan komórek.

3.5 Dane wyjściowe

- Pliki `.png`, w których znajduje się graficzna interpretacja każdej kolejnej generacji;
- Pliki w formacie `.txt`, w których zapisane są kolejne generacje;
- W przypadku uruchomienia opcji dialogu z użytkownikiem, dane wyjściowe będą wypisywane również w oknie terminala.

4 Scenariusz działania programu

4.1 Scenariusz ogólny

1. Uruchomienie programu z terminala z argumentami określającymi dalsze jego działanie;
2. Sprawdzenie przez program argumentów oraz poprawności danych wejściowych;
3. Zapisanie danych z argumentów do pamięci programu;
4. Stworzenie mapy automatu komórkowego;
5. Tworzenie kolejnych generacji automatu;
6. W przypadku wybrania możliwości dialogu z użytkownikiem:
   - Wyświetlenie stworzonej generacji na ekranie powłoki tekstowej;
   - Pytanie o zezwolenie na zapis danych wyjściowych do `.txt` i `.png`;
   - Ewentualny zapis lub pomięcie zapisu danej generacji.
7. W przypadku uruchomienia programu bez opcji dialogu z użytkownikiem, zapis danych zgodnie z podanymi argumentami lub wartościami domyślnymi;
8. Po każdym zapisie, utworzenie nowej mapy na podstawie zmian automatu komórkowego i powtórzenie kroków 6 lub 7 (zależnie od sposobu uruchomienia programu);

9. Zakończenie pracy programu z komunikatem o ukończeniu pracy.

4.2 Scenariusz szczegółowy

1. Uruchomienie programu z terminala z argumentami określającymi dalsze jego działanie:
    - W przypadku podania błędnych danych wejściowych (błędne argumenty) program zostaje przerwany oraz zostaje wyświetlona poprawna składnia programu;
    - W przypadku podania nieistniejącego pliku wejściowego, lub braku uzyskania do niego dostępu, program zakończy się z komunikatem o błędzie.
2. Sprawdzenie przez program argumentów oraz poprawności danych wejściowych:
    - Sprawdzenie danych wejściowych pod kątem błędnych informacji na temat mapy automatu komórkowego. W przypadku znalezienia nieprawidłowych znaków w macierzy (znaki niebędące opisem komórek, czyli inne niż '0' i '1') program zgłosi problem użytkownikowi, a tym samym jego działanie się zakończy.
3. Zapisanie danych z argumentów do pamięci programu:
    - Zapisanie danych z argumentów do zmiennych i tablic w programie.
4. Stworzenie mapy automatu komórkowego:
    - Stworzenie mapy na podstawie podanych danych wejściowych.
5. Tworzenie kolejnych generacji automatu:
    - Na podstawie wybranego sąsiedztwa, tworzenie kolejnych generacji po przejściach komórek zgodnie z wybranym typem sąsiedztwa.
6. W przypadku wybrania możliwości dialogu z użytkownikiem:
    - Wyświetlenie stworzonej generacji na ekranie powłoki tekstowej;
    - Pytanie o zezwolenie na zapis danych wyjściowych do `.txt` i `.png`;
    - Ewentualny zapis lub pomięcie zapisu danej generacji.
7. W przypadku uruchomienia programu bez opcji dialogu z użytkownikiem, zapis danych zgodnie z podanymi argumentami lub wartościami domyślnymi:

- Na podstawie danych o ilości generacji i numerze pierwszej stworzonej generacji następuje zapis danych wynikowych do wybranego katalogu w formatach `.txt` i `.png`.

8. Po każdym zapisie, utworzenie nowej mapy na podstawie zmian automatu komórkowego i powtórzenie kroków 6 lub 7 (zależnie od sposobu uruchomienia programu);

9. Zakończenie pracy programu z komunikatem o ukończeniu pracy.

5 Testowanie

Testować będziemy każdą funkcję z osobna, potem każdy z modułów programu, a na końcu cały program. Testy będą "ręczne", rezultaty będziemy sprawdzać z pomocą debuggera (gdb). Poprawność generacji planszy będziemy sprawdzać za pomocą strony `https://www.mimuw.edu.pl/~ajank/zycie/`, na której można śledzić działanie automatu komórkowego.