Politechnika Warszawska
Wydział Elektryczny

Specyfikacja funkcjonalna symulatora automatu komórkowego *Life*

Autorzy:
J. Korczakowski, nr albumu 291079
B. Suchocki, nr albumu 291111
Grupa projektowa nr 11

3 marca 2018

1 Opis ogólny

1.1 Nazwa programu

Nasz program będzie nazywał się `AutomatKomorkowyLIFE`.

1.2 Poruszany problem

Zadaniem naszego programu będzie symulacja automatu komórkowego Life, czyli symulacja popularnej „Gry w życie" wymyślonej przez Johna Conwaya w 1970 roku. Gra toczy się na planszy podzielonej na kwadratowe komórki. Każda z komórek może znajdować się w stanie martwym lub żywym. Przy przejściu do następnej generacji komórek, każda z nich może przeżyć, umrzeć lub ożyć w zależności od bieżącego stanu jej sąsiadów. Przyjmujemy, że komórka ma 8 (sąsiedztwo Moore'a) lub 4 (sąsiedztwo von Neumanna) sąsiadów, w zależności od przyjętych zasad sąsiedztwa. To, ile żywych sąsiadów musi mieć komórka, aby przeżyć, umrzeć lub ożyć określają możliwe do zmiany zasady gry.

Dodatkowo, w naszym programie zakładamy, że plansza jest ograniczona i komórki poza nią są martwe. Naszym zdaniem uprości to symulację i zwiększy jej czytelność.

1.3 Cel projektu

Celem tego projektu jest nauka i praktyka programowania w języku C przy jednoczesnym zapoznaniu się z tematem symulacji automatu komórkowego.

2 Opis funkcjonalności

2.1 Korzystanie z programu

Program uruchamiany jest z wiersza poleceń. Obok nazwy programu użytkownik może podać dodatkowe parametry.

2.2 Uruchomienie

Ogólny schemat polecenia do uruchomienia programu:

```
./AutomatKomorkowyLIFE [-i plik_konfig] [-g ile_ generacji]
[-o plik_wyjsciowy]
```

Gdzie:

- `plik_konfig` – plik wejściowy konfiguracyjny (opisujący stan początkowej generacji),

- `ile_generacji` – liczba (naturalna) generacji do przeprowadzenia,
- `plik_wyjsciowy` – plik wyjściowy opisujący konfigurację ostatniej generacji (po symulacji),
- `ile_obrazow` – liczba obrazów przedstawiających stan wybranych generacji.

2.3 Możliwości programu:

- zmiana obowiązujących podczas symulacji zasad sąsiedztwa,
- przeprowadzenie symulacji automatu komórkowego na podstawie wczytanego pliku konfiguracyjnego i zasad sąsiedztwa,
- wyświetlanie bieżących generacji na standardowe wyjście,
- wygenerowanie obrazów *.png przedstawiających stan wybieranych na bieżąco przez użytkownika generacji,
- wygenerowanie pliku wyjściowego opisującego stan ostatniej generacji.

3 Format danych i struktura plików

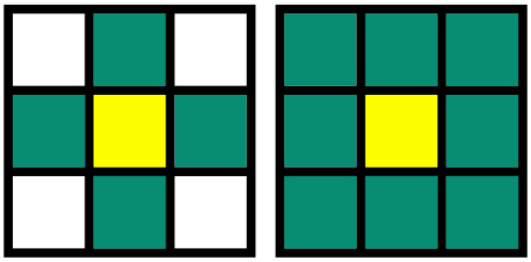
3.1 Pojęcia

Komórka – podstawowa jednostka automatu komórkowego.

Generacja – jeden cykl, w którym komórki rodzą się lub umierają.

Stan komórki – komórka może znajdować się w dwóch stanach, żywym (włączonym) lub martwym(wyłączonym).

Sąsiedzi komórki – komórki przylegające do niej bokami i rogami (sąsiedztwo Moore'a) lub tylko bokami(sąsiedztwo von Neumanna).



Rysunek 1: Sąsiedztwo von Neumanna(po lewej) oraz Moore'a(po prawej).

3.2 Struktura katalogów

Projekt będzie znajdował się w katalogu `Symulator`. Pliki z kodem źródłowym i plikami nagłówkowymi znajdą się w katalogu `src`. W katalogu `data` umieścimy domyślny plik wejściowy. Dane testowe będą znajdowały się w katalogu `test` w katalogu `data`. Wyniki działania naszego programu(plik wyjściowy i pliki .png) umieścimy w katalogu `bin`. W katalogu głównym naszego projektu znajdzie się również plik `Makefile`.

Graficzne przedstawienie struktury katalogów w naszym projekcie:

```
Symulator
+--- src
+--- data
|    +--- test
+--- bin
+--- Makefile
```

3.3 Przechowywanie danych w programie

Plansza przeznaczona do symulacji będzie trzymana w strukturze. Wyróżnimy w niej pola takie jak, dwuwymiarowa tablica typu `char` oraz jej szerokość i wysokość.

3.4 Dane wejściowe

Plik wejściowy konfiguracyjny zawiera w pierwszym wierszu szerokość s i wysokość w planszy a w kolejnych w wierszach po s zer i jedynek. Liczba 0 odpowiada martwej komórce, a 1 żywej.

Plik ustawień zawiera w pierwszym wierszu literę M lub N (domyślnie M), odpowiadającą użytej zasadzie sąsiedztwa (Moore'a lub von Neumanna), a w drugim wierszu opis ustawień reguł „gry w życie" w formacie $x_1x_2 \ldots x_n$ /$y_1y_2 \ldots y_m$, gdzie $x_1x_2 \ldots x_n$-liczby komórek w sąsiedztwie, dla których żywe komórki przeżywają, $y_1y_2 \ldots y_m$ – liczby komórek w sąsiedztwie, dla których martwe komórki ożywają.

3.5 Dane wyjściowe

Pliki wyjściowe naszego programu dzielą się na dwa rodzaje. Jeden rodzaj to stan bieżącej generacji zapisany w formacie analogicznym do pliku wejściowego przyjmowanego przez nasz program. Drugi rodzaj to pliki .png zawierające obrazy generacji, których zażądał użytkownik.

4 Scenariusz działania programu

4.1 Scenariusz ogólny

1. Uruchomienie programu z wiersza poleceń.
2. Sprawdzenie parametrów wywołania.
3. Otwarcie plików: wejściowego konfiguracyjnego, ustawień oraz pliku wyjściowego.
4. Wygenerowanie planszy.
5. Symulacja podanej liczby generacji, wypisywanie bieżącej i ewentualne (decyduje o tym użytkownik) generowanie plików .png przedstawiających stan aktualnej generacji.
6. Wypisanie opisu ostatniej generacji do pliku wyjściowego.
7. Zakończenie działania programu.

4.2 Scenariusz szczegółowy

1. Uruchomienie programu z wiersza poleceń
2. Sprawdzenie podanych przez użytkownika parametrów:
   - Jeśli użytkownik nie poda nazwy wejściowego pliku konfiguracyjnego, przypisana zostanie nazwa pliku domyślnego (`generation_config`).
   - Jeśli użytkownik nie poda liczby generacji do zasymulowania, przypisana zostanie wartość domyślna (20).
   - Jeśli parametr oznaczający liczbę generacji będzie błędny (np. będzie literą), program wypisze komunikat: `Nieprawidłowa liczba generacji` na standardowe wyjście i zakończy działanie.
   - Jeśli użytkownik nie poda nazwy pliku wyjściowego, przypisana zostanie wartość domyślna (`last_generation_config`)
   - Jeśli użytkownik poda nieobsługiwany parametr (np. poprzedzony zamiast `-g` prefiksem `-u`, program go zignoruje
3. Próba otwarcia plików: wejściowego konfiguracyjnego, ustawień oraz pliku wyjściowego
   - Jeśli program nie będzie mógł otworzyć któregoś z pliku (np. przez brak uprawnień), wypisze komunikat: `Nie można otworzyć pliku nazwa_pliku` na standardowe wyjście i zakończy działanie.

4. Wygenerowanie planszy

   4.1. Alokacja pamięci potrzebnej do przechowywania planszy w strukturze.

   - Jeśli wygenerowanie planszy będzie niemożliwe z powodu braku pamięci, program wypisze komunikat: `Wygenerowanie planszy zakończone niepowodzeniem. Spróbuj podać mniejsze rozmiary` na standardowe wyjście i zakończy działanie.

   4.2. Wypełnienie struktury przeprowadzone na podstawie zawartości wejściowego pliku konfiguracyjnego.

   - Jeśli używany do generowania planszy plik wejściowy konfiguracyjny będzie w niewłaściwym formacie, program wypisze komunikat: `Plik wejściowy nazwa_pliku jest w niewłaściwym formacie` na standardowe wyjście i zakończy działanie.

5. Symulacja podanej liczby generacji i równoległe generowanie obrazów przedstawiających stan wybranych przez użytkownika generacji.

   - Jeśli plik ustawień ma błędny format, program zwróci komunikat: `Błędny format pliku ustawień.` na standardowe wyjście i zakończy działanie.

   5.1. Przejście do następnej generacji na podstawie zasad pobranych z pliku ustawień i wypisanie jej stanu na standardowe wyjście.

   5.2. Umożliwienie użytkownikowi podania komendy `n` lub `s`.

   - Jeżeli użytkownik poda jako komendę literę `s`, program wygeneruje plik .png przedstawiający stan bieżącej generacji i przejdzie do symulacji następnej generacji.
     - Jeśli próba wygenerowania któregoś obrazu zakończy się niepowodzeniem, program wypisze komunikat o błędzie na standardowe wyjście, lecz nie zakończy działania.
   - Jeżeli użytkownik poda jako komendę literę `n`, program od razu przeprowadzi symulację następnej generacji.
   - Jeżeli użytkownik wprowadzi komendę `n liczba`, gdzie `liczba` jest liczbą całkowitą, program przeprowadzi symulację tylu generacji, ile wynosi `liczba`, bez bieżącego wypisywania i proszenia o wpisanie komendy.
     - Jeśli `liczba` będzie większa od liczby pozostałych do zasymulowania generacji, program zasymuluje dokładnie taką ilość generacji, jaka pozostała do wykonania.
   - Jeżeli użytkownik poda inną komendę, program wypisze komunikat: `Błędna komenda` na standardowe wyjście i będzie czekać na podanie poprawnej.

6. Wypisanie opisu ostatniej generacji do pliku wyjściowego

7. Zakończenie działania programu

    7.1. Wypisanie komunikatu: `Program zakończył działanie` na standardowe wyjście.

5 Testowanie

Działanie programu w wypadkach błędnie podawanych danych i parametrów przetestujemy podając na wejście (wcześniej przygotowane) błędnie sformatowane lub takie do których program nie ma uprawnień pliki i nieprawidłowe parametry. Następnie przeanalizujemy jak zachowa się nasz program.

Moduł wczytywania macierzy, a także generowanie przez nasz program plików wyjściowych oraz .png sprawdzimy ręcznie, analizując to jak zachowa się nasz program. Działanie symulacji automatu komórkowego przetestujemy poprzez generację przez nasz program macierzy, która będzie znana przez nas z innych źródeł. Porównamy obie macierze i jeśli będą się różnić, to poprawimy nasz program.