Politechnika Warszawska
Wydział Elektryczny

SPECYFIKACJA FUNKCJONALNA
PROJEKT: AUTOMAT KOMÓRKOWY

Autorzy:

Anna Głowińska
numer indeksu: 291070
anna.glowinska98@gmail.com

Adam Czajka
numer indeksu: 291063
czajka.adam147@gmail.com

Praca wykonana w ramach przedmiotu: Języki i metody programowania 2

8 kwietnia 2018

Spis treści

1 Opis ogólny

1.1 Nazwa programu

Life - automat komórkowy symulujący „Grę w życie" Johna Conwaya zbudowany w języku C.

1.2 Podstawowe pojęcia

Komórka - najmniejsza jednostka automatu komórkowego; może znajdować się w stanie martwym (przypisujemy jej kolor biały = 0) lub w stanie żywym (kolor czarny = 1).

Automat komórkowy - uporządkowany zbiór komórek, z których każda znajduje się w jednym z kilku dozwolonych stanów (np. martwy i żywy). Komórki przylegają do siebie tworząc siatki.

Siatka - dwuwymiarowy kawałek płaszczyzny o konkretnych wymiarach, podzielony liniami tworzącymi macierz, składającą się z pojedynczych komórek.

Generacja - Stan wszystkich komórek w danej chwili, ale też czynność zmiany ich stanów.

Sąsiedztwo - graniczenie komórki z innymi komórkami, warunkujące nowy stan tej komórki, wyróżniamy:

- sąsiedztwo Moore'a: 8 przylegających komórek (znajdujących się: na południu, na południowym-zachodzie, na zachodzie, na północnym-zachodzie, na północy, na północnym-wschodzie, na wschodzie i na południowym-wschodzie),
- sąsiedztwo von Neumanna: 4 przylegające komórki (na południu, zachodzie, północy i wschodzie).

1.3 Poruszany problem

Zadaniem automatu komórkowego *Life* jest symulacja „Gry w życie" wymyślonej i opracowanej przez Johna Hortona Conwaya. Gra toczy się na dwuwymiarowej płaszczyźnie podzielonej na kwadratowe komórki. W zależności od przyjętych zasad każda komórka ma czterech sąsiadów przylegających do

niej bokami (sąsiedztwo von Neumanna) lub ośmiu sąsiadów (sąsiedztwo Moore'a), czyli komórki przylegające do niej bokami i rogami. Każda komórka może znajdować się w jednym z dwóch stanów: może być albo żywa (włączona), albo martwa (wyłączona).

Stany komórek zmieniają się przy każdej generacji. Dana generacja jest używana do obliczenia następnej generacji. Po obliczeniu wszystkie komórki zmieniają swój stan dokładnie w tym samym momencie. Stan komórki po generacji zależy tylko od liczby jej żywych sąsiadów.

Zestaw zasad przy tworzeniu nowej generacji (przejścia do następnego stanu) jest następujący:

1. Martwa komórka, która ma dokładnie 3 żywych sąsiadów, staje się żywa w następnym opisie (rodzi się).
2. Żywa komórka z 2 albo 3 żywymi sąsiadami pozostaje nadal żywa; przy innej liczbie sąsiadów umiera (z samotności albo zatłoczenia).

2 Opis funkcjonalności

2.1 Korzystanie z programu

Program *Life* jest kompilowany oraz uruchamiany za pomocą programu `make`, który czyta co ma zrobić z pliku o nazwie `Makefile`, znajdującego się w katalogu `life`.

2.2 Uruchomienie programu

Program `make` wywoływany jest bez poleceń z linii komend. Wszelkie parametry podane przez użytkownika przy wywoływaniu zostaną przez program zignorowane.

2.3 Możliwości programu

Zadania jakie wykonuje program:

- wczytywanie do programu początkowej konfiguracji generacji z pliku w formacie TXT,
- wczytanie danych wejściowych z wbudowanej w program biblioteki,
- przeprowadzenie zadanej liczby generacji,

- zapisywanie bieżącej generacji do pliku TXT, który może zostać potem wczytany,
- zapisywanie bieżącej generacji do pliku PBM.

3 Format danych i struktury plików

3.1 Struktura katalogów

Szkielet projektu został przedstawiony na poniższym schemacie:

```
life
+— dane
|    +— dane.txt
|    +— biblioteka
|    +— wyniki
+— src
|    +— testy
+— bin
+— Makefile
```

gdzie:
`life` - katalog główny,
`dane` - katalog zawierający dane testowe (dane.txt) oraz katalog,
`biblioteka` - zawierający ciekawe przykłady danych wejściowych,
`wyniki` - katalog w którym zostają zapisane bieżące generacje w formacie TXT oraz obrazy przedstawiające stan wybranych generacji w postaci plików PBM,
`src` - katalog zawierający pliki z treścią kodu,
`testy` - katalog zawierający pliki z kodem poszczególnych testów,
`bin` - katalog zawierający plik wykonywalny,
`Makefile` - plik zawierający treść reguły programu `make`.

3.2 Przechowywanie danych

Na dysku: Pliki z danymi tekstowymi oraz obrazy przedstawiające stan wybranych generacji są zapisywane w katalogu `life/dane/wyniki`. Zostaną nazwane zgodnie z odpowiadającymi im numerami generacji.

W programie: Zostanie utworzona struktura przechowująca dane o wielkości płaszczyzny zadanej przez użytkownika oraz wektor przechowujący kolejne liczby (0 i 1), odpowiadające kolejnym martwym i żywym komórkom.

3.3 Dane wejściowe

Program będzie operował na pliku wejściowym o formacie TXT, gdzie dane przedstawione są w postaci:

```
x y
1 0 1 1 ...
0 0 0 1 ...
0 1 0 0 ...
... ... ...
```

gdzie:

- `x`, `y` to odpowiednio podane szerokość oraz długość dwuwymiarowej płaszczyzny na której odbędzie się symulacja „Gry w życie",
- `0` – odpowiada komórce martwej, `1` - żywa.

3.4 Dane wyjściowe

Program dzięki swojej pracy generuje :

1. Obrazy przedstawiających stan wybranych generacji w postaci plików PBM.
2. Bieżącą generację w pliku TXT.

Te pliki zostaną zapisane w katalogu: `life/dane/wyniki`.

4 Scenariusz działania programu

4.1 Scenariusz ogólny

Przebieg działania programu:

1. Kompilacja (o ile jest konieczna - zgonie z zasadami działania programu `make`) i uruchomienie.

2. Pobranie od użytkownika informacji o rodzaju danych początkowych (własnych albo z biblioteki), sąsiedztwie, którego zamierza użyć dla kolejnych generacji oraz o liczbie generacji, jaką chce przeprowadzić.

3. Przebieg generacji.

4. Pobranie od użytkownika informacji o zapisie bieżącej generacji w pliku TXT lub w formie graficznej (plik PBM) oraz jej ewentualny zapis.

5. Powtórzenie punktów 3. oraz 4. tyle razy, ile zadał użytkownik.

6. Zakończenie działania programu.

4.2 Scenariusz szczegółowy

Szczegółowy przebieg działania programu:

1. Uruchomienie programu przebiega poprzez wywołanie programu `make` - bez podawania argumentów przez użytkownika. W przypadku ich podania zostają one całkowicie zignorowane.

2. Program oferuje możliwość wyboru między sąsiedztwem Moore'a oraz von Neumanna poprzez wpisanie odpowiednio liczby `1` lub `2`. Następnie pyta o liczbę generacji do wykonania oraz o rodzaj danych wejściowych. W przypadku wybrania danych z biblioteki program wypisze na ekran pełną listę nazw dostępnych przykładowych danych, które znajdują się w katalogu `life/dane/biblioteka`, z których należy wybrać jeden wpisując odpowiednią liczbę.

3. Każdorazowo po przebiegu generacji i wyświatleniu jej na ekran program umożliwia użytkownikowi zapis bieżącej generacji do pliku tekstowego lub graficznego. Następuje to poprzez wpisaniu odpowiedniego znaku. Wpisanie `s` zapisuje plik PBM, `t` zapisuje plik TXT, `q` natychmiast kończy działanie programu, a znaki od `1` do `9` przechodzą o odpowiednią liczbę generacji.

4. Po wykonaniu odpowiedniej liczby generacji program samoczynnie kończy działanie.

5 Testowanie

5.1 Ogólny przebieg testowania

Do weryfikacji poprawności kodu zastosowane zostaną testy cząstkowe sprawdzające prawidłowość działania odpowiednich funkcji programu *Life*.

Pliki zawierające kod poszczególnych testów będą się znajdować w katalogu `life/src/testy`.