Języki i Metody Programowania 2 projekt I: Automat Komórkowy

Patryk Milewski
Paweł Zarczuk

6 marca 2017

1 Specyfikacja funkcjonalna

1.1 Przeznaczenie

Program przeprowadza symulację Gry w życie Johna Conwaya w siatce dwuwymiarowej wykorzystując dwa rodzaje sąsiedztwa(Moore oraz von Neumanna). Generuje obrazy poszczególnych generacji w postaci plików PNG oraz zapisuje ostatnią generację w postaci postaci pliku, który program może później wczytać.

1.2 Funkcjonalność

Działanie poszczególnych funkcjonalności uzależnione jest od parametrów, z którymi uruchamiany jest program oraz wartości ustawionych w pliku konfiguracyjnym.

Program oferuje dwie opcje wyboru początkowej generacji: wczytanie początkowej konfiguracji z pliku(z rozszerzeniem .life) lub wygenerowanie losowej konfiguracji o wymiarach ustawionych w pliku konfiguracyjnym). Aby wybrać pierwszą z tych opcji, należy przy uruchamianiu programu podać odpowiedni parametr.

Ponadto umożliwia generowanie ustalonej przez użytkownika liczby generacji N, które standardowo zdefiniowane są w pliku konfiguracyjnym, lecz mogą zostać zmienione poprzez podanie odpowiedniego parametru programu z liczbą generacji N.

Umożliwia ustawienie zachowania się programu na brzegach. Dostępne są następujące opcje: brzegi puste, brzegi pełne, brzegi naprzemienne.

Program generuje N+1 obrazów przedstawiających generację początkową i kolejne N generacji, a następnie zapisuje je w postaci pliku formacie PNG, o ustalonym przez użytkownika rozmiarze komórek(w pikselach) w folderze o nazwie podanej przy uruchamianiu programu.

Dla ostatniej generacji program generuje plik .life, który zawiera opis bieżącej generacji(który może zostać potem wczytany). Jest on również zapisywany w folderze o nazwie podanej przy uruchamianiu programu.

1.3 Wywołanie

Program wywołujemy pisząc:

```
Life.exe [opcja]...
```

gdzie dopuszczalne opcje to:

`-w` Nazwa katalogu do zapisu wyników

`[-l ]` (Opcjonalnie) nazwa pliku z siatką zerowej generacji,

`[-g ]` (Opcjonalnie) liczb generacji do wygenerowania (nie może być większa niż 999)

1.4 Dane wejściowe

`Plik tekstowy` z rozszerzeniem .life, o postaci

```
n
c11 c12 c13 c14 ...
c21 c22 c23 c24 ...
...
```

gdzie:

`n` to wymiary generowanej siatki n x n,

`c` to cyfry 1, 0 oznaczające martwe(0)i żywe(1) komórki.

Separatorem kolejnych komórek w linii jest spacja.

`Plik konfiguracyjny` znajdujący się w katalogu z programem: life.conf. Struktura pliku konfiguracyjnego:

[nazwaOpcji] = [wartość]

Opis poszczególnych parametrów:

`neighborhoodSettings`

Ustawienia sąsiedztwa: dla wartości 0 przyjmuje sąsiedztwo Von Neumann'a, a dla wartości 1 przyjmuje sąsiedztwo Moore'a

`defaultBoardSize`

Ustawienia wielkości planszy(uzywane tylko w przypadku losowego generowania komórek)

`edgeSettings`

Ustawienie brzegów planszy, przyjmuje wartość 0- dla pustych brzegów, 1- dla pełnych brzegów, 2- dla brzegów naprzemiennych.

`defaultGenerationCount`

Domyślna liczba generowanych generacji. Liczba generacji nie może być większa niż 999.

`cellSize`

Wielkość pojedynczej komórki na obrazie PNG generacji (w pikselach)

1.5 Format wyników

Plik o rozszerzeniu .life o takiej samej strukturze jak dane wejściowe oraz N+1 obrazów generacji w formacie png.

2 Architektura kodu

Program składa się z 5 modułów: main, gameOperator, boardGenerator, gameSettings, pngOperator.

W module main znajdują się głównie wywołania funkcji, w module gameOperator znajdują się najważniejsze funkcje programu, odpowiedzialne za symulację poszczególnych generacji. boardGenerator to moduł zawierający dwie funkcje służące do generowania planszy. GameSettings to moduł zawierający obsługę pliku konfiguracyjnego, zaś pngOperator służy do tworzenia plików PNG.

`main`

W module tym znajdują się wywołania poszczególnych funkcji, wczytywanie parametrów uruchomienia programu oraz obsługa wypisywania informacji o błędach.

`gameOperator`

Moduł składający się z funkcji createBoard - która w zależności od podanych parametrów decyduje czy generować losową planszę czy wczytać planszę z pliku; funkcji gameSimulation, która jest odpowiedzialna za podstawową funkcjonalność programu. Odpowiada ona za symulację kolejnych generacji, wykorzystując do tego funkcje checkAlive, checkCellMooreNeighborhood, checkCellVonNeumannNeighborhood, findFileName(również zawarte w tym module - są opisane poniżej). Najważniejszym elementem tej

funkcji są dwie pętle wykonywane dla każdej generacji oraz funkcje zapisującą dane w postaci obrazu PNG - printFile:

```
        for (int i = 1; i < boardSize - 1; i++)      // i=1 and, i <
            boardSize-1, cause cells are stored from index 1 to
            for (int j = 1; j < boardSize - 1; j++){ // index equal
                boardSize-1, other space is designed for border
                int aliveNeighbours;
                if (settings.neighborhoodSettings == 1)
                    aliveNeighbours = checkCellMooreNeighborhood(gameBoard
                        ->fields, i, j);
                else
                    aliveNeighbours = checkCellVonNeumannNeighborhood(
                        gameBoard->fields, i, j);
                newFields[i][j]=checkAlive(aliveNeighbours,gameBoard->
                    fields[i][j]);
            }
        freeFields(gameBoard); //Free old generation array
        gameBoard->fields=newFields; //puts new generation into struct
            gameBoard
        findFileName(saveName,1+1,fileName);
        errorCommand=printFile(fileName, gameBoard,settings.cellSize);
            generate visualization of generation in PNG format
```

Funkcja checkAlive sprawdza czy komórka powinna zmienić swój stan z żywej na martwą lub odwrotnie, w zależności od liczby komórek żywych. Funkcje checkCellMooreNeighborhood i checkCellVonNeumannNeighborhood służą do zliczania żywych komórek według zasad odpowiedniego sąsiedztwa. Funkcja findFileName znajduje nazwy dla kolejnych obrazów PNG. Do modułu tego należy także struktura gameBoard_t :

```
typedef struct gameBoard{
    int boardSize;
    short **fields;
}   gameBoard_t;
```

`boardGenerator`

Moduł składający się z funkcji boardGenerator - służącej do generowania losowej mapy, oraz z funkcji setBorders służącej do tworzenia brzegów według ustawień.

`gameSettings`

Moduł składający się z funkcji służącej do wczytywania zawartości pliku konfiguracyjnego do struktury settings typu gameSettings_t:

```
typedef struct gameSettings{
    int defaultBoardSize;
    int neighborhoodSettings;
    int edgeSettings;
    int isBoardLoaded;
    int cellSize;
    int defaultGenerationCount;
} gameSettings_t;
```

`pngOperator`

Moduł składający się z funkcji służącej do generowania z informacji o planszy zawartych w strukturze gameBoard_t obrazów w formacie PNG o wielkosci poszczególnych komórek pobranych ze struktury typu gameSettings_t.

3 Testy

Test I: Test

```
18                                                                  1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 2
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 4
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 6
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0                                 7
0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0                                 8
0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0                                 9
0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0                                 10
0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0                                 11
0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0                                 12
0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0                                 13
0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0                                 14
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0                                 15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 16
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 17
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 19
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 20
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                                 21
```