

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

25 kwietnia 2018

1 Opis ogólny

1.1 Nazwa programu

Nasz program będzie nazywał się **WireWorld**.

1.2 Poruszany problem

Zadaniem naszego programu będzie symulacja automatu komórkowego WireWorld wymyślonego przez Brian Silvermanna w 1987 roku. Kolejne symulacje polegają na zmianie stanów komórek ułożonych na prostokątnej planszy według określonych zasad. Każda komórka może znajdować się w jednym z czterech stanów: ogon elektronu (kolor żółty), głowa elektronu (kolor czerwony), przewodnik (kolor czarny), pusta komórka (kolor biały). Zasady symulacji:

- Komórka pozostaje Pusta, jeśli była Pusta.
- Komórka staje się Ogonem elektronu, jeśli była Głową elektronu.
- Komórka staje się Przewodnikiem, jeśli była Ogonem elektronu.
- Komórka staje się Głową elektronu tylko wtedy, gdy dokładnie 1 lub 2 sąsiadujące komórki są Głowami Elektronu.
- Komórka staje się Przewodnikiem w każdym innym wypadku.

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

1.3 Cel projektu

Celem tego projektu jest nauka i praktyka programowania w języku JAVA przy jednoczesnym zapoznaniu się z tematem automatu komórkowego WireWorld Briana Silvermanna.

2 Format danych i struktura plików

2.1 Pojęcia

Komórka – podstawowa jednostka automatu komórkowego.

Generacja – jeden cykl, w którym komórki zmieniają swój stan.

Stan komórki – komórka może znajdować się w czterech stanach: ogon elektronu, głowa elektronu, przewodnik lub pusta komórka.

Sąsiedzi komórki – komórki przylegające do niej bokami i rogami.

Bramka logiczna - element konstrukcyjny realizujący prostą funkcję logiczną.

Bramka logiczna NOR - bramka logiczna realizująca funkcję alternatywy wykluczającej.

Bramka logiczna AND - bramka logiczna realizująca funkcję koniunkcji.

Bramka logiczna OR - bramka logiczna realizująca funkcję alternatywy.

2.2 Struktura katalogów

Projekt będzie znajdował się w katalogu **WireWorldr**. Pliki z kodem źródłowym przechowywane będą w katalogach odpowiadającym ich zastosowaniu (np. pliki odpowiedzialne za graficzny interfejs użytkownika będą w katalogu **gui**). Katalogi te będą z kolei przechowywane w folderze **src**, w którym znajdzie się również katalog **data** zawierający przykładowe pliki wejściowe oraz katalog **test** zawierający pliki testowe dla poszczególnych modułów programu. Wyniki działania naszego programu (plik wyjściowy) oraz pliki wynikowe z rozszerzeniem **.class** umieścimy w katalogu **bin**. W katalogu **External Libraries** znajdują się biblioteki, z których korzysta nasz program. Graficzne przedstawienie struktury katalogów w naszym projekcie:

```
WireWorld
+---- src
|   +---- gui
|   +---- board
|   +---- inne pakiety z plikami źródłowymi
|   +---- test
|   +---- data
+---- bin
+---- External Libraries
```

2.3 Przechowywanie danych w programie

Plansza przeznaczona do przechowywania stanów wszystkich komórek będzie reprezentowana przez klasę zawierającą dwuwymiarową tablicę liczb całkowitych, zmienne zapamiętującą jej wymiary (wysokość i szerokość) oraz metody ją obsługujące.

2.4 Dane wejściowe

Plik wejściowy zawiera w każdym wierszu definicję obiektu lub stanu pojedynczej komórki. Definicja powinna zostać podana w postaci:

Nazwa_obiektu_lub_stanu_komórki: X,Y

Gdzie:

`nazwa_obiektu_lub_stanu_komorki` – nazwa tworzonego obiektu lub nazwa stanu tworzonej komórki,

`ile_generacji` – współrzędne oznaczające położenie początku obiektu lub położenie komórki.

Dostępne nazwy obiektów/stanów komórek:

- Diode - dioda,
- OR - bramka logiczna OR,
- NOR - bramka logiczna NOR,
- AND - bramka logiczna AND,
- Wire - drut
- ElectronHead - głowa elektronu,
- ElectronTail - ogon elektronu,
- Blank - pusta komórka,
- Conductor - przewodnik.

W przypadku definiowania drutu można (opcjonalnie) podać jeszcze dodatkowe dwa parametry:

`N` - długość drutu (ilość pól planszy, które będzie zajmował), domyślnie ustawiany na: 2,

`orientation` - orientacja drutu określająca czy kolejne pola przez niego zajmowane będą w linii poziomej (`horizontal`) czy pionowej (`vertical`). Domyślnie parametr ustawiony jest jako: `horizontal`.

Przykład definicji drutu:

`Wire: 0,3 3 vertical`

W przypadku kilkukrotnego wpisania obiektu/stanu komórki na te same współrzędne, program weźmie pod uwagę ostatnią definicję.

2.5 Dane wyjściowe

Plik wyjściowy programu (opisujący stan wybranej generacji przy zastopowanej symulacji) sporządzony zostanie według tej samej struktury co plik wejściowy, dzięki czemu będzie mógł być potem użyty jako plik wejściowy.

3 Testowanie

Działanie poszczególnych części programu przetestujemy przy użyciu biblioteki **AssertJ** oraz przykładowych plików wejściowych. Tylko część graficzną (GUI) przetestujemy ręcznie np. sprawdzając czy program wykona odpowiednią akcję po naciśnięciu przycisku.