

# Specyfikacja implementacyjna automatu komórkowego

GameOfLife: Gra w życie Johna  
Conwaya

Aleksandra Michalska, Natalia Olszewska

09.03.2021

## Spis treści

<b>1</b>	<b>Informacje ogólne</b>	<b>2</b>
1.1	Opis dokumentu . . . . .	2
1.2	Środowisko implementacyjne . . . . .	2
1.3	Informacje o wyświetlanych generacjach . . . . .	2
<b>2</b>	<b>Opis modułów</b>	<b>2</b>
2.1	Main . . . . .	2
2.1.1	Struktura generacji . . . . .	2
2.1.2	Obsługa parametrów wejściowych . . . . .	2
2.1.3	Odczyt i zapis plików . . . . .	3
2.1.4	Uruchomienie odpowiedniego trybu . . . . .	3
2.2	Modes . . . . .	3
2.2.1	SBS . . . . .	3
2.2.2	Fast . . . . .	3
2.3	SaveImage . . . . .	4
2.3.1	SaveGeneration . . . . .	4
2.4	CreateNew . . . . .	4
2.4.1	Rules . . . . .	4
2.4.2	New . . . . .	4
2.5	HelpCreate . . . . .	5
2.5.1	CreateNeighbourhood . . . . .	5
2.6	Neighbor . . . . .	5
2.6.1	MooreFlatWorld . . . . .	5
2.6.2	MooreSphereWorld . . . . .	5
2.6.3	NeumannFlatWorld . . . . .	6
2.6.4	NeumannSphereWorld . . . . .	6
<b>3</b>	<b>Testowanie</b>	<b>6</b>
<b>4</b>	<b>Diagram modułów</b>	<b>6</b>

# 1 Informacje ogólne

## 1.1 Opis dokumentu

Dany dokument koresponduje z poprzednią specyfikacją - Specyfikacją funkcjonalną

## 1.2 Środowisko implementacyjne

Docelowa gra została napisana w środowiskach Linuxowych, w języku C.

## 1.3 Informacje o wyświetlanych generacjach

Wyświetlana postać generacji została utworzona z dwukolorowych kwadratów. Stan żywy komórki został oznaczony białym kolorem, natomiast stan martwy kolorem czarnym. Kolorowe kwadraty zostały wyświetlone przy pomocy Unico-du, żywa - u2b1c, martwa - u2b1b.

# 2 Opis modułów

## 2.1 Main

W module głównym, dalej *Main*, została zaimplementowana podstawowa część programu. Można tam odczytać postać struktury pojedynczej *generacji*. *Main* odpowiada za obsługę parametrów, czytanie pliku wejściowego oraz zapis końcowej generacji, a także uruchomienie odpowiedniego trybu programu (SBS, FAST). Moduł główny nawiązuje do modułu *Modes*.

### 2.1.1 Struktura generacji

Struktura *generacji* zawiera:

- zmienną **gen** typu **int\*\*** - dwuwymiarową tablicę zawierającą liczby z zakresu 0-1, oznaczające stan danej komórki.
- zmienną **r** typu **int** - oznaczającą liczbę wierszy tablicy.
- zmienną **c** typu **int** - oznaczającą liczbę kolumn tablicy.
- zmienną **Nr** typu **int** - oznaczającą liczbę kolejnej generacji.

### 2.1.2 Obsługa parametrów wejściowych

W module *Main* funkcja *main* pobiera parametry z wywołania wsadowego. Parametry te następnie są porządkowane i sprawdzone dzięki funkcji *Switch*. Dzięki tej funkcji parametry wywołania zostają zapisane do zmiennych programu, które następnie przekazywane są do poszczególnych funkcji. W przypadku błędnego typu lub zapisu danych zwracane zostają kody błędów. Dla ułatwienia działania programu dla niektórych parametrów zostały utworzone własne typy (przy pomocy *enum*), np. dla parametru "-m" typ danych *modes* może zawierać wartości *sbs* lub *fast*.

### 2.1.3 Odczyt i zapis plików

Na podstawie pliku wejściowego utworzona zostaje początkowa generacja, która następnie zostaje przekazana pozostałym funkcją jako argument. Ostatnia generacja natomiast zostaje przekazana z powrotem do funkcji *main* przez, wybraną funkcję trybu jako wartość zwracana. Zostaje ona następnie zapisana do pliku tekstowego, którego nazwę podano przy uruchomieniu programu.

### 2.1.4 Uruchomienie odpowiedniego trybu

Zależnie od wybranego trybu (sbs, fast) funkcja *main* wywołuje odpowiednią funkcję *SBS* lub *Fast*.

## 2.2 Modes

W danym module zostały zaimplementowane funkcje obsługi poszczególnych trybów. Nawiązuje on do modułów *SaveImage* oraz *CreateNew*.

### 2.2.1 SBS

Funkcja *SBS* jako argumenty przyjmuje:

- zmienną **first** typu **generation** - strukturę zawierającą pierwszą generację.
- zmienną **count** typu **int** - proszoną liczbę generacji do wykonania.
- zmienną **how** typu **neighbour** - wybrany tryb sąsiedztwa (Mf, Ms, Nf, Ns).
- zmienną **toSave** typu **char** - określającą wybrany tryb zapisu.
- zmienną **howManyToSave** typu **int** - określającą ile lub którą generację zapisać zależnie od wybranego trybu zapisu.

Funkcja ta po pojedynczym wykonaniu oczekuje na wartość *c* podaną przez użytkownika. W przypadku, gdy wyniesie ona **e**, tryb *step by step* zostanie wyłączony, i zostanie uruchomiony tryb *fast*. W przeciwnym wypadku zostanie przedstawiona kolejna generacja. Wartość zwracana jest **ostatnią generacją** typu **generation**.

### 2.2.2 Fast

Funkcja *Fast* jako argumenty przyjmuje:

- zmienną **first** typu **generation** - strukturę zawierającą pierwszą generację.
- zmienną **count** typu **int** - proszoną liczbę generacji do wykonania.
- zmienną **how** typu **neighbour** - wybrany tryb sąsiedztwa (Mf, Ms, Nf, Ns).
- zmienną **toSave** typu **char** - określającą wybrany tryb zapisu.

- zmienną **howManyToSave** typu **int** - określającą ile lub którą generację zapisać zależnie od wybranego trybu zapisu.

Funkcja działa w trybie szybkim. Kolejne generacje są wyświetlane po sobie bez możliwości ingerencji użytkownika. Wartość zwracana jest **ostatnią generacją** typu **generation**.

## 2.3 SaveImage

W tej części programu zostały utworzone funkcje zapisu danej *generacji* do postaci obrazu. Dany moduł nie korzysta z żadnych innych modułów.

### 2.3.1 SaveGeneration

Funkcja *SaveGeneration* jako argumenty przyjmuje:

- zmienną **generationToSave** typu **generation** - strukturę zawierającą generację przeznaczoną do zapisu.
- zmienną **numberOfGeneration** typu **int** - liczbę odpowiadającą kolejnej generacji.

Funkcja ta tworzy zapis generacji w pliku o rozszerzeniu *bmp*, przy pomocy bitmapy. *SaveGeneration* jest typu *void*.

## 2.4 CreateNew

W module znajduje się funkcja tworząca nową *generację*, według zasad opisanych również w danym module. Nawiązuje do modułu *HelpCreate*.

### 2.4.1 Rules

Funkcja *Rules* opisuje zasady umierania i ożywiania komórek. Jako argumenty przyjmuje:

- zmienną **howManyNeighbours** typu **int** - liczba sąsiadów komórki.
- zmienną **isAlive** typu **int** - określającą stan komórki, gdzie 0 - martwa, 1 - żywa.

Funkcja zwraca zmienną typu **int** określającą stan komórki w nowej generacji.

### 2.4.2 New

Funkcja *New* przyjmuje argumenty:

- zmienną **oldGeneration** typu **generation** - struktura starszej generacji.
- zmienną **how** typu **neighbour** - określającą typ sąsiedztwa.

Dzięki użyciu funkcji z modułu *HelpCreate* funkcja *New* tworzy tablicę sąsiedztwa, z której korzysta wraz z funkcją *Rules* do utworzenia nowej generacji. Funkcja zwraca nową generację typu **generation**.

## 2.5 HelpCreate

W tym module zostały zaimplementowane funkcje pomocnicze przy tworzeniu nowej *generacji*. Nawiązuje do modułu *Neighbor*.

### 2.5.1 CreateNeighbourhood

Funkcja tworzy tablicę sąsiedztwa. Przyjmuje argumenty:

- zmienną **whichGeneration** typu **generation** - struktura generacji przeznaczona do utworzenia sąsiedztwa.
- zmienną **how** typu **neighbour** - określającą typ sąsiedztwa.

Funkcja powoduje się do modułu *Neighbor* w celu obliczenia liczby sąsiadów dla każdej komórki. Zwracana jest tablica typu **int\*\*** gdzie wartość w danej komórce odpowiada liczbie sąsiadów w wybranym sąsiedztwie.

## 2.6 Neighbor

W tej części napisane zostały funkcje potrzebne do sprawdzania liczby sąsiadów pojedynczej komórki. Ten moduł nie powołuje się na inne moduły.

### 2.6.1 MooreFlatWorld

Funkcja liczy liczbę sąsiadów danej komórki przy według zasad sąsiedztwa Moore'a oraz założenia płaskiego świata. Oznacza to, że żywa komórka która znajdzie się na brzegu planszy umiera.

Funkcja przyjmuje argumenty:

- zmienną **worldGeneration** typu **generation** - struktura generacji do policzenia sąsiedztwa.
- zmienną **row** typu **int** - indeks wiersza komórki, której chcemy policzyć sąsiedztwo.
- zmienną **column** typu **int** - indeks kolumny komórki, której chcemy policzyć sąsiedztwo.

Funkcja zwraca liczbę typu **int** sąsiadów komórki o podanych indeksach.

### 2.6.2 MooreSphereWorld

Funkcja liczy liczbę sąsiadów danej komórki przy według zasad sąsiedztwa Moore'a oraz założenia kulistego świata. Oznacza to, że dla komórki znajdującej się na brzegu tablicy, sąsiad, którego szukalibyśmy poza rozmiarem tablicy będzie pojawiał się po przeciwnej stronie tablicy. W ten sposób tablica jest zawijana.

Funkcja przyjmuje argumenty:

- zmienną **worldGeneration** typu **generation** - struktura generacji do policzenia sąsiedztwa.

- zmienną **row** typu **int** - indeks wiersza komórki, której chcemy policzyć sąsiedztwo.
- zmienną **column** typu **int** - indeks kolumny komórki, której chcemy policzyć sąsiedztwo.

Funkcja zwraca liczbę typu **int** sąsiadów komórki o podanych indeksach.

#### 2.6.3 NeumannFlatWorld

Funkcja liczy liczbę sąsiadów danej komórki przy według zasad sąsiedztwa von Neumanna'a oraz założenia płaskiego świata.

Funkcja przyjmuje argumenty:

- zmienną **worldGeneration** typu **generation** - struktura generacji do policzenia sąsiedztwa.
- zmienną **row** typu **int** - indeks wiersza komórki, której chcemy policzyć sąsiedztwo.
- zmienną **column** typu **int** - indeks kolumny komórki, której chcemy policzyć sąsiedztwo.

Funkcja zwraca liczbę typu **int** sąsiadów komórki o podanych indeksach.

#### 2.6.4 NeumannSphereWorld

Funkcja liczy liczbę sąsiadów danej komórki przy według zasad sąsiedztwa von Neumanna'a oraz założenia kulistego świata.

Funkcja przyjmuje argumenty:

- zmienną **worldGeneration** typu **generation** - struktura generacji do policzenia sąsiedztwa.
- zmienną **row** typu **int** - indeks wiersza komórki, której chcemy policzyć sąsiedztwo.
- zmienną **column** typu **int** - indeks kolumny komórki, której chcemy policzyć sąsiedztwo.

Funkcja zwraca liczbę typu **int** sąsiadów komórki o podanych indeksach.

## 3 Testowanie

## 4 Diagram modułów