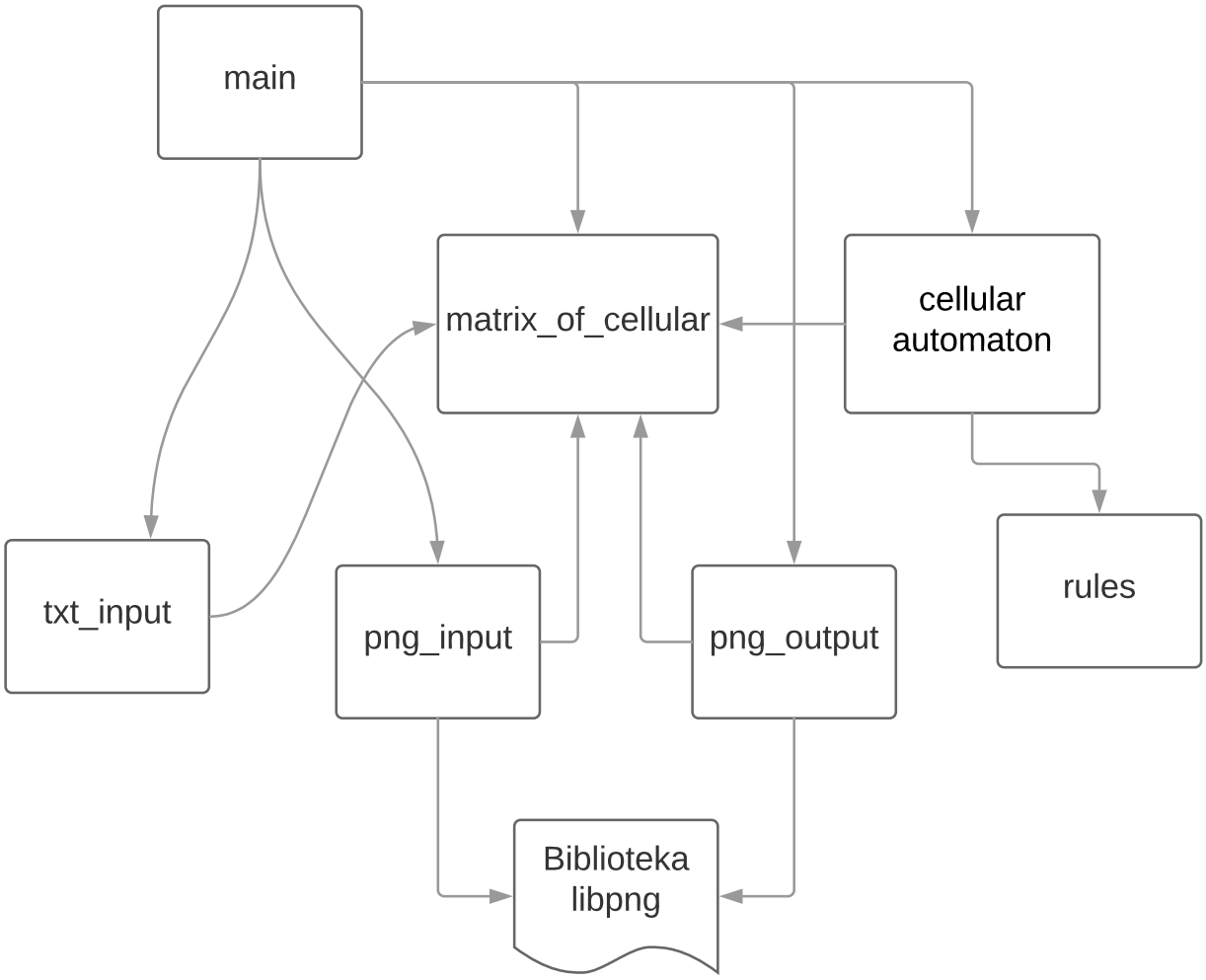
Politechnika Warszawska
Wydział Elektryczny
Kierunek Informatyka

Specyfikacja implementacyjna

Wykonali: Piotr Jeleniewicz, Aliaksandr Karolik
Warszawa, 19.03.2018

Spis treści

1 Diagram Modułów



RYS.01 DIAGRAM MODUŁÓW

2 Opis funkcji i typów danych

2.1 Moduł matrix_of_cellular.

Moduł ten będzie definiował i inicjalizował główny kontener danych w programie, służący do przechowywania informacji dotyczących wymiarów oraz samej macierzy przechowującej informacje o obecnej generacji. Strukrura ma postać:

```
typedef struct matrix
{
 int width;
 int height;
 int ** matrix;
} matrix_S;
```

Będzie też zawierać funkcję inicjalizującą tą strukturę danych oraz funkcję zwalniającą pamieć:

```
int init_T ( matrix_S * generation_T, int w, int h );

void free_T ( matrix_S * generation_T );
```

2.2 Moduł txt_input.

Będzie to moduł wczytujący pierwszą generację z pliku TXT. Będzię się składał z następujących funkcji:

```
int read_txt ( char * file_name, matrix_S * generation_T);
```

Funkcja ta będzie wczytywać z pliku TXT, którego nazwa przekazana jest w zmiennej file_name pierwszą generację. Następnie zostanie wywołana funkcja alokująca pamięć dla struktury zdefiniowanej w module matrix_of_cellular oraz zostaną wpisane do tej struktury dane reprezentujące wczytaną generację.

2.3 Moduł png_input.

Będzie to moduł wczytujący pierwszą generację z pliku PNG. Będzię się składał z następujących funkcji:

```
1) int  read_png ( char * file_name, matrix_S * generation_T );
```

Funkcja ta będzie wczytywać z pliku PNG, którego nazwa przekazana jest w zmiennej file_name pierwszą generację. Następnie zostanie wywołana funkcja alokująca pamięć dla struktury zdefiniowanej w module matrix_of_cellular oraz zostaną wpisane do tej struktury dane reprezentujące wczytaną generację.

```
2) void abort_(const char * s, ...);
```

Funkcja wymagana do obsługi plików png.

2.4 Moduł cellular_automaton.

Będzie to moduł tworzący następne generacje.

```
1) int neighbours( matrix_S * generation_T, int x, int y );
```

Funkcja ta będzie zliczać liczbę żywych komórek znajdujących się obok komórki o współrzędnych x oraz y.

```
2) void evolve( matrix_S * generation_T );
```

Funkcja generujące następnę pokolenie, na podstawie tablicy zdefiniowanej w strukturze pochodzącej z modułu matrix_of_cellular. Nowe pokolenie jest zapisywane w strukturze na miejscu poprzedniego.

```
3) int simulation( matrix_S * generation_T, char * name,
                   int generation_nr, int series_or_one,
                   int (*write_output)(  matrix_S * generation_T,
                   char* file_name, int number) );
```

Funkcja ta będzie sterować generowaniem kolejnych pokoleń oraz wywoływaniem funkcji generującej plik PNG z modułu png_output przedstawiający następne pokolenia. Argument generation_nr oznacza ilość żądanych generacji. Argument series_or_one będzie informował o tym czy wygenerować pliki png przedstawiające wszystkie generację czy jedynie konkretną żądaną.

2.5 Moduł png_output.

Będzie to moduł tworzący pliki PNG przedstawiające kolejne generacje.

```
void write_png( matrix_S * generation_T, char* file_name, int number);
```

Funkcja ta będzie generować pliki PNG o nazwach powstałych z połączenia zawartości zmiennej file_name oraz numeru generacji.

2.6 Moduł rules.

Będzie to moduł definiujący reguły automatu komórkowego.

```
int rules ( int cellular, int neighbours );
```

Funkcja ta będzie sprawdzać czy dla reguł w niej opisanych, sprawdzana komórka będzie w następnej generacji martwa lub żywa. Jeżeli komórka będzie żywa w następnej generacji, funkcja zwróci wartość 1 , w przeciwnym przypadku 0.

3 Przepływ sterowania.

Program zostanie wywołany z argumentami definującymi tryb pracy programu. W zależności od pliku źródłowego zostanie uruchomiona funkcja read_txt z modułu txt_output lub funkcja read_png z modułu png_output, która za pomocą funkcji z modułu matrix_of_cellular zainicujuje strukturę danych przechowująca początkową generację. Następnie struktura wraz z funkcją generującą pliki wyjściowe zostanie przekazana do funkcji simulation z modułu cellular_automaton, która przeprowadzi proces tworzenia nowych generacji a następnie w zależności od argumentu podanego przy uruchomienu wywoła funkcję przekazaną jako argument write_output, która będzie generować pliki wyjściowe dla wszystkich, bądź dla wybranej generacji.

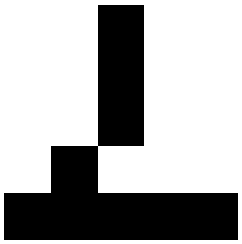
Funkcja simulation będzie składać się z pętli, w której będzie wywoływana funkcja evolve odpowiedzialna za generowanie następnej generacji. W funkcji evolve, będzie wywoływana dla każdej komórki funkcja neighbours, która wyznaczy ilość żywych sąsiadów. Następnie ilość ta wraz ze stanem badanej obecnie komórki zostanie przekazana do funkcji rules, która w zależności od zdefiniowanych w niej reguł automatu komórkowego wyznaczy stan tej komórki dla następnego pokolenia. Następnie wartość ta zostanie zapisana do tablicy tymczasowej utworzonej w funkcji evolve. Kiedy algorytm wygeneruje całą nową generację zostanie ona przepisana do stuktury danych całego programu. Po wygenrowaniu nowego pokolenia zostanie uruchomiona funkcja generująca dane wyjściowe, podana jako argument write_output funkcji simulation. Pętla funkcji simulation wykona się tyle razy ile generacji zażąda użytkownik.

4 Testy jednostkowe.

4.1 Test modułu png_input

Dla modułu png_input zostaną przeprowadzone dwa testy. Będa one polegały na wczytaniu pliku PNG, a następnie przetowrzeniu go na macierz składającą się z 0 i 1 oraz na wyznaczeniu wymiarów tej macierzy. Test zostanie zaliczony jeśli jedynki macierzy będą odwzorowaniem czarnych pikseli pliku PNG a zera odwzorowaniem pikseli białych Testy mają postać:

1) Wczytywany plik PNG:



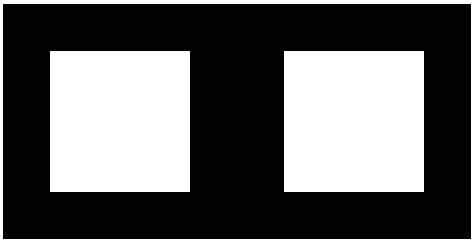
RYS.02 Plik wej. do 1 testu png_input

Oczekiwany wynik:

```
height = 5
width = 5
```

```
0 0 1 0 0
0 0 1 0 0
0 0 1 0 0
0 1 0 0 0
1 1 1 1 1
```

2) Wczytywany plik PNG:



RYS.03 Plik wej. do 2 testu png_input

Oczekiwany wynik:

```
height = 5
width = 10
                    1 1 1 1 1 1 1 1 1 1
                    1 0 0 0 1 1 0 0 0 1
                    1 0 0 0 1 1 0 0 0 1
                    1 0 0 0 1 1 0 0 0 1
                    1 1 1 1 1 1 1 1 1 1
```

4.2 Test modułu png_output

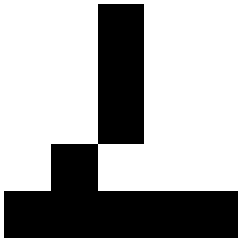
Dla modułu png_output zostaną przeprowadzone dwa testy. Będa one polegały na generowaniu pliku na podstawie wczytanej struktury danych, a następnie na podstawie tej struktury będą wygenerowane pliki PNG. Test zostanie zaliczony, gdy plik PNG będzie odwzorowaniem danych zdefiniowanych w strukturze.

1) Zawartość struktury będzie odpowiadała danym przedstawionym poniżej:

```
height = 5
width = 5
```

```
0 0 1 0 0
0 0 1 0 0
0 0 1 0 0
0 1 0 0 0
1 1 1 1 1
```

Oczekiwany wynik:



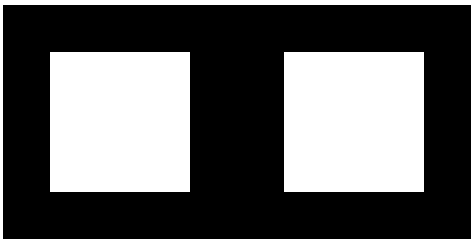
RYS.03 Plik wyj. do 1 testu png_output

2) Dane dla testu drugiego:

```
height = 5
width = 10
```

```
1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 1 0 0 0 1
1 0 0 0 1 1 0 0 0 1
1 0 0 0 1 1 0 0 0 1
1 1 1 1 1 1 1 1 1 1
```

Oczekiwany wynik:



RYS.4 Plik wyj. do 2 testu png_input

4.3 Test modułu txt_input

Dla modułu txt_input zostanie przeprowadzy jeden test. Będzie on polegał na wczytaniu pliku TXT, a następnie przetworzeniu go na macierz składającą się z zer i jedynek oraz na wyznaczeniu wymiarów tej macierzy. Test zostanie zaliczony jeśli macierz oraz jej wymiary w strukturze będą odpowiadały macierzy z pliku TXT.

Zawartość pliku TXT jest następująca:

```
1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 1 0 0 0 1
1 0 0 0 1 1 0 0 0 1
1 0 0 0 1 1 0 0 0 1
1 1 1 1 1 1 1 1 1 1
```

Oczekiwany wynik:

```
height = 5
width = 10
                    1 1 1 1 1 1 1 1 1 1
                    1 0 0 0 1 1 0 0 0 1
                    1 0 0 0 1 1 0 0 0 1
                    1 0 0 0 1 1 0 0 0 1
                    1 1 1 1 1 1 1 1 1 1
```

4.4 Test modułu rules

Test modułu będzie polegał na sprawdzeniu wszystkich kombinacji liczby sąsiadów oraz stanów komórek dla zasad gry w życie Johna Conway'a (John Conway's Game of life). W przypadku naszego automatu komórkowego zachodzą następujące reguły:

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

Prototyp funkcji ma postać:

```
int rules ( int cellular, int neighbours );
```

gdzie,

```
cellular - oznacza stan komórki.
Jeżeli jest żywa ma wartość 1, a jeżeli martwa 0.

neighbours - liczba żywych sąsiadów badanej komórki.
```

Testy będą polegały na wywoływaniu funkcji dla wybranych argumentów. Testowe argumenty wyglądają następująco:

1. rules (0, 1) - oczekiwana zwracana wartość 0
2. rules (0, 3) - oczekiwana zwracana wartość 1
3. rules (1, 8) - oczekiwana zwracana wartość 0
4. rules (1, 2) - oczekiwana zwracana wartość 1
5. rules (1, 3) - oczekiwana zwracana wartość 1

4.5 Test modułu cellular_automaton

Dla modułu cellular_automaton zostanie przeprowadzony jeden test. Będzie on polegał na wyznaczeniu następnej generacji

1) Zawartość struktury będzie odpowiadała danym przedstawionym poniżej:

```
height = 5
width = 5
```

```
0 0 1 0 0
0 0 0 1 0
0 1 1 1 0
0 0 0 0 0
0 0 0 0 0
```

Oczekiwany wynik następnej generacji:

```
0 0 0 0 0
0 1 0 1 0
0 0 1 1 0
0 0 1 0 0
0 0 0 0 0
```