Dokumentacja projektu 3.4

**Treść projektu**

Napisz program realizujący prosty automat komórkowy - symulator życia „Life” na powierzchni zamkniętej typu torus. Program powinien posiadać wygodny interfejs użytkownika, dzięki któremu będzie możliwe łatwe wprowadzanie i modyfikacja położenia punktów początkowych (mile widziane użycie kursorów do przemieszczania się po planszy).

**Game of life**

Gra toczy się na planszy (płaszczyźnie) podzielonej na kwadratowe komórki. Każda komórka ma ośmiu „sąsiadów” (tzw. 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 w pewnych jednostkach czasu. Stan wszystkich komórek w pewnej jednostce czasu jest używany do obliczenia stanu wszystkich komórek w następnej jednostce. Po obliczeniu wszystkie komórki zmieniają swój stan dokładnie w tym samym momencie. Stan komórki zależy tylko od liczby jej żywych sąsiadów. W grze w życie nie ma graczy w dosłownym tego słowa znaczeniu. Udział człowieka sprowadza się jedynie do ustalenia stanu początkowego komórek.

**Reguły gry według Conwaya**

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

https://pl.wikipedia.org/wiki/Gra\_w\_%C5%BCycie

**Zmienne globalne**

* *x* – ilość kolumn planszy,
* *y* – ilość rzędów planszy,
* *aktualna* – tablica przechowująca aktualny stan punktów (1 – żyje, 0 – nie żyje),
* *pokolenia* – ilość iteracji symulacji,
* *alive* – zmienna przechowująca aktualną liczbę żywych punktów,
* *czas* – zmienna przechowująca długość opóźnienia pomiędzy iteracjami,
* *zapis* - zmienna przechowująca co ile pokoleń stan tablicy będzie przekazany do pliku wyjściowego
* *i,j* – zmienne pomocnicze do liczenia obrotów pętli.

**Funkcje**

* *start()* – funkcja typu void, inicjuje struktury biblioteki *ncurses*, tworzy menu wyboru pomiędzy losowym układem startowym a własnym wyborem punktów startowych,
* *ustawienia()* – funkcja typu void, wywoływana z argumentem stan (może przyjmować wartość 1 lub 2), tworzy menu, w którym użytkownik ustawia parametry symulacji takie jak:
  + ilość kolumn,
  + ilość rzędów,
  + ilość iteracji symulacji,
  + długość opóźnienia pomiędzy kolejnymi iteracjami,
* *gra()* – funkcja typu void, wywoływana z argumentem *poko* (przyjmuje wartość taką samą jak *pokolenia*),na początku użytkownik podaje nazwę pliku w którym będzie zapisany wynik symulacji, następnie funkcja realizuje algorytm gry w życie, przy czym funkcję można przerwać klikając spację, uruchamia to funkcję *menu\_stop()*,
* *menu\_stop()* – funkcja typu int, wywoływana z argumentem poko, tworzy menu podczas pauzy w funkcji *gra()*, użytkownik może wybrać pomiędzy zmianą aktualnych stanów symulacji, wyjściem z symulacji lub jej kontynuacją, funkcja uruchamia funkcję *wlasne()* przy wyborze zmiany punktów, zwraca 1 jeśli użytkownik wybrał kontynuację symulacji lub 2 jeśli wybrał jej zakończenie,

Funkcje c.d.

* *losowe()* – funkcja typu void, tworzy losowy układ tablicy aktualna,
* *wlasne()* – funkcja typu void, przy użyciu kursorów użytkownik wybiera punkty w tablicy aktualna i przy pomocy spacji zmienia ich stan na przeciwny,
* *wypisz()* – funkcja typu void, wyświetla na ekranie aktualny stan tablicy aktualna,
* *kontynuacja()* – funkcja typu void, tworzy menu wyboru pomiędzy rozpoczęciem nowej symulacji a wczytaniem stanu ostatniej, funkcja ta jest uruchamiana tylko wtedy jeśli po zakończeniu poprzedniej symulacji w tablicy aktualna znajdują się jakiekolwiek żyjące komórki,
* *czyszczenie()* – funkcja typu void, wykorzystywana do zerowania tablicy aktualna i ustawiania *alive* na 0,
* dane\_plik() – funkcja typu int, pobiera liczbę z pliku wejściowego i ją zwraca,
* dane() – funkcja typu int, przyjmuje liczbę wpisaną przez użytkownika i ją zwraca,
* zapisywanie() – funkcja typu void, wywoływana z argumentem poko, zapisuje w pliku wynikowym, który jest zgodny ze standardem HTML, aktualny stan symulacji (tablicy aktualna),
* wczytaj() – funkcja typu void, wczytuje dane z podanego przez użytkownika pliku wejściowego, obsługuje błędy napotkane w pliku wejściowym, jeżeli wszystkie dane są poprawne uruchamia funkcję gra().

Plik wejściowy

Aby opcja wczytania z pliku wykonała się poprawnie należy odpowiednio przygotować plik z danymi, musi być to plik tekstowy, który zawiera wszystkie ustawienia i układ początkowy symulacji. Musi mieć on taką strukturę:

s (liczba oznaczająca szerokość planszy)

w (liczba oznaczająca wysokość planszy)

o (liczba będąca długością opóźnienia pomiędzy iteracjami, podana w mikrosekundach)

i (liczba oznaczająca ilość pokoleń)

n (liczba oznaczająca co ile pokoleń ma być zapisywany stan symulacji)

(tablica snaw zawierająca ciąg 0 lub 1 odpowiadających stanom poszczególnych pól)

Przykład:

s 10

w 5

o 1000000

i 10

n 1

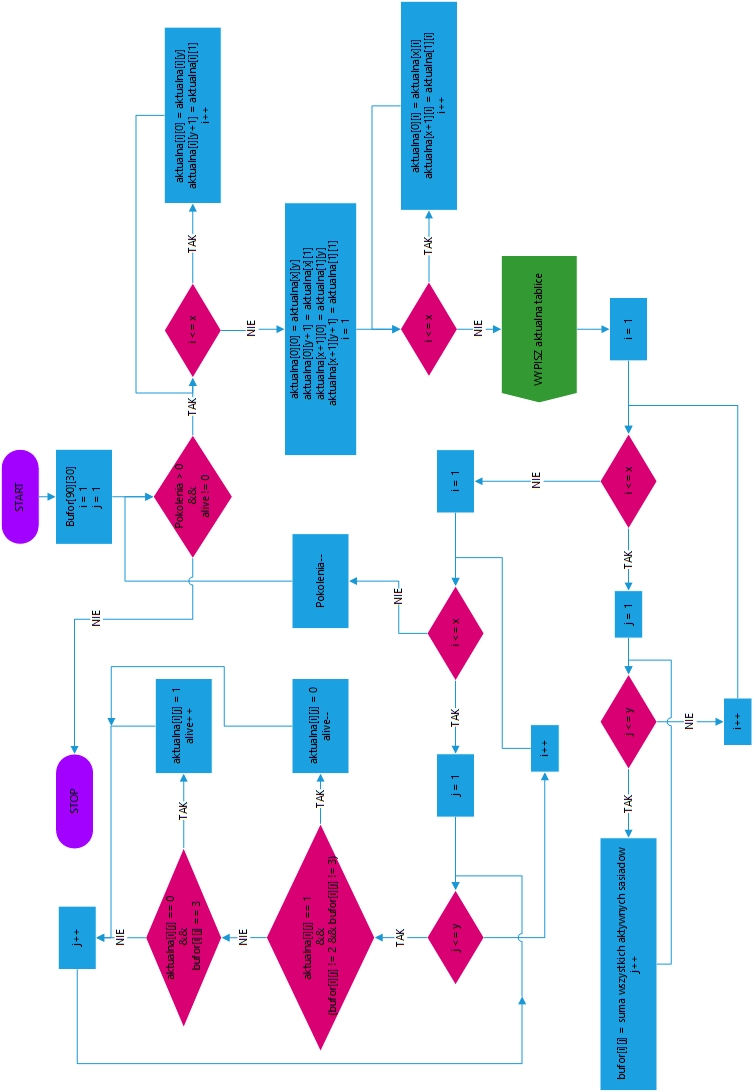
0000000000

0000111000

0000100000

0000010000

0000000000



Schemat blokowy Algorytmu gry w życie