Specyfikacja Funkcjonalna

Rozdział 1: Opis Ogólny

Podrozdział 1.1: Nazwa programu.

Nazwa programu : „Conway's Game Of Life"

Podrozdział 1.2: Poruszany problem.

Zadaniem programu będzie symulacja automatu komórkowego stworzonego przez brytyjskiego matematyka Johna Conway'a. Program w trakcie swojej pracy będzie drukował podane na wejściu iteracje gry do pliku *.png.

Podrozdział 1.3: Użytkownik docelowy.

Program jest tworzony z myślą o osobach pasjonujących się zagadkami matematycznymi. Symulacja umożliwi im dokładną analizę działania automatu zaproponowanego przez Conway'a.

Rozdział 2: Opis Funkcjonalności

Podrozdział 2.1: Jak korzystać z programu?

Do poprawnego funkcjonowania programu wymagane są 4 dane wejściowe typów odpowiednio Integer, Integer, FILE, Integer podane jako argumenty wywołania.

Podrozdział 2.2: Uruchomienie programu.

W celu uruchomienia programu należy wpisać w konsoli unixowej polecenie :

./game 250 350 points 25

Objaśnienia :
250 – wymiar x-owy planszy. Maksymalny akceptowalny rozmiar to 2000.
350 – wymiar y-owy planszy. Maksymalny akceptowalny rozmiar to 2000.
points – plik zawierający współrzędne 'żywych' komórek.
25 – liczba początkowych iteracji to wydrukowania

Podrozdział 2.3: Możliwości programu.

a) Program będzie w stanie stwierdzić poprawność danych wejściowych, w przypadku ich błędu zostanie zwrócony odpowiedni komunikat.
b) Wczyta z pliku zawierającego współrzędne 'żywych' komórek potrzebne naniesienia na planszę.
c) Korzystając z odpowiednich bibliotek program utworzy pliki *.png obrazujące n początkowych iteracji, gdzie n jest 4 argumentem podanym przy wywołaniu.

Rozdział 3: Format danych i struktura plików

Podrozdział 3.1: Pojęcia i pola formularza

grid – plansza na której umieszczane są ‘żywe’ komórki podane w pliku przy wywołaniu.

sim – plansza pomocnicza przetrzymująca aktualny stan planszy. Na jej bazie tworzony jest następny stan planszy.

Podrozdział 3.2: Struktura katalogów.

/src – katalog z plikami źródłowymi.

/bin – katalog z plikiem wykonywalnym.

/tests – katalog z testami dla programu

Makefile – plik makefile znajdujący się w folderze głównym programu, zawierający instrukcje wykonania kompilacji całego programu oraz przykładowe i testowe polecenia wykonawcze

Podrozdział 3.3: Przechowywanie danych w programie.

W programie znajdować się będą struktury odpowiednio do:

- przetrzymywania planszy
- przetrzymywania tymczasowego stanu planszy
- przechowywania ‘żywych’ komórek znajdujących się na planszy

```
typedef struct Grid
{
  char** v;
  unsigned int x, y;
}*Grid_t;
```

W strukturze grid przechowywane będą :

- wektor ‘v’ elementów typu char*
- zmienna ‘x’ typu unsigned integer odpowiadająca za szerokość trzymanej planszy
- zmienna ‘y’ typu unsigned integer odpowiadająca za wysokość trzymanej planszy

```
typedef struct PointList
{
  int x, y;
  struct PointList * next;
}*Plist_l;
```

W strukturze PointList przechowywane będą :

- zmienna ‘x’ typu integer trzymająca pierwszą współrzędną ‘żywej’ komórki
- zmienna ‘y’ typu integer trzymająca drugą współrzędną ‘żywej’ komórki
- wskaźnik na następny element listy ‘żywych’ komórek na planszy

```
typedef struct Simulator
```

```
{
  Plist_l pointlist;
  Grid_t grid;
  Grid_t nextgrid;
}*Simulator_t;
```

W strukturze Simulator przechowywane będą :
- zmienna ‘grid’ typu struct Grid* trzymająca aktualny stan planszy
- zmienna ‘nextgrid’ typu struct Grid* trzymająca obraz nowej planszy, na niej wykonywane będą operacje w celu uzyskania kolejnego stanu planszy
- zmienna pointlist typu struct PointList* odpowiadająca za ‘żywe’ komórki znajdujące się na planszy

Podrozdział 3.4: Dane wejściowe

Program jako dane wejściowe będzie dostawał parę liczb x,y odpowiadające za rozmiar planszy. Dodatkowo wczytywane będą pary liczb znajdujące się w pojedynczym pliku – będą to współrzędne ‘żywych’ komórek. Ostatnim parametrem wymaganym do poprawnego działania programu jest liczba it odpowiadająca za początkową ilość wyrysowanych iteracji.

Przykład do danych wejściowych dla pliku z współrzędnymi:

```
9 7
9 8
9 9
8 9
7 8
100 5
18 10
1 -1
2 5
8 16
```

Uwaga:
W pierwszej kolumnie znajdują się współrzędne x-owe,
w drugiej zaś współrzędne y-owe.

Podrozdział 3.5: Funkcje wewnętrzne

Grid_t initGrid(int x, int y);

Funkcja inicjuje planszę o zadanych rozmiarach po czym zwraca wskaźnik na utworzoną strukturę.

void destroyGrid(Grid_t g);

Funkcja zwalnia pamięć wcześniej zaalokowaną na potrzeby utworzenia planszy w programie.

void clearGrid(Grid_t g);

Funkcja wypełnia zadaną planszę zerami - 'uśmierca' wszystkie komórki, przygotowując planszę do zaktualizowania jej stanu.

void printGrid(Grid_t g);

Funkcja wypisuje stan planszy na ekran konsoli.

void readFile(FILE * in, Simulator_t sim);

Funkcja wczytuje dane z plików do struktur wewnętrznych.

void createPng(char* filename, Grid_t grid);

Funkcja tworzy plik *.png obrazujący stan planszy w danej iteracji.

int countNeighbours(int x, int y, Grid_t grid);

Funkcja liczy ilość 'żywych' komórek sąsiadujących z komórką o współrzędnych x,y otrzymanych jako parametry.

Plist_l initPlist(int x, int y);

Funkcja inicjuje listę 'żywych' komórek potrzebną do aktualizacji stanu planszy.

Plist_l addToPlist(Plist_l list, int x, int y);

Funkcja dodaje komórkę o współrzędnych x,y do list 'żywych' komórek.

void destroyPlist(Plist_l list);

Funkcja czyści listę 'żywych' komórek, po czym dealokuje używaną pamięć.

Simulator_t initSimulator(int x, int y);

Funkcja inicjuje simulator wraz z inicjacją jego pól
– dwóch plansz oraz listy komórek.

Simulator_t prepareSimulation(FILE* file);

Funkcja przygotowuje simulator do przeprowadzenia symulacji gry.

int simulate(FILE * file, int x, int y, int iterations);

Funkcja otrzymuje plik z współrzędnymi komórek, wymiary planszy oraz liczbę iteracji do wykonania. Wewnątrz wywoływane są funkcje tworzące simulator, na którego bazie przeprowadza się właściwą symulację.
W jej trakcie tworzy się pliki *.png odpowiednich iteracji gry, wywołując funkcję createPng().

Simulator_t calculateFrame(Simulator_t sim);

Funkcja mając aktualny stan planszy, listę 'żywych' komórek tworzy następny stan planszy.

void destroySimulator(Simulator_t simulator);

Funkcja niszczy simulator przez dealokację pamięci.

Podrozdział 3.6: Dane wyjściowe

Generowanymi przez program danymi wyjściowymi będą pliki *.png, obrazujące kolejne iteracje gry.

Rozdział 4: Scenariusz działania programu

Podrozdział 4.1: Scenariusz ogólny

Na wstępie program wczyta dane z pliku wejściowego opisującego współrzędne 'żywych' komórek podanego jako argument wywołania. Następnie przeprowadzona zostanie symulacja zachowania się każdej komórki w zależności od jej sąsiadów.
W trakcie trwania symulacji tworzona będzie graficzna reprezentacja każdej z n początkowych iteracji.

Podrozdział 4.2: Scenariusz szczegółowy

1) Wczytywanie danych do programu
- Zainicjowane zostaną na początku struktury potrzebne do przetrzymywania danych. Następnie zostaną one pobrane z plików wejściowych i umieszczone w przygotowanych wcześniej strukturach. Nastąpi również weryfikacja danych w poszczególnych przypadkach:

- **nie podania argumentów wywołania zwrócony zostanie komunikat z prośbą o ich wpisanie**
- **podania niewłaściwych danych wyświetlony zostanie komunikat informujący o błędnych danych i zwrócony będzie kod błędu.**

2) Przeprowadzenie symulacji gry

- Korzystając z Conway'owskich zasad gry w życie przeprowadzane będą prognozy stanów komórek. Wykorzystana do tego zostanie struktura przetrzymująca aktualny stan planszy, listę 'żywych' komórek oraz planszę pomocniczą na której wyznaczony będzie następny stan gry.

3) Generowanie obrazów iteracji

- W trakcie trwania symulacji gry generowane będą pliki *.png reprezentujące dane iteracje gry. Zapisywane zostaną one do katalogu ./ obok pliku Makefile.

Rozdział 5: Testowanie

Podrozdział 5.1: Ogólny przebieg testowania

W celu testowania programu użyte zostaną wcześniej przygotowane paczki testów. Testowanie polegać będzie na wykorzystaniu skryptu porównującego dwa pliki o rozszerzeniach png. Zalecany środowiskiem testowania jest system Linux.