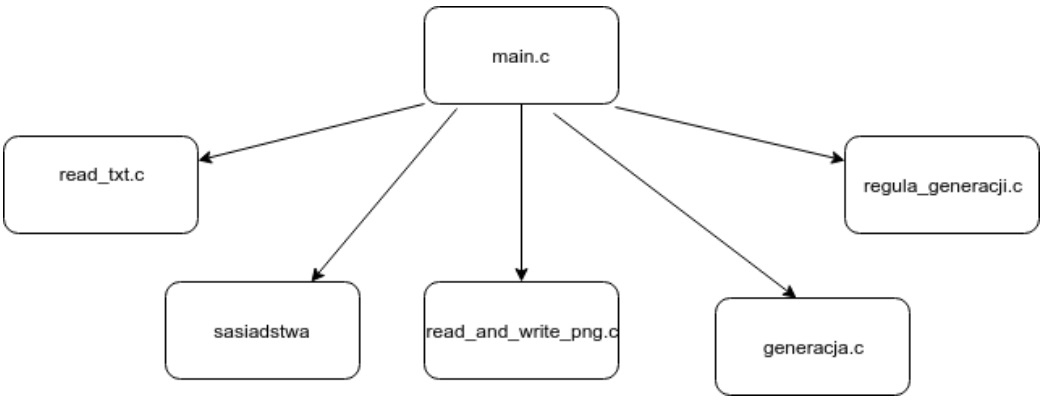
SPECYFIKACJA IMPLEMENTACYJNA "Gra w życie" DLA JĘZYKA PROGROMOWANIA C

Wykonali:
1. Ivan Prakapets

Spis treści

1 Diagram modułów



Rysunek 1: Diagram modułów

1.1 Opis diagrama

Poniżej jest przedstawiony opis diagrama:

`main` jest głównym plikiem z rozszerzeniem `.c`. Plik `main.c` wiąże między sobą wszystkie poszeczególne moduły w programie ”Gram w życie”.

2 Opis i spis funkcji

2.1 Moduł read_and_write_png

Moduł `read_and_write_png` składa się z trzech funkcji:

* `void read_png_file (char * )`
* `void write_png_file (char *, int )`
* `void process_file ( int ** )`

Można powiedzieć, że ten moduł jest najgłówniejszym, dlatego że, korzystając z poniszczysz funkcji, będziemy obrabiać zdjęcia.

1. Funkcja `read_png_file`. Celem tej funkcji jest sczytywanie wszystkich danych z plika `.png`. Dla uruchomienia tej funkcji trzeba podać nazwę plika oraz rozszerzenie png. Dla nas główniejsze, że po uruchomieniu będziemy mieli takie wartości, jak `width, height`, `color_type`, które będą używane następnie dla naszej gry. Funkcja `read_png_file.` zawiera obsługę błędów.

2. Funkcja `write_png_file`. Celem tej funkcji jest zapisywanie wszystkich naszych nowych danych z gry do plików `.png` z których będzie stworzony `plik.gif`. Dla jej uruchomienia trzeba podać nazwę plika z rozszerzeniem `png` oraz number. Funkcja `write_png_file` zawiera obsługę błędów.

3. Funkcja `process_file`. Celem tej funkcji jest sprawdzanie pliku `png` (musi być stworzony w GIMPie) oraz wdrożenie funkcji `read_png_file` do naszej gry. Dla uruchomienia trzeba podać naszą tablicę.

2.2 Moduł read_txt

Moduł `read_txt` składa się z jednej funkcji. Celem tej funkcji jest sczytywanie wszystkich danych z plika `.txt`. Dla uruchomienia tej funkcji trzeba podać nazwę plika oraz rozszerzenie txt. Dla nas główniejsze, że po uruchomieniu będziemy mieli takie wartości, jak `width, height`, które będą używane następnie dla naszej gry. Funkcja `read_txt.` zawiera obsługę błędów.

2.3 Moduł generacja

Moduł generacja składa się z dwóch funkcji:

* `void print_gen_to_file (int ** )`
* `void write_png_file (char *, int )`
* `void print_gen ( int ** )`

1. Funkcja `print_gen_to_file`. Celem tej funkcji jest podsumowanie naszych danych i oddawania ich do funkcji `write_png_file`.
2. Funkcja `print_gen`. Celem tej funkcji jest podsumowanie naszych danych i wypisanie ich w terminali.

2.4 Moduł sasiadstwa

Moduł sasiadstwa składa się z dwóch funkcji z jednym imieniem

* `int count_neighbours ( int , int , int ** )`

To jest zrobiono dla ułatwienia programu, dlatego że musimy zrobić taki samy program w dwa różnych sposoby. Funkcja `count_neighbours` liczy sumę sąsiedzi dla każdej komórki w automacie. W zależności od sposobu gry(Murra czy Neumanna) liczy się różna suma. Dla jej uruchomienia trzeba podać x, y oraz naszą tablicę.

3 Przechowywanie danych

Wszystkie dane (program w języku C, zdjęcie początkowe, zdjęcia wygenerowane, plik `gif`) przechowywane w folderu `projekt_c`.

Dane będa przechowywanie w tablice dwuwymiarowej `matrix`.

4 Wymagania systemowe

Projekt „Gra w życie" jest stworzony na systemie operacyjnym Linux (Ubuntu 16.04.4 LTS). Testowania są przeprowadzone też w systemie operacyjnym Linux za pomocą zewnętrznej programy „Valgrind". Będziemy korzystać z kompilatora C99 używając zewnętrznej biblioteki C „libpng". Źródło: http://www.libpng.org/pub/png/libpng.html

5 Wersjonowanie

W projekcje „Gra w życie" jest używany system kontroli wersji (oprogramowanie służące do śledzenia zmian głównie w kodzie źródłowym oraz pomocy programistom w łączeniu zmian dokonanych w plikach przez wiele osób w różnym czasie). Dla kontroli wersji jest używany „Git". Za pomocą repozytoria Repozytorium_2018L_JIMP2_P_Zawadzki" który został utworzony 2018-02-22 przez prowadzącego: mgr inż. Paweł Zawadzki będziemy wykorzystać branch.

Dla stworzenia branch potrzebna jest komenda git branch ivan. Zostanie utworzony branch ivan, żeby przyjść do niego potrzebna jest komenda git checkout ivan. Zostaniesz przekierowany do branch ivan. W dowolnej chwili

możemy sprawdzić w jakim branchu aktualnie się znajdujemy po napisaniu git branch gwiazdka pokaże gdzie się znajdujesz. Żeby dodać coś do branch potrzebna jest komenda git add. Potem jest potrzebna komenda git commit -m 'Komentarz'. Na końcu dodajemy komendę dla wysyłania wszystkich zrobionych zmian git push origin ivan. Potem trzeba przyjść do mastera za pomocą komendy git checkout maste, teraz możemy dociągnąć zmiany za pomocą polecenia git merge ivan.

6 Testy

Program jest w stanie developerskim, dlatego testy są tylko na sprawdzanie działania programu lub niektórych funkcji takich jak: wczytywanie plika `png`, wczytywanie plika `txt`, wdrożenie danych do plika `png`, tworzenie plika gif z plików `png`, wybór metody(Murra lub Neumanna), wybór wczytywania plika `txt` lub `png`.

7 Algorytm

Algorytm jest opisany w specyfikacji funkcjonalnej.