Systemy wbudowane i czasu rzeczywistego
Gra w życie

Andrzej Kwasiborski nr indeksu 271069

28 marca 2017

Spis treści

1 Funkcjonalność programu **1**

2 Opis funkcji i algorytmów **1**

3 Wnioski **2**

1 Funkcjonalność programu

Celem projektu było napisanie programu w języku C, który zasymuluje *Grę w życie*, automat komórkowy wymyślony przez Johna Conway'a. Program miał się uruchamiać w środowisku QNX.
Program jest uruchamiany z poziomu konsoli. Następuje to po wpisaniu polecenia *./life -s size -n number_of_iterations -f*, gdzie *s* to rozmiar macierzy jaka ma zostać wygenerowana, *n* to liczba generacji jaka ma być przeprowadzona, zaś *f* określa czy program ma wyświetlać generacje jedna po drugiej z odstępami czasowymi czy wyświetlić od razu wszystko na raz. W przypadku brak flag program uruchomi się z wartościami domyślnymi. Gdy zaś zostanie uruchomiony z błędnie podanymi flagami, wyświetli komunikat informujący o tym jak należy prawidłowo używać programu.

2 Opis funkcji i algorytmów

Program zawiera w sobie 4 ważne funkcje i jedna poboczną. Są to *main*, *neighbours*, *generate*, *print_board* oraz jako ostatnia *get_terminal_size*. Funkcja *main* odpowiada za sterowanie programem, wczytywanie danych, uruchamianie innych funkcji i kontakt z użytkownikiem. *Neighbours* zgodnie z sąsiedztwem Moore'a zlicza ile sąsiadów ma konkretna komórka. *Generate* zarządza kolejnymi generacjami. To tutaj decydowane jest, czy komórka

przeżyje czy umrze po bieżącej generacji. *Print_board* odpowiada za wyświetlanie bieżącego stanu macierzy z komórkami. *Get_terminal_size* jest natomiast funkcją wybitnie pomocniczą i wspomaga mechanizm uniemożliwiający wygenerowanie macierzy większej niż terminal jest w stanie wyświetlić bez konieczności załamywania linii.
Kod programu może wydawać się krótki, a jest tak ze względu na to, iż zastosowałem mechanizm *#define*, dzięki któremu byłem w stanie znacznie szybciej i wydajniej pisać kod. Zastąpiłem tak dwa bloki *for*, utworzyłem zmienną *CELL(x,y)*, oraz mini-funkcje *SET_LIVE*, *SET_DEAD*, oraz *MIN(X,Y)*.
Po uruchomieniu program rozpoznaje z jakimi flagami został uruchomiony i dostosowuje swoje działanie do nich. Następnie zajmowana jest pamięć niezbędna na przeprowadzenie generacji. Kolejnym krokiem jest wygenerowanie planszy z komórkami. Stany komórek generowane są losowo. Po tym następuje proces n-iteracji, na który składa się wypisanie na standardowe wyjście początkowej planszy, macierzy z komórkami, oraz generacja kolejnego stanu komórek. Potem zwalniana jest zajęta pamięć i program kończy swoje działanie.

3 Wnioski

Projekt był ciekawym, aczkolwiek delikatnie irytującym, powrotem do programowania w języku C, który nie jest preferowanym przeze mnie językiem. Niemniej jednak, główną trudnością okazało się spełnienie wymagać środowiska QNX, które nie miało kilku lubianych przeze mnie bibliotek. QNX wymagał również skrupulatności przy pisaniu kodu programu. Sama tematyka *gry w życie* była już mi znajoma, lubię ją, więc nie miałem większych problemów z napisaniem programu. Projektowanie oprogramowania na system QNX wymaga nieco większej uwagi właśnie ze względu na potrzebę dokładnego przyjrzenia się kwestiom kompatybilności, lecz w gruncie rzeczy nie różni się bardzo od oprogramowania pisanego na maszyny używane przez nas na co dzień co czyni pracę z systemami QNX dostępną dla wielu programistów.