Automat komórkowy

Czym jest automat komórkowy?

Automat komórkowy jest to system składający się z pojedynczych komórek, sąsiadujących ze sobą według ustalonego wzorca. Liczba stanów, które każda komórka może przyjąć jest skończona, lecz dowolnie duża. Stan komórki jest zmieniany na podstawie reguł zawartych w programie, które mówią w jaki sposób w zależności od sąsiedztwa oraz obecnego stanu zależy jej kolejny stan.

Działanie programu

Nasz program został napisany w całości w języku C. Program uruchamiamy za pomocą konsoli tekstowej. Przykłady działania programu:

1. Komenda „make” kompiluje nam cały program, powoduje pojawienie się pliku wykonywalnego moor, który po uruchomieniu „./moor nazwa_pliku ilość_iteracji” (gdzie nazwa pliku jest plikiem tekstowym z podaną macierzą 0 i 1 oraz podanymi kolumnami i rzędami) wywołuje program „gry w życie” z użyciem sąsiedztwa Moor’a. Kolejne obrazy w formacie pbm zostają tworzone w folderze „outputs” i jest ich tyle ile podano iteracji. Poniżej zamieszczamy przykładowe działanie programu oraz przykładowy plik tekstowy:

```
root@DESKTOP-VGQ1CLS:~/automatkomorkowy/Automat-Komorkowy# ./moor test/good 10
root@DESKTOP-VGQ1CLS:~/automatkomorkowy/Automat-Komorkowy# cd outputs
root@DESKTOP-VGQ1CLS:~/automatkomorkowy/Automat-Komorkowy/outputs# ls
info.txt     output10.pbm  output3.pbm  output5.pbm  output7.pbm  output9.pbm
output1.pbm  output2.pbm   output4.pbm  output6.pbm  output8.pbm
root@DESKTOP-VGQ1CLS:~/automatkomorkowy/Automat-Komorkowy/outputs#
```

```
P1
5 5
0 0 0 0 0
0 1 0 1 0
0 0 1 0 0
0 1 0 1 0
0 0 0 0 0
```

2. Komenda „make neumann” powoduje to samo co wyżej zamieszczone, z tym że zamiast pliku „moor” pojawia się plik „neumann”, a gra bazuje na sąsiedztwie Neumanna. Zasady gry pozostają niezmienne.
3. Komenda „make clean” czyści folder outputs, a także pliki wykonywalne oraz pliki *.o.

Jak dodać swoją grę?

Najlepiej skorzystać z funkcji (zmienić jej działanie) breedAndKill, znajdującej się w pliku world/gameOfLife.c, natomiast nic nie stoi na przeszkodzie w dołączeniu kolejnej funkcji w oddzielnym pliku, przy czym należy pamiętać aby umieścić pliki nagłówkowe takie jak „include „mat.h”” oraz „include „games.h”” w podanej kolejności, następnie umieścić prototyp swojej funkcji w pliku nagłówkowym „games.h”. Następnie najwygodniej jest zamienić wszelkie użycia funkcji breedAndKill w innych wykorzystujących ją plikach na tą, z której chcielibyśmy skorzystać. Po wykonaniu powyższych kroków należy dodać własną implementację tego pliku w pliku Makefile.

Autorzy: Michał Ankiersztajn, Mikołaj Guryn
Projekt powstał na potrzeby zajęć na Politechnice Warszawskiej