Gra w życie - elohim.c

Michał Balas, Jan Dobrowolski

2 marca 2019

Spis treści

Cel projektu

Program elohim.c jest przykładem automatu komórkowego symulującego "grę w życie" Johna Conwaya. Ma on na celu wygenerowanie określonego zbioru komórek w stanie końcowym automatu oraz opisu chwilowego dla N przejść symulacji (oraz ich możliwej wizualizacji w pliku graficznym).

Dane wejściowe

Do poprawnego działania programu wymagane są:

- informacja o stanie początkowym układu (w postaci użycia flagi –random lub –input przy wywołaniu programu);
- liczba generacji N, gdzie $100 \geq N > 0$.

Dodatkową funkcjonalnością zawartą w programie jest tworzenie planszy o zadanych przez użytkownika wymiarach i losowe jej wypełnianie.

Parametry wywołania programu

Program elohim.c może być wywołany z następującymi parametrami:

- "–help" otwiera pomoc i opis komend programu;
- "–defchar [char]" determinuje jeden znak, który będzie użyty przy generowaniu lub wczytywaniu planszy jako symbol aktywnej, lub żywej komórki (do poprawnego wczytania planszy z pliku wymagana jest zgodność użytych znaków z podanym tu argumentem!). Domyślnie jest to znak "1", a więc plik wejściowy (parametr "–input [plik.txt]") powinien zawierać ciąg znaków "0" (niezmienne) i "1";

- “–random x y” determinuje generowanie początkowej planszy o zadanych wymiarach (x,y), gdzie punktem startowym jest lewy-górny róg ekranu (współrzędna (0,0)). Następnie plansza wypełniania jest losowo zerami i jedynkami. $100 \geq x > 0, 100 \geq y > 0$;
- “–input [plik.txt]” określa plik w formacie tekstowym, który będzie służył jako stan początkowy automatu. Warunkiem poprawności integracji pliku z programem jest wypełnienie go wyłącznie symbolami “0” i “1” ułożonymi w kształt dowolnego prostokąta. Rozmiar prostokąta staje się wówczas rozmiarem symulowanej planszy;
- “–ngen n” jest liczbą generacji, czyli zmian stanów automatu. n>0;
- “–output “ścieżka”” zmienia docelową lokalizację pliku/plików z danymi wyjściowymi. Domyślnie jest to folder “./output/ “;
- “–backup [nazwa.txt]” tworzy plik tekstowy w bieżącym folderze, który potem może zostać wczytany jako argument wejścia programu. Domyślnie tablica z n-tą generacją zostanie wypisana do stdout.

Przykładowa zawartość pliku plik.txt (przy wywołaniu można użyć flagi “ –input 1 ”):
0 0 0 0 0 0 0 0 0 0
0 1 0 0 1 1 1 1 0 0
0 1 0 0 1 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0
0 1 1 1 1 1 1 1 0 0
0 0 0 0 1 0 0 1 0 0
0 0 0 0 1 0 0 1 0 0
0 1 1 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0

Przykładowe wywołanie programu:

1. **./elohim –help**
   Otwiera pomoc i opis komend programu

2. **./elohim –defchar # –random 10 10 –ngen 5 –output ./data/myfiles/**
   W tym przypadku utworzona zostanie plansza 10x10, która zostanie losowo wypełniona znakami "0" oraz "#". Następnie przeprowadzone zostanie 5 generacji, z których każda zostanie zapisana w pliku [numer generacji].png w ścieżce "./data/myfiles/*".

3. **./elohim –input plik.txt –ngen 20 –backup brb.txt**
   Tutaj jako stan początkowy układu pobrana zostanie zawartość pliku plik.txt (pod warunkiem, że zawartość pliku spełnia kryteria programu) oraz wygenerowane zostanie 20 kolejnych stanów, z których każdy zostanie zapisany w oddzielnym pliku graficznym w folderze "./output/". Ponadto w folderze bieżącym (w którym znajduje się plik wykonywalny elohim) utworzony zostanie plik tekstowy brb.txt zawierający dwudziesty (ostatni) stan automatu.

Teoria

Automat komórkowy to model, który operuje na skończonym zbiorze komórek, które przylegają do siebie, tworząc dwuwymiarową lub trójwymiarową siatkę. Każda komórka opisywana jest przez stan, który zazwyczaj oznaczony jest poprzez kolor. Stan wszystkich komórek w danej chwili nazywamy "generacją". Gdy stan generacji jest ustalony, możliwe jest utworzenie nowej (potomnej) generacji komórek. Nowy stan danej komórki zależy od jej obecnego stanu oraz od obecnych stanów jej sąsiadów.

Dla każdego automatu istnieje zbiór reguł określających jak mają się zmieniać stany komórek. To, które komórki brane są pod uwagę przy określaniu stanu komórki danej określa wybrany model sąsiedztwa. Dla siatek dwuwymiarowych najpopularniejsze takie modele to:

- sąsiedztwo Moore'a: 8 przylegających komórek (znajdujących się: na południu, na południowym-zachodzie, na zachodzie, na północnym-zachodzie, na północy, na północnym-wschodzie, na wschodzie i na południowym-wschodzie).

- sąsiedztwo von Neumanna: 4 przylegające komórki (na południu, zachodzie, północy i wschodzie).

“Gra w życie” Johna Conwaya to przykład automatu komórkowego, korzystającego z sąsiedztwa Moore’a. Każda z komórek, na której on operuje, znajduje się w jednym z dwóch stanów: “żywym” (oznaczanym przez kolor czarny lub cyfrę 1) lub “martwym” (oznaczanym przez kolor biały lub cyfrę 0).
Zestaw reguł obowiązujących w “Grze w życie” jest następujący:

- Martwa komórka, która ma dokładnie 3 żywych sąsiadów, staje się żywa w następnej generacji (rodzi się);
- Żywa komórka z 2 albo 3 żywymi sąsiadami pozostaje żywa w następnej generacji; przy innej liczbie sąsiadów umiera (z "samotności" albo "zatłoczenia").

Komunikaty błędów

Program “elohim” może wyeliminować część błędnych argumentów wywołania, poprzez zignorowanie niepoprawnego żądania użytkownika i zastąpienie go domyślną wartością lub przerwanie pracy programu. Lista potencjalnych błędów oraz obsługujące je działania lub komunikaty:

1. Wywołanie programu bez flagi –random lub –input
   “Brak parametrów wejściowych. Działanie programu zostanie przerwane.”
2. Nieistniejący plik wejściowy lub brak zdefiniowanego pliku po parametrze ‘input’.
   “Brak pliku wejściowego. Działanie programu zostanie przerwane”.
3. Zły format pliku wejściowego.
   “nazwapliku.format: Zły format pliku wejściowego. Program czyta pliki w formacie txt. Działanie programu zostanie przerwane.”
4. Niedozwolone znaki w zawartości pliku wejściowego.
   “nazwapliku.format W pliku znaleziono niedozwolone znaki. Działanie programu zostanie przerwane.”
5. Rozmiary planszy wykraczające poza zakres lub nie będące liczbami naturalnymi.
   “Niewłaściwy rozmiar planszy! Ustawiam domyślne wartości 10x10”

6. Nieprawidłowa liczba generacji (wykraczająca poza zakres lub nienaturalna)
   "Błędna liczba generacji. Ustawiam N = [liczba]" gdzie n jest zaokrągleniem w dół podanego argumentu lub górną/dolną granicą zakresu (w zależności do której z nich podana liczba ma bliżej)

7. Flaga "–defchar" - podano więcej, niż jeden znak.
   "Podano błędny argument flagi defchar, przywracam domyślną wartość"

8. Podanie dwóch takich samych flag:
   "Zdublowana flaga [nazwa flagi]. Używam pierwszego wywołania flagi."

9. Użycie flagi –help z innymi flagami
   Wyświetlenie pomocy i zignorowanie pozostałych flag.

Dane wyjściowe

W przypadku braku podania flagi –backup, n-ta generacja symulacji zostanie wypisana do stdout (jeżeli jej rozmiar nie przekracza 20x20). Ponadto utworzone zostaną pliki graficzne w folderze ./output/ zawierające wizualizacje kolejnych generacji automatu.

Użycie flagi –backup spowoduje wypisanie n-tej generacji do pliku tekstowego o zadanej nazwie, który zostanie utworzony zostanie w folderze output lub w lokalizacji określonej przy fladze –output.