|  |  |
| --- | --- |
| JIMP1 | Inf. Stos. 2023Z |
| Mrówka Langtona | 18.01.2024 |
| Oliwia Pawelec, Jakub Żebrowski G1 | |

# Problem

Mrówka Langtona to prosty automat komórkowy. Opiera się na następujących zasadach:

*Mrówka chodzi po dwuwymiarowej siatce. Może poruszać się w jednym z 4-ech kierunków (góra, dół, lewo, prawo), zgodnie z następującymi zasadami:*

*• Mrówka znajduje się w komórce białej, wykonuje: obrót o 90 stopni w prawo, zmienia kolor komórki na przeciwny, przesuwa się o jedną komórkę do przodu*

*• Mrówka znajduje się w komórce czarnej, wykonuje: obrót o 90 stopni w lewo, zmienia kolor komórki na przeciwny, przesuwa się o jedną komórkę do przodu*

Za zadanie mieliśmy napisać program pokazujący kroki mrówki Langtona na planszach. Każdą planszę (krok) należało wypisać na konsolę lub do wskazanego pliku.

Nasz program prezentuje następujące funkcjonalności:

* Tworzy symulację działania automatu dla i-operacji (podane przez użytkownika)
* Wytwarza planszę z zapełnionymi polami, będącymi jakimś % całości – podane przez użytkownika
* Zapisuje te działa do plików z początkową nazwą podaną przez użytkownika – kolejne będą się nazywać *nazwa\_nriteracji*
* Umożliwia wczytanie dowolnego stanu planszy do programu i działanie na nim
* Wywołanie funkcji *help*, która opisuje działanie programu

# Struktura programu

Program składa się z 4 plików \*.c i 5 plików nagłówkowych.

Struktura:

* constants.h
* field.c field.h
* ant.c ant.h
* manager.c manager.h
* main.c

constants.h:

Plik constants.h zawiera stałe dla programu – znaki specjalne używane do wypisywania planszy i mrówki do plików oraz maksymalne wielkości planszy – 1000.

field.h field.c

Pliki field.h i field.c zawierają informację o strukturze planszy, którą będzie przechodzić mrówka. Składa się z białych i czarnych pól (wartości 0 i 1).

typedef struct {

int m, n;

char field[M\_MAX][N\_MAX];

} field\_t;

W tym pliku znajduje się również generator losowych przeszkód (czarnych pól).

# Wywołanie programu

Program można skompilować za pomocą komendy make:

make

Lub ręcznie skompilować:

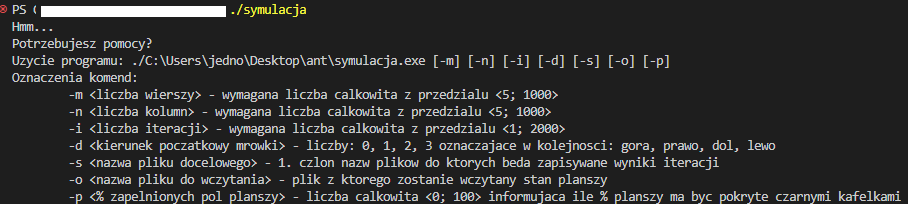
gcc -Wall field.c ant.c manager.c main.c -o mrowka

Następnie wywołać za pomocą:

./mrowka [komendy]

Aby wywołać komendę help (komendę z poleceniami) należy wpisać:

./symulacja



Rys. Działanie wywołania ./symulacja (plik wykonywalny programu)

Program można użyć

./mrowka [-m] [-n] [-i] [-d] [-s] [-o] [-p]

Gdzie:

* m <liczba wierszy> - wymagana liczba całkowita z przedziału <5; 1000>
* n <liczba kolumn> - wymagana liczba całkowita z przedziału <5; 1000>
* i <liczba iteracji> - wymagana liczba całkowita z przedziału <1; 2000>
* d <kierunek początkowy mrówki> - liczby: 0, 1, 2, 3 oznaczające w kolejności: góra, prawo, dół, lewo
* s <nazwa pliku docelowego> - 1. człon nazw plików do których Beda zapisywane wyniki iteracji
* o <nazwa pliku do wczytania> - plik z którego zostanie wczytany stan planszy
* p <% zapełnionych poł planszy> - liczba całkowita <0; 100> informująca ile % planszy ma być pokryte czarnymi kafelkami

# Przykładowe działanie programu

dla różnych ustawień

# Wnioski

Problem automatu nie był sam w sobie trudny – instrukcje były trywialne, a sama implementacja chodzącej mrówki była przyjemna (trójkąt-mrówka zmieniająca swoje pozycje między plikami wyglądała przezabawnie).

Największym wyzwaniem było ~~pisanie tego sprawozdania~~, wprowadzenie funkcji getopt do programu – pierwszy raz mieliśmy z tym styczność – testowanie kodu oraz używanie gita w nie-indywidualna praca.