Łukasz Jarzęcki Paweł Skierkowski

# MRÓWKA LANGTONA

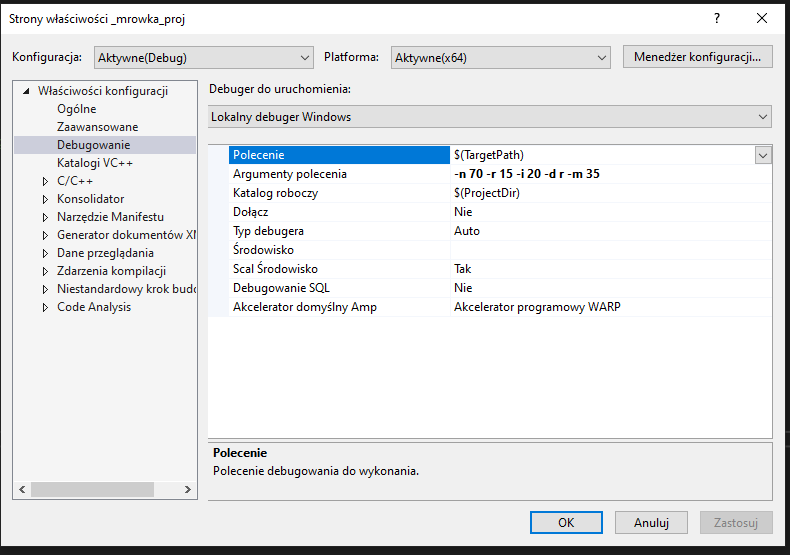
Zasady działania algorytmu:

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

* Jeżeli mrówka znajduje się na białej komórce wykonuje: obrót o 90 stopni w prawo (zgodnie z ruchem wskazówek zegara), zmienia kolor komórki na przeciwny oraz przesuwa się o jedną komórkę do przodu.
* Jeżeli mrówka znajduje się na czarnej komórce wykonuje: obrót o 90 stopni w lewo (przeciwnie do ruchu wskazówek zegara), zmienia kolor komórki na przeciwny oraz przesuwa się o jedną komórkę do przodu
* Mrówka nie może wyjść poza granicę planszy, dlatego w przypadku, gdy mrówka miałaby przemieścić się poza granicę algorytm kończy swoje działanie

Wywoływanie programu:

Aby wywołać program z powłoki konsoli należy skompilować program a następnie wywołać go z odpowiednimi parametrami podanymi poniżej.

Innym sposobem na wywołanie programu jest stworzenie projektu w visual studio, dodanie plików źródłowych, a następnie wejście we właściwości projektu i podanie parametrów w oknie ”argumenty polecenia”:

Program jest przystosowany do uruchomienia na systemie Windows.

Parametry możliwe do podania podczas uruchamiania to:

-m -liczba rzędów planszy /domyślnie 50

-n -liczba kolumn planszy /domyślnie 50

-i -ilość iteracji /domyślnie 20

-f -przedrostek pliku wyjściowego /domyślnie program wypisuje na stdout /nie podajemy rozszerzenia a jedynie nazwę – domyślnie tworzy pliki .txt

-l nazwa pliku wejściowego /jeżeli chcemy aby plansza została wygenerowana automatycznie pomijamy parametr

-r <0,100>- procentowa ilość czarnych pól wybranych losowo /domyślnie 0 /pomijane przy wczytywaniu planszy

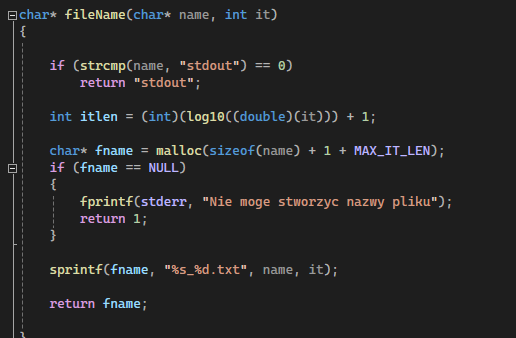
-d <u/d/l/r> - początkowy kierunek mrówki /domyślnie mrówka skierowana jest w górę /pomijane przy wczytywaniu planszy

Podział programu na moduły:

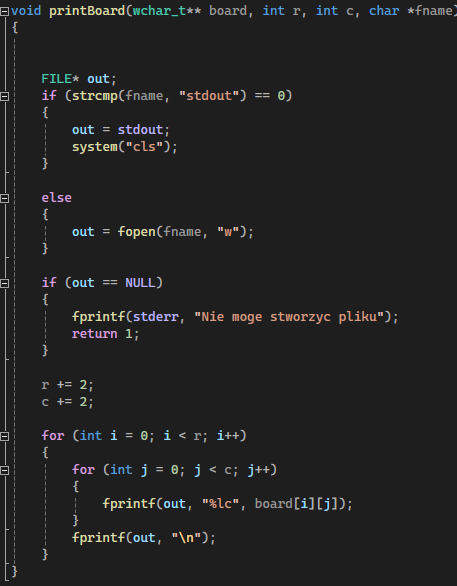
Program podzielony jest na moduły:

* board.c board.h //odpowiedzialny za inicjację, generowanie lub wczytywanie planszy
* fhandling.c fhandling.h //odpowiedzialne za obsługę plików
* macro.h //zawiera specjalne znaki oraz załącza biblioteki
* main.c //główne ciało programu
* move.c move.h //odpowiedzialne za przechodzenie mrówki po planszy
* getopt.c getopt.h //implementacja getopta dla visual studio

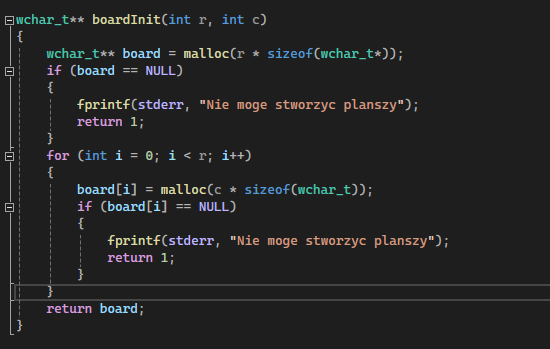
Przykładowe funkcje wraz z opisem:



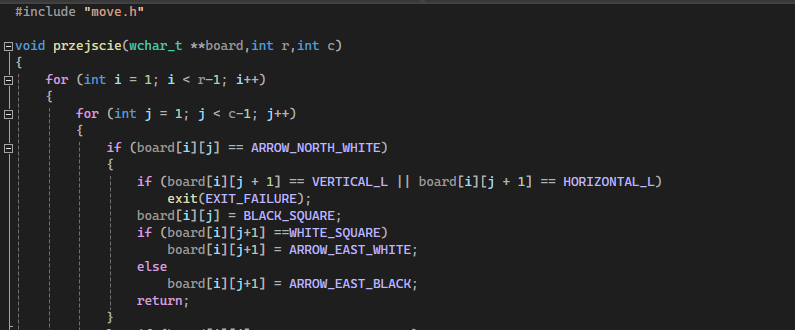
Jest to funkcja służąca do zwracania nazwy pliku po podania przedrostka i iteracji.



Funkcja printBoard służy do wypisywania elementów tablicy (w zależności od podanych argumentów robi to w konsoli rób zapisuje do pliku)



Funkcja boardInit służy do inicjacji planszy. Wykorzystuje parametry r i c będące wymiarami planszy, a następnie przydziela tablicy pamięć.

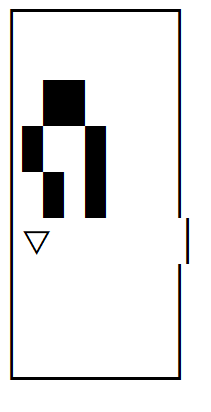
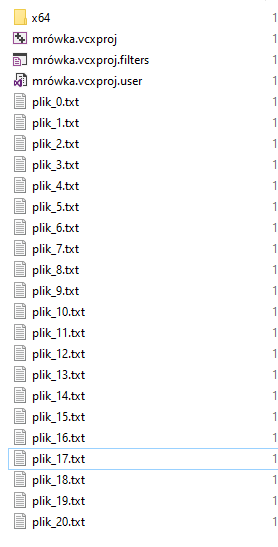


Powyżej widnieje fragment funkcji przejście odpowiadającej za główne działanie algorytmu. Przyjmuje ona planszę oraz jej wymiary. Następnie za pomocą pętli przechodzi po kolejnych wartościach tablicy dwuwymiarowej (zaczynając od drugiego wiersza i drugiej kolumny ponieważ zajęte one są przez obramowanie). Funkcja zgodnie z założeniami algorytmu porównuje pola planszy. Po pierwsza sprawdza czy kolejne pole nie jest granicą i kończy działanie programu, kiedy napotka krawędź (exit(EXIT\_FAILURE)). Zmienia kolor pola na którym się znajduje na przeciwny, obraca się w odpowiednim kierunku i przesuwa się na kolejne pole.

Przykładowe działanie programu dla różnych ustawień:



Dla wpisanego przedrostka pliku program utworzył 20 plików +1 z początkowym położeniem (plik\_0.txt) (z powodu wejścia na granicę planszy nie ma pliku plik\_21.txt)



W przypadku pominięcia przedrostka w argumentach wywołania program wypisuje kolejne przejścia w konsoli.

Obraz zawierający tekst, zrzut ekranu, komputer, oprogramowanie

Opis wygenerowany automatycznie

Po każdym przejściu konsola jest czyszczona

Wnioski:

Praca w grupie nad projektem ma wiele zalet. Podział obowiązków, przystosowywanie się do przyszłej pracy, w której będziemy musieli współpracować z innymi programistami. Dużym ułatwieniem jest korzystanie z repozytorium na github. W bardzo prosty sposób mogliśmy zamieszczać gotowe rozwiązania i konsultować swoje bieżące postępy w pracy nad modułami. Algorytm pomimo braku dużej złożoności był ciekawy. Nieraz napotkaliśmy problemy, jednak po długich staraniach i wspólnych wysiłkach udało się je rozwiązać.