Dokumentacja Funkcjonalna Projektu Labirynt

Polina Nesterova
01192875@pw.edu.pl

Martyna Kochalska
01187553@pw.edu.pl

25 lutego 2024

1 Opis programu

Program umożliwia znalezienie najkrótszej ścieżki labiryntu. Labirynt jest dostarczany w formie pliku tekstowego, zawierającego definicje punktów reprezentujących różne elementy labiryntu (P, K, X, spacja). Dodatkowo, program jest ograniczony do użycia maksymalnie 512 kB pamięci.

2 Parametry wejściowe

Aby uruchomić program, należy w konsoli wpisać nazwę programu razem z dodaniem nazwy pliku wczesniej wygenerowanego labiryntu.

```
maze_way.out -f [nazwapliku.txt]
```

- `-f [nazwapliku.txt]` parametr obowiązkowy. Odpowiada za dodanie pliku wejściowego. Wymaga podania nazwy ścieżki do pliku z odpowiednio sformatowanym labiryntem.
- `-o [nazwapliku.txt]` parametr opcjonalny. Po dodaniu parametru wynik działania programu zostanie przekierowany do wskazanego pliku. W przypadku braku jego istnienia plik zostanie utworzony.
- `-h` parametr opcjonalny. Wyświetla krótką pomoc do programu.

3 Format pliku wejsciowego

Plik labiryntu powinien zostać wygenerowany za pomocą strony tob.iem.pw.edu.pl/maze/ Labirynt powinien zostać podany w postaci pliku tekstowego zawierającego definicje punktów reprezentujących odpowiednio:

- P - punkt wejścia do labiryntu,

- K punkt wyjścia z labiryntu,
- X – ściana,
- spacja – miejsce, po którym można się poruszać.

Program nie przyjmuje labiryntów o wielkości ponad 1024 x 1024 (liczonych wg ilości ścieżek, po których można się poruszać. Labirynt podany w niewlasciwym formacie zwróci bląd o numerze 4. Przykładowy format labiryntu 5 x 4:

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
P               xxx     xxx     xxx     x
x     xxx       xxx     xxx     xxx     x
x     xxx               xxx             x
x  x  xxx  x    xxx     xxx     xxx     x
x               xxx             xxx     x
x     xxx       xxx     xxx     xxx     x
x     xxx               xxx     xxx     K
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

4 Parametry wyjściowe

1. Lista wykonanych kroków, takich jak:
   - START
   - FORWARD 1
   - TURNLEFT
   - FORWARD 4
   - TURNRIGHT
   - FORWARD 3
   - STOP
2. Informacje o błędach

5 Obsługa błędów

Obsługa przypadków wyjątkowych:

- Kod 2: "Brak możliwości wczytania pliku" komunikat o błędzie, zakończenie działania. Powodem tego komunikatu może być podanie nieprawidłowej ścieżki do pliku .
- Kod 3: "Brak wystarczającej pamięci" komunikat o błędzie, zakończenie działania. Program dostał niewsytarczającą ilość pamięci. (<512kB)

- Kod 4: "Nieprawidłowy format pliku" komunikat o błędzie, zakończenie działania. Plik wejściowy został podany w niewłaściwym formacie. Należy sprawdzić, czy plik na pewno zawiera labirynt w formacie opisanym w sekcji `Format pliku wejsciowego`

6 Teoria - Proces działania programu

1. Wczytanie pliku z labiryntem
2. Analiza danych i zbudowanie grafu reprezentującego labirynt
3. Zastosowanie algorytmu przeszukiwania grafu dla znalezienia najkrótszej ścieżki od wejścia do wyjścia
4. Generowanie listy wykonanych kroków
5. Wyświetlenie listy lub informacji o błędach

7 Moduły

1. Wczytywanie pliku
   - Odpowiada za wczytanie pliku z definicją labiryntu
   - Przetwarzanie informacji z pliku na strukturę danych zrozumiałą dla programu

   Plik – `file.c`
2. Analizy labiryntu
   - Przetwarza dane labiryntu ze struktur danych
   - Tworzy reprezentację grafu

   Plik - `maze_analysis.c`
3. Algorytm przeszukiwania grafu
   - Implementacje algorytmów przeszukiwania grafu (np. BFS lub DFS)
   - Wykorzystuje graf labiryntu do znalezienia ścieżki

   Plik - `graph_search.c`
4. Generowanie listy kroków
   - Na podstawie wyników działania algorytmu przeszukiwania generuje listę kroków
   - Tworzy strukturę danych zawierającą kolejność kroków potrzebnych do przejścia

Plik - `path_generation.c`

5. Wyświetlanie wyników
   - Prezentacja wyników działania użytkownikowi
   - Wyświetla listę kroków / komunikaty o błędach

   Plik - `result_display.c`

6. Główny
   - Uruchamia poszczególne moduły w odpowiedniej kolejności
   - Zarządza interakcją z użytkownikiem (np. pobiera nazwę pliku)

   Plik – `main.c`

7. Makefile
   - Plik konfiguracyjny

   Plik – `Makefile`

8. Utils
   - Funkcje pomocnicze (np. obsługa błędów, alokacja pamięci itd.)

   Plik – `utils.c`