Labirynt
Specyfikacja Funkcjonalna

Tomasz Rogalski Jakub Sokolik

23 kwietnia 2024

Spis treści

1 Opis ogólny

Program znajduje dowolną ścieżkę wyjścia z podanego na wejściu labiryntu, w postaci pliku tekstowego lub binarnego. W związku z tym, będzie on wyświetlał ten labirynt z zaznaczonym rozwiązaniem oraz wypisze instrukcję "krok po kroku"jak się z niego wydostać.

2 Opis funkcjonalności

2.1 Jak korzystać z programu?

W celu uruchomienia programu, najpierw należy go skompilować poleceniem *make*. Do działania programu potrzebujemy labiryntu w pliku tekstowym lub binarnym w tym samym folderze, w którym znajduje się program.

2.2 Uruchomienie programu

Program uruchamiamy przy użyciu komendy:

```
./program
```

Domyślnie zostanie pobrany plik tekstowy z przykładowym labiryntem i zostanie rozwiązany. Jeśli chcemy, aby program rozwiązał nasz labirynt, należy wybrać tryb, w zależności od rodzaju pliku, *-b* to plik binarny, a *-t* to plik tekstowy. Jeśli chcemy skorzystać z pliku tekstowego, to nie ma potrzeby wpisywania *-t*, ponieważ program domyślnie korzysta z tego rodzaju plików. Następnie należy wpisać po *-p* nazwę pliku i potwierdzić. Przykładowa próba uruchomienia programu wygląda tak:

```
./program -t -p bigmaze.txt
```

lub:

```
./program -p bigmaze.txt
```

W obu przypadkach otrzymamy ten sam wynik.

2.3 Dane wejściowe

Program przyjmuje labirynt w postaci pliku tekstowego lub binarnego. W tym pierwszym labirynt powinien się składać z: "X ścian, " pustych pól, po których można się poruszać oraz "P"i "K", czyli początku i końca labiryntu. Dane wejściowe w formacie binarnym składają się na 4 główne sekcje: nagłówek pliku, sekcja kodująca zawierająca powtarzające się słowa kodowe,

nagłówek sekcji rozwiązania oraz sekcja rozwiązania zawierająca powtarzające się kroki które należy wykonać aby wyjść z labiryntu.

2.4 Dane wyjściowe

Wynikiem programu będzie wypisany labirynt wraz z rozwiązaniem i instrukcja przejścia tego labiryntu. Poza tym zostanie utworzony plik, zawierający labirynt ze ścieżką rozwiązania. W przypadku plików binarnych, dodatkowo program wykorzysta miejsce do stworzenia jeszcze jednego pliku, przechowującego zdekodowane dane z pliku binarnego.

3 Scenariusz działania programu

3.1 Scenariusz główny

1. Uruchomienie.
2. Sprawdzenie wybranego trybu i nazwy pliku przez użytkownika.
3. Wyznaczenie wymiarów labiryntu oraz współrzędnych P oraz K.
4. Wypisanie labiryntu.
5. Oznaczenie i wyświetlenie labiryntu ze ścieżką.
6. Wypisanie instrukcji.
7. Zakończenie działania programu.

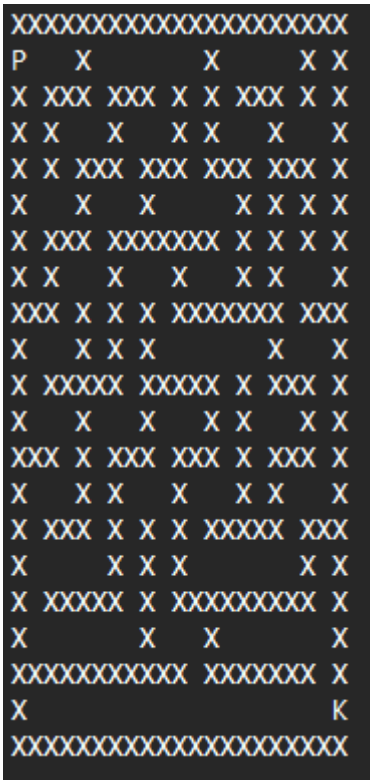
3.2 Scenariusz szczegółowy

1. Uruchomienie.
2. Sprawdzenie nazwy pliku podanej przez użytkownika
   (a) Jeżeli nie podał argumentu, wyświetl komunikat o błędzie, zakończ.
   (b) Sprawdź czy dany plik da się otworzyć i czy istnieje, jeśli nie, zakończ.
3. Sprawdzenie wybranego trybu przez użytkownika
   (a) Jeżeli brak trybu, to przyjmuj pliki tekstowe.
   (b) Jeżeli podano oba tryby, wyświetl komunikat o błędzie, zakończ.
   (c) Jeżeli wybrano plik binarny to przypisz dane z pliku do zmiennych i przekaż je do pliku tekstowego.
   (d) Jeżeli wybrano plik tekstowy to określ wymiary labiryntu oraz pozycję P i K w labiryncie.
4. Wypisanie labirynt.
5. Oznaczenie i wyświetlenie labiryntu ze ścieżką.
6. Wypisanie instrukcji.
7. Zakończenie działania programu.

4 Testowanie

Do przetestowania kodu w środowisku Unixowym wykorzystaliśmy Linux Ubuntu i skrajne przypadki labiryntów, tzn. bardzo mały, bardzo duży, z kilkoma możliwymi rozwiązaniami oraz z pliku binarnego. Jeżeli pojawią się błędy, będziemy w stanie je zlokalizować dość szybko przez różnicę badanych plików. Testowe labirynty:

- smallmaze.txt - labirynt 10x10,



Rysunek 1: Labirynt smallmaze.txt

- maze.txt - labirynt 20x20,
- bigmaze.txt - labirynt 250x250,
- hugemaze.txt - labirynt 512x512,
- maze1024_1024.txt - labirynt 1024x1024,
- maze20%.txt - labirynt 50x50, mający w 20% usunięte ściany,

- maze.bin - labirynt 256x256 zapisany w pliku binarnym.