

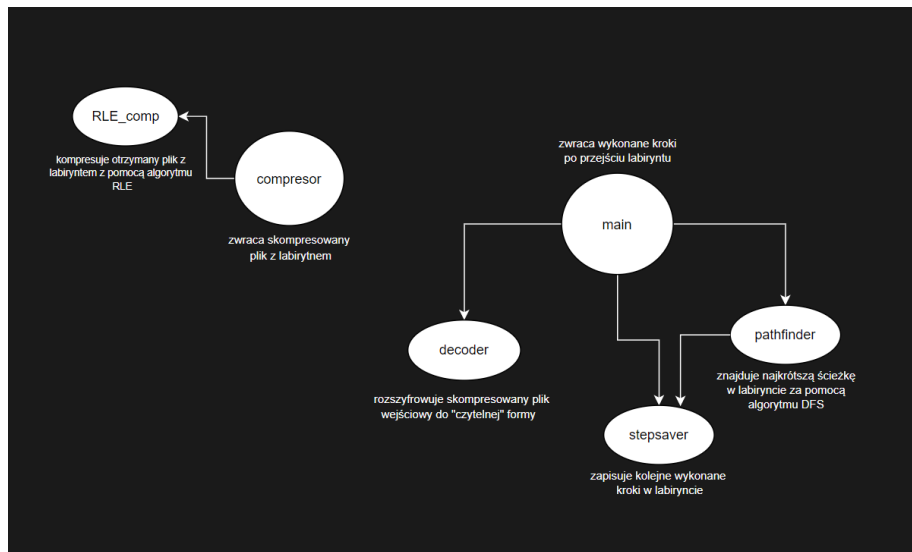
A-MAZE-ING

Kacper Warpechowski i Adam Boros

25.03.2024r.

Specyfikacja implementacyjna

1. Diagram modułów



2. Opis struktur danych

Labirynt będzie reprezentowany jako dwuwymiarowa tablica punktów, gdzie każdy punkt ma przypisaną wartość odpowiadającą typowi (P dla punktu wejścia, K dla punktu wyjścia, X dla ściany, [spacja] dla przestrzeni). Kolejno, struktura kroków będzie reprezentowana poprzez wektor (np. ["FORWARD 2", "TURN-LEFT"]). Dzięki tym strukturom, program będzie efektywnie zarządzał informacjami o labiryncie, umożliwiając znajdowanie najkrótszej ścieżki oraz generowanie listy kroków.

3. Lista funkcji w poszczególnych modułach

Compresor

main():

Główna funkcja, która przekazuje plik z labiryntem do modułu RLE.comp, w wyniku czego z otrzymanego tekstu tworzy nowy skompresowany plik z labiryntem.

RLE.comp

codeRLE(ciąg_znakow: string)

Funkcja kompresująca labirynt do postaci tekstu, wykorzystując algorytm kompresji RLE (Run-Length Encoding). Kompresja polega na zliczaniu kolejnych powtórzeń znaków w labiryncie i zapisywaniu ich jako liczbe wystąpień, poprzedzona odpowiednim znakiem. Zwraca skompresowany labirynt jako tekst.

```
4
5 char* codeRLE(const char* input) {
6
7     int length = strlen(input);
8     char* encoded = malloc((2 * length + 1) * sizeof(char));
9     if (!encoded) {
10         printf("Błąd alokacji pamięci.\n");
11         exit(EXIT_FAILURE);
12     }
13
14     int index = 0;
15     int combo = 1;
16     char current_char = input[0];
17     for (int i = 1; i <= length; i++) {
18         if (input[i] == current_char) {
19             combo++;
20         } else {
21             encoded[index++] = current_char;
22             if (combo > 1) {
23                 encoded[index++] = combo + '0'; // zamiana liczby na znak
24             }
25             current_char = input[i];
26             combo = 1;
27         }
28     }
29     encoded[index] = '\0'; // zakończ ciąg znakiem null
30
31     return encoded;
32 }
```

funkcja codeRLE

decoder

`decode_maze(dane_skompresowane: string) -> Labirynt: Macierz`

Funkcja dekompresująca dane skompresowane algorytmem RLE do postaci labiryntu (macierzy). Odczytuje dane skompresowane z pliku i dekoduje je, przywracając pierwotny labirynt.

pathfinder

`find_path(labirynt: Macierz) -> Lista Kroków: Wektor`

Funkcja znajdująca najkrótszą ścieżkę od punktu wejścia do punktu wyjścia w labiryncie. Wykorzystuje algorytm DFS w celu znalezienia najkrótszej ścieżki. Zwraca listę kroków potrzebnych do przejścia przez labirynt jako wektor.

stepsaver

`save_steps(Lista kroków: wektor, nazwa_pliku: string):`

Funkcja zapisująca wektor wykonanych kroków do pliku tekstowego. Zapisuje listę kroków w czytelnej formie, która może być odczytana przez użytkownika w celu śledzenia przebytej ścieżki.

Main

`main():`

Główna funkcja uruchamiająca właściwy program szukający ścieżki w labiryncie. Odczytuje skompresowany labirynt z pliku, dekompresuje go, znajduje najkrótszą ścieżkę przy użyciu funkcji `find_path`, a następnie zapisuje listę wykonanych kroków do pliku. Dodatkowo zawiera przełącznik `-h` z pomocnymi informacjami dla użytkownika.