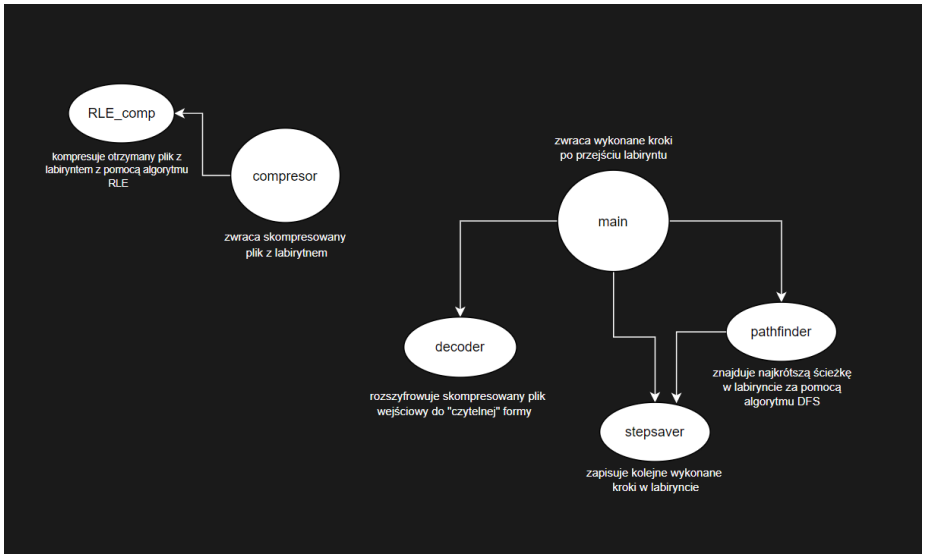
A-MAZE-ING

Kacper Warpechowski i Adam Boros

25.03.2024r.

Specyfikacja implementacyjna

1. Diagram modułów



2. Opis struktur danych

Labirynt bedzie reprezentowany jako dwuwymiarowa tablica punktów, gdzie każdy punkt ma przypisana wartość odpowiadajaca typowi (P dla punktu wejścia, K dla punktu wyjścia, X dla ściany, [spacja] dla przestrzeni). Kolejno, struktura kroku bedzie reprezentować pojedyncza operacje wykonana w labiryncie, z informacja o rodzaju kroku (np. "FORWARD", "TURNLEFT") oraz opcjonalnie ilości kroków. Ścieżka bedzie zawierać liste kroków potrzebnych do przejścia przez labirynt. Dzieki tym strukturom, program bedzie efektywnie zarzadzał

informacjami o labiryncie, umożliwiajac 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(ciag_znakow: string)`
Funkcja kompresujaca labirynt do postaci tekstu, wykorzystujac algorytm kompresji RLE (Run-Length Encoding). Kompresja polega na zliczaniu kolejnych powtórzeń znaków w labiryncie i zapisywaniu ich jako liczbe wystapień, 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:`
Funkcja dekompresujaca dane skompresowane algorytmem RLE do postaci labiryntu. Odczytuje dane skompresowane z pliku i dekoduje je, przywracajac pierwotny labirynt.

pathfinder

`find_path(labirynt: Labirynt) -> Lista Kroków:`
Funkcja znajdujaca najkrótsza ścieżke od punktu wejścia do punktu wyjścia w labiryncie. Wykorzystuje algorytm DFS w celu znalezienia najkrószej ścieżki. Zwraca liste kroków potrzebnych do przejścia przez labirynt.

stepsaver

`save_steps(sciezka: Lista Kroków, nazwa_pliku: string):`
Funkcja zapisujaca liste wykonanych kroków do pliku tekstowego. Zapisuje liste kroków w czytelnej formie, która może być odczytana przez użytkownika w celu śledzenia przebytej ścieżki.

Main

`main():`
Główna funkcja uruchamiajaca właściwy program szukajacy ścieżki w labiryncie. Odczytuje skompresowany labirynt z pliku, dekompresuje go, znajduje najkrótsza ścieżke przy użyciu funkcji `find_path`, a nastepnie zapisuje liste wykonanych kroków do pliku. Dodatkowo zawiera przelacznik -h z pomocnymi informacjami dla użytkownika.