Mapa Miasta – Klasy i Struktury Danych

Paweł Rapacz

1. Klasa App – klasa obsługująca interfejs użytkownika i działanie programu
   1. Pola klasy
      * CliOptions options\_ - zawiera opcje linii poleceń
      * char\*\* argv\_ - wskaźnik do tablicy argumentami linii poleceń
      * int argc\_ - liczba argumentów linii poleceń
      * std::vector results\_ - zawiera wyniki działania programu
   2. Metody klasy
      * Domyślny konstruktor i destruktor
      * run() – uruchamia działanie programu
      * handleCli() – obsługuje linię poleceń
2. Struktura CliOptions – struktura zawierająca wartości opcji linii poleceń podanych przez użytkownika
   1. Pola struktury
      * coordinatesFile – plik wejściowy z współrzędnymi
      * connectionsTableFile – plik wejściowy z tablicą połączeń
      * queriesFile – plik wejściowy z trasami do wyznaczenia
      * outputFile – plik wynikowy
3. Klasa FileManager – obsługuje wczytywanie i zapisywanie plików
   1. Metody klasy
      * Domyślny konstruktor i destruktor
      * loadCoordinates() – wczytuje plik z koordynatami
      * loadConnections() – wczytuje plik z tablicą połączeń
      * loadQueries() – wczytuje plik z trasami do wyznaczenia
      * writeOutput() – zapisuje wyniki działania programu
4. Klasa CityGraph – klasa reprezentująca graf z połączeniami miejsc w mieście
   1. Pola klasy
      * std::unordered\_map verticies\_ - zawiera nazwy miejsc w mieście i dane z nimi związane czyli: współrzędne oraz połączenia z innymi miejscami (Klasa VertexData)
   2. Metody klasy
      * Domyślny konstruktor i destruktor
      * size() – zwraca ilość wierzchołków grafu
      * empty() – sprawdza czy graf jest pusty
      * clear() – wymazuje wszystkie elementy grafu
      * data() – zwraca referencję do pola verticies\_
      * operator[] – zwraca referencję do współrzędnych danego miejsca
      * insertVertex() – tworzy nowy wierzchołek
      * removeVertex() – usuwa wierzchołek
      * addEdge() – dodaje połączenie między wierzchołkami
      * removeEdge() – usuwa połączenie między wierzchołkami
      * adjadent() – sprawdza czy dwa wierzchołki są ze sobą połączone lub zwraca sąsiadów danego wierzchołka (przeciążenie)
      * contains() – sprawdza czy graf zawiera dany wierzchołek
5. Struktura VertexData – zawiera dane powiązane z wierzchołkiem grafu
   1. Pola Struktury
      * Point val – współrzędne
      * std::vector edges – lista sąsiadujących wierzchołków (połączeń)
6. Struktura Point – zawiera współrzędne punktu na mapie
   1. Pola struktury
      * int x – współrzędna x
      * int y – współrzędna y
   2. Metody
      * Point() – konstruktor domyślny
      * Point(x, y) – konstruktor ustawiający współrzędne
      * ~Point() – destruktor domyślny
7. Klasa Path – klasa bazowa przechowująca znalezioną trasę
   1. Pola klasy
      * std::vector pathPoints – lista miejsc przez które prowadzi ścieżka
      * double distance – długość ścieżki
   2. Metody klasy
      * Domyślny konstruktor i destruktor wirtualny
      * virtual pathType() – zwraca informację czy jest to ścieżka wyznaczona dla pieszych czy dla samochodów
      * describe() – zwraca łatwy do zrozumienia opis ścieżki (wynikowy)
8. Klasa PedestrianPath - dziedziczy z klasy Path i odpowiednio implementuje jej wirtualne metody. Reprezentuje ścieżkę wyznaczoną dla pieszych.
9. Klasa CarPath - dziedziczy z klasy Path i odpowiednio implementuje jej wirtualne metody. Reprezentuje ścieżkę wyznaczoną dla samochodów.

