Politechnika Śląska

Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Infrastruktura Drogowa

Autor:	Agnieszka Góral
Prowadzący:	dr inż. Paweł Foszner
Rok akademicki:	2020/2021
Kierunek:	informatyka
Rodzaj studiów:	SSI
Semestr:	1
Termin laboratorium:	poniedziałek 10:30 – 12:45 wtorek 12:00 – 14:14
Sekcja:	22
Termin oddania	2020-11-08

1 . Treść zadania:

Napisz program, który zaproponuje optymalny (najtańszy) sposób połączenia miast siecią drogową. Zaproponowana sieć drogowa musi umożliwić przemieszczanie się pomiędzy dowolnie wybranymi miastami. Połączenia między miastami są zapisane w pliku w następujący sposób:

<miasto 1> <miasto2> <cena>

W wyniki działania programu zostanie utworzony plik zawierający zaproponowaną strukturę sieci w formacie:

<miasto 1> <miasto 2> <cena>

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

-i plik wejściowy
-o plik wyjściowy

2 . Analiza zadania:

Zadanie przedstawia problem połączenia sieci miast, które pobieramy z pliku tekstowego siecią drogową. Szukamy takiego połączenia miast, które będzie najtańsze. Musimy pamiętać, że każde miasto nie musi być połączone z każdym, niektóre mogą być połączone za pośrednictwem innego z miast.

2.1 Struktury danych:

W programie wykorzystana została stworzona przeze mnie struktura „węzeł”, która przechowuje:

- nazwę miasta węzłowego (zmienną typu string);
- mapę połączeń (listę par, która przechowuje zmienne typu string oraz double), składającą się z nazwy miasta docelowego i ceny połączenia miasta węzłowego i danego miasta;
- zmienną typu bool, która używana jest podczas algorytmu do sprawdzania czy dane miasto węzłowe zostało już odwiedzone i czy zostały już sprawdzone połączenia z tym miastem;
- zmienną typu double, która znajduje zastosowanie przy użyciu algorytmu Dijkstry.

- Parę, która składa się ze zmiennej typu string oraz double, która będzie wykorzystywana podczas algorytmu i będzie potrzebna do przechowania najtańszego połączenie z danego węzła wraz z jego nazwą.

Struktura ta ilustruje jedno miasto węzłowe (część grafu) oraz przechowuje wszystkie miasta, z którymi możemy je połączyć oraz ceny tych połączeń.

2.2 Algorytmy:

Program szuka najtańszych połączeń pomiędzy węzłami za pomocą zmodyfikowanego algorytmu Dijkstry. Najpierw algorytm wszystkim cenom połączeń przyporządkowuje wartkość maksymalną dla danego typu zmiennej (w tym przypadku double). Następnie szuka, czy istnieje jakieś tańsze połączenie do węzła (miasta). Tańsze połączenia są zapamiętywane przez węzeł. Później sprawdza, czy miasto zostało odwiedzone i sprawdza najtańsze połączenia z nim. Algorytm wykona się dla każdego węzła i w ten sposób znajdzie najtańsze połączenia.

3 . Specyfikacja zewnętrzna:

Program jest uruchamiany z linii poleceń, za pomocą następujących przełączników:

-i nazwa_pliku_wejściowego.txt
-o nazwa_pliku_wyjściowego.txt

Użytkownik musi pamiętać, że pliki muszą posiadać rozszerzenie „.txt”. Jeśli użytkownik poda błędne parametry lub program nie będzie w stanie otworzyć pliku to w konsoli pojawi się komunikat informujący o błędzie oraz dokładna instrukcja wprowadzania danych.

4 . Specyfikacja wewnętrzna:

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikacji z użytkownikiem) od logiki aplikacji.

4.1 Ogólna struktura programu

Program za pomocą funkcji *czytanie_z_wiersza_polecen* pobiera parametry, które zostały podane przez użytkownika i sprawdza ich poprawność. Gdy wprowadzone parametry są niepoprawne to wyświetlany jest komunikat o błędzie, wraz z instrukcją obsługi parametrów. Następnie, jeśli parametry zostały wczytane poprawnie, czyli jeśli funkcja *czytanie_z_wiersza_polecen* zwróciła wartość *true* to funkcja *zczytaj_z_pliku* pobiera dane z pliku, który użytkownik wskazał za pomocą parametrów i umieszcza je w strukturze (po zakończonej operacji zamyka plik), funkcja ta zwraca wektor węzłów (listę miast wraz z ich możliwymi połączeniami). Następnie lista miast jest przekazywana to funkcji *algorytm* (która wykorzystuje funkcje: *przygotuj_dane*, *relaksacja_krawedzi* oraz *znajdz_najblizsze_miasto*). W tej funkcji program sprawdza po kolei miasta i szuka najtańszych połączeń pomiędzy nimi. Jako ostania jest wywoływana fukcja *zwróć_do_pliku*, która zapisuje wynik działania

programu do pliku, a następnie go zamyka. Przed zakończeniem działania programu pamięć jest zwalniana aby zapobiec wyciekom pamięci.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 . Testowanie

Program został przetestowany na różnych plikach wejściowych. Włącznie z plikami, w których nie wszystkie dane są prawidłowe (część linijek jest pusta, w pliku znajdują się losowe znaki lub brakuje jakiejś danej). W takim przypadku program zlicza ilość linijek pustych oraz tych zapisanych w niepoprawny sposób, a następnie po odczytaniu pozostałych linijek (tych, w których dane są zapisane w sposób poprawny, jeśli takie istnieją) wyświetla komunikat o stanie pliku wejściowego np.:

„|| Liczba wszystkich linijek: 30 || Liczba pustych linijek: 5 || Liczba niepoprawnie sformatowanych linijek: 2 ||”

Podczas testowania został wyeliminowany przypadek, gdy w pliku wejściowym znajduje się graf rozłączny (np. w pliku znajdują się dwie grupy miast ale nie są one ze sobą w żaden sposób połączone) oraz dodano opcję wyświetlania komunikatu błędu, gdy użytkownik próbuje podać dwa razy ten sam parametr lub wprowadza niepoprawne parametry.

Program został przetestowany pod kątem wycieków pamięci. Pomimo dużych rozmiarów plików na jakich program był testowany, nie pojawił się komunikat o naruszeniu ochrony pamięci.

6 . Wnioski :

Zadane zadanie zostało zrealizowane w całości. Najtrudniejszym etapem zadania okazało się ułożenie algorytmu oraz stworzenie struktury, która miała by obrazować połączenia. Podczas tworzenia programu poszerzyłam swoją wiedzę z zakresu tworzenia struktur oraz funkcji, gdyż korzystałam w tym celu z wielu zewnętrznych źródeł.

Literatura:

- „C++ Przewodnik Dla Początkujących” Alex Allain
- „Podstawy Programowania w Języku C++” Józef Zieliński