Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Planowanie podróży autobusowych

autor	Filip Miera
prowadzący	dr inż. Artur Pasierbek
rok akademicki	2020/2021
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	wtorek, 14:30 – 16:45
sekcja	5
termin oddania sprawozdania	2020-11-09

1 Treść zadania

Napisać program wyznaczający wszystkie możliwe trasy dojazdu z jednego miasta do drugiego autobusem o wybranej przez użytkownikiem godzinie. Wyznaczone trasy zostaną zapisane do pliku wynikowego razem z ogległością jaką się przejedzie, ceną jaką bedzie trzeba zapłacić przewoźnikowi oraz godziną przyjazdu. Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

- `-t` plik wejściowe z opisem tras
- `-k` plik wyjściowy z kursami autobusów
- `-p` miasto początkowe
- `-d` miasto docelowe
- `-g` godzina odjazdu
- `-w` nazwa pliku wynikowego

2 Analiza zadania

Zagadnienie przedstawia problem przeszukiwania plików tekstowych w poszukiwaniu odpowiedniej trasy i godziny.

2.1 Struktury danych

W programie wykorzystano graf do znalezienia połączeń między poszczególnymi miastami. Graf ma wierzchołki które mogą mieć wielu sąsiadów z którymi są połączeni krawędziami. Może występować on jako graf skierowany który może się poruszać tylko w wyznaczonym kierunku krawędzi lub nieskierowany który nie ma tego ogranicznia.

2.2 Algorytmy

Program wybiera wszytskie możliwe trasy dojazdu z punktu A do punktu B najwcześniej jak się da od godziny wybranej przez użytkownika, oraz sumuje odległość, koszt przejazdu oraz pokazuje godzinę przewidywanego dojazdu na miejsce.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików wejściowych z trasami, rozkładem oraz wynikowego po odpo-

wiednich przełącznikach (odpowiednio: `-k` dla pliku z kursami autobusów, `-t` dla pliku z opisem tras oraz `-w` dla pliku z wyznaczonymi trasami), np.

```
-k rozklad.txt -t trasa.txt -w czy_doajde
-t trasa.txt -w czy_dojade -k rozklad.txt
-w czy_dojade.txt -k rozklad.txt -t trasa.txt
```

Pliki są plikami tekstowymi. Przełączniki mogą być podane w dowolnej kolejności. Uruchomienie programu bez żadnego parametru powoduje wyświetlenie krótkiej pomocy. Uruchomienie programu z nieprawidłową ilością parametrów powoduje wyświetlenie komunikatu:

```
Podaj własciwa ilość argumentów.
```

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym.

4.1 Ogólna struktura programu

W programie wywołane są funkcje wczytywanie_trasy oraz wczytywanie_godzin . Funkcje te sprawdzają, czy do programu zostały wczytane odpowiednie pliki. Gdy program nie został wywołany prawidłowo, zostaje wypisany stosowny komunikat i program się kończy. Program przegląda po kolei miasta tak jak są zapisane w pliku tekstowym zaznaczając każde miasto które sprawdził jako sprawdzone, gdy znajdzie miasto pasujące do początkowej trasy, trasy do przesiadki lub końcowej to zapisuje je, ich numer trasy, odległość od poprzedniego miasta oraz koszt przejazdu. Następnie przeszukuje wszytskie godziny z pliku z rozkładem które są przypisane do pasującego numeru trasy i wybiera najbliższą większą godzinę niż podał użytkownik. Jeżeli autobus nie kursuje już tego dnia to pasażer musi zaczekać na kolejny który wyruszy następnego dnia.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załaczniku.

5 Testowanie

Program został przetestowany na różnych rodzajach tras z różnymi mistami początkowymi oraz końcowymi, dla różnych godzin i danych w plikach tekstowych.

6 Wnioski

Napisanie tego programu wiązało się z trudnościami, podstawową było odpowiednie uporządkowanie danych z plików tekstowych w pamięci programu. Zastosowanie grafu pozwalało mi znaleźć wszyskie możliwe ścieżki pomiędy miastami.

Dodatek
Szczegółowy opis typów i funkcji

Zadanie programistyczne

Wygenerowano przez Doxygen 1.8.20

Rozdział 1

projekt_autobus

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy graf

Klasa przechowujaca metody i pola potrzebne do zapisu i edycji grafu.

`#include <funkcje.h>`

Metody publiczne

- graf (int n)

  Konstruktor z parametrem okreslajacym maksymalna wielkosc grafu.
- void dodaj_sciezke (int v, int u)

  Funkcja dodajaca sciezki.
- void dodaj_cene_odleglosc_nr_trasy (int v, int u, int nr_trasy, float cena, float odleglosc)

  Funkcja, ktora zapisuje w polach klasy pozadane parametry.
- void znajdz_sciezki (int s, int d)

  Fukcja znajdujaca wszyskie mozliwe sciezki.

Atrybuty publiczne

- std::vector< std::vector< int > > znalezione

  Wektor z znalezionymi polaczeniami.
- std::vector< std::vector< int > > nr_trasy

  Wektor przechowywujacy numery tras.
- std::vector< std::vector< float > > cena

  Wektor przechowywujacy ceny przejazdu.
- std::vector< std::vector< float > > odleglosc

  Wektor przechowywujacy odleglosci pomiedzy miastami.

4.1.1 Opis szczegółowy

Klasa przechowujaca metody i pola potrzebne do zapisu i edycji grafu.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 graf()

```
graf::graf (
            int n )
```

Konstruktor z parametrem okreslajacym maksymalna wielkosc grafu.

Parametry

n	Maksymalna wielkosc zapamietywanych sciezek.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 dodaj_cene_odleglosc_nr_trasy()

```
void graf::dodaj_cene_odleglosc_nr_trasy (
            int v,
            int u,
            int nr_trasy,
            float cena,
            float odleglosc )
```

Funkcja, ktora zapisuje w polach klasy pozadane parametry.

Parametry

v	Poczatek sciezki.
u	Koniec sciezki.
nr_trasy	Numer trasy danej sciezki.
cena	Cena danej sciezki.
odleglosc	Odleglosc danej sciezki.

4.1.3.2 dodaj_sciezke()

```
void graf::dodaj_sciezke (
            int v,
            int u )
```

Funkcja dodajaca sciezki.

Parametry

v	Poczatek sciezki.
u	Koniec sciezki.

4.1.3.3 znajdz_sciezki()

```
void graf::znajdz_sciezki (
            int s,
            int d )
```

Fukcja znajdujaca wszyskie mozliwe sciezki.

Parametry

s	Poczatkowy wezel.
d	Koncowy wezel.

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/projekty/projekt_autobus/wazne/funkcje.h
- D:/projekty/projekt_autobus/wazne/funkcje.cpp

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku D:/projekty/projekt_autobus/wazne/funkcje.h

```
#include <vector>
#include <list>
#include <string>
```

Komponenty

- class graf

  Klasa przechowujaca metody i pola potrzebne do zapisu i edycji grafu.

Funkcje

- std::vector< std::vector< std::vector< std::string > > > wczytywanie_trasy (std::string filename)

  Funkcja, ktora otwiera plik z trasami i zapisuje je do wektora trasy, jezeli operacja sie nie powiedzie to wypisze stosowna informacje.
- std::vector< std::vector< std::string > > odczyt_trasy (std::string line)

  Funkcja, porzadkuje wczytana linijke pliku tekstowego i zapisuje ja do wektora.
- std::vector< std::vector< float > > wczytywanie_godzin (std::string filename)

  Funkcja, ktora otwiera plik z rozkladem i zapisuje je do wektora rozklad, jezeli operacja sie nie powiedzie to wypisze stosowna informacje.
- std::vector< float > odczyt_godzin (std::string line)

  Funkcja, ktora usuwa numer porzadkowy i zapisuje godziny w wektorze.
- float godzina_na_float (std::string h)

  Funkcja, ktora usuwa dwukropki w rozkladzie i zamienia go na wektor float'owy.
- std::string godzina_na_string (float godzina)

  Funkcja zamieniajaca wartosci float'owe godzina z powrotem na string.
- std::string porownanie_godzin (float godz_uzytkow, std::vector< std::vector< float >> rozklad, float numer↩ _trasy)

  Funkcja porownuje godzine ktora wprowadzil uzytkownik z godzinami z rokzladu i wybiera najblizsza mozliwa godzine odjazdu.
- std::string przybycie (float odleglosc, std::string godzina)

  Funkcja oblicza czas przyjazdu na podstawie odleglosci.
- std::string float_to_string_with_prec (float liczba, int precision)

  Funkcja ktora zamienia liczbe typu float na string z odpowiednia precyzja.
- void wypisanie_koncowych_tras (std::vector< std::vector< std::vector< std::string >>> trasy, std::vector< std::vector< float >> rozklad, std::string plik_wynikowy, std::string poczatek, std::string koniec, float godzina_wyjazdu)

  Funkcja wypisuje do konsoli oraz pliku znalezione polaczenia.

5.1.1 Dokumentacja funkcji

5.1.1.1 float_to_string_with_prec()

```
std::string float_to_string_with_prec (
            float liczba,
            int precision )
```

Funkcja ktora zamienia liczbe typu float na string z odpowiednia precyzja.

Parametry

liczba	Liczba ktora funkcja ma zaokraglic.
precision	Liczba miejsc po przecinku zaokraglenia.

Zwraca

Zwraca string z okreslona precyzja.

5.1.1.2 godzina_na_float()

```
float godzina_na_float (
            std::string h )
```

Funkcja, ktora usuwa dwukropki w rozkladzie i zamienia go na wektor float'owy.

Parametry

h	Godzina odjazdu lub godzina uzytkownika w potaci string(hh:mm).

Zwraca

Zwraca godzine zamieniona na float.

5.1.1.3 godzina_na_string()

```
std::string godzina_na_string (
            float godzina )
```

Funkcja zamieniajaca wartosci float'owe godzina z powrotem na string.

Parametry

godzina	Godzina jako wartosc float np.(10:12 => 10.2).

Zwraca

Zwraca godzine zamieniona na string.

5.1.1.4 odczyt_godzin()

```
std::vector<float> odczyt_godzin (
            std::string line )
```

Funkcja, ktora usuwa numer porzadkowy i zapisuje godziny w wektorze.

Parametry

line	Pojedyncza linijka odczytana z pliku.

Zwraca

Zwraca wektor w ktorym znajduja sie godziny odjazdu oraz numer trasy.

5.1.1.5 odczyt_trasy()

```
std::vector<std::vector<std::string> > odczyt_trasy (
            std::string line )
```

Funkcja, porzadkuje wczytana linijke pliku tekstowego i zapisuje ja do wektora.

Parametry

line	Pojedyncza linijka odczytana z pliku.

Zwraca

Zwraca 2D wektor zawierajacy uporzadkowane parametry danej trasy.

5.1.1.6 porownanie_godzin()

```
std::string porownanie_godzin (
            float godz_uzytkow,
```

```
            std::vector< std::vector< float >> rozklad,
            float numer_trasy )
```

Funkcja porownuje godzine ktora wprowadzil uzytkownik z godzinami z rokzladu i wybiera najblizsza mozliwa godzine odjazdu.

Parametry

godzina_uzytkow	Godzina od ktorej program zacznie szukac mozliwych polaczen.
rozklad	Rozklad autobusowy w postaci float.
numer_trasy	Numer trasy kursu autobusu.

Zwraca

Zwraca godzine odjazdu autobusu.

5.1.1.7 przybycie()

```
std::string przybycie (
            float odleglosc,
            std::string godzina )
```

Funkcja oblicza czas przyjazdu na podstawie odleglosci.

Parametry

odleglosc	Odleglosc pomiedzy miastem poczatkowym a docelowym.
godzina	Godzina wyjazdu autobusu.

Zwraca

Zwraca godzine dojazdu.

5.1.1.8 wczytywanie_godzin()

```
std::vector<std::vector<float> > wczytywanie_godzin (
            std::string filename )
```

Funkcja, ktora otwiera plik z rozkladem i zapisuje je do wektora rozklad, jezeli operacja sie nie powiedzie to wypisze stosowna informacje.

Parametry

filename	Nazwa pliku z rozkladem.

Zwraca

Zwraca rozklad autobusowy zapisany jako odpowiedni wektor.

5.1.1.9 wczytywanie_trasy()

```
std::vector<std::vector<std::vector<std::string> > > wczytywanie_trasy (
            std::string filename )
```

Funkcja, ktora otwiera plik z trasami i zapisuje je do wektora trasy, jezeli operacja sie nie powiedzie to wypisze stosowna informacje.

Parametry

filename	Nazwa pliku z trasami.

Zwraca

Zwraca trasy, odleglosc miedzy miastami oraz koszt przejazdu zapisane jako odpowiedni wektor.

5.1.1.10 wypisanie_koncowych_tras()

```
void wypisanie_koncowych_tras (
            std::vector< std::vector< std::vector< std::string >>> trasy,
            std::vector< std::vector< float >> rozklad,
            std::string plik_wynikowy,
            std::string poczatek,
            std::string koniec,
            float godzina_wyjazdu )
```

Funkcja wypisuje do konsoli oraz pliku znalezione polaczenia.

Parametry

trasy	Wektor przechowujacy dane tras.
rozklad	Wektor przechowujacy dane rozkladu.
plik_wynikowy	Nazwa pliku wynikowego.
poczatek	Nazwa miasta poczatkowego wybranego przez uzytkownika.
koniec	Nazwa miasta koncowego wybranego przez uzytkownika.
godzina_wyjazdu	Godzina wyjazdu wybrana przez uzytkownika.

Indeks