Politechnika Śląska w Gliwicach
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Budowa infrastruktury drogowej

Autor	Zbigniew Witkowski
Prowadzący	Dr hab. inż. Wojciech Sułek
Rok akademicki	2022/2023
Kierunek	Teleinformatyka
Rodzaj studiów	SSI
Semestr	1
Termin laboratorium	Środa 8:15-10:45
Grupa	1
Sekcja	1
Data oddania sprawozdania	2023-02-18

1. Treść zadania

Napisać program który zaproponuje optymalny (najtańszy) sposób połączenia miast siecią drogową. Zaproponowana sieć drogowa musi umożliwić przemieszczanie się pomiędzy dowolnie wybranymi miastami. Następnie najtańszy sposób połączania miast zostanie zapisany do pliku.

-i plik wejściowy
-o plik wyjściowy

2. Analiza zadania

Zagadnienie przedstawia problem optymalizowania połączeń między miastami zapisanych w pliku wejściowym.

2.1 **Struktury danych**

W programie są wykorzystywane następujące struktury danych:

- wektor do przechowywania dróg pomiędzy miastami,
- mapa do przechowywania relacji między miastami,
- para do przechowywania kosztu drogi oraz nazw miast z tą droga związanych,
- tuple do przechowywania 2 typów zmiennych,
- string do przechowywania nazw miast.

2.2 **Algorytm**

Program wykorzystuje algorytm Kruskala. Algorytm ten łączy wszystkie krawędzi (drogi) w taki sposób, aby suma ich wag była jak najmniejsza. Krawędzie są dodawane do momentu kiedy wszystkie wierzchołki (miasta) są ze sobą połączone.

3. Specyfikacja zewnętrzna programu:

- Program pobiera dane z pliku wejściowego zawierającego informacje o kosztach połączeń między miastami.
- Program zapisuje wynik do pliku wyjściowego.
- Program akceptuje opcjonalne argumenty w postaci flagi "-i nazwaplikuwejsciowego" dla nazwy pliku wejściowego oraz "-o nazwaplikuwyjsciowego" dla nazwy pliku wyjściowego.
- Jeśli nie zostanie podany plik wejściowy, program wyświetla "Brak pliku wejściowego" i kończy działanie.

- Program wyświetla połączenia między miastami oraz ich koszty na ekranie.

4.Specyfikacja wewnętrzna

Program służy do przetwarzania danych wejściowych w postaci połączeń między miastami i ich kosztów, a następnie zapisania tylko unikalnych połączeń o najmniejszym koszcie między miastami do pliku wyjściowego.

4.1Szczegółowy opis kodu

Program szuka w folderze Projekt PPK pliku wejściowego i nazwie Plikwejściowy.csv.

```
double pobierajkoszt(Droga droga) {                 //Pobiera liczbe z linijki
    return droga.first;
}
string pobierajMiasto1(Droga droga) {
//pobiera miasto z pozycji 1
    return droga.second.first;
}

string pobierajMiasto2(Droga droga) {
//pobiera miasto z pozycji 2
    return droga.second.second;
}


string zwrockorzen(string miasto) {

    if (Miasta.count(miasto)) {
        return Miasta[miasto] = zwrockorzen(Miasta[miasto]);
    }
  else return miasto;
}
//zwraca miasto łączące 2 miasta
```

Powyższy fragment kodu jest odpowiedzialny za pobieranie z Plikuwejśćowego.csv nazw miast i kosztów związanych z podróżą między nimi.

```
int main(int argn, char* args[]) {
    vector<Droga> Drogi;
    bool jestplikwejsciowy = false;
    bool jestplikwyjsciowy = false;
    char* Plikwejsciowy = NULL;
    char* Plikwyjsciowy = NULL;
    for (int i = 1; i < argn - 1; i++) {
        int len = strlen(args[i]);
        if (len >= 2 && args[i][0] == '-') {
//przypisuje pliki wejściowy i wyjściowy do wartości char
            if (args[i][1] == 'i') {
                Plikwejsciowy = args[i + 1];
                jestplikwejsciowy = true;
            }
            if (args[i][1] == 'o') {
                Plikwyjsciowy = args[i + 1];
```

```
                jestplikwyjsciowy = true;
            }
        }
    }
```

Ten kod weryfikuje argumenty wywołania programu (argn, args[]). Jeśli argument zaczyna się od "-" i następny argument jest dostępny, to przypisuje plik wejściowy lub plik wyjściowy do zmiennej Plikwejsciowy lub Plikwyjsciowy, odpowiednio. Dodatkowo, ustawia zmienne jestplikwejsciowy lub jestplikwyjsciowy na wartość false, aby potem można było sprawdzić, czy plik wejściowy lub wyjściowy został przypisany.

```
if (!jestplikwejsciowy) {
        cout << "Brak pliku wejsciowego" << endl;
        return 0;
    }
```

Ten fragment sprawdza czy zastała przypisana wartość do jestplikwejsciowy, jeżeli nie wyświetla komunikat Brak pliku wejściowego i kończy działanie programu.

```
ifstream fi(Plikwejsciowy);
    cout << "Dane wejsciowe:" << endl;
    while (true) {
        Droga road;
        double Koszt;                                          //czyta linijke polinijce
i zapisuje nazwy miast jako startmiasto i endmiasto
        string startmiasto = "";
        string endmiasto = "";
        fi >> startmiasto >> endmiasto >> Koszt;
        if (startmiasto == "" || endmiasto == "") {
            break;
        }
        cout << startmiasto << " " << endmiasto << " " << Koszt << endl;
        Drogi.push_back(make_pair(Koszt, make_pair(startmiasto, endmiasto)));
    }
```

Następnie jest otwierany plik wejściowy (Plikwejsciowy) i odczytuje zawartość pliku linia po linii. Dla każdej linii, odczytuje pierwsze i drugie słowo jako nazwy miast początkowego i końcowego, a następnie odczytuje trzecie słowo jako koszt drogi między tymi miastami. Te wartości są zapisywane w strukturze Droga i dodawane do wektora Drogi. Na końcu, dane wejściowe są wypisywane na ekranie.

```
  ofstream fo;
    if (jestplikwyjsciowy) {
        fo.open(Plikwyjsciowy);
    }
```

Zostaje utworzona funkcja do zapisywania w pliku, jeżeli istnieje plik wyjściowy to zostaje on otwarty.

```
sort(Drogi.begin(), Drogi.end());
```

Drogi zostają posortowane według kosztów od najmniejszego do największego.

```
string PierwszeMiasto = pobierajMiasto1(Drogi[i]);
        string DrugieMiasto = pobierajMiasto2(Drogi[i]);
        double Koszt = pobierajkoszt(Drogi[i]);
        string PierwszeMiastoRoot = zwrockorzen(PierwszeMiasto);
        string DrugieMiastoRoot = zwrockorzen(DrugieMiasto);
```

Następnie pobiera wartości startowego miasta, końcowego miasta i kosztu z wektora Drogi, po czym oblicza korzeń startowego i końcowego miasta za pomocą funkcji zwrockorzen. Wyniki są przypisywane do zmiennych PierwszeMiastoRoot i DrugieMiastoRoot.

```
  if (PierwszeMiastoRoot != DrugieMiastoRoot) {
            cout << PierwszeMiasto << " " << DrugieMiasto << " " << Koszt << endl;
//wyświetla połączenia i ich koszty
            if (jestplikwyjsciowy) {
                fo << PierwszeMiasto << " " << DrugieMiasto << " " << Koszt << endl;
//zapisuje po pliku połączenia i ich koszty
            }
            Miasta[PierwszeMiastoRoot] = DrugieMiastoRoot;
        }
    }
    fo.close();
    fi.close();
    return 0;
```

W ostatniej części kodu wszystkie drogi optymalne pomiędzy miastami zostają zapisane do plikuwyjściowego.csv (fo) oraz wyświetlone na ekran konsoli. Ostatnie 3 linijki odpowiadają za zamknięcie pliku wejściowego i wyjściowego po czym program kończy swoje działanie.

5.Testowanie programu

Program został testowany pod kątem wczytywania danych w innej kolejności. Taka operacja powoduje błędne wpisanie i wyświetlenie tej konkretnej linii przy wypisywaniu pliku wyjściowego. Linijki w pliku wejściowym powinny wyglądać tak jak w treści zadania, żeby program działał poprawnie.
Kiedy plik wejściowy jest pusty program wyświetla komunikaty takie same jak przy pełnym pliku wejściowym(Plik wejściowy:, Plik wyjściowy:). Do pliku wyjściowego nic nie zostaje zapisane (pozostaje pusty). Maksymalny koszt drogi jest ograniczany przez typ double.

6.Podsumowanie

Program opiera się na algorytmie Kruskala. Program wykonuje podstawowe działania związane z obliczaniem optymalnej (najtańszej) drogi pomiędzy miastami. Nie jest on przygotowany na zmianę struktury pliku wejściowego. Skuteczność programu jest uwarunkowana poprawną strukturą danych w pliku wejściowym.