|  |  |  |  |
| --- | --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | | |
| «Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра теоретической и прикладной информатики | | | |
|  | | | |
| Курсовая работа | | | |
| по дисциплине «Структуры данных и алгоритмы» | | | |
|  | | | |
|  | | | |
|  | | | |
|  | Факультет: | ПМИ | |
| Группа: | ПМИ-92 | |
| Студент: | Львов Денис |
| Преподаватель: | Еланцева Ирина Леонидовна | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  | | | |
| Новосибирск 2020 | | | |
|  | | | |

1. **Условие задачи:**

В системе двусторонних дорог за проезд каждой дороги взимается некоторая пошлина. Найти путь из города A в город B с минимальной величиной S+P, где S – сумма длин дорог пути, а P - сумма пошлин проезжаемых дорог.

1. **Анализ задачи:**

Дано: файл map.txt, который содержит в себе карту городов (описание графа). Первая строка – названия всех городов (количество не должно превышать двадцати, название города – один символ) среди которых должны быть города A и B. В конце первой строки обязательно должна присутствовать точка. Остальные строки задают дороги между введёнными городами (рёбра графа) и имеют вид: название первого города, название второго города, стоимость проезда, длина дороги.

Результат: вывод на экран названий городов оптимального маршрута (пути из города A в город B с минимальной величиной S+P, где S – сумма длин дорог пути, а P - сумма пошлин проезжаемых дорог.), затрат в пути и общей продолжительность дорог. Или вывод на экран сообщения о наличии ошибки.

Метод решения: в соответствии с первой строкой файла создадим граф (неориентированный, взвешенный, циклический псевдограф), состоящий только из вершин (пока без рёбер). Потом добавим в граф рёбра на основании информации из следующих строк файла. В полученном графе найдём города A и B и путём перебора всех маршрутов (исключая заранее невыгодные) найдём самый оптимальный маршрут. Выведем полученный ответ на экран.

1. **Структура данных:**

сity – вершина графа: struct city{char name=NULL; city \*cities[MAX\_N];

int cost[MAX\_N]; int length[MAX\_N]; int countOut = 0;};

name – название города .

cities – массив указателей на города к которым есть дорога.

cost – массив цен за проезд по соответствующим дорогам.

length – массив длин соответствующих дорог.

countOut – количество дорог выходящих из города.

Константы:

MAX\_N – максимально допустимое количество городов

MAX\_ROADS - максимально допустимое количество дорог между городами (подсчитано с помощью формулы для нахождения суммы элементов алгебраической прогрессии )

Глобальные переменные:

way – массив указателей на вершины графа. Используется при поиске оптимального маршрута.

minWay – массив указателей на вершины графа, в котором после выполнения подпрограммы поиска оптимального пути будут храниться вершины этого пути.

min – переменная для хранения минимального значения выражения S+P, где S – сумма длин дорог пути, а P - сумма пошлин проезжаемых дорог.

minC – переменная для хранения суммы пошлин дорог оптимального маршрута.

minL – переменная для хранения суммы длин дорог оптимального маршрута.

sumC - переменная для хранения суммы длин дорог. Используется при поиске оптимального маршрута.

sumL - переменная для хранения суммы пошлин дорог. Используется при поиске оптимального маршрута.

Подпрограмма search:

A, B – указатели на города между которыми нужно найти оптимальный маршрут.

check – переменная логического типа данных для проверки существования дорог в массиве выходных дорог.

Подпрограмма main:

graph – файл с картой городов и дорог между ними.

countCities – количество городов на карте.

cost – переменная для считывания пошлины за проезд из файла.

length – переменная для считывания длины дороги файла.

aIn, bIn – переменная для считывания городов между которыми задаётся дорога.

check – переменная логического типа данных для предотвращения ошибок.

A, B – указатели для проведения рёбер между вершинами графа.

cities – массив указателей на все вершины графа.

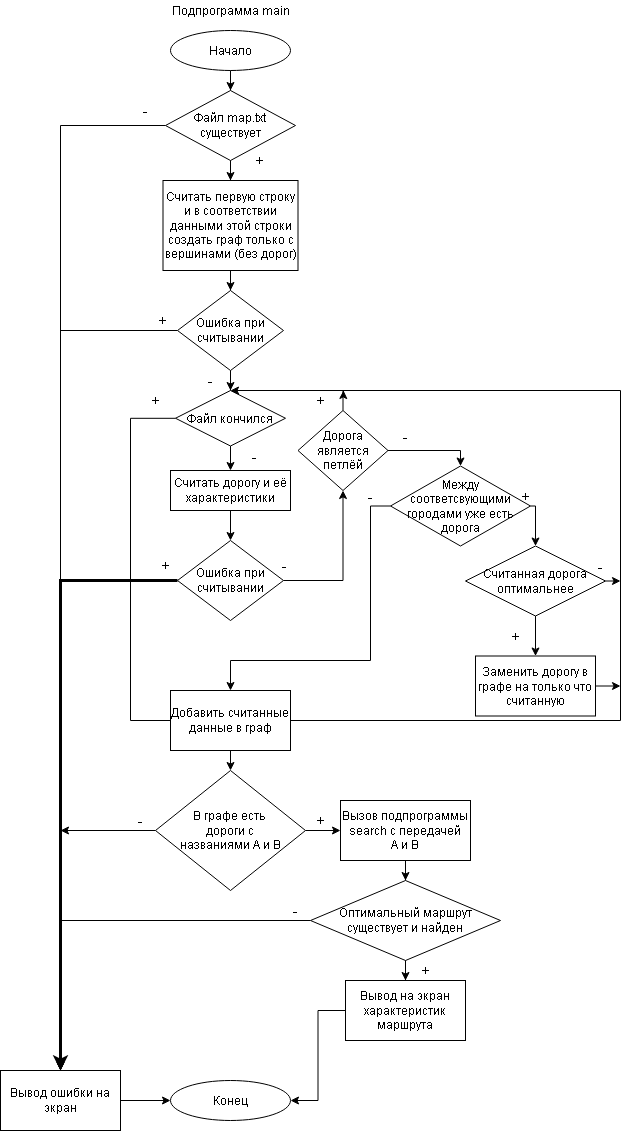
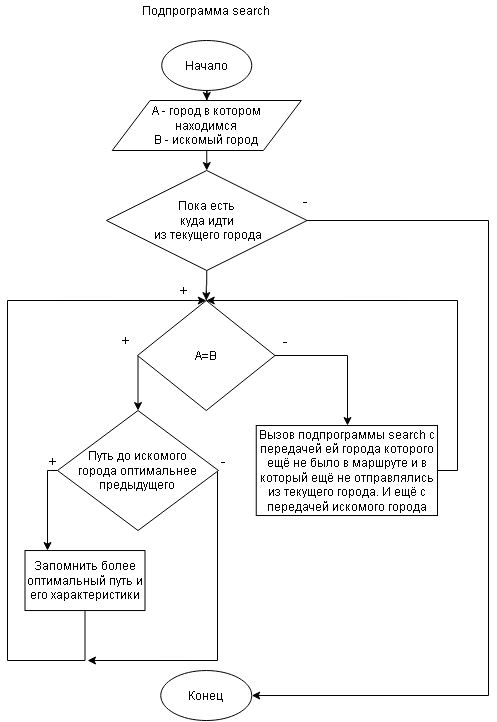
numA, numB – переменные целочисленного типа данных для нахождения самой оптимальной дороги среди кратных.

1. **Структура программы:**

Для поиска оптимального маршрута между двумя городами была задействована рекурсивная процедура search.

Аргументы: A, B – указатели на города, между которыми надо найти оптимальный маршрут.

**5. Алгоритм:**



1. **Код программы:**

#include<iostream>

#include<string>

#include<fstream>

#include<locale.h>

#define MAX\_N 20

#define MAX\_ROADS 210

#define INPUT "map.txt"

using namespace std;

struct city

{

char name=NULL;

city \*cities[MAX\_N];

int cost[MAX\_N];

int length[MAX\_N];

int countOut = 0;

};

city\* way[MAX\_ROADS];

city\* minWay[MAX\_ROADS];

int min = -1,minC,minL;

int wayCount = 0;

int minWayCount = 0;

int sumC = 0;

int sumL = 0;

void search(city\* A, city\* B)

{

int i, j;

bool check;

way[wayCount] = A;

wayCount++;

for (i = 0; i < A->countOut; i++)

{

check = true;

for (j = 0; j < wayCount; j++) if (A->cities[i] == way[j]) check = false;

if (A == B)

{

if ((min == (-1)) || ((sumC + sumL) < min))

{

for (int k = 0; k < MAX\_N; k++)

{

minWay[k] = way[k];

min = (sumC + sumL);

minWayCount = wayCount;

minC = sumC;

minL = sumL;

}

}

check = false;

}

if (check)

{

sumC = sumC + A->cost[i];

sumL = sumL+A->length[i];

if (((sumC+sumL)<min)||(min==(-1))) search(A->cities[i], B);

sumC = sumC - A->cost[i];

sumL = sumL - A->length[i];

}

if (wayCount>=0) way[wayCount] = NULL;

}

wayCount--;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int i,countCities=0,cost,length,numA,numB;

char aIn = NULL, bIn = NULL;

bool check;

city \*cities[MAX\_N],\*A,\*B;

fstream graph(INPUT);

for (i = 0; i < MAX\_N; i++) cities[i] = NULL;

if (graph)

{

graph >> aIn;

check = false;

while (aIn != '.')

{

if ((graph.eof())|| (countCities == MAX\_N))

{

check = true;

break;

}

for (i = 0; i < MAX\_N; i++)

{

if (cities[i] == NULL)

{

cities[i] = new city;

cities[i]->name = aIn;

countCities++;

break;

}

else

if (aIn == cities[i]->name)

{

cout << "Город " << aIn << " уже есть." << endl;

break;

}

}

graph >> aIn;

}

if (check)

{

cout << "Ошибка во входном файле." << endl;

}

else

{

while (!graph.eof())

{

aIn = bIn = NULL;

cost = length = -1;

graph >> aIn >> bIn;

graph >> cost >> length;

if ((aIn == NULL) || (bIn == NULL) || (cost < 0) || (length < 0))

{

cout << "Ошибка во входном файле." << endl;

break;

}

A = B = NULL;

for (i = 0; i < countCities; i++)

{

if (aIn == cities[i]->name) A = cities[i];

if (bIn == cities[i]->name) B = cities[i];

}

if ((A != NULL) && (B != NULL))

{

if (A != B)

{

check = false;

for (i = 0; i < A->countOut; i++)

if (A->cities[i] == B)

{

if ((A->cost[i] + A->length[i]) >= (cost + length))

{

check = true;

numA = i;

for (int j = 0; j < A->countOut; j++)

if (B->cities[j] == A) numB = j;

}

}

if (!check)

{

A->cities[A->countOut] = B;

B->cities[B->countOut] = A;

A->cost[A->countOut] = B->cost[B->countOut] = cost;

A->length[A->countOut] = B->length[B->countOut] = length;

A->countOut++;

B->countOut++;

}

else

{

A->cities[numA] = B;

B->cities[numB] = A;

A->cost[numA] = B->cost[numB] = cost;

A->length[numA] = B->length[numB] = length;

}

}

}

else

{

cout << "Ошибка во входном файле.1" << endl;

}

}

graph.close();

}

A = B = NULL;

for (i = 0; i < countCities; i++)

{

if ('A' == cities[i]->name) A = cities[i];

else if ('B' == cities[i]->name) B = cities[i];

}

if ((A != NULL) && (B != NULL))

{

search(A, B);

if (min != (-1))

{

cout << "Самый оптимальный путь: ";

for (i = 0; i < minWayCount; i++) cout << minWay[i]->name << " ";

cout << " Длина пути: " << minL << ". Потрачено денег на дорогу: " << minC << "." << endl;

}

else

{

cout << "Между городами A и B нет пути.";

}

}

else

{

cout << "Нет города A или B.";

}

}

else

{

cout << "Входной файл отсутствует.";

}

return 0;

}

1. **Тесты:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Визуальное представление графа  (Синим цветом обозначен оптимальный путь) | Входные данные | Вывод на экран | Примечание |
| 1 |  | A B C D F.  A B 1000 1000  A C 110 15  A D 30 90  B F 30 60  C F 100 50  D F 10 10  B C 70 90  C D 50 70 | Самый оптимальный путь:  A D F B Длина пути: 160. Потрачено денег на дорогу: 70. | Корректно составленный граф |
| 2 |  | A C.  A C 10 10 | Нет города A или B. | Нет города B |
| 3 |  | B C.  B C 10 10 | Нет города A или B. | Нет города А |
| 4 |  | A B C.  B C 10 10 | Между городами A и B нет пути. | Между городами A и B нет пути |
| 5 |  | A B C.  B C 10 10  A C | Ошибка во входном файле.  Между городами A и B нет пути. | Данные в файле введены некорректно |
| Номер теста | Визуальное представление графа  (Синим цветом обозначен оптимальный путь) | Входные данные | Вывод на экран | Примечание |
| 6 |  |  | Ошибка во входном файле.  Нет города A или B. | Пустой файл |
| 7 |  | “отсутствие входного файла” | Входной файл отсутствует. | Отсутствие входного файла |
| 8 |  | A B C D F.  A B 1000 1000  A C 110 15  A D 30 100  B F 30 60  A D 30 90  C F 100 50  D F 10 10  B C 70 90  C D 50 70  A A 10 10 | Самый оптимальный путь:  A D F B Длина пути: 160. Потрачено денег на дорогу: 70. | В графе присутствуют петли и кратные рёбра |

1. **Результат работы программы:**

На всех тестах программа выдала ожидаемый результат. Программа выполнена правильно.

1. **Литература:**

|  |  |
| --- | --- |
| 1. Хиценко В. П., Шапошникова Т. А. Структуры данных и алгоритмы: методические указания к курсовой работе для 1 курса ФПМиИ (направление 010500 - Прикладная математика и информатика, специальность 010503 - Математическое обеспечение и администрирование информационных систем) дневного отделения: методическое пособие. – Новосибирск: НГТУ, 2008. |  |