Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

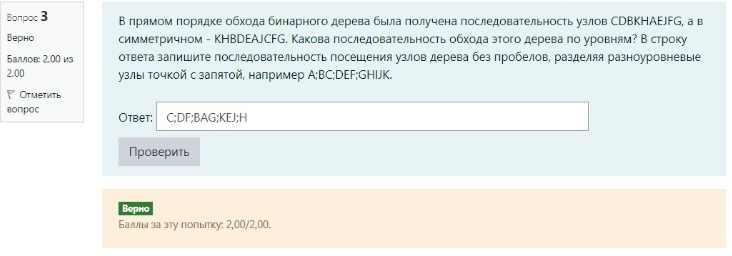
Кафедра О7 «Информационные системы и программная инженерия»

**Практическая работа №2**по дисциплине «Структуры и организация данных»  
на тему «Нелинейные структуры данных»  
  
вариант 25

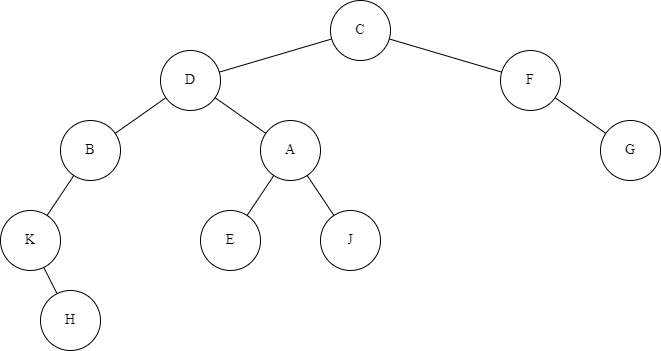
Выполнил:  
Студент Фокин В. К.  
Группа И924Б  
  
Преподаватель:  
Гладевич А. А.

Санкт-Петербург  
2023 г.

**Задание 1.**

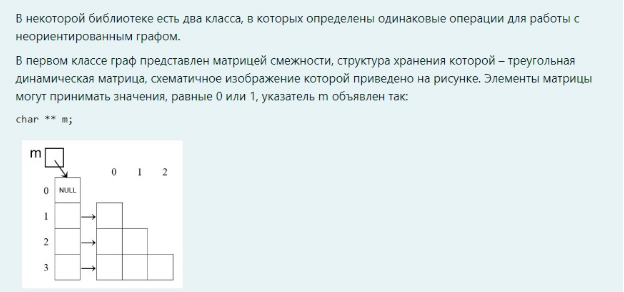
**

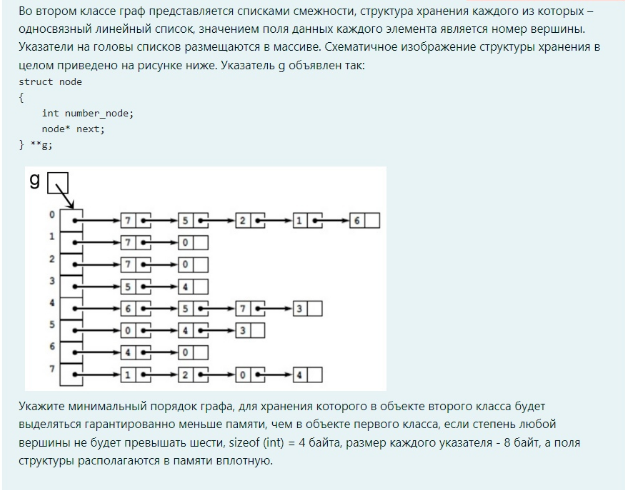
Полученное дерево:



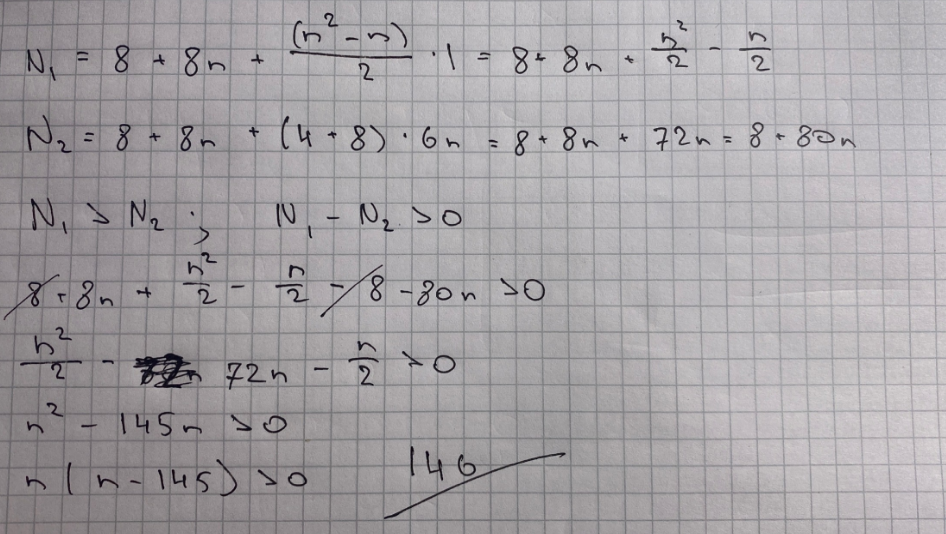
Ответ: C;DF;BAG;KEJ;H

**Задание 2.**

**



Решение задачи:

**

Ответ:146

**Задание 3.** Написать программудля решения задачи.

Уровень сложности – **Низкий.**

Задачу достаточно решить для одного фиксированного набора данных. Граф представляется матрицей смежности (или матрицей весов), структура хранения – статический двумерный массив.

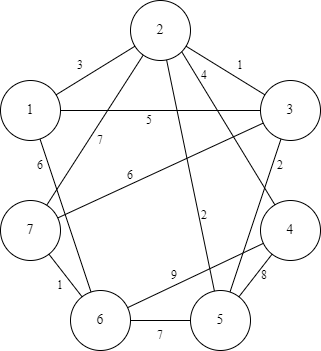
Вариант – 25.

Заданы две системы двусторонних дорог с одним и тем же множеством городов (железные и шоссейные дороги). Найдите минимальный по длине путь из города А в город В (который может проходить как по железным, так и по шоссейным дорогам), и места пересадок с одного вида транспорта на другой на этом пути. А и В вводятся с клавиатуры.

Тестовый граф:

Система шоссейных дорог

Граф:



Матрица:

0 3 5 0 0 6 0

3 0 1 4 2 0 7

5 1 0 0 2 0 6

0 4 0 0 8 9 0

0 2 2 8 0 7 0

6 0 0 9 7 0 1

0 7 6 0 0 1 0

Список смежности:

1: (2,3), (3,5), (6,6)

2: (1,3), (3,1), (4,4), (5,2), (7,7)

3: (1,5), (2,1), (5,2), (7,6)

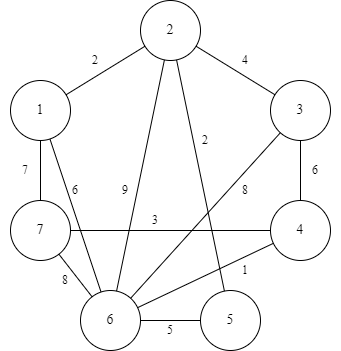
4: (2,4), (5,8), (6,9)

5: (2,2), (3,2), (4,8), (6,7)

6: (1,6), (4,9), (5,7), (7,1)

7: (2,7), (3,6), (6,1)

Система железных дорог:



Матрица:

0 2 0 0 0 6 7

2 0 4 0 2 9 0

0 4 0 6 0 8 0

0 0 6 0 0 1 3

0 2 0 0 0 5 0

6 9 8 1 5 0 8

7 0 0 3 0 8 0

Список смежности:

1: (2,2), (6,6), (7,7)

2: (1,2), (3,4), (5,2), (6,9)

3: (2,4), (4,6), (6,8)

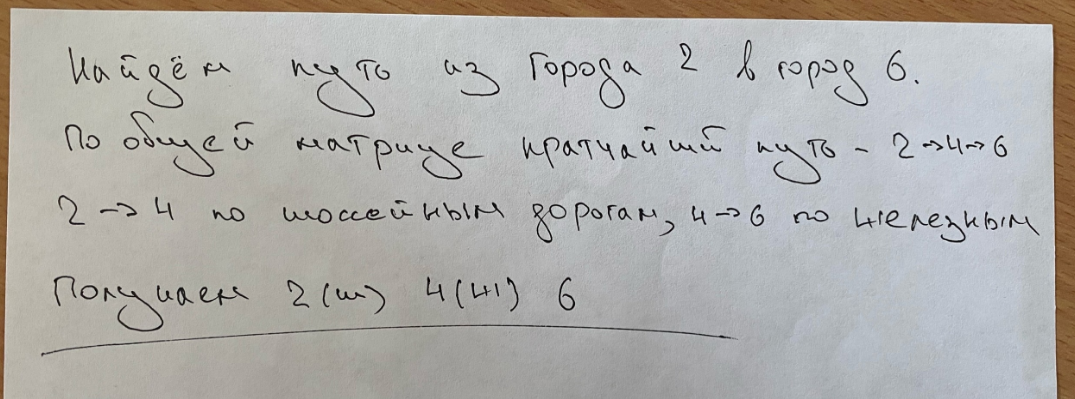
4: (3,6), (6,1), (7,3)

5: (2,2), (6,5)

6: (1,6), (2,9), (3,8), (4,1), (5,5), (7,8)

7: (1,7), (4,3), (6,8)

Ожидаемый результат работы программы:



Структура хранения для представления графа матрицей смежности или весов:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 3 | 5 | 0 | 0 | 6 | 0 |
| 3 | 0 | 1 | 4 | 2 | 0 | 7 |
| 5 | 1 | 0 | 0 | 2 | 0 | 6 |
| 0 | 4 | 0 | 0 | 8 | 9 | 0 |
| 0 | 2 | 2 | 8 | 0 | 7 | 0 |
| 6 | 0 | 0 | 9 | 7 | 0 | 1 |
| 0 | 7 | 6 | 0 | 0 | 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 2 | 0 | 0 | 0 | 6 | 7 |
| 2 | 0 | 4 | 0 | 2 | 9 | 0 |
| 0 | 4 | 0 | 6 | 0 | 8 | 0 |
| 0 | 0 | 6 | 0 | 0 | 1 | 3 |
| 0 | 2 | 0 | 0 | 0 | 5 | 0 |
| 6 | 9 | 8 | 1 | 5 | 0 | 8 |
| 7 | 0 | 0 | 3 | 0 | 8 | 0 |

Применяемые алгоритмы:

Для поиска кратчайшего пути в матрице смежности используется алгоритм Дейкстры.

Текст программы:

#include <iostream>

#include <fstream>

#include <windows.h>

using namespace std;

const int MAX = 100;

int minDistance(int dist[], bool visit[], int n) {

int minDist = MAX;

int minIndex = -1;

for (int i = 1; i <= n; i++) {

if (!visit[i] && dist[i] <= minDist) {

minDist = dist[i];

minIndex = i;

}

}

return minIndex;

}

void dijkstra(int matr[MAX][MAX], int a, int b, int n, int matrix1[MAX][MAX], int matrix2[MAX][MAX]) {

int dist[MAX], vertex[MAX], transfer[MAX], count = 0;

bool visited[MAX] = { false };

for (int i = 1; i <= n; i++) {

dist[i] = MAX;

}

dist[a] = 0;

for (int i = 1; i <= n; i++) {

int ind = minDistance(dist, visited, n);

visited[ind] = true;

for (int v = 1; v <= n; v++) {

if (!visited[v] && matr[ind][v] && dist[ind] != MAX && dist[ind] + matr[ind][v] < dist[v]) {

dist[v] = dist[ind] + matr[ind][v];

vertex[v] = ind;

}

}

}

cout << "Кратчайший путь из города " << a << " в город " << b << ": " << dist[b] << endl;

int current = b;

while (current != a) {

transfer[count] = current;

current = vertex[current];

count++;

}

transfer[count] = a;

cout << "Маршрут с пересадками: ";

for (int i = count; i >= 0; i--) {

cout << transfer[i];

if (i > 0) {

if (matrix1[transfer[i]][transfer[i - 1]] != 0 && matrix2[transfer[i]][transfer[i - 1]] == 0 || matrix1[transfer[i]][transfer[i - 1]] < matrix2[transfer[i]][transfer[i - 1]]) {

cout << "(ш) ";

}

else if (matrix1[transfer[i]][transfer[i - 1]] == 0 && matrix2[transfer[i]][transfer[i - 1]] != 0 || matrix2[transfer[i]][transfer[i - 1]] < matrix1[transfer[i]][transfer[i - 1]]) {

cout << "(ж) ";

}

}

}

cout << endl;

}

int main()

{

SetConsoleOutputCP(1251);

int n, a, b;

int matrix1[MAX][MAX], matrix2[MAX][MAX], Matrix[MAX][MAX];

ifstream file1("C:\\Data Structures\\pr2\_3\\programm2\_3 mat\\matrix1.txt");

ifstream file2("C:\\Data Structures\\pr2\_3\\programm2\_3 mat\\matrix2.txt");

if (!file1 || !file2) {

cout << "Error opening";

return 0;

}

file1 >> n;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

file1 >> matrix1[i][j];

}

}

file1.close();

file2 >> n;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

file2 >> matrix2[i][j];

}

}

file2.close();

cout << "---------------------------" << endl;

cout << "Система шоссейних дорог: " << endl;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

cout << matrix1[i][j] << " ";

}

cout << endl;

}

cout << "---------------------------" << endl;

cout << "Система железных дорог: " << endl;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

cout << matrix2[i][j] << " ";

}

cout << endl;

}

cout << "---------------------------" << endl;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

if (matrix1[i][j] == 0 && matrix2[i][j] != 0) {

Matrix[i][j] = matrix2[i][j];

}

else if (matrix1[i][j] != 0 && matrix2[i][j] == 0) {

Matrix[i][j] = matrix1[i][j];

}

else if (matrix1[i][j] == 0 && matrix2[i][j] == 0) {

Matrix[i][j] = 0;

}

else {

Matrix[i][j] = min(matrix1[i][j], matrix2[i][j]);

}

}

}

cout << "Общая система всех дорог: " << endl;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

cout << Matrix[i][j] << " ";

}

cout << endl;

}

cout << "---------------------------" << endl;

do {

cout << "Введите номер города, из которого надо найти путь (от 1 до " << n << "): ";

cin >> a;

if (a < 1 || a > n) {

cout << "Город не найден" << endl;

}

} while (a < 1 || a > n);

do {

cout << "Введите номер города, до которого надо найти путь (от 1 до " << n << "): ";

cin >> b;

if (b < 1 || b > n) {

cout << "Город не найден" << endl;

}

} while (b < 1 || b > n);

dijkstra(Matrix, a, b, n, matrix1, matrix2);

}

Результаты работы программы:

При запуске программы выводятся система шоссейных дорог и система железных дорог. Обе системы представлены матрицами смежности. Затем эти матрицы объединяются и выводится система общих дорог. Для того, чтобы найти маршрут из вершины A в вершину B, нужно ввести эти вершины с клавиатуры, что показано на рисунке 1.

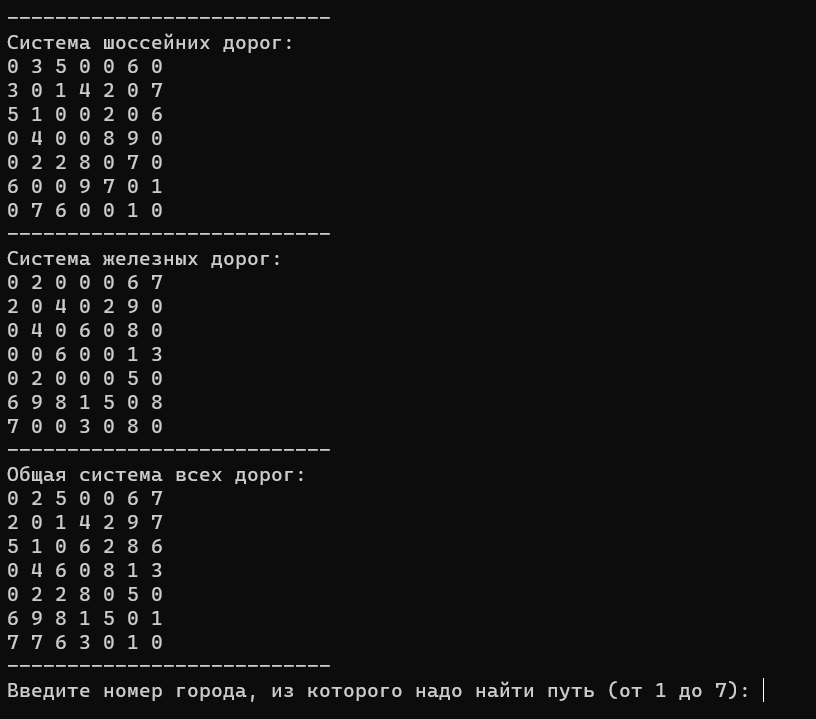


Рисунок 1 – Запуск программы

После ввода вершин A и B, выводится кратчайшее расстояние между этими вершинами, а также весь маршрут, в котором после вершины выводится (ш), если путь прошел по первой матрице, или (ж), если путь прошел по второй матрице, что показано на рисунке 2.

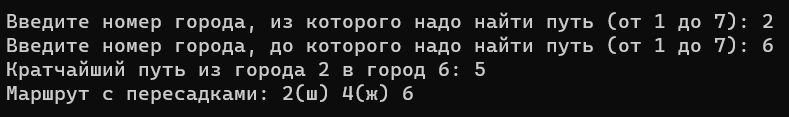


Рисунок 2 – Работа программы

Если ввести вершину, которая меньше или больше имеющихся в матрице, выведется сообщение об этом, что показано на рисунке 3.

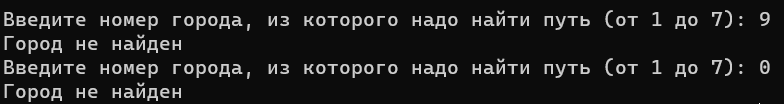


Рисунок 3 – Ввод несуществующей вершины