Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра ИИТ

**Лабораторная работа №6**

По дисциплине: “Алгоритмы и структуры данных”

Тема: ”Алгоритм Дейкстры”

**Выполнил:**

Студент 1-го курсы

Группы ПО-7

Лобан К.Ю.

**Проверила:**

Глущенко Т.А.

Брест, 2021

**Ход работы.**

**Постановка задачи:** Требуется реализовать алгоритм Дейкстры поиска кратчайшего пути от заданной вершины до всех остальных. Для хранения графа использовать матрицу смежности.

На экран вывести пути и их длины.

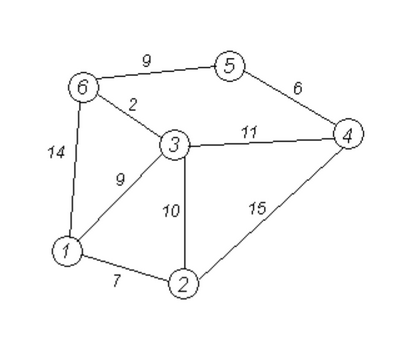


Таблица весов выглядит следующим образом:

Изображение выглядит как текст

Автоматически созданное описание

**Код программы:**

#include <iostream>

#include <fstream>

#include <limits>

using namespace std;

struct node{

int weight = std::numeric\_limits<int>::max()-1; //вес между рёбер

bool visit = 0; //посещена ли?

};

struct data\_info{

int amount; //количество вершин

node \*\_node;

int \*\*array;

};

data\_info input(){ //считывание из файла

int amount;

ifstream in("C:\\Users\\Admin\\Documents\\Универ\\table.txt");

in>>amount;

int \*\*array = new int\*[amount];

for (int i=0; i<amount; i++){

array[i] = new int [amount];

for (int j=0; j<amount; j++){

in>>array[i][j];

}

}

in.close();

node \*\_node = new node[amount];

\_node[0].weight = 0;

data\_info \_data;

\_data.array = array;

\_data.amount = amount;

\_data.\_node = \_node;

return \_data;

}

int algorithm (data\_info \_data){ //сам алгоритм

int amount = \_data.amount; //записываем

node \*\_node = \_data.\_node; //всю

int \*\*array = \_data.array; //информацию

int min\_weight = std::numeric\_limits<int>::max(); //все вершины оооочень далеко

int min\_i = -1;

for (int i=0; i<\_data.amount; i++){ //выявляем связи между вершинами и присваиваем им новые значения

if (\_node[i].weight <= min\_weight && \_node[i].visit == 0){

min\_weight = \_node[i].weight;

min\_i = i;

}

}

if (min\_i == -1){

return 0;

}

for (int i=0; i<amount; i++){ //находим кратчайший путь

if (array[min\_i][i] != 0 &&

\_node[i].visit == 0 &&

array[min\_i][i] + \_node[min\_i].weight < \_node[i].weight){

\_node[i].weight = array[min\_i][i] + \_node[min\_i].weight;

}

}

\_node[min\_i].visit = 1; //вершина посещена

cout<<min\_i + 1<<": "<<\_node[min\_i].weight<<endl;

algorithm (\_data);

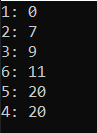
}

int main(){

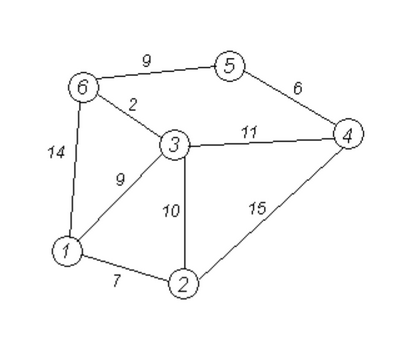
data\_info \_data = input();

algorithm (\_data);

}

**Результаты программы:**

**Граф после выполнения программы выглядит следующим образом:**



**20**

**20**

**9**

**11**

**7**

**0**

Рассматриваем вершины в следующем порядке: 1 -> 3 -> 2 -> 6 -> 5 -> 4