Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет (институт) Информационных технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра Прикладная математика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И.Потупчик

(подпись преподавателя) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Отчет

по лабораторной работе №5

\_\_\_\_\_\_\_\_\_\_\_\_Разработка и анализ программ обхода графа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(название лабораторной (практической) работы)

по дисциплине \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Типы и структуры данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование дисциплины)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ЛР 09.03.04.21.000 ОТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(обозначение документа)

Студент группы \_ПИ-02\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Р.А.Чередов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(инициалы, фамилия)

Преподаватель \_доцент, доцент А.И.Потупчик\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2022

**Вариант 21(1)**

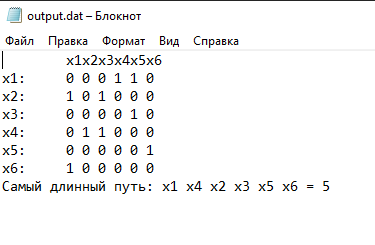
**Задание:**

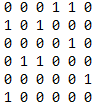
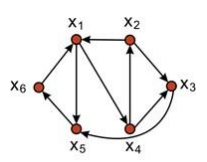
1. Разработать и отладить программу на языке C++, реализующую работу с графом в соответствии с вариантом. Выполнить оценку временной и емкостной сложности программы.
2. Исходные данные поместить в файл input.dat
3. Результаты вывести на экран. Исходные данные и результаты вывести также в выходной файл output.dat

**Задача:**

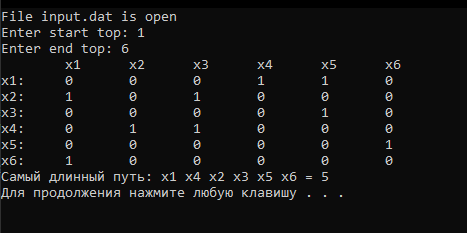
1. Найти самый длинный путь, соединяющий две данные вершины графа.

**Результат работы программы:**





**Исходный граф Данные файла input.dat Данные файла output.dat**



**Результат работы программы в консоли**

**Код программы:**

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <unordered\_set>

using namespace std;

string OpenPath = "input.dat"; // Название файла из которого достаются данные

string SavePath = "output.dat"; // Название файла в который сохраняются данные

template<typename T>

class Graf {

vector<vector<T>> graf; // Матрица смежности

public:

unordered\_set<int> MostLongWay; // Контейнер для хранения вершин самого длинного найденного пути

int size; // Размер матрицы

Graf() {

size = 0;

readFile(OpenPath);

}

// Вывод матрицы смежности

void printGraf() {

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

cout << "\tx" << i + 1;

}

cout << endl;

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

cout << "x" << i + 1 << ":\t";

for (int j = 0; j < size; ++j) { cout << graf[i][j] << "\t"; }

cout << endl;

}

}

// Создание пути

void NewWay(int row, int column, T data) { graf[row][column] = data; }

// Функция чтения данных из файла - незакончена

bool readFile(string FileName) {

ifstream fin;

fin.exceptions(ifstream::badbit | ifstream::failbit);

try {

fin.open(FileName);

cout << "File " << OpenPath << " is open" << endl;

}

catch (const exception& ex) {

cout << "Error: file " << OpenPath << " not found" << endl;

return false;

}

vector<T> temp;

// Находится размер матрицы и загружаются все данные в контейнер temp(одной строкой)

while (!fin.eof()) {

temp.push\_back(0);

fin >> temp[size];

++size;

}

size = sqrt(size);

// Создаётся пустая матрица смежности размером size X size

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

vector<T> t;

for (int i = 0; i < size; i++) { t.push\_back(0); }

graf.push\_back(t);

}

// Данные из контейнера temp помещаются в матрицу смежности

int tempCounter = 0;

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

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

NewWay(i, j, temp[tempCounter]);

++tempCounter;

}

}

fin.close();

return true;

}

// Функция сохранения матрицы в файл

void saveFile(int startWay, T maxWay, string FileName = SavePath) {

ofstream fout;

fout.open(FileName);

fout << "\t";

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

fout << "x" << i + 1;

}

fout << endl;

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

fout << "x" << i + 1 << ":\t";

for (int j = 0; j < size; ++j) { fout << graf[i][j] << " "; }

fout << endl;

}

fout << "Самый длинный путь: x" << startWay + 1 << " ";

for (auto& item : MostLongWay) {

fout << "x" << item << " ";

}

fout << "= " << maxWay << endl;

fout.close();

}

// Поиск самого длинного пути из вершины start

int FindMaxWay(int start, int end, vector<bool> way, unordered\_set<int>& nameWay, int &max) {

way[start] = true; // Флаг на текущий элемент = true

int count = 0; // Счётчик пути

// Алгоритм поиска

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

// Если в матрице смежности найдена 1

if (graf[start][i] != 0 && way[i] == false) {

way[i] = true; // Флаг на пройденую вершину = true

if (i != end) {

nameWay.emplace(i + 1); // Запись вершины

count = FindMaxWay(i, end, way, nameWay, max); // Вызов поиска самого длинного пути для найденой вершины

++count;

}

else {

++count;

nameWay.emplace(i + 1); // Запись вершины

if (count > max)

{

max = count;

MostLongWay = nameWay;

}

break;

}

--count;

}

}

return count;

}

// Посик самого длинного пути в графе

void FindMostLongWay(int start\_top, int end\_top) {

if (start\_top == end\_top) { return; }

int max = 0; // Максимальный путь

vector<bool> way; // Массив пройденных вершин

unordered\_set<int> ThisWay; // Переменная для хранения пути

// Массив вершин заполняется 0 (мы не прошли ни одной вершины)

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

way.emplace\_back(false);

}

// Алгоритм поиска

way[start\_top] = true; // Флаг текущего элемента ставится true

FindMaxWay(start\_top, end\_top, way, ThisWay, max); // поиск самого длинного пути из start\_top вершины в end\_top

}

};

int main() {

setlocale(LC\_ALL, "ru");

Graf<int> graf; // Граф

int maxWay; // Самый длинный путь в графе

int start = 0, end = 0;

// Ввод первой вершины

while (start < 1 || start > graf.size) {

cout << "Enter start top: ";

cin >> start;

}

--start;

// Ввод второй вершины

while (end < 1 || end > graf.size) {

cout << "Enter end top: ";

cin >> end;

}

--end;

graf.printGraf(); // Вывод графа в консоль

graf.FindMostLongWay(start,end); // Поиск самого длинного пути

maxWay = graf.MostLongWay.size();

// Вывод всех вершин

cout << "Самый длинный путь: x" << start+1 << " ";

for (auto& item : graf.MostLongWay) {

cout << "x" << item << " ";

}

cout << "= " << maxWay << endl;

graf.saveFile(start, maxWay); // Сохранение результатов в файл

system("pause");

}