Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №9

по дисциплине: «Логика и основы алгоритмизации в инженерных задачах.»

на тему: «Поиск расстояний в графе.»

Выполнили:

студенты группы 21ВВ4

Колокольцева У. А.

Нагорная Д. А.

Принял:

Акифьев И. В.  
Юрова О. В.

Пенза, 2022

**Цель работы:** разработать программный код для поиска расстояний в графе.

**Лабораторная работа:**

**Задание 1.**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При  реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

**Задание 2**.

1. \* Реализуйте процедуру поиска расстояний на основе обхода в глубину.
2. \* Оцените время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков.

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale>

#include <iostream>

#include <conio.h>

#include <time.h>

#include <stack>

#include <queue>

#include <stdio.h>

#include <Windows.h>

#include <stdlib.h>

using namespace std;

int N, j, i, m;

int\* dist; // дистанция

int vis[]; //выделяем память под булевы значения для матрицы

int\*\* graph;

queue<int>Q; //очередь

int start;

void BFSD(int vertex, int N, int\*\* graph) { // алгоритм в ширину

Q.push(vertex); //вносим в очередь вершину

dist[vertex] = 0; // обозначаем расстояние до нее

while (!Q.empty()) {

vertex = Q.front(); // вершину переносим в начало очереди

printf(" %d", vertex + 1);

Q.pop(); // выносим из очереди

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

if ((graph[vertex][i] == 1) && (dist[i] == -1)) {

Q.push(i);//вносим удовлетворяющую условию вершину в очередь

dist[i] = dist[vertex] + 1;//считаем дистанцию до вершины

}

}

}

}

void DFSD(int vertex, int N, int\*\* graph, bool vis[]) { // алгоритм в глубину

vis[vertex] = 1;

printf(" %d", vertex + 1);

for (int i = 0; i < N; i++)

{

if ((graph[vertex][i] == 1) && (vis[i] == 0)) {

DFSD(i, N, graph, vis);

}

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

int min = 10000;

printf("Ведите размер матрицы: ");

scanf\_s("%d", &N);

graph = new int\* [N];//выделяем память под матрицу

dist = new int[N]; // выделяем память для значений дистанции вершин друг от друга

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

graph[i] = new int[N];

}

for (int i = 0; i < N; i++) { //задаем изначальные значения дистанций

dist[i] = -1;

}

printf("\n");

for (i = 0; i < N; ++i) //заполняем матрицу

{

for (j = i; j < N; ++j)

{

graph[i][j] = graph[j][i] = rand() % 2;

graph[i][i] = graph[j][j] = 0; // чтобы петля(узел) не создавалась

}

}

printf(" ");

for (j = 0; j < N; j++)

{

printf("%4d ", j + 1); //горизонтальные нумерация сверху матрицы

}

printf("\n\n");

for (i = 0; i < N; i++) {

printf(" %d ", i + 1); // вертикальная нумерация слева матрицы

for (j = 0; j < N; j++)

{

printf("%4d ", graph[i][j]);

}

printf("\n\n");

}

printf("\n Введите вершину, с которой нужно начинать обход: ");

scanf\_s("%d", &start);

printf("\n Результат обхода в ширину: ");

srand(time(NULL));

BFSD(start - 1, N, graph);

cout << "\nВремя работы реализаций алгоритмов поиска расстояний на основе обхода в ширину = " << clock() / 1000.0 << endl; // время работы программы

printf("\n Вершины: ");

printf(" ");

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

printf(" %d ", i);

}

printf("\n Расстояние: ");

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

printf(" %d ", dist[i]);

}

printf("\n\n");

printf(" Результат обхода в глубину: ");

bool\* vis = new bool[start];

for (int i = 0; i < N; i++)

{

vis[i] = { 0 };

}

srand(time(NULL));

DFSD(start - 1, N, graph, vis);

cout << "\nВремя работы реализаций алгоритмов поиска расстояний на основе обхода в глубину = " << clock() / 1000.0 << endl; // время работы программы

printf("\n Вершины: ");

printf(" ");

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

printf(" %d ", i);

}

printf("\n Расстояние: ");

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

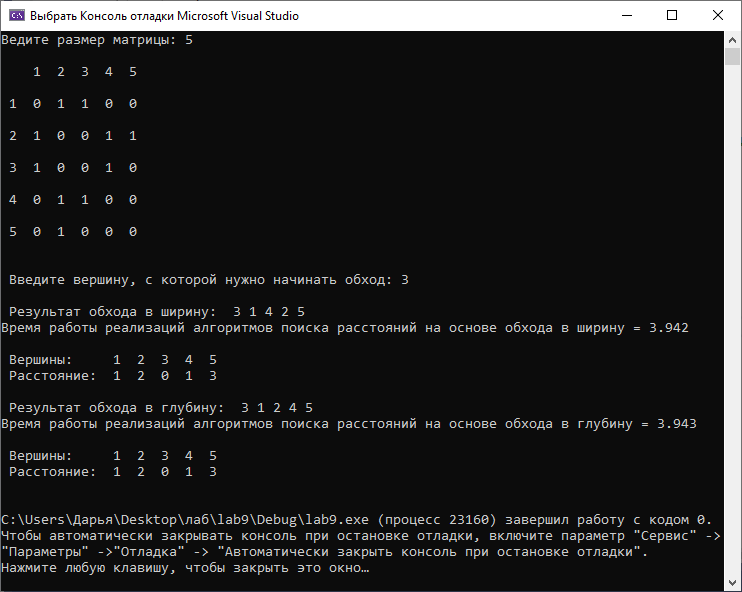
printf(" %d ", dist[i]);

}

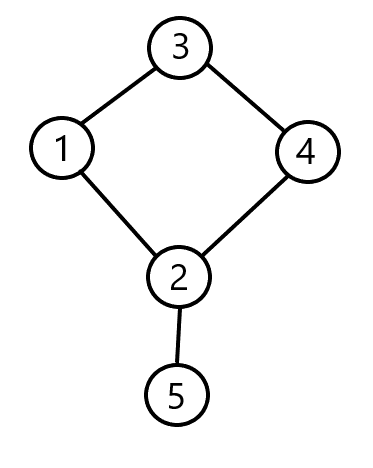
printf("\n\n");

}

Результат работы программы:



Проверка результатов:



**Вывод:** в ходе данной лабораторной работы был разработан программный код для поиска расстояний в графе, используя обход графа в ширину и глубину. А также оценили время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков.