Федеральное государственное бюджетное образовательное учреждение высшего образования «Пензенский Государственный университет»

Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной работе №9

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему «Поиск расстояний в графе»

Выполнили: ст. гр. 21ВВ1

Сурков А.А

Хорошильцев Е.А

Кривенков И.В

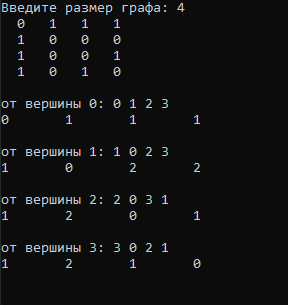
Приняли: Юрова О.В.

Акифьев И.В.

2022

**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++.



**Вывод:**

Выполняя данную лабораторную работу, мы научились реализовывать и применять алгоритма поиска расстояний в графе.

**Листинг**

**Задание 1:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <time.h>

#include <locale.h>

#include <queue>

using namespace std;

void adj(int\*\* G, int n) {

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

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

if (i == j) {

G[i][j] = 0;

}

if (i < j) {

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

G[j][i] = G[i][j];

}

}

}

}

void print\_adj(int\*\* G, int n) {

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

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

printf("%3d ", G[i][j]);

}

printf("\n");

}

printf("\n");

}

void BFSD(int v,int\* dist, int\*\* G, int n) {

queue <int> q;

q.push(v);

dist[v] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

printf("%d ", v);

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

if ((G[v][i] == 1) && dist[i] == -1) {

q.push(i);

dist[i] = dist[v] + 1;

}

}

}

printf("\n");

}

int main() {

int\*\* G; // указатель на указатель на строку элементов

int\* dist;

int i, j, n;

setlocale(LC\_ALL, "rus");

srand(time(NULL));

printf("Введите размер графа: ");

scanf("%d", &n);

G = (int\*\*)malloc(n \* sizeof(int\*));

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

G[i] = (int\*)malloc(n \* sizeof(int));

}

dist = (int\*)malloc(n \* sizeof(int));

adj(G, n);

print\_adj(G, n);

for (j = 0; j < n; j++) {

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

dist[i] = -1;

}

printf("от вершины %d: ", j);

BFSD(j, dist, G, n);

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

if (dist[i] != -1)

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

else

printf("нет прохода\t");

}

printf("\n\n");

}

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

free(G[i]);

}

free(G);

free(dist);

getchar();

getchar();

}