|  |
| --- |
| Министерство образования Российской Федерации  Пензенский государственный университет  Кафедра «Вычислительная техника» |
| Отчет  по лабораторной работе №6  по курсу «Логика и основы алгоритмизации в инженерных задачах»  на тему «Поиск расстояний в графе» |
|  |
|  |
| Выполнила студентка группы 19ВВ2:  Ходакова М.А.  Принял:  Митрохин М. А  Юрова О.В. |
| Пенза 2020 |

**Задание 1**

1. Сгенерировать (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Вывести матрицу на экран.

 Для сгенерированного графа осуществить процедуру поиска расстояний, реализованную в соответствии с описанием. При  реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

**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

bool\* visited = new bool[m]; //выделяем память под булевы значения для матрицы

int\*\* graph;

queue<int>Q; //очередь

int start;

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

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

}

}

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

int min = 10000;

printf("Ведите размер матрицы: ");

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

srand(time(NULL));

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 начальную вершину: ");

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

printf(" Результат обхода в ширину: ");

BFSD(start - 1, N, graph);

printf("\n\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:**

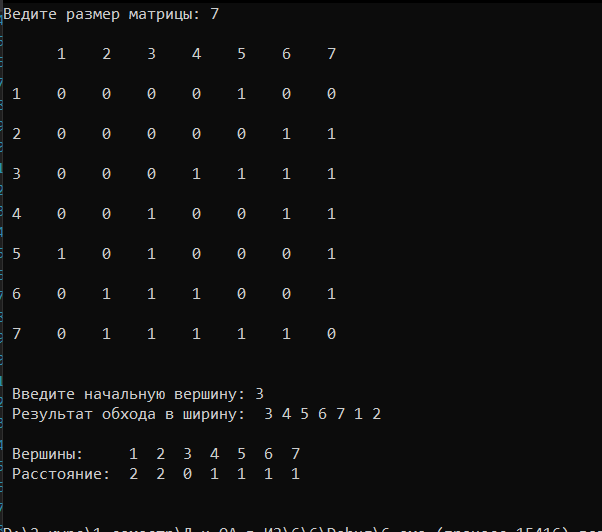


Рис. 1 Задание 1(пункты 1 и 2)

**Описание работы программы:**

При запуске программы нас просят ввести размер матрицы смежности. После чего выводится сама матрица. Для упрощения проверки работы алгоритма на экране появляется вершина и смежная(-ые) ей, если такие есть. Далее нужно ввести вершину, с которой мы хотим начать обход графа в ширину. Если мы введем не существующую вершину, то появится сообщение с просьбой выбрать вершину заново. Потом выведется результат обхода в ширину. Следом за ним каждая вершина графа, а под ней расстояние от выбранной нами ранее вершины.

**Вывод:**

В результате работы был реализован поиск расстояний в графе на основе обхода графа в ширину.