Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Л и ОА в ИЗ»

на тему «Обход графа в ширину»

**Выполнил:**

студент группы 19ВВ1

Васютин М.

**Приняли:**

Митрохин М.А.

Юрова О.В.

Пенза 2020

### **Задание 1**

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

**3.**\* Реализуйте процедуру обхода в ширину для графа, представленного списками смежности.

### **Задание 2\***

1. Для матричной формы представления графов реализуйте алгоритм обхода в ширину с использованием очереди, построенной на основе структуры данных «список», самостоятельно созданной в лабораторной работе № 3

**Листинг:**

#define \_CRT\_SEvE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

#include <locale.h>

#include <iostream>

#include <vector>

#include <time.h>

#include <stack>

#include <string>

#include <limits>

#include <queue>

using namespace std;

void BFS(int st, int N);

void BFS\_LIST(int st);

void BFS\_MY(int st, int N);

int\*\* graph;

int\* vis;

int\* vis3;

int\* qq;

int N, i, j, st, v;

using Edge = int; // ребро - целое число, указывающее, куда её ведут

using Vertex = vector<Edge>; // вершина - набор рёбер

using Graph = vector<Vertex>; // граф - набор вершин

Graph graph\_1;

vector<bool> visited; // посещённые вершины

queue <int> Q;

void BFS(int st, int N)

{

int r;

Q.push(st);

vis[st] = true;

while (!Q.empty())

{

st = Q.front();

Q.pop();

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

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

if ((graph[st][r] != 0) && (vis[r] == 0)) // ребро существует и вершина не посещена

{

Q.push(r);

vis[r] = true;

}

}

}

void BFS\_LIST(int st)

{

queue<int> q;

q.push(st); //в качестве начальной вершины используем введённую

visited[st] = true;

while (!q.empty())

{

v = q.front(); //извлекаем из очереди текущую вершину

q.pop();

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

for ( int a : graph\_1[v]) //добавляем всех непосещённых

{

if (!visited[a])

{

q.push(a);

visited[a] = true;

}

}

}

}

void BFS\_MY(int st, int N)

{

int first = 0;

int last = 0;

qq[last] = st;

last++;

vis[st] = true;

while (last != first)

{

st = qq[first];

first++;

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

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

{

if ((graph[st][j] == 1) && (vis[j] != 1))

{

qq[last] = j;

last++;

vis[j] = 1;

}

}

}

}

void main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

printf("\n Введите размер матрицы: ");

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

printf("\n");

graph = new int\* [N];

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

{

graph[i] = new int[N];

}

vis = new int[N];

qq = new int[N];

vis3 = new int[N];

int ch = 0;

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

{

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

{

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

graph[i][i] = graph[j][j] = 0; //Чтобы не было возврата

}

}

printf(" ");

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

{

printf("%4d ", j + 1); //горизонталь

}

printf("\n\n");

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

{

printf(" %d ", i + 1); // вертикаль

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

{

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

}

printf("\n\n");

}

graph\_1.resize(N); //размер графа

for (int i = 0; i < N; i++) //заполнение матрицами

{

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

{

if (graph[i][j] == 1)

graph\_1[i].push\_back(j);

}

}

printf("\n");

for (int i = 0; i < graph\_1.size(); i++) // вывод списка

{

printf(" Вершина %d смежна с ",i + 1);

for (int j = 0; j < graph\_1[i].size(); j++)

{

printf(" %d", graph\_1[i][j] + 1);

}

printf(" \n");

}

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

vis[i] = 0; //отмена посещения

printf(" \n");

printf("\n Выберите вершину ");

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

printf("\n Результат обхода в ширину матрицы через класс queue\n ");

clock\_t start1 = clock();

BFS(st-1, N);

clock\_t end1 = clock();

printf(" %.3f мс\n", difftime(end1, start1));

printf("\n\nРезультат обхода в ширину списка смежности\n ");

visited.resize(N);

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

{

visited[i] = 0;

}

clock\_t start3 = clock();

BFS\_LIST(st - 1);

clock\_t end3 = clock();

printf(" %.3f мс\n", difftime(end3, start3));

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

{

vis[i] = 0; //отмена посещения

}

printf("\n\n Результат обхода в ширину матрицы через самостоятельно очередь\n ");

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

vis3[i] = 0; //отмена посещения

clock\_t start2 = clock();

BFS\_MY(st-1, N);

clock\_t end2 = clock();

printf(" %.3f мс\n", difftime(end2, start2));

system("pause");

}

**Результат работы программы:**

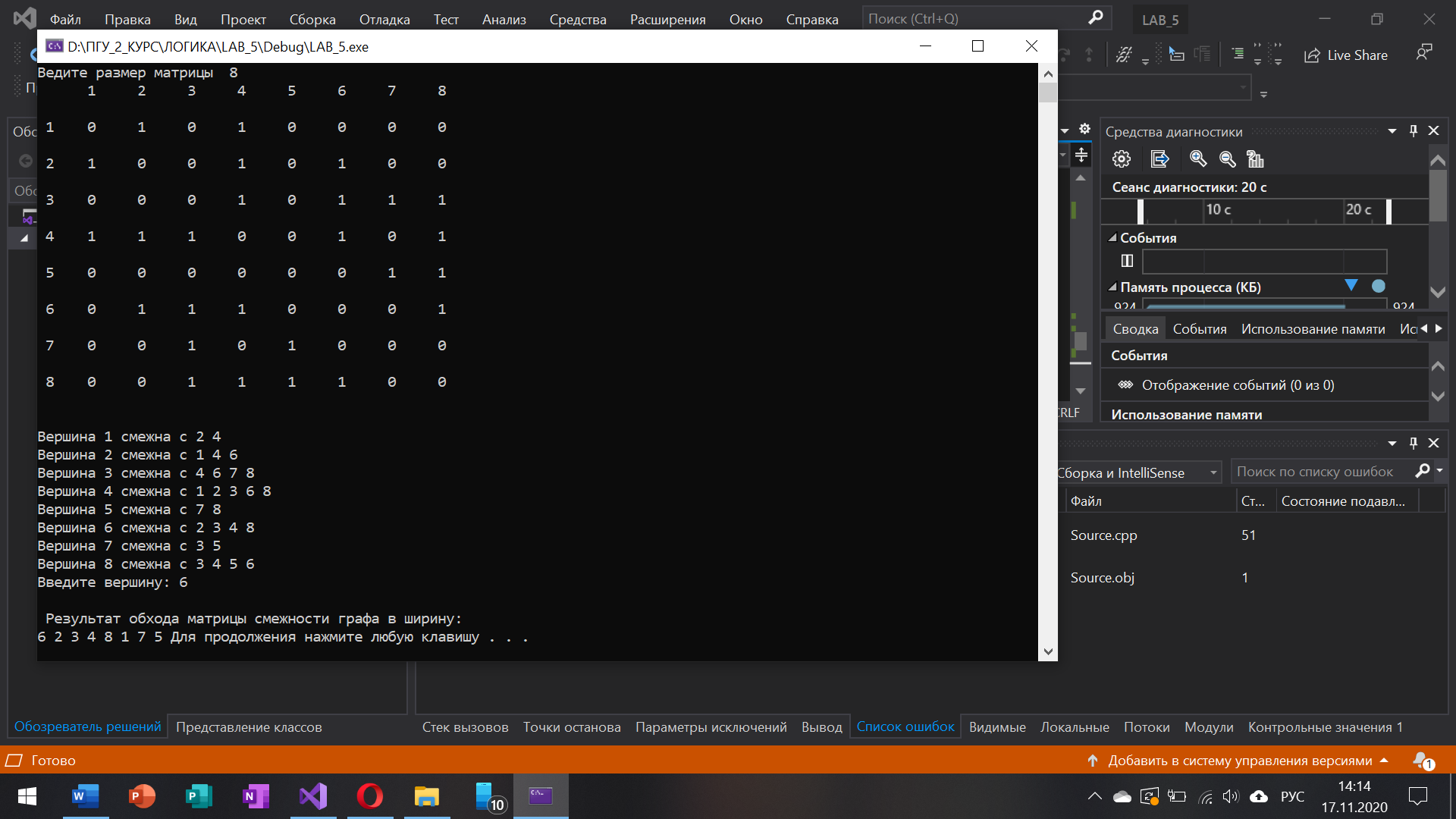


Рис. 1 Задание 1(пункты 1 и 2)

**Вывод:**

В результате работы был реализован обход в ширину с помощью работы с очередью. Идея обхода в ширину заключается в посещении вершин на каждом уровне удаленности.