Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Л и ОА в ИЗ»

на тему «Обход графа в глубину»

**Выполнил:**

студент группы 19ВВ1

Васютин М.

**Приняли:**

Митрохин М.А.

Юрова О.В.

Пенза 2020

**Цель работы:**

Изучить алгоритм обхода графа в глубину.

**Ход работы:**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу

смежности для неориентированного графа G. Выведите сгенерированные матрицы на экран.

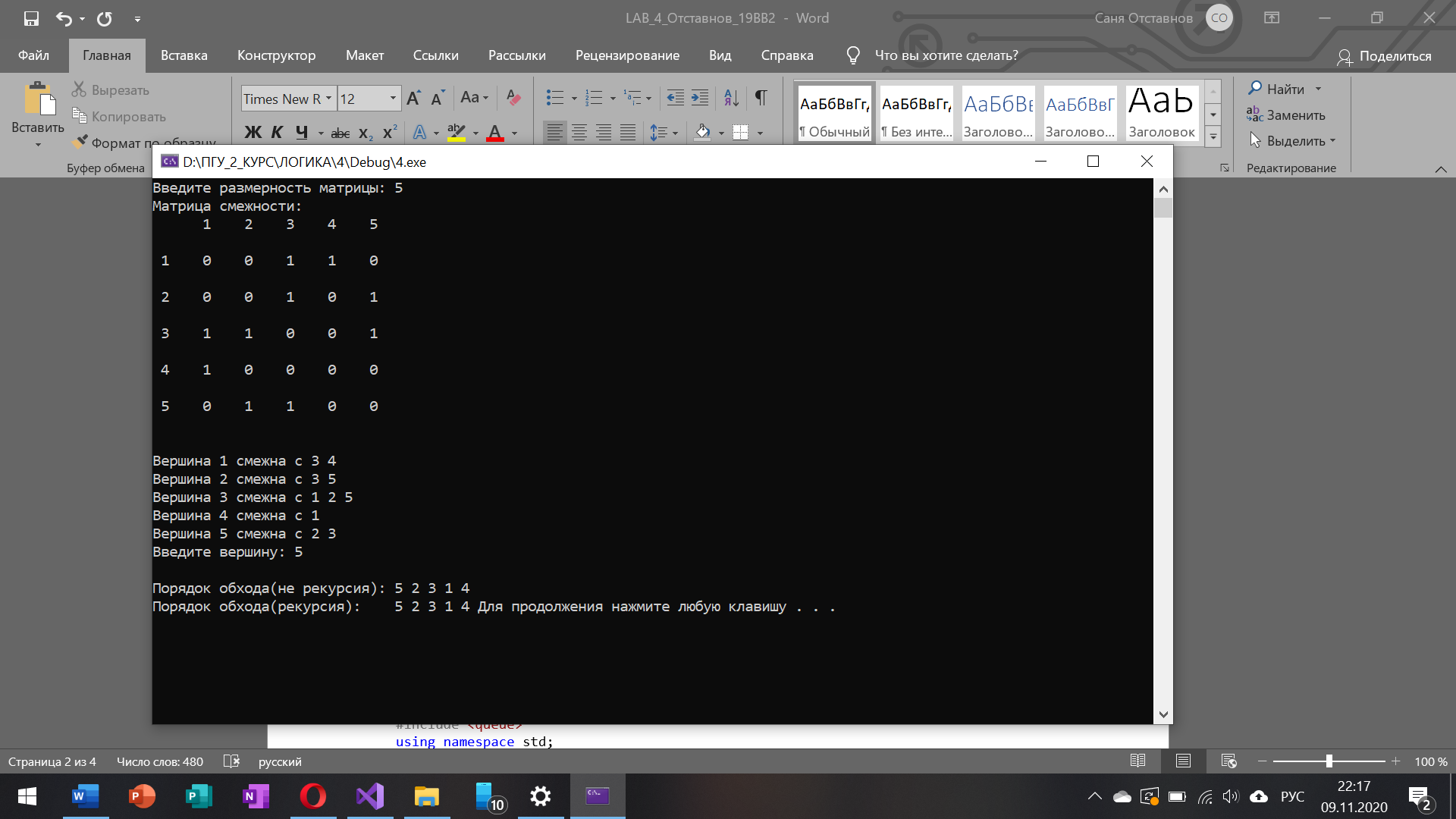
2. Для сгенерированного графа осуществите процедуру обхода в

глубину, реализованную в соответствии с приведенным выше описанием.

**Задание 2\***

1. Для матричной формы представления графов выполните

преобразование рекурсивной реализации обхода графа к не рекурсивной.



Результат работы программы

**Объяснение вывода программы:**

Для начала работы, программа просит ввести размерность матрицы смежности. В нашем случае это 5×5. После чего выводится сама матрица смежности. После выводится каждая вершина, а также вершины смежные с ней. Далее программа просит ввести номер вершины, с которой Вы бы хотели начать обход. В нашем случае это вершина 5. После чего на экран выводится порядок обхода. Сначала результат нерекурсивного обхода, а потом рекурсивного.

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <conio.h>

#include <time.h>

#include <stack>

#include <queue>

#include <vector>

#include <locale>

using namespace std;

using Edge = int;

using Vertex = vector<Edge>;

using Graph = vector<Vertex>;

int i, j, m, k;

bool\* visited = new bool[m];

bool\* visit = new bool[m];

int\*\* graph;

stack <int> s;

stack <int> ss;

Graph GRAPH\_1;

queue <int> sss;

void DFS\_NO\_REC(int st)

{

visited[st] = true;

ss.push(st);

int v;

while (!ss.empty()) //записывает в стек последнюю вершину, затем вытаскивает её и записывает следущую

{

v = ss.top();

ss.pop();

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

for (i = m; i >= 0; i--)

{

if ((graph[v][i] == 1) && (!visited[i]))

{

visited[i] = 1;

ss.push(i);

}

}

}

}

void DFS\_REC(int st)

{

int r;

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

visited[st] = true;

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

if ((graph[st][r] == 1) && (visited[r] == false))

DFS\_REC(r);

}

void DFS\_REC\_LIST(int v)

{

{

if (visit[v])

{

return;

}

visit[v] = 1;

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

for (int i : GRAPH\_1[v]) // цикл, основанный на диапазоне

{

DFS\_REC\_LIST(i); //новый обход от каждого рядом стоящего

}

}

}

void DFS\_main()

{

setlocale(LC\_ALL, "Rus");

printf("Введите размерность матрицы: ");

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

graph = new int\* [m];

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

{

graph[i] = new int[m];

}

int start;

printf("Матрица смежности:\n");

srand(time(NULL));

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

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

{

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

graph[i][i] = graph[j][j] = 0; //петлю убиракм

}

printf(" ");

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

{

printf(" %d ", j + 1); //горизонталь

}

printf("\n\n");

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

{

printf(" %d ", i + 1); //вертикаль

visited[i] = false;

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

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

printf("\n\n");

}

GRAPH\_1.resize(m);

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

{

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

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

{

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

{

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

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

}

}

}

printf("\nВведите вершину: ");

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

while (start > m)

{

printf("\nТакая вершина не существует!\n");

printf("\nВведи вершину: ");

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

}

bool\* vis = new bool[m];

printf("\nПорядок обхода матрицы(не рекурсивный): ");

DFS\_NO\_REC(start - 1);

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

{

visited[i] = false;

}

printf("\nПорядок обхода матрицы (рекурсивный): ");

DFS\_REC(start - 1);

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

{

visit[i] = false;

}

printf("\nПорядок обхода списка (рекурсивный): ");

DFS\_REC\_LIST(start - 1);

system("pause");

}

void main(void)

{

DFS\_main();

}

### Вывод

В ходе выполнения лабораторной работы были изучен и реализован обход графа в глубину.