Министерство образования Российской Федерации   
Пензенский государственный университет   
Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной №3   
На тему: «Обход графа в глубину»  
По дисциплине: «Логика и основы алгоритмизации в инженерных задачах»

Выполнили студенты группы 20ВВ2:

Борунов Д.А

Сурков М.С

Приняли:

Митрохин М.А. Юрова О.В.

**Пенза 2020г**

**Лабораторное задание**

**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

глубину, реализованную в соответствии с приведенным выше описанием.

**3.\*** Реализуйте процедуру обхода в глубину для графа, представленного

списками смежности.

**Задание 2\***

1. Для матричной формы представления графов выполните

преобразование рекурсивной реализации обхода графа к не рекурсивной.

**Метод выполнения задания**

**Задание 1.1 и 1.2.**

Создать матрицу смежности графа (двумерный динамический массив). Описать функцию:

DFS(int st, int\*\* b, int num\_of\_elem, bool\* visited)

int st **–** стартовая вершина

int\*\* b – указатель на сгенерированную матрицу смежности

bool\* visited – указатель на массив посещенных вершин

int num\_of\_elem – количество элементов в сгенерированной матрице

Перед вызовом функции пометить все вершины сгенерированного графа как не помеченные (False) и с читать с клавиатуры стартовой вершины графа для последующей передачи ее в функцию.

Функция **DFS –** рекурсивная и вызывает саму себя, если находит еще не посещенную смежную вершину.

**Задание 1.3**

Описать функцию:

void DFS\_noRekr(int st, int\*\* b, int num\_of\_elem)

int st **–** стартовая вершина

int\*\* b – указатель на сгенерированную матрицу смежности

int num\_of\_elem – количество элементов в сгенерированной матрице

В теле функции создается очередь Q при помощи библиотеки <queue>  
и массив не посещенных вершин, который заполняется нулями (0-вершина не посещена, 1 - посещена, индекс элемента = номер элемента). Стартовая вершина помещается в очередь (push()). Далее следует проверка на наличие элементов в очереди (функция empty()). При выполнении условия, в очередь помещается не посещенная вершина, которая смежна вершине на данной итерации цикла. Перед началом следующей итерации печатается, а затем удаляется последний элемент очереди.

**Листинг программы**

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <cstdlib>

#include <windows.h>

#include <conio.h>

#include <ctime>

#include <cmath>

#include <stack>

#include <queue>

using namespace std;

int max\_size = 6, i, j, start;

bool\* vektor = (bool\*)malloc(max\_size \* sizeof(bool));

void DFS(int st, int\*\* b, int num\_of\_elem, bool\* visited) {

int g;

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

visited[st] = true;

for (g = 0; g < max\_size; g++) {

if (b[st][g] != 0 && !visited[g])

DFS(g, b, num\_of\_elem, visited);

}

}

void DFS\_noRekr(int st, int\*\* b, int num\_of\_elem)

{

queue<int> Q;

int\* mass;

int t;

mass = (int\*)malloc(num\_of\_elem \* sizeof(int));

for (int i = 0; i < num\_of\_elem; i++) {

mass[i] = 0;

}

Q.push(st);

mass[st] += 1;

while (!Q.empty())

{

t = Q.front();

Q.pop();

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

for (int i = 0; i < num\_of\_elem; i++)

if (b[t][i] && !mass[i])

{

Q.push(i);

mass[i] += 1;

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int\*\* a = new int\* [max\_size];

for (i = 0; i < max\_size; i++)

{

a[i] = new int[max\_size];

}

for (i = 0; i < max\_size; i++)

{

for (j = i + 1; j < max\_size; j++)

{

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

a[j][i] = a[i][j];

}

}

for (i = 0; i < max\_size; i++)

{

vektor[i] = false;

for (j = 0; j < max\_size; j++)

{

if (i == j)

a[i][j] = 0;

printf("%2d", a[i][j]);

}

printf("\n");

}

///////////////////////////////////////

printf("\nОбход графа в глубину методом рекурсии\ncтартовая вершина - ");

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

printf("Вывод: ");

DFS(start-1, a, max\_size, vektor);

printf("\n");

printf("\nОбход графа в глубину методом рекурсии\ncтартовая вершина - ");

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

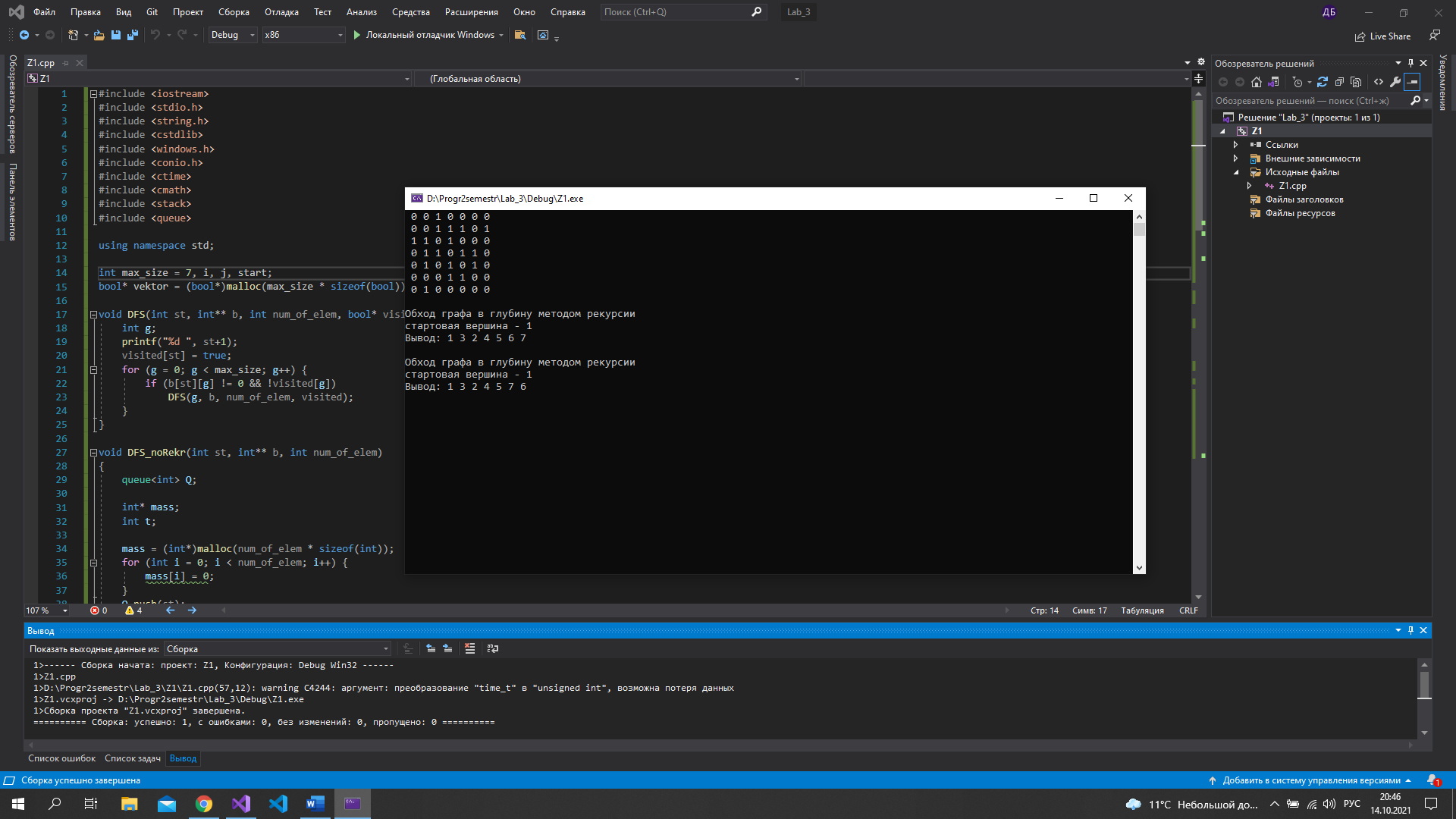
printf("Вывод: ");

DFS\_noRekr(start-1, a, max\_size);

\_getch();

}

**Результат работы программы**



**Вывод:** в ходе выполнения лабораторной реализовали алгоритм обхода графа в глубину несколькими способами