Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Логика и основы алгоротмизирования в ИС»

на тему «Обход графа в глубину»

Выполнили

студенты группы 20ВВ2:

Тумасов Вадим

Портнов Никита

Принял:

д.т.н Митрохин М.А.

д.т.н Юрова О.В.

Пенза 2021

**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.

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <ctime> // Библиотека времени

#include <cmath>

#include <locale.h>

#include <iostream> // Библиотека Ввода,Вывода

#include <conio.h>

#include <stack>

int n;

int\*\* matrix = NULL;

using namespace std;

bool\* visited = NULL;

struct spisok\_smzh {

int versh;

spisok\_smzh\* next;

};

struct spiski\_smzh {

spisok\_smzh\* spisok;

spiski\_smzh\* next;

};

spiski\_smzh\* p\_spiski = NULL;

spiski\_smzh\* n\_spiski = NULL;

spiski\_smzh\* p\_spiski\_start = NULL;

spisok\_smzh\* p\_spisok = NULL;

spisok\_smzh\* p\_spisok\_start = NULL;

spisok\_smzh\* n\_spisok = NULL;

void DFSS(int st)

{

stack<int> s;

s.push(st);

visited[st] = true;

while (!s.empty())

{

int v = s.top();

s.pop();

cout << v <<" ";

for (int i = n-1; i > -1; --i)

{

if ((visited[i] == false) && (matrix[v][i]==1))

{

s.push(i);

visited[i] = true;

}

}

}

}

void DFS(int st)

{

int r;

p\_spiski = p\_spiski\_start;

spisok\_smzh\* p\_spisok\_loc = NULL;

cout << st << " ";

visited[st] = true;

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

p\_spiski = p\_spiski->next;

p\_spisok\_loc = p\_spiski->spisok;

while (p\_spisok\_loc != NULL)

{

if (!visited[p\_spisok\_loc->versh])

{

DFS(p\_spisok\_loc->versh);

}

p\_spisok\_loc = p\_spisok\_loc->next;

}

}

void main()

{

setlocale(LC\_ALL, "Russian"); // подключение русского

srand(time(NULL)); // новый отсчёт времени

int i, j, m;

int start;

cout << "n=";

cin >> n;

//cout << "Матрица: \n";

int z;

matrix = new int\* [n]; // матрица

visited = new bool[n];

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

{

matrix[i] = new int[n];

}

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

{

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

{

matrix[i][m] = rand() % 2;

}

}

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

matrix[i][i] = 0;

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

{

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

{

matrix[i][j] = matrix[j][i];

cout << matrix[i][j]<<" ";

}

cout << endl;

}

/\*for (int i = 0; i < n; i++)

{

cout << "\n";

cout << "{";

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

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

{

cout << j << " ";

}

cout << "}" << " Вершина" << i;

}

\*/

cout << "\n";

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

{

n\_spiski = (spiski\_smzh\*)malloc(sizeof(spiski\_smzh));

p\_spisok = NULL;

p\_spisok\_start = NULL;

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

{

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

{

n\_spisok = (spisok\_smzh\*)malloc(sizeof(spisok\_smzh));

n\_spisok->versh = j;

if (p\_spisok == NULL)

{

p\_spisok = n\_spisok;

p\_spisok\_start = p\_spisok;

}

else

{

p\_spisok->next = n\_spisok;

p\_spisok = n\_spisok;

}

n\_spisok->next = NULL;

}

}

n\_spiski->spisok = p\_spisok\_start;

if (p\_spiski == NULL)

{

p\_spiski = n\_spiski;

p\_spiski\_start = p\_spiski;

}

else

{

p\_spiski->next = n\_spiski;

p\_spiski = n\_spiski;

}

}

p\_spiski = p\_spiski\_start;

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

{

p\_spisok = p\_spiski->spisok;

p\_spisok\_start = p\_spisok;

cout << "\n";

cout << "{";

while (p\_spisok != NULL)

{

cout << p\_spisok->versh << " ";

p\_spisok = p\_spisok->next;

}

cout << "}" << " Вершина" << i;

p\_spisok = p\_spisok\_start;

p\_spiski = p\_spiski->next;

}

cout << "\n";

p\_spiski = p\_spiski\_start;

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

{

visited[i] = false;

}

cout << "Стартовая вершина » ";

cin >> start;

bool\* vis = new bool[n];

cout << "Порядок обхода DFSS: ";

DFSS(start);

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

{

visited[i] = false;

}

cout << "\n";

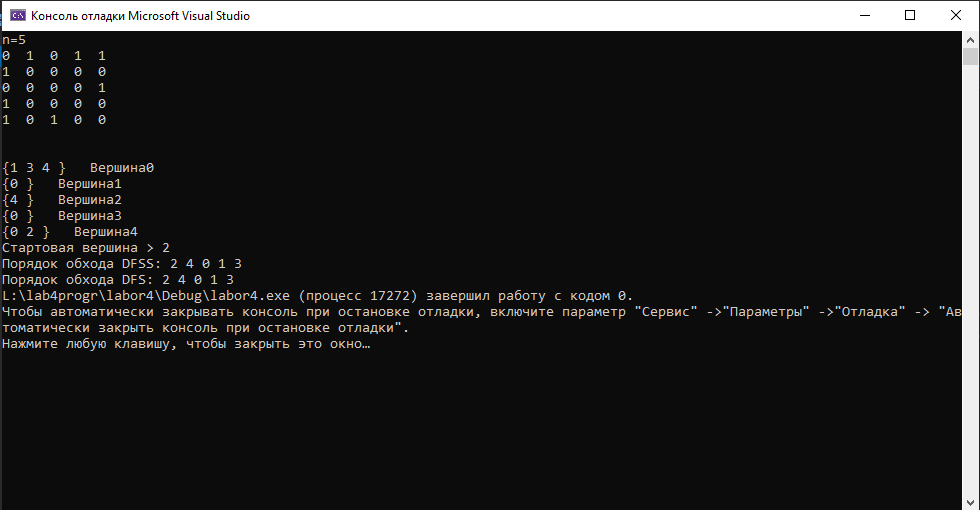
cout << "Порядок обхода DFS: ";

DFS(start);

//delete[]visited;

}

**Результат работы программы:**



**Вывод**: Мы сгенерировали матрицу смежности для неориентированного графа, организовали обход в глубину с помощью dfs.Научились представлять неориентированный граф списком смежности.Также произвели изменения в работе самого алгоритма обхода в глубину,сменив его фунционирование с помощью рекурсии на работу со стеком.