Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

# по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»

на тему: «Обход графа в глубину»

Выполнили:

студенты группы 20ВВ3

Горожанин Я. А.

Тихонов А. А.

Скирдова В. М.

Приняли:

Юрова О.В.

Митрохин М.А.

Пенза 2021

**Название**

Обход графа в глубину

**Цель работы**

Разработка алгоритма обхода в глубину

**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**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <iostream>

#include <stack>

using namespace std;

struct node {

int numb;

node\* next;

};

struct graf {

node\*\* nodes;

int size;

};

void deep(int\*\* a, int num, int\* versh, int size) {

versh[num] = 1;

cout << num << " -> ";

for (int i = 0; i < size; i++) {

if (a[num][i] != 0 && versh[i] != 1)

deep(a, i, versh, size);

}

}

void deepDarkAss(graf\* grafon, int num, int\* versh){

versh[num] = 1;

node\* buf = grafon->nodes[num];

cout << num << " -> ";

while (buf != NULL) {

if (versh[buf->numb] == 0)

deepDarkAss(grafon, buf->numb, versh);

buf = buf->next;

}

}

void norec(int\*\* a, int num, int\* versh, int size) {

stack <int> steck;

versh[num] = 1;

cout << " " << num << " -> ";

steck.push(num);

int q, i;

while (!steck.empty()) {

for (i = 0; i < size; i++) {

if (a[num][i] != 0 && versh[i] != 1) {

cout << i << " -> ";

steck.push(num);

versh[i] = 1;

num = i;

break;

}

}

if (i == size) {

num = steck.top();

steck.pop();

}

}

}

void norec\_cunt\_fuck(graf\* grafon, int num, int\* versh, int size) {

stack <int> steck;

versh[num] = 1;

node\* buf = grafon->nodes[num];

cout << " " << num << " -> ";

steck.push(num);

int q;

while (!steck.empty()) {

buf = grafon->nodes[num];

while (buf != 0) {

if (versh[buf->numb] == 0) {

cout << buf->numb << " -> ";

steck.push(num);

versh[buf->numb] = 1;

num = buf->numb;

break;

}

else

buf = buf->next;

}

if (buf == NULL) {

num = steck.top();

steck.pop();

}

}

}

graf\* sozdat(int versh) {

graf\* grafon = new graf;

grafon->size = versh;

grafon->nodes = new node\* [versh];

for (int i = 0; i < versh; i++) {

grafon->nodes[i] = NULL;

}

return grafon;

}

node\* sozdatnode(int index) {

node\* newnode = new node;

newnode->numb = index;

newnode->next = NULL;

return newnode;

}

void addgran(graf\* grafon, int from, int to) {

node\* newnode = sozdatnode(from);

if (grafon->nodes[to] == 0) {

grafon->nodes[to] = newnode;

newnode = NULL;

}

node\* buf = grafon->nodes[to];

while (buf->next != NULL) {

buf = buf->next;

}

buf->next = newnode;

newnode = sozdatnode(to);

if (grafon->nodes[from] == 0) {

grafon->nodes[from] = newnode;

return;

}

buf = grafon->nodes[from];

while (buf->next != NULL) {

buf = buf->next;

}

buf->next = newnode;

}

void main(){

srand(time(0));

setlocale (LC\_ALL, "Russian");

int size;

cout << " Введите размерность матрицы: "<< " ";

cin >> size;

cout << endl;

int\*\* arr = new int\* [size];

for (int count = 0; count < size; ++count)

arr[count] = new int[size];

for(int row = 0;row< size;row++){

arr[row][row] = 0;

for(int col = row+1;col< size;col++){

arr[row][col] = rand()%2;

arr [col][row] = arr[row][col];

}

}

cout << " ";

for(int row = 0;row < size;row++)

{

for(int col = 0; col < size;col++){

cout << arr[row][col] << " " ;

}

cout << endl << " ";

}

cout << endl;

auto versh = new int[size];

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

versh[i] = 0;

cout << " Введите номер вершины, с которой хотите начать обход: ";

int num;

cin >> num;

cout << endl << " ";

deep(arr, num, versh, size);

cout << endl << endl << " Обход в глубину списков смежности" << endl;

cout << " " << endl;

graf\* grafon = sozdat(size);

int j = 1;

for (int i = 0; i < size; i++) {

for (j; j < size; j++) {

if (arr[i][j] == 1) {

addgran(grafon, i, j);

}

}

j = j - size + i + 1;

}

cout << " ";

for (int i = 0; i < size; i++) {

node\* temp = grafon->nodes[i];

cout << i;

while (temp) {

cout << " -> " << temp->numb;

temp = temp->next;

}

cout << endl << " ";

}

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

versh[i] = 0;

cout << endl << " Введите номер вершины, с которой хотите начать обход: ";

num = 0;

cin >> num;

cout << endl << " ";

deepDarkAss(grafon, num, versh);

cout << endl << endl << " Обход в глубину нерекурсивным способом матриц" << endl;

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

versh[i] = 0;

cout << endl << " Введите номер вершины, с которой хотите начать обход: ";

num = 0;

cin >> num;

cout << endl;

auto stack = new int(size);

norec(arr, num, versh, size);

cout << endl;

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

versh[i] = 0;

cout << endl << " Обход в глубину нерекурсивным способом списков смежности" << endl;

cout << endl << " Введите номер вершины, с которой хотите начать обход: ";

num = 0;

cin >> num;

cout << endl;

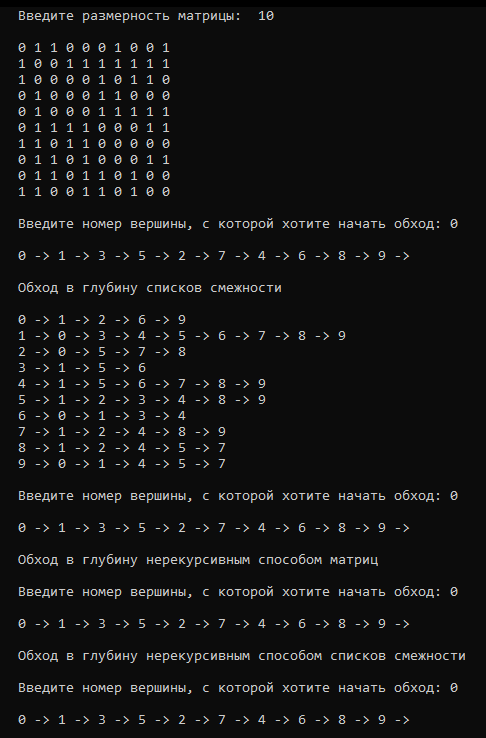
stack = new int(size);

norec\_cunt\_fuck(grafon, num, versh, size);

cout << endl;

}

**Результат работы программы**



**Вывод**

В ходе выполнения данной лабораторной работы, были получены навыки разработки алгоритма в глубину матрицами и списками смежности, а также преобразование рекурсивной реализации обхода графа к не рекурсивной для матричной формы представления графов.