Министерство образования и науки Российской Федерации

Пензенский государственный университет

Факультет вычислительной техники

**Отчёт**

По лабораторной работе №3

По курсу "ЛиОАвИЗ"

Выполнили:

Студент группы 20ВВ1

Репин И.В.

Приняла:

Юрова О.В.

Пенза 2021

**Название:** Обход графа в глубину

**Цель работы:** выполнить ряд заданий

**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.

**Описание метода решения задачи:**

Задание 1.

1. Заполняем двумерный массив случайными 0 и 1, следим за тем что бы значения были симметричны относительно главное оси
2. Реализуем следующую функцию:

DFS(v):

пометить v как посещенную&;

NUM[v] = True;

вывести на экран v;

ДЛЯ i = 1 ДО size\_G ВЫПОЛНЯТЬ

ЕСЛИ G(v,i) = = 1И NUM[i] = = False

ТО

{

DFS(i);

}

1. Аналогичную функцию реализуем для графа

Задание 2.

Реализуем стэк

**Листинг:**

// lb4.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

int\* vis;

int\* norvis;

int\* vislist;

int\* prev;

int num;

int\*\* m;

int norDFS(int n);

void DFS(int num);

struct list\* init(int a);

struct list\* addelem(list\* lst, int number, int id, int line);

void listprint(list\* lst);

void DFSlst(int n, list\* lst);

struct list

{

int field, id, line; // поле данных

struct list\* ptr; // указатель на следующий элемент

};

struct list\* init(int a) // а- значение первого узла

{

struct list\* lst;

// выделение памяти под корень списка

lst = (struct list\*)malloc(sizeof(struct list));

lst->field = a;

lst->id = a;

lst->line = -1;

lst->ptr = NULL; // это последний узел списка

return(lst);

}

struct list\* addelem(list\* lst, int number, int id, int line)

{

struct list\* temp, \* p;

temp = (struct list\*)malloc(sizeof(list));

p = lst->ptr; // сохранение указателя на следующий узел

lst->ptr = temp; // предыдущий узел указывает на создаваемый

temp->field = number; // сохранение поля данных добавляемого узла

temp->id = id;

temp->line = line;

temp->ptr = p; // созданный узел указывает на следующий элемент

return(temp);

}

void listprint(list\* lst)

{

struct list\* p;

p = lst->ptr;

int line, id;

line = 0;

id = 0;

do {

if (p->line > line) {

printf("\n");

line++;

id = 0;

}

printf("%4i", p->id);

p = p->ptr; // переход к следующему узлу

} while (p != NULL);

}

void DFS(int n) {

vis[n] = 1;

printf("%4i", n);

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

if ((m[n][i] == 1) && (vis[i] == 0)) {

DFS(i);

}

}

}

void DFSlst(int n, list\* lst) {

struct list\* p;

int i;

p = lst->ptr;

printf("%4i", n);

vislist[n] = 1;

while (p->line != n) {

p = p->ptr;

}

do{

i = p->id;

if (vislist[i] == 0) {

DFSlst(i, lst);

}

p = p->ptr;

} while (p != NULL);

}

int norDFS(int n) {

int cnt = 0;

if(norvis[n] != 1){

norvis[n] = 1;

printf("%4i", n);

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

if ((norvis[i] == 0)&&(m[n][i] == 1)){

cnt++;

}

}

if (cnt != 0) {

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

if ((norvis[i] == 0) && (m[n][i] == 1)) {

int prevc = 0, j=0;

while (prevc == 0) {

if (prev[j] == 0) {

prev[j] = n;

prevc = 1;

}

j++;

}

return i;

}

}

}

else {

int prevn;

int prevc = 0, j = 0;

while (prevc == 0) {

if (prev[j] == 0) {

prevn = j - 1;

prevc = 1;

}

j++;

}

return prev[prevn];

}

}

else {

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

if ((norvis[i] == 0) && (m[n][i] == 1)) {

cnt++;

}

}

if (cnt != 0) {

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

if ((norvis[i] == 0) && (m[n][i] == 1)) {

int prevc = 0, j = 0;

while (prevc == 0) {

if (prev[j] == 0) {

prev[j] = n;

prevc = 1;

}

j++;

}

return i;

}

}

}

else {

int prevn;

int prevc = 0, j = 0;

while (prevc == 0) {

if (prev[j] == 0) {

prevn = j - 2;

prev[j - 1] = 0;

prevc = 1;

}

j++;

}

return prev[prevn];

}

}

}

int main()

{

//generate array

int generate;

printf("Input array size: ");

scanf("%i", &num);

m = new int\* [num];

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

m[i] = new int[num];

}

srand(time(NULL));

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

for (int j = 0; j < num; j++) {

if (j == i) {

m[i][j] = 0;

}

else if (j > i) {

generate = rand() % 100;

if (generate >= 30) {

m[i][j] = 1;

}

else {

m[i][j] = 0;

}

}

else {

m[i][j] = m[j][i];

}

}

}

//print array

printf("array:");

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

printf("\n");

for (int j = 0; j < num; j++) {

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

}

}

//generate list

struct list \*lst = init(0);

for (int i = (num-1); i >= 0; i--) {

for (int j = (num-1); j >= 0; j--) {

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

addelem(lst, 1, j, i);

}

}

}

//print list

printf("\nList: \n");

listprint(lst);

//rounds

vis = new int [num];

vislist = new int[num];

norvis = new int[num];

prev = new int[num+1];

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

vis[i] = 0;

vislist[i] = 0;

norvis[i] = 0;

prev[i] = 0;

}

prev[num] = 0;

//result

printf("\n");

printf("\n");

printf("recursive matrix:\n");

DFS(0);

printf("\n");

printf("\n");

printf("list:\n");

DFSlst(0, lst);

printf("\n");

printf("\n");

printf("not recursive:\n");

int complited = 0, actn = 0;

while (complited == 0) {

actn = norDFS(actn);

int check = 1;

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

if (norvis[i] == 0) {

check = 0;

}

}

complited = check;

}

}

// Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"

// Отладка программы: F5 или меню "Отладка" > "Запустить отладку"

// Советы по началу работы

// 1. В окне обозревателя решений можно добавлять файлы и управлять ими.

// 2. В окне Team Explorer можно подключиться к системе управления версиями.

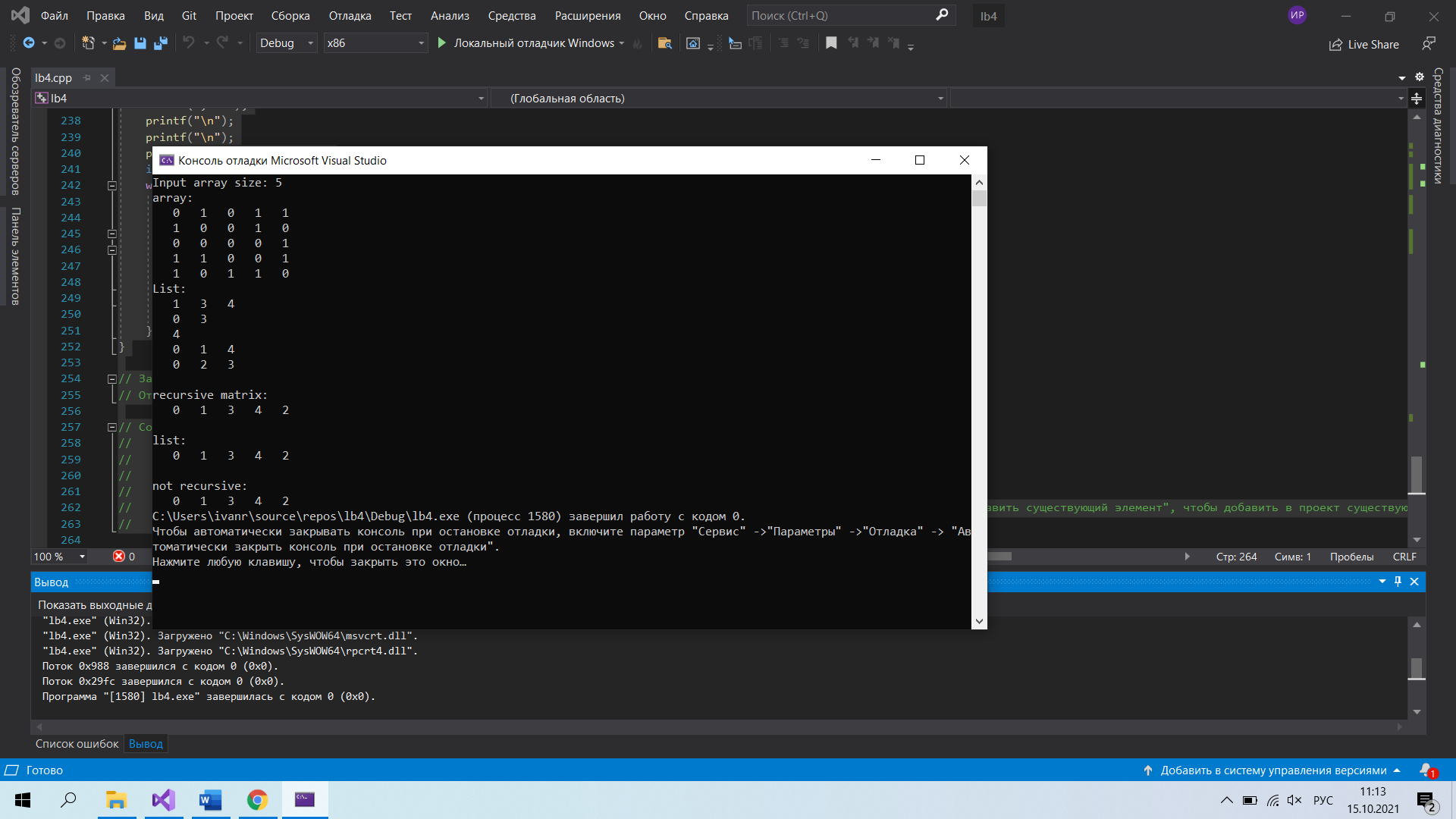
// 3. В окне "Выходные данные" можно просматривать выходные данные сборки и другие сообщения.

// 4. В окне "Список ошибок" можно просматривать ошибки.

// 5. Последовательно выберите пункты меню "Проект" > "Добавить новый элемент", чтобы создать файлы кода, или "Проект" > "Добавить существующий элемент", чтобы добавить в проект существующие файлы кода.

// 6. Чтобы снова открыть этот проект позже, выберите пункты меню "Файл" > "Открыть" > "Проект" и выберите SLN-файл.

**Результаты работы программы:**



**Вывод:** научились реализовывать обход в глубину с матрицами и списками, рекурсивной и не рекурсивной функцией