Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Л и ОА в ИЗ»

на тему «Унарные и бинарные операции над графами»

**Выполнили:**

студентом группы 19ВВ3

Земляков В.Д.

**Принял:**

Митрохин М.А.

Пенза 2020

**Цель работы:**

Изучить унарные и бинарные операции над графами.

**Ход работы:**

**Листинг:**

**Header.h:**

#pragma once

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <conio.h>

#include<locale.h>

#include <String.h>

#include <time.h>

void transfer(int\*\* Array, int N);

**lab3.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Header.h"

void output(int\*\* Array, int N) {

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

{

for (int j = 0; j < N ; j++)

{

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

}

printf("\n");

}

}

void Gen(int\*\* Array, int N) {

printf("\n");

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

{

for (int j = 0; j < N; j++)

{

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

Array[j][i] = Array[i][j];

if (i == j) {

Array[i][j] = 0;

}

}

}

output(Array, N);

}

int identification(int \*\*Array,int N,int u,int v) {

int check = 0;

int M = N;

u--;

v--;

if ((u<0)||(u>N)||(v<0)||(v>N)) {

printf("Вершины не существует");

}

int index = u;

if (v > u) {

index = v;

}

int\*\* Temp = NULL;

Temp = (int\*\*)malloc(M \* sizeof(int\*));

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

{

Temp[i] = (int\*)malloc(M \* sizeof(int));

}

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

Array[i][u] = Array[i][v] || Array[i][u];

Array[u][i] = Array[v][i] || Array[u][i];

}

for (int i = 0; i < N-1; i++)

{

for (int j = 0; j < N-1; j++)

{

if (j>=index && i<index) {

Temp[i][j] = Array[i][j+1];

}

else if (j >= index && i >= index) {

Temp[i][j] = Array[i+1][j + 1];

}

else if (j < index && i >= index) {

Temp[i][j] = Array[i+1][j];

}

else {

Temp[i][j] = Array[i][j];

}

}

}

output(Temp,N-1);

return \*\*Temp;

}

void contraction(int N, int\*\* Array, int u, int v) {

u--;

v--;

int M = N;

int\*\* Temp = NULL;

Temp = (int\*\*)malloc(M \* sizeof(int\*));

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

{

Temp[i] = (int\*)malloc(M \* sizeof(int));

}

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

{

for (int j = 0; j < M; j++)

{

Temp[i][j] = Array[i][j];

}

}

if (Temp[u][v] == 0) {

printf("Ребра не существует\n");

return;

}

else if ((u < 0) || (u > N) || (v < 0) || (v > N)) {

printf("Вершины не существует\n");

return;

}

else {

identification(Temp, M, u + 1, v + 1);

}

}

int split(int\*\* Array,int N,int M,int u) {

u--;

int\*\* Temp = NULL;

Temp = (int\*\*)malloc(M \* sizeof(int\*));

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

{

Temp[i] = (int\*)malloc(M \* sizeof(int));

}

for (int i = 0; i < N+1;i++) {

for (int j = 0; j < N+1; j++) {

Temp[i][j] = 0;

}

}

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

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

Temp[i][j] = Array[i][j];

}

}

for (int i = 0; i < N+1; i++) {

for (int j = 0; j < N+1; j++) {

if (j<N/2 && Temp[u][j] == 1) {

Temp[N][j] = 1;

Temp[u][j] =0;

}

if (i < N / 2 && Temp[i][u] == 1) {

Temp[i][N] = 1;

Temp[i][u] = 0;

}

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

}

printf("\n");

}

return \*\*Temp;

}

int Union(int\*\* Array,int\*\* Array2,int N,int M) {

int max = N;

int min = M;

if (N<M) {

min = N;

max = M;

}

int\*\* Temp = NULL;

Temp = (int\*\*)malloc(max \* sizeof(int\*));

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

{

Temp[i] = (int\*)malloc(max \* sizeof(int));

}

printf("\nОбъеденение\n");

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

{

for (int j = 0; j < max; j++)

{

if (N>M) {

Temp[i][j] = Array[i][j];

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

{

for (int j = 0; j < min; j++)

{

Temp[i][j] = Array[i][j] || Array2[i][j];

}

}

}

else {

Temp[i][j] = Array2[i][j];

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

{

for (int j = 0; j < min; j++)

{

Temp[i][j] = Array[i][j] || Array2[i][j];

}

}

}

}

}

output(Temp, max);

return \*\*Temp;

}

int Crossing(int\*\* Array, int\*\* Array2, int N,int M) {

int max = N;

int min = M;

if (N<M) {

min = N;

max = M;

}

int\*\* Temp = NULL;

Temp = (int\*\*)malloc(max \* sizeof(int\*));

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

{

Temp[i] = (int\*)malloc(max \* sizeof(int));

}

printf("\nПересечение\n");

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

{

for (int j = 0; j < max; j++)

{

if (N>M) {

Temp[i][j] = Array[i][j];

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

{

for (int j = 0; j < min; j++)

{

Temp[i][j] = Array[i][j] && Array2[i][j];

}

}

}

else {

Temp[i][j] = Array2[i][j];

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

{

for (int j = 0; j < min; j++)

{

Temp[i][j] = Array[i][j] && Array2[i][j];

}

}

}

}

}

output(Temp, max);

return \*\*Temp;

}

int Circle(int\*\* Array, int\*\* Array2, int N,int M) {

int max = N;

int min = M;

if (N < M) {

min = N;

max = M;

}

int\*\* Temp = NULL;

Temp = (int\*\*)malloc(max \* sizeof(int\*));

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

{

Temp[i] = (int\*)malloc(max \* sizeof(int));

}

printf("\nКольцевая сумма\n");

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

{

for (int j = 0; j < max; j++)

{

if (N > M) {

Temp[i][j] = Array[i][j];

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

{

for (int j = 0; j < min; j++)

{

if ((Array[i][j] == 1 && Array2[i][j] == 0) || (Array[i][j] == 0 && Array2[i][j] == 1)) {

Temp[i][j] = 1;

}

else {

Temp[i][j] = 0;

}

}

}

}

else {

Temp[i][j] = Array2[i][j];

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

{

for (int j = 0; j < min; j++)

{

if ((Array2[i][j] == 1 && Array[i][j] == 0) || (Array2[i][j] == 0 && Array[i][j] == 1)) {

Temp[i][j] = 1;

}

else {

Temp[i][j] = 0;

}

}

}

}

}

}

output(Temp, N);

return \*\*Temp;

}

int composition(int\*\* Array, int\*\* Array2, int N, int M) {

int Z = N \* M;

int\*\* Temp = NULL;

Temp = (int\*\*)malloc(Z \* sizeof(int\*));

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

{

Temp[i] = (int\*)malloc(Z \* sizeof(int));

}

printf("\nДекартово произведение\n");

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

for (int j = 0; j < N; j++, j++){

for (int k = 0; k < N; k++) {

for (int l = 0; l < N; l++) {

if ((i=k)) {

}

}

}

}

}

output(Temp, N);

return \*\*Temp;

}

int composition(int\*\* Array, int\*\* Array2, int N, int M) {

int z1 = -1;

int z2 = -1;

int Z = N \* M;

int\*\* Temp = NULL;

Temp = (int\*\*)malloc(Z \* sizeof(int\*));

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

{

Temp[i] = (int\*)malloc(Z \* sizeof(int));

}

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

{

for (int j = 0; j < Z; j++)

{

Temp[i][j] = 0;

}

printf("\n");

}

printf("\nДекартово произведение\n");

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

z1++;

for (int k = 0; k < N; k++){

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

for (int l = 0; l < N; l++) {

z2++;

if (i=k) {

Temp[z1][z2] = Array[i][j];

}

if (j=l) {

Temp[z1][z2] = Array2[k][l];

}

if (z1 = z2) {

Temp[z1][z2] = 0;

}

}

}

}

}

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

{

for (int j = 0; j < Z; j++)

{

Temp[j][i] = Temp[i][j];

}

printf("\n");

}

output(Temp, Z);

return \*\*Temp;

}

void main() {

srand(time(NULL));

setlocale(LC\_ALL, "RUS");

int\*\* ArrayM1=NULL;

int\*\* ArrayM2 = NULL;

int\*\* ArrayM3 = NULL;

int M = 0;

int N = 0;

int u= 0,v=0;

int x1 = 0, x2 = 0;

//1.1//

printf("Введите размерность матриц:");

scanf("%d", &N);

ArrayM1 = (int\*\*)malloc(N \* sizeof(int\*));

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

{

ArrayM1[i] = (int\*)malloc(N \* sizeof(int));

}

ArrayM3 = (int\*\*)malloc(N \* sizeof(int\*));

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

{

ArrayM3[i] = (int\*)malloc(N \* sizeof(int));

}

Gen(ArrayM1, N);

printf("Введите размерность матриц:");

scanf("%d", &M);

ArrayM2 = (int\*\*)malloc(M \* sizeof(int\*));

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

{

ArrayM2[i] = (int\*)malloc(M \* sizeof(int));

}

Gen(ArrayM2, M);

Union(ArrayM1,ArrayM2,N,M);

Crossing(ArrayM1,ArrayM2,N,M);

Circle(ArrayM1, ArrayM2, N,M);

composition(ArrayM1, ArrayM2, N, M);

ArrayM3 = ArrayM1;

printf("Отождествление:\n");

printf("Введите первую вершину:\n");

scanf("%d", &u);

printf("Введите вторую вершину:\n");

scanf("%d", &v);

identification(ArrayM3,N,u,v);

printf("Стягивание:\n");

printf("Введите первую вершину:\n");

scanf("%d", &u);

printf("Введите вторую вершину:\n");

scanf("%d", &v);

contraction(N,ArrayM1,u,v);

printf("Расщепление:\n");

printf("Введите вершину:\n");

scanf("%d", &u);

int Z = N + 1;

split(ArrayM1,N,Z,u);

transfer(ArrayM2, M);

\_getch();

}

**Graph.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Header.h"

typedef struct Node {

int index;

struct Node\* next;

}LIST;

LIST\* GM[50];

LIST\* create\_node(int name)

{

LIST\* tmp = (LIST\*)malloc(sizeof(LIST));

tmp->index = name;

tmp->next = NULL;

return(tmp);

}

void add\_element\_end(int data, LIST\* head)

{

LIST\* tmp = (LIST\*)malloc(sizeof(LIST));

tmp->index = data;

tmp->next = NULL;

LIST\* p = head;

while (p->next != NULL)

p = p->next;

p->next = tmp;

}

LIST\* remove\_all(LIST\* head)

{

while (head != NULL)

{

LIST\* p = head;

head = head->next;

free(p);

}

return NULL;

}

LIST\* remove\_element(int data, LIST\* head)

{

LIST\* tmp = head, \* p = NULL;

if (head == NULL)

return NULL;

while (tmp && tmp->index != data)

{

p = tmp;

tmp = tmp->next;

}

if (tmp == head)

{

free(head);

return tmp->next;

}

if (!tmp)

return head;

p->next = tmp->next;

free(tmp);

return head;

}

void print(LIST\* head) {

LIST\* v = head;

if (v != NULL) {

printf("%d: ", v->index + 1);

v = v->next;

}

while (v != NULL)

{

printf("%d ", v->index + 1);

v = v->next;

}

}

void transfer(int\*\*Array,int N) {

printf("Список Смежности:\n");

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

Node\* tmp = create\_node(i);

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

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

add\_element\_end(j, tmp);

}

GM[i] = tmp;

}

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

print(GM[i]);

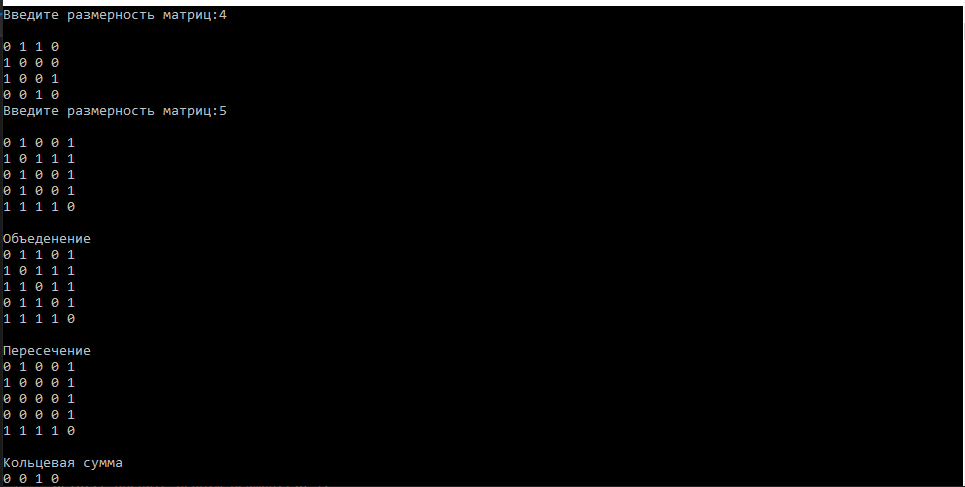
printf("\n");

}

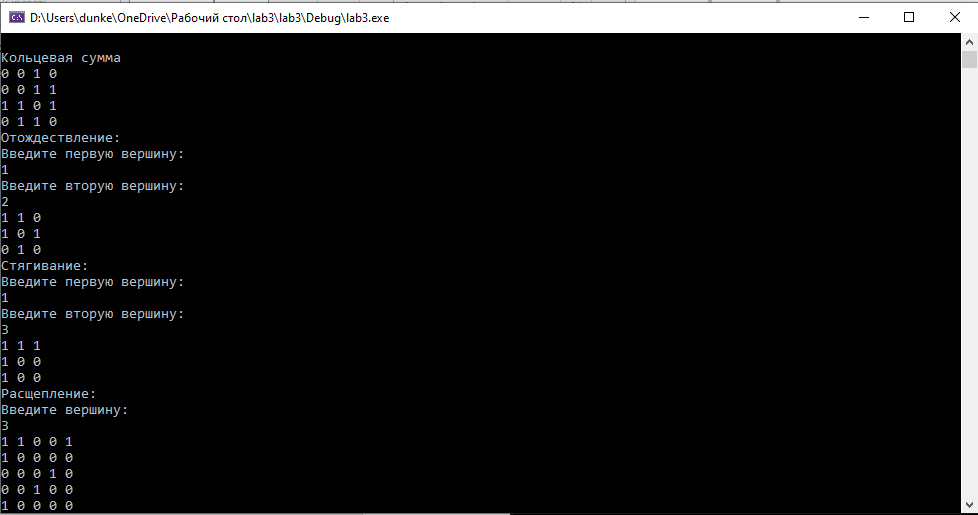
printf("\n");

}

### Результат работы программы:



**Рисунок1. Результат №1**



**Рисунок 2.Результат №2**

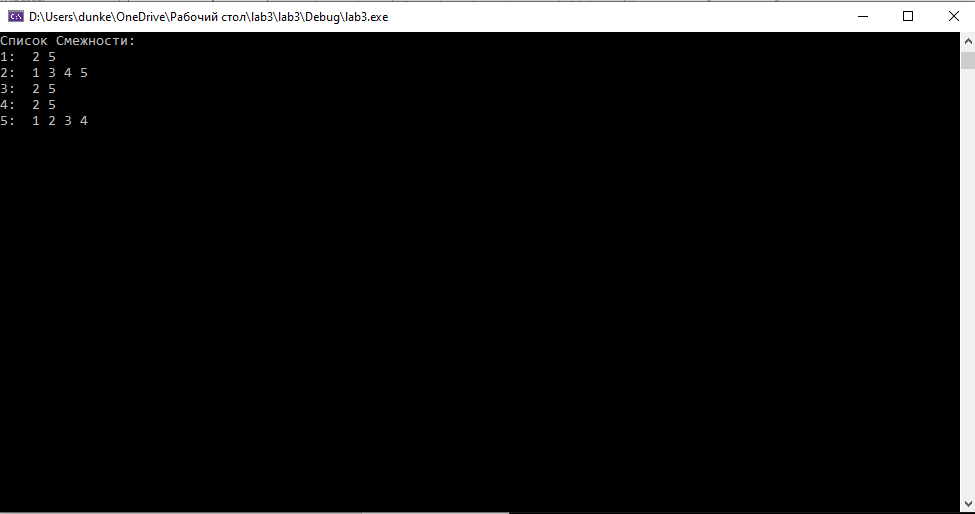


Рисунок 3. Результать №3

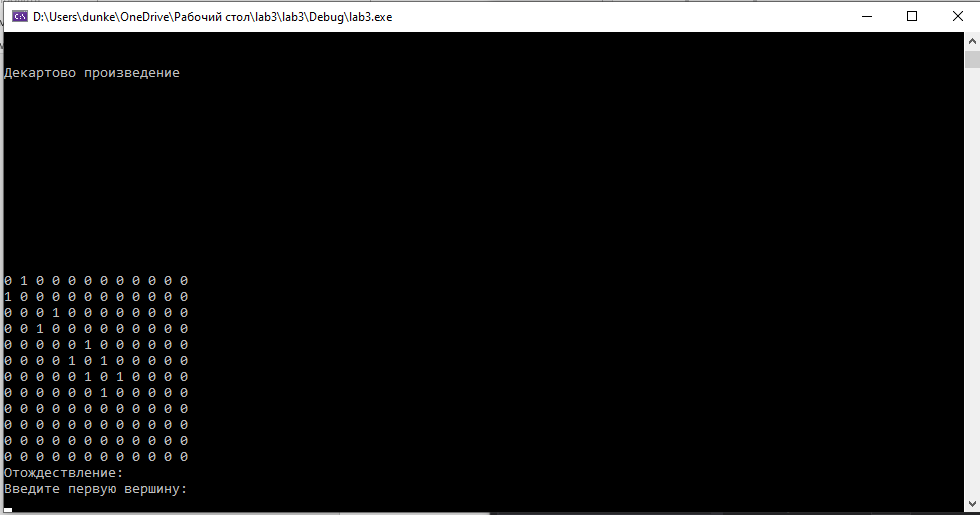


Рисунок 4. Декартово произведение.

### Выводы

В ходе выполнения лабораторной работы изучил унарные и бинарные операции над графами.