Пензенский государственный университет

Кафедра "Вычислительной техники"

**Отчёт**

По лабораторной №3

по дисциплине "Л и ОА в ИТ"

на тему "Унарные и бинарные операции над графами"

***Выполнили студенты группы 19вв1:***

*Григорьева Д. Д.*

*Мереняшева Е. А.*

***Приняли:***

*Митрохин М. А.*

*Юрова О. В.*

Пенза 2020

# Цель работы: выполнить над графами унарные и бинарные операции.

**Лабораторное задание:**

### **Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) две матрицы *M*1*, М*2 смежности неориентированных помеченных графов *G*1, *G*2. Выведите сгенерированные матрицы на экран.

### **Задание 2**

1. Для матричной формы представления графов выполните операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

**Задание 3**

1. Для матричной формы представления графов выполните операцию:

а) объединения *G* = *G*1  *G*2

б) пересечения *G* = *G*1  *G*2

в) кольцевой суммы *G* = *G*1  *G*2

Результат выполнения операции выведите на экран.

**Теоретическая часть:**

Операции над графами делятся на унарные и бинарные.

Унарные

Отождествление вершин. В графе *G*1 выделяются вершины *и,v.* Определяют окружение *Q*1 вершины *u*,и окружение *Q*2 вершины *v,* вычисляют их объединение *Q* = *Q1* * Q2.* Затем над графом *G*1 выполняются следующие преобразования:

* из графа *G*1 удаляют вершины *u,* *v (H*1 *= G*1 *- u - v);*
* к графу *Н*1присоединяют новую вершину *z (H*1 *= H*1 *+z);*
* вершину *z* соединяют ребром с каждой из вершин *w*1*Q*

*(G*2 *= H*1 *+ zwi*, *i =* 1,2,3*,…).*

Стягивание ребра*.* Данная операция является операцией отождествления смежных вершин *и, v* в графе *G*1.

Наиболее важными бинарными операциями являются: объединение, пересечение, декартово произведение и кольцевая сумма.

Бинарные

Объединение. Граф G называется объединением или наложением графов G1 и G2, если VG = V1V2; UG = U1 U2 .

Объединение графов G1 и G2 называется дизъюнктным, если V1V2 = . При дизъюнктном объединении никакие два из объединяемых графов не должны иметь общих вершин.

Пересечение. Граф G называется пересечением графов G1, G2, если VG = V1V2 и UG = U1U2 . Операция "пересечения" записывается следующим образом: G = G1G2.

Декартово произведение. Граф G называется декартовым произведением графов G1 и G2 если VG = V1V2 —декартово произведение множеств вершин графов G1, G2, а множество ребер Uc задается следующим образом: вершины (zi, vk) и (zj, vl) смежны в графе G тогда и только тогда, когда zi = zj(i = j), a vk и vl смежны в G2 или vk = vl(k = l), смежны в графе G1.

Кольцевая сумма графов представляет граф, который не имеет изолированных вершин и состоит из ребер, присутствующих либо в первом исходном графе, либо во втором. Кольцевая сумма определяется следующим соотношением: G = G1  G2.

**Практическая часть:**

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "locale"

#include "stdio.h"

#include "conio.h"

#include <locale.h>

#include <stdlib.h>

#include <malloc.h>

void main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int v1; //Размерность матрицы 1

int v2; //Размерность матрицы 2

int\*\* M1;//Матрица 1

int\*\* M2;//Матрица 2

int\*\* N3;//Увеличенная дополнительная матрица для расщепления

int\*\* N2;//

int\*\* N;//

int\*\* E;

printf("Введите размерность матрицы М1\n");

scanf("%d", &v1);

printf("Введите размерность матрицы М2\n");

scanf("%d", &v2);

int l = 0, k = 0;

int vr = 0;

int q1, q2;

int w = v2 + 1;

M1 = (int\*\*)malloc(v1 \* sizeof(int\*));

M2 = (int\*\*)malloc(v2 \* sizeof(int\*));

N = (int\*\*)malloc(v2 \* sizeof(int\*));

N2 = (int\*\*)malloc(v2 \* sizeof(int\*));

N3 = (int\*\*)malloc(w \* sizeof(int\*));

E = (int\*\*)malloc(50 \* sizeof(int\*));

for (int i = 0; i < 50; i++) // цикл по строкам

{

// Выделение памяти под хранение строк

E[i] = (int\*)malloc(50 \* sizeof(int));

}

for (int i = 0; i < w; i++) // цикл по строкам

{

// Выделение памяти под хранение строк

N3[i] = (int\*)malloc(w \* sizeof(int));

}

for (int i = 0; i < v2; i++) // цикл по строкам

{

// Выделение памяти под хранение строк

N[i] = (int\*)malloc(v2 \* sizeof(int));

}

for (int i = 0; i < v2; i++) // цикл по строкам

{

// Выделение памяти под хранение строк

N2[i] = (int\*)malloc(v2 \* sizeof(int));

}

if ((v2==1) || (v2==0)) {

printf("Матрица не может быть размерностью 0 или 1, там дальше по заданиям \n нужно вводить 2 веришины, а откуда мы их возьмём, если массив 1 или 0\n\n");

};

//----------------------------------------------------------------------

printf("задание 1\n");

for (int i = 0; i < v1; i++) {

M1[i] = (int\*)malloc(v1 \* sizeof(int));

for (int j = 0; j < v1; j++) {

M1[i][j] = rand() % 2;

printf("%5d ", M1[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

for (int i = 0; i < v2; i++) {

M2[i] = (int\*)malloc(v2 \* sizeof(int));

for (int j = 0; j < v2; j++) {

M2[i][j] = rand() % 2;

}

}

for (int i = 0; i < v2; i++) // цикл по строкам

{

for (int j = 0; j < v2; j++) // цикл по столбцам

{

printf("%5d ", M2[i][j]); // 5 знакомест под элемент массива

}

printf("\n");

}

printf("Задание 2\n");

printf("Отождествление вершин\n");

printf("\nВведите две вершины\n");

scanf("%d%d", &q1, &q2);

for (int j = 0; j < v2; j++) {

for (int i = 0; i < v2; i++)

{

N[i][j] = M2[i][j];

vr = M2[i][q1];

if ((N[i][q2] || vr) == 1) {

N[i][q2] = 1;

}

else

{

N[i][q2] = 0;

};

N[i][q1] = 0;

}

}

for (int i = 0; i < v2; i++) {

for (int j = 0; j < v2; j++)

{

vr = M2[q1][j];

if ((N[q2][j] || vr) == 1) {

N[q2][j] = 1;

}

else

{

N[q2][j] = 0;

};

N[q1][j] = 0;

}

}

for (int i = 0; i < v2; i++) {

k = 0;

if (i == q1) {

continue;

};

for (int j = 0; j < v2; j++)

{

if (j != q1) {

N2[l][k] = N[i][j];

k++;

};

}

l++;

}

for (int i = 0; i < v2 - 1; i++) {

for (int j = 0; j < v2 - 1; j++)

{

printf("%5d", N2[i][j]);

}

printf("\n");

}

//----------------------------------------------------------------------

printf("\nСтягивание ребра");

printf("\nВведите 2 вершины\n");

scanf("%d%d", &q1, &q2);

vr = 0;

for (int j = 0; j < v2; j++) {

for (int i = 0; i < v2; i++)

{

N[i][j] = M2[i][j];

vr = M2[i][q1];

if ((N[i][q2] || vr) == 1) {

N[i][q2] = 1;

}

else

{

N[i][q2] = 0;

};

N[i][q1] = 0;

}

}

for (int i = 0; i < v2; i++) {

for (int j = 0; j < v2; j++)

{

vr = M2[q1][j];

if ((N[q2][j] || vr) == 1) {

N[q2][j] = 1;

}

else

{

N[q2][j] = 0;

};

N[q1][j] = 0;

}

}

for (int i = 0; i < v2; i++) {

for (int j = 0; j < v2; j++)

{

if (N[q2][q2] == 1) {

N[q2][q2] = 0;

};

}

}

l = 0;

for (int i = 0; i < v2; i++) {

k = 0;

if (i == q1) {

continue;

};

for (int j = 0; j < v2; j++)

{

if (j != q1) {

N2[l][k] = N[i][j];

k++;

};

}

l++;

}

for (int i = 0; i < v2 - 1; i++) {

for (int j = 0; j < v2 - 1; j++)

{

printf("%5d", N2[i][j]);

}

printf("\n");

}

printf("\n");

//----------------------------------------------------------------------

int vver;

int nov = 0;

printf("\nРасщепление вершины");

printf("\nВведите вершину\n");

scanf("%d", &vver);

for (int j = 0; j < v2; j++) {

for (int i = 0; i < v2; i++)

{

N3[i][j] = M2[i][j];

if (i == v2) {

N3[i][j] = 0;

}

}

}

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

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

{

if (j == v2) {

N3[i][j] = 0;

}

}

}

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

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

{

if (i == v2) {

N3[i][j] = 0;

}

}

}

for (int j = 0; j < v2; j++) {

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

if (j % 2 == 0 && N3[vver][j] == 1)

{

N3[vver][j] = 0;

N3[v2][j] = 1;

}

else {

continue;

}

}

}

for (int i = 0; i < v2; i++) {

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

if (i % 2 == 0 && N3[i][vver] == 1)

{

N3[i][vver] = 0;

N3[i][v2] = 1;

}

else {

continue;

}

}

}

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

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

{

if (i == vver)

{

N3[i][v2] = 1;

}

if (i == v2)

{

N3[i][vver] = 1;

}

}

}

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

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

printf("%5d,", N3[i][j]);

}

printf("\n");

}

//----------------------------------------------------------------------

printf("\n\nЗадание 3. Объединение графов \n");

printf("\nОбъединение графов\n");

int r;

if (v1 >= v2)

{

w = v2;

r = v1;

}

else {

w = v1;

r = v2;

}

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

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

{

if (v2 >= v1)

{

E[i][j] = M2[i][j];

}

else {

E[i][j] = M1[i][j];

}

}

}

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

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

if ((M1[i][j] || M2[i][j]) == 1) {

E[i][j] = 1;

}

else

{

E[i][j] = 0;

}

//printf("%5d", E[i][j]);

}

//printf("\n");

}

printf("\n\n");

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

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

printf("%5d,", E[i][j]);

}

printf("\n");

}

//----------------------------------------------------------------------

printf("\nПересечение графов \n");

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

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

{

E[i][j] = 0;

}

}

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

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

if ((M1[i][j] && M2[i][j]) == 1) {

E[i][j] = 1;

}

else

{

E[i][j] = 0;

}

//printf("%5d", N[i][j]);

}

printf("\n");

}

printf("\n\n");

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

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

printf("%5d,", E[i][j]);

}

printf("\n");

}

//----------------------------------------------------------------------

printf("\nКольцевая сумма графов\n");

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

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

{

if (v2 <= v1)

{

E[i][j] = M1[i][j];

}

else {

E[i][j] = M2[i][j];

}

}

}

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

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

if ((M1[i][j] ^ M2[i][j]) == 1) {

E[i][j] = 1;

}

else

{

E[i][j] = 0;

}

}

printf("\n");

}

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

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

printf("%5d,", E[i][j]);

}

printf("\n");

}

for (int i = 0; i < v2; i++) // цикл по строкам

free(M2[i]); // освобождение памяти под строку

free(M2);

getchar(); getchar();

for (int i = 0; i < v2; i++) // цикл по строкам

free(N2[i]); // освобождение памяти под строку

free(N2);

getchar(); getchar();

for (int i = 0; i < v2; i++) // цикл по строкам

free(N[i]); // освобождение памяти под строку

free(N);

getchar(); getchar();

for (int i = 0; i < w; i++) // цикл по строкам

free(N3[i]); // освобождение памяти под строку

free(N3);

getchar(); getchar();

for (int i = 0; i < r; i++) // цикл по строкам

free(E[i]); // освобождение памяти под строку

free(E);

getchar(); getchar();

//\_getch();

}

**Декартово произведение**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "locale"

#include "stdio.h"

#include "conio.h"

#include <locale.h>

#include <stdlib.h>

#include <malloc.h>

#include <iostream>

void main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int size1;

printf("Введите размерность матрицы 1: ");

scanf("%d", &size1);

int size2;

printf("Введите размерность матрицы 2: ");

scanf("%d", &size2);

int\*\* m1;

m1 = (int\*\*)malloc(size1 \* sizeof(int\*));

for (int i = 0; i < size1; i++) // цикл по строкам

{

m1[i] = (int\*)malloc(size1 \* sizeof(int));

}

for (int i = 0; i < size1; i++) {

for (int j = 0; j < size1; j++) {

m1[i][j] = rand() % 2;

if (i == j) {

m1[j][i] = 0;

}

printf("%3d,", m1[i][j]);

}

printf("\n");

}

printf("\n");

int\*\* m2;

m2 = (int\*\*)malloc(size2 \* sizeof(int\*));

for (int i = 0; i < size2; i++) // цикл по строкам

{

m2[i] = (int\*)malloc(size2 \* sizeof(int));

}

int temp;

for (int i = 0; i < size2; i++)

{

for (int j = 0; j < size2; j++)

{

m2[i][j] = rand() % 2;

if (i == j) {

m2[j][i] = 0;

}

printf("%3d,", m2[i][j]);

}

printf("\n");

}

printf("\n");

int\*\* n;

n = (int\*\*)malloc(size1 \* size2 \* sizeof(int\*));

for (int i = 0; i < size1 \* size2; i++) // цикл по строкам

{

n[i] = (int\*)malloc(size1 \* size2 \* sizeof(int));

}

for (int i = 0; i < size1 \* size2; i++) {

for (int j = 0; j < size1 \* size2; j++) {

n[i][j] = 2;

}

}

for (int i = 0; i < size1; i++) {

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

int ioffset = size2 \* i;

int joffset = size2 \* j;

for (int k = ioffset; k < ioffset + size2; k++) {

for (int l = joffset; l < joffset + size2; l++) {

int val = m1[i][j] == 0 ? m2[k - ioffset][l - joffset] : k - ioffset == l - joffset ? 1 : 0;

n[k][l] = val;

n[l][k] = val;

}

}

}

}

for (int i = 0; i < size1 \* size2; i++) {

for (int j = 0; j < size1 \* size2; j++) {

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

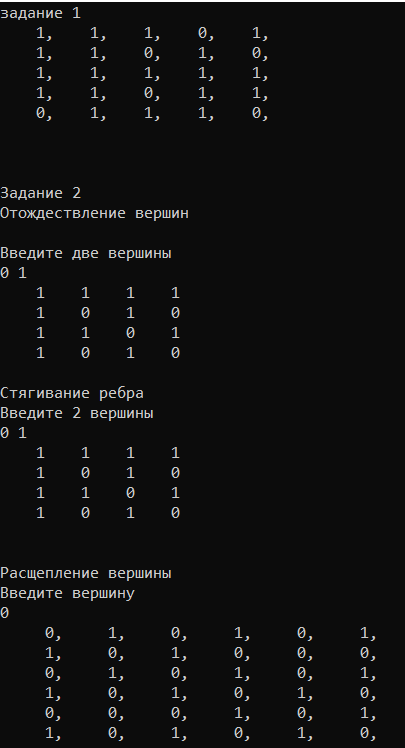
}

printf("\n");

}

}

**Результат работы программы:**



**Вывод:**

1. Выполнили унарные и бинарные операции над графами.
2. Сделали динамический ввод матрицы.
3. Выполнили декартово произведение двух матриц.