Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

# по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Унарные и бинарные операции над графами

**Выполнили:**

студент группы 20ВВ3

Пантюшов Егор

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

Унарные и бинарные операции над графами.

**Цель работы** – изучение алгоритмов реализации унарных и бинарных графов.

# Методические указания

# Все унарные операции над графами можно объединить в две группы. Первую группу составляют операции, с помощью которых из исходного графа G 1, можно построить граф G 2 с меньшим числом элементов. В группу входят операции удаления ребра или вершины, отождествления вершин, стягивание ребра. Вторую группу составляют операции, позволяющие строить графы с большим числом элементов. В группу входят операции расщепления вершин, добавления ребра.

# Лабораторное задание

# Задание 1

# 1. Сгенерируйте (используя генератор случайных чисел) две матрицы M 1 и 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

# Листинг

#define \_CRT\_NONSTDC\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <queue>

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <time.h>

#include <stack>

void task\_1\_1(int\*\* G1, int n1, int\*\* G2, int n2)

{

printf("\nЗадание 1.\n\nПункт 1.\nМатрицы смежности:\n");

printf("Матрица 1:");

printf("\n");

for (int i = 0; i < n1; i++)

{

G1[i] = (int\*)malloc(n1 \* sizeof(int));

for (int j = 0; j < n1; j++)

{

G1[i][j] = rand() % 2;

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (i == j)

{

G1[i][j] = 0;

}

else

{

G1[i][j] = G1[j][i];

}

printf(" %d ", G1[i][j]);

}

printf("\n");

}

printf("\n");

printf("Матрица 2:");

printf("\n");

for (int i = 0; i < n2; i++)

{

G2[i] = (int\*)malloc(n2 \* sizeof(int));

for (int j = 0; j < n2; j++)

{

G2[i][j] = rand() % 2;

}

}

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

if (i == j)

{

G2[i][j] = 0;

}

else

{

G2[i][j] = G2[j][i];

}

printf(" %d ", G2[i][j]);

}

printf("\n");

}

}

int\*\* UT(int\*\* G, int n)

{

int v1, v2;

int a = 0;

int\*\* U = (int\*\*)malloc(n \* sizeof(int\*));

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

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

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

{

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

U[i][j] = G[i][j];

}

printf("Введите номера вершин матрицы:");

scanf("%d", &v1);

scanf("%d", &v2);

if (U[v1][v2] == 1)

a = 1;

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

{

if (U[v2][i] == 1)

{

U[v1][i] = U[v2][i];

U[i][v1] = U[i][v2];

}

}

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

{

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

U[i][j] = U[i + 1][j];

}

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

{

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

U[i][j] = U[i][j + 1];

}

n--;

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

U[i] = (int\*)realloc(U[i], n \* sizeof(int));

free(U[n]);

if (a == 1)

U[v1][v1] = 1;

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

{

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

{

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

}

printf("\n");

}

return(U);

}

int\*\* RAS(int\*\* G, int n)

{

int v;

int\*\* U = (int\*\*)calloc((n + 1), sizeof(int\*));

for (int i = 0; i < (n + 1); i++)

U[i] = (int\*)calloc((n + 1), sizeof(int));

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

{

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

U[i][j] = G[i][j];

}

printf("Введите номер вершины матрицы:");

scanf("%d", &v);

for (int j = 0; j < n / 2; j++)

{

U[j][n] = U[v][j];

U[n][j] = U[j][v];

U[v][j] = 0;

U[j][v] = 0;

}

for (int i = 0; i < (n + 1); i++)

{

for (int j = 0; j < (n + 1); j++)

{

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

}

printf("\n");

}

return(U);

}

int\*\* ST(int\*\* G, int n)

{

int v1, v2;

int\*\* U = (int\*\*)malloc(n \* sizeof(int\*));

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

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

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

{

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

U[i][j] = G[i][j];

}

printf("Введите номера вершин матрицы:");

scanf("%d", &v1);

scanf("%d", &v2);

if (U[v1][v2] != 1) {

printf("Нельзя выполнить операцию!");

return(U);

}

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

{

if (U[v2][i] == 1)

{

U[v1][i] = U[v2][i];

U[i][v1] = U[i][v2];

}

}

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

{

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

U[i][j] = U[i + 1][j];

}

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

{

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

U[i][j] = U[i][j + 1];

}

n--;

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

U[i] = (int\*)realloc(U[i], n \* sizeof(int));

free(U[n]);

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

{

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

{

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

}

printf("\n");

}

return(U);

}

int\*\* OB(int\*\* G1, int n1, int\*\* G2, int n2)

{

if (n2 > n1)

{

int\*\* U = (int\*\*)malloc(n2 \* sizeof(int\*));

for (int i = 0; i < n2; i++)

U[i] = (int\*)malloc(n2 \* sizeof(int));

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

U[i][j] = G2[i][j];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (G1[i][j] == 1)

U[i][j] = G1[i][j];

}

}

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

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

}

printf("\n");

}

return(U);

}

else

{

int\*\* U = (int\*\*)malloc(n1 \* sizeof(int\*));

for (int i = 0; i < n1; i++)

U[i] = (int\*)malloc(n1 \* sizeof(int));

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

U[i][j] = G1[i][j];

}

}

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

if (G2[i][j] == 1)

U[i][j] = G2[i][j];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

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

}

printf("\n");

}

return(U);

}

}

int\*\* PER(int\*\* G1, int n1, int\*\* G2, int n2)

{

if (n2 > n1)

{

int\*\* U = (int\*\*)calloc(n1, sizeof(int\*));

for (int i = 0; i < n1; i++)

U[i] = (int\*)calloc(n1, sizeof(int));

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if ((G1[i][j] == 1) && (G2[i][j] == 1))

U[i][j] = G2[i][j];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

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

}

printf("\n");

}

return(U);

}

else

{

int\*\* U = (int\*\*)calloc(n2, sizeof(int\*));

for (int i = 0; i < n2; i++)

U[i] = (int\*)calloc(n2, sizeof(int));

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

if ((G1[i][j] == 1) && (G2[i][j] == 1))

U[i][j] = G1[i][j];

}

}

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

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

}

printf("\n");

}

return(U);

}

}

int\*\* CS(int\*\* G1, int n1, int\*\* G2, int n2)

{

if (n2 > n1)

{

int\*\* U = (int\*\*)calloc(n2, sizeof(int\*));

for (int i = 0; i < n2; i++)

U[i] = (int\*)calloc(n2, sizeof(int));

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if ((G1[i][j] == 1) && (G2[i][j] == !1))

U[i][j] = G1[i][j];

else

if ((G2[i][j] == 1) && (G1[i][j] == !1))

U[i][j] = G2[i][j];

}

}

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

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

}

printf("\n");

}

return(U);

}

else

{

int\*\* U = (int\*\*)calloc(n1, sizeof(int\*));

for (int i = 0; i < n1; i++)

U[i] = (int\*)calloc(n1, sizeof(int));

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

if ((G1[i][j] == 1) && (G2[i][j] == !1))

U[i][j] = G1[i][j];

else

if ((G2[i][j] == 1) && (G1[i][j] == !1))

U[i][j] = G2[i][j];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

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

}

printf("\n");

}

return(U);

}

}

void task\_menu(int\*\* G1, int n1, int\*\* G2, int n2)

{

printf("1.Отождествление вершин.\n2.Стягивание ребра.\n3.Расщепление вершин.\n4.Объединение графов.\n5.Пересечение графов.\n6.Кольцевая сумма графов.\n7.Выход.\n");

int s, t;

printf("Введите номер операции:");

scanf("%d", &s);

switch (s) {

case 1:

{printf("Введите номер матрицы:");

scanf("%d", &t);

if (t == 1)

UT(G1, n1);

else

UT(G2, n2);

break;

}

case 2:

{printf("Введите номер матрицы:");

scanf("%d", &t);

if (t == 1)

ST(G1, n1);

else

ST(G2, n2);

break; }

case 3:

{ printf("Введите номер матрицы:");

scanf("%d", &t);

if (t == 1)

RAS(G1, n1);

else

RAS(G2, n2);

break; }

case 4:

{

OB(G1, n1, G2, n2);

break;

}

case 5:

{

PER(G1, n1, G2, n2);

break;

}

case 6:

{

CS(G1, n1, G2, n2);

break;

}

case 7:

exit(0);

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

printf("Введите размерность матрицы1: ");

int n1, n2;

scanf("%d", &n1);

printf("Введите размерность матрицы2: ");

scanf("%d", &n2);

int\*\* G1 = (int\*\*)malloc(n1 \* sizeof(int\*));

int\*\* G2 = (int\*\*)malloc(n2 \* sizeof(int\*));

task\_1\_1(G1, n1, G2, n2);

while (true)

task\_menu(G1, n1, G2, n2);

system("PAUSE");

return 0;

}

# Результат работы программы

Задание 1.

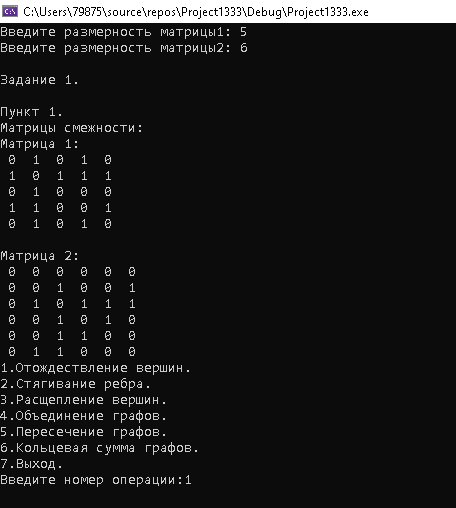


Рисунок 1Задание 1

Задание 2

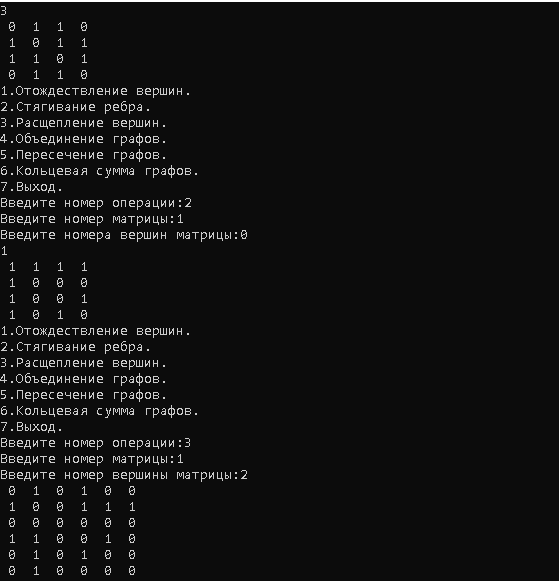
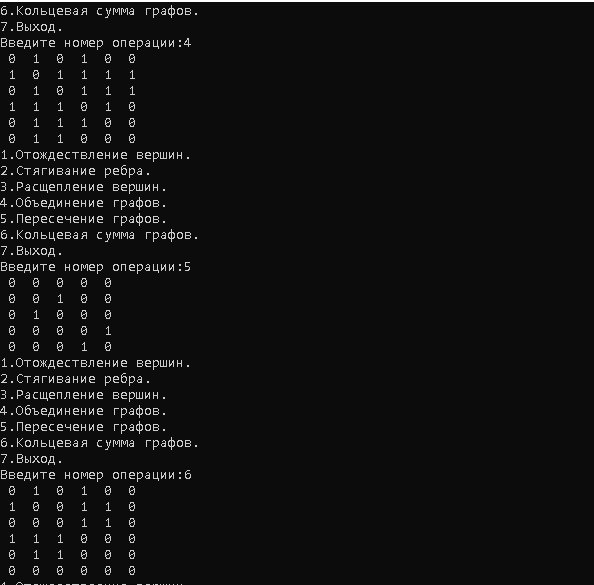


Рисунок 2Задание 2

Задание 3



# Вывод

# В ходе выполнения лабораторной работы былы изучены алгоритмы реализаций бинарных и унарных операций над графами.