Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

## на тему «Унарные и бинарные операции над графами»

Выполнили:

студенты группы 20ВВ2

Киреев Б.П.

Верховский М.В.

Лукин В.Д.

Приняли:

д.т.н. Митрохин М.А.

к.т.н. доцент Юрова О.В.

Пенза 2021

Название: Унарные и бинарные операции над графами.

Цель работы: Научиться осуществлять унарные и бинарные операции над графами для их матричной формы представления.

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 = G1G2

б) пересечения G = G1G2

в) кольцевой суммы G = G1  G2

Результат выполнения операции выведите на экран.

**Задание 1, 3:**

Листинг:

Файл Lab3(1,3).cpp

#include <iostream>

#include <ctime>

#include <malloc.h>

#include <conio.h>

using namespace std;

int n, \*\* Matrix1 = NULL, \*\* Matrix2 = NULL;

int pogr = 0;

int choose = 0, \*\* Matrix3 = NULL;

void Obiedinenie() {

cout << "Объединение матриц 1 и 2:" << endl;

//Объединение

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

for (int m = 0; m < n; m++) {

if (Matrix1[i][m] == 0 && Matrix2[i][m] == 0) {

Matrix3[i][m] = 0;

}

else {

Matrix3[i][m] = 1;

}

}

}

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

for (int m = 0; m < n; m++) {

cout << Matrix3[i][m] << " ";

}

cout << endl;

}

}

void Peresechenie() {

cout << "Пересечение матриц 1 и 2:" << endl;

//Пересечение

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

for (int m = 0; m < n; m++) {

if (Matrix1[i][m] == 1 && Matrix2[i][m] == 1) {

Matrix3[i][m] = 1;

}

else {

Matrix3[i][m] = 0;

}

}

}

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

for (int m = 0; m < n; m++) {

cout << Matrix3[i][m] << " ";

}

cout << endl;

}

}

void ShapeSumm() {

cout << "Кольцевая сумма матриц 1 и 2:" << endl;

//Кольцевая сумма

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

for (int m = 0; m < n; m++) {

if ((Matrix1[i][m] == 0 && Matrix2[i][m] == 0) || (Matrix1[i][m] == 1 && Matrix2[i][m] == 1)) {

Matrix3[i][m] = 0;

}

else {

Matrix3[i][m] = 1;

}

}

}

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

for (int m = 0; m < n; m++) {

cout << Matrix3[i][m] << " ";

}

cout << endl;

}

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(0));

//Задание 1//

cout << "Введите размерность матриц: ";

cin >> n;

if (n <= 0) { cout << "\nНекорректный ввод. Повторите попытку" << endl; return 0; }

Matrix1 = (int\*\*)calloc(n, 3);

Matrix2 = (int\*\*)calloc(n, 3);

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

Matrix1[j] = (int\*)calloc(n, 3);

Matrix2[j] = (int\*)calloc(n, 3);

}

cout.precision(3 \* n);

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

for (int m = 0 + pogr; m < n; m++) {

Matrix1[i][m] = 0 + rand() % 2;

Matrix1[m][i] = Matrix1[i][m];

Matrix1[i][i] = 0;

Matrix2[i][m] = 0 + rand() % 2;

Matrix2[m][i] = Matrix2[i][m];

Matrix2[i][i] = 0;

}

pogr++;

}

//Задание 3//

while (choose != 4) {

system("cls");

cout << "\nМатрица №1:" << endl;

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

for (int m = 0; m < n; m++) {

cout << Matrix1[i][m] << " ";

}

cout << endl;

}

cout << "\nМатрица №2:" << endl;

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

for (int m = 0; m < n; m++) {

cout << Matrix2[i][m] << " ";

}

cout << endl;

}

cout << "\n1.Объединение" << endl;

cout << "2.Пересечение" << endl;

cout << "3.Кольцевая сумма" << endl;

cout << "4.Выход\n" << endl;

cout << "Выберите операцию для выполнения: ";

cin >> choose;

cout << endl;

Matrix3 = (int\*\*)calloc(n, 3);

for (int j = 0; j < n; j++) { Matrix3[j] = (int\*)calloc(n, 3); }

switch (choose) {

case 1://Объединение

Obiedinenie();

\_getch();

break;

case 2://Пересечение

Peresechenie();

\_getch();

break;

case 3: //Кольцевая сумма

ShapeSumm();

\_getch();

break;

case 4:

cout << "\nВыход из программы" << endl;

break;

default:

cout << "\nНекорректный ввод. Повторите попытку." << endl;

break;

}

}

return 0;

}

Пояснительный текст к программе:

Данная программа осуществляет: генерацию двух матриц смежности неориентированных графов (Рис.№1); для матричной формы графов выполняет операцию объединения (Рис.№2), пересечения (Рис.№3), кольцевой суммы (Рис.№4).

Результаты работы программы:

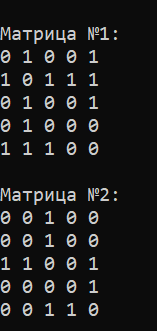


Рисунок №1 – Работа программы из задания №1.

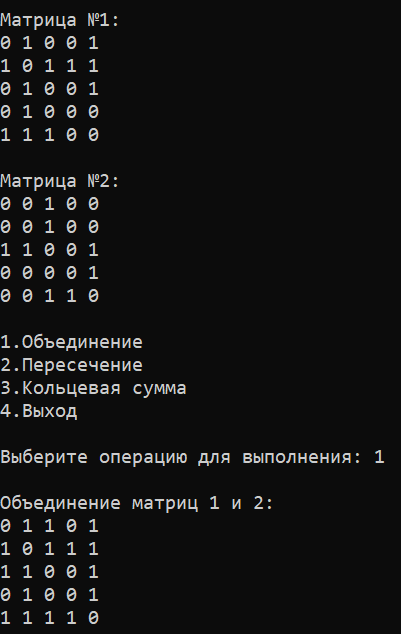


Рисунок №2 – Работа программы из задания №3 (объединение).

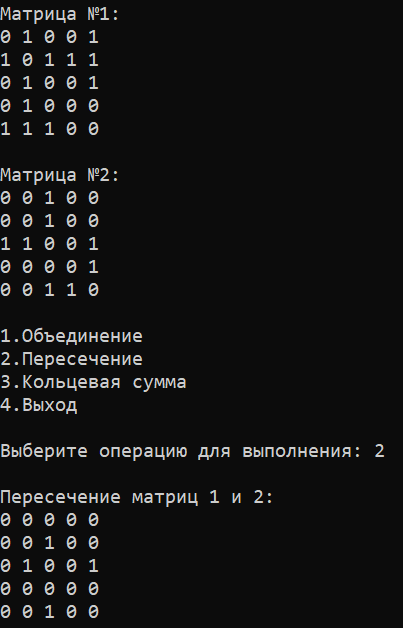


Рисунок №3 – Работа программы из задания №3 (пересечение).

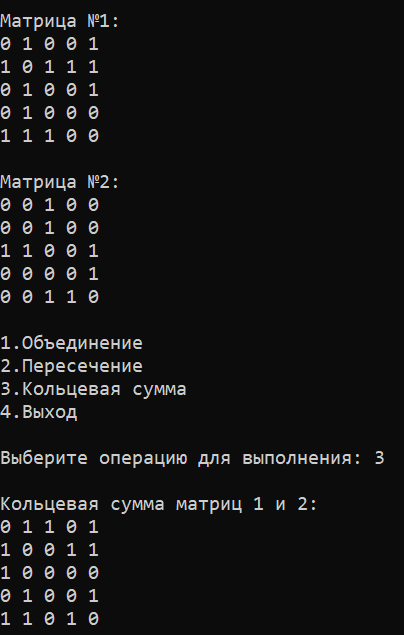


Рисунок №4 – Работа программы из задания №3 (кольцевая сумма).

**Задание 2:**

Листинг:

Файл Lab3(2).cpp

#include <iostream>

#include <ctime>

#include <malloc.h>

#include <conio.h>

#include <string.h>

using namespace std;

int n, \*\* Matrix1 = NULL;

int pogr = 0, dopo = 0;

int choose = 0, \*\* Matrix3 = NULL, usrentr = 0, usrentr2 = 0;

void Otogdestvlenie() {

if (n == 0) { return; } //Нельзя отождествить матрицу 1 на 1

Matrix3 = (int\*\*)calloc(n, 3);

pogr = 0;

for (int j = 0; j < n; j++) { Matrix3[j] = (int\*)calloc(n, 3); }

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

for (int m = 0; m < n; m++) { Matrix3[i][m] = 2; Matrix3[m][m] = 0; }

}

cout << "Введите номера отождествляемых вершин: \n";

cout << "Первая: ";

cin >> usrentr;

cout << "Вторая: ";

cin >> usrentr2;

usrentr--;

usrentr2--;

n++; //Возвращаем n до нормального(большего на 1) значения матрицы

for (int j = 0; j < n; j++) { Matrix1[usrentr][j] = Matrix1[usrentr2][j]; Matrix1[j][usrentr] = Matrix1[usrentr2][j]; } //Цикл для переноски значений в нужную вершину

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

for (int m = 0 + pogr; m < n; m++) {

if (i == m || i == usrentr2 || m == usrentr2) {}

else if (m > usrentr2 && i < usrentr2) { Matrix3[i][m - 1] = Matrix1[i][m]; Matrix3[m - 1][i] = Matrix1[m][i]; }

else if (m > usrentr2 && i > usrentr2) { Matrix3[i - 1][m - 1] = Matrix1[i][m]; Matrix3[m - 1][i - 1] = Matrix1[i][m]; }

else { Matrix3[i][m] = Matrix1[i][m]; Matrix3[m][i] = Matrix1[m][i]; }

}

pogr++;

}

cout << endl;

n--; //Уменьшаем n для взаимодействия со старой матрицей(на n-1 и n-1)

Matrix1 = (int\*\*)calloc(n, 3);

for (int j = 0; j < n; j++) { Matrix1[j] = (int\*)calloc(n, 3); }

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

for (int m = 0; m < n; m++) {

Matrix1[i][m] = Matrix3[i][m];

Matrix1[m][m] = 0;

cout << Matrix1[i][m] << " ";

}

cout << endl;

}

}

void Stuagivanie() {

if (n == 0) { return; } //Нельзя отождествить матрицу 1 на 1

Matrix3 = (int\*\*)calloc(n, 3);

pogr = 0;

for (int j = 0; j < n; j++) { Matrix3[j] = (int\*)calloc(n, 3); }

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

for (int m = 0; m < n; m++) { Matrix3[i][m] = 2; Matrix3[m][m] = 0; }

cout << endl;

}

cout << "Введите номера стягиваемых вершин: \n";

cout << "Первая: ";

cin >> usrentr;

cout << "Вторая: ";

cin >> usrentr2;

usrentr--;

usrentr2--;

n++; //Возвращаем n до нормального(большего на 1) значения матрицы

for (int j = 0; j < n; j++) { Matrix1[usrentr][j] = Matrix1[usrentr2][j]; Matrix1[j][usrentr] = Matrix1[usrentr2][j]; } //Цикл для переноски значений в нужную вершину

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

for (int m = 0 + pogr; m < n; m++) {

if (i == m || i == usrentr2 || m == usrentr2) {}

else if (m > usrentr2 && i < usrentr2) { Matrix3[i][m - 1] = Matrix1[i][m]; Matrix3[m - 1][i] = Matrix1[m][i]; }

else if (m > usrentr2 && i > usrentr2) { Matrix3[i - 1][m - 1] = Matrix1[i][m]; Matrix3[m - 1][i - 1] = Matrix1[i][m]; }

else { Matrix3[i][m] = Matrix1[i][m]; Matrix3[m][i] = Matrix1[m][i]; }

}

pogr++;

}

cout << endl;

n--; //Уменьшаем n для взаимодействия со старой матрицей(на n-1 и n-1)

Matrix1 = (int\*\*)calloc(n, 3);

for (int j = 0; j < n; j++) { Matrix1[j] = (int\*)calloc(n, 3); }

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

for (int m = 0; m < n; m++) {

Matrix1[i][m] = Matrix3[i][m];

Matrix1[m][m] = 0;

cout << Matrix1[i][m] << " ";

}

cout << endl;

}

}

void Raschep() {

Matrix3 = (int\*\*)calloc(n, 3);

for (int j = 0; j < n; j++) { Matrix3[j] = (int\*)calloc(n, 3); }

//Расширяем матрицу на одну строку и столбец

cout << "Введите номер вершины, которую хотите расщепить: ";

cin >> usrentr;

usrentr--;

n--;//Уменьшаем n для взаимодействия со старой матрицей(на n-1 и n-1)

if (usrentr > n || usrentr < 0) { cout << "\nНекорректный ввод. Повторите попытку." << endl; return; }

//Ограничиваем ввод пользователя

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

for (int m = 0; m < n; m++) {

if (i == usrentr) { Matrix3[n][m] = Matrix1[i][m]; Matrix3[m][n] = Matrix3[n][m]; }

else { Matrix3[i][m] = Matrix1[i][m]; Matrix3[m][i] = Matrix1[m][i]; }

}

}

Matrix3[usrentr][n] = 1;

Matrix3[n][usrentr] = 1;

//У расщепляемых вершин должно быть соединяющее ребро

n++; //Возвращаем n до нормального(большего на 1) значения матрицы

Matrix1 = (int\*\*)calloc(n, 3);

for (int j = 0; j < n; j++) { Matrix1[j] = (int\*)calloc(n, 3); }

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

for (int m = 0; m < n; m++) {

Matrix1[i][m] = Matrix3[i][m];

Matrix1[m][m] = 0;

cout << Matrix1[i][m] << " ";

}

cout << endl;

}

cout << "\nВершина " << usrentr + 1 << " была расщеплена в вершину " << n << endl;

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(0));

//Задание 1//

cout << "Введите размерность матрицы: ";

cin >> n;

if (n <= 0) { cout << "\nНекорректный ввод. Повторите попытку.\n"; return 0; } //Ограничитель на ввод

Matrix1 = (int\*\*)calloc(n, 3);

for (int j = 0; j < n; j++) { Matrix1[j] = (int\*)calloc(n, 3); }

cout.precision(3 \* n);

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

for (int m = 0 + pogr; m < n; m++) {

Matrix1[i][m] = 0 + rand() % 2;

Matrix1[m][i] = Matrix1[i][m];

Matrix1[i][i] = 0;

}

pogr++;

}

//Задание 2//

while (choose != 4) {

system("cls");

cout << "\nМатрица №1:" << endl;

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

for (int m = 0; m < n; m++) { cout << Matrix1[i][m] << " "; }

cout << endl;

}

cout << "\n1.Отождествить вершины" << endl;

cout << "2.Стянуть рёбра" << endl;

cout << "3.Расщепление вершины" << endl;

cout << "4.Выход\n" << endl;

cout << "\nВыберите операцию для выполнения: ";

cin >> choose;

switch (choose) {

case 1://Отождествление вершин

n--;

Otogdestvlenie();

\_getch();

break;

case 2://Стягивание ребра

n--;

Stuagivanie();

\_getch();

break;

case 3: //Расщепление вершины

n++;

Raschep();

\_getch();

break;

case 4:

cout << "\nВыход из программы." << endl;

break;

default:

cout << "\nНекорректный ввод. Повторите попытку." << endl;

break;

}

}

return 0;

}

Пояснительный текст к программе:

Данная программа для матричной формы графов выполняет операцию отождествления вершин (Рис.№5), стягивания ребра (Рис.№6), расщепления вершины (Рис.№7).

Результаты работы программы:

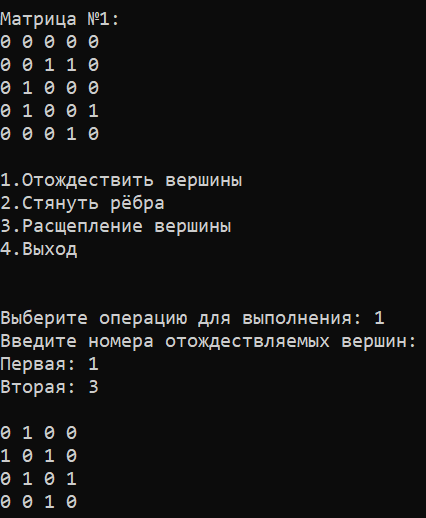


Рисунок №5 – Работа программы из задания №2 (отождествление вершин).

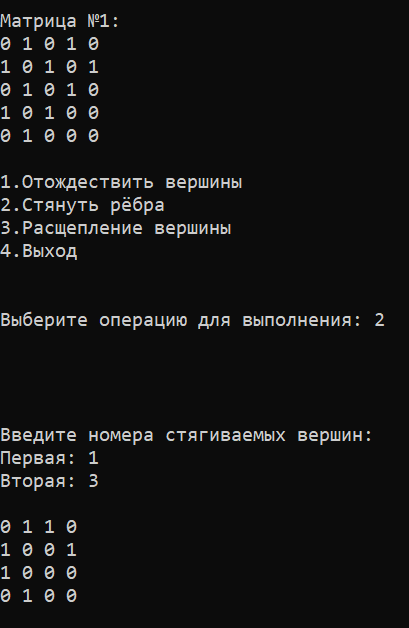


Рисунок №6 – Работа программы из задания №2 (стягивание ребра).

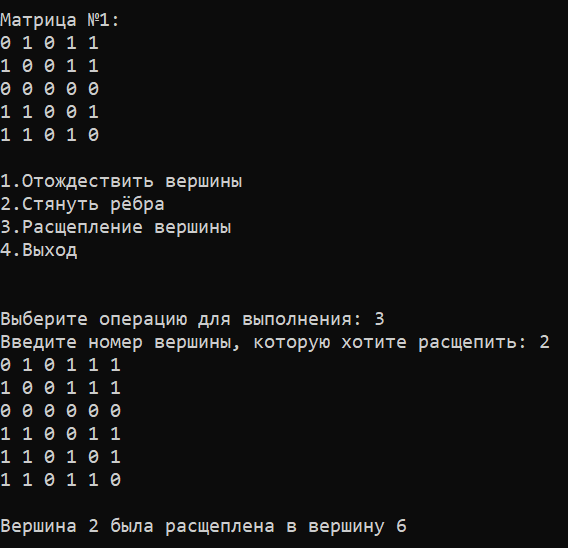


Рисунок №7 – Работа программы из задания №2 (расщепления вершины).

Вывод: При выполнении данной лабораторной работы мы смогли научиться осуществлять унарные и бинарные операции над графами для их матричной формы представления.