Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в ИС»

на тему «Унарные и бинарные операции над графами»

Выполнили

студенты группы 20ВВ2:

Тумасов Вадим

Портнов Никита

Принял:

д.т.н Митрохин М.А.

д.т.н Юрова О.В.

Пенза 2021

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.
2. \* Для указанных графов преобразуйте представление матриц смежности в списки смежности. Выведите полученные списки 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.

1. \* Для представления графов в виде списков смежности 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.

**Задание 4 \***

1. Для матричной формы представления графов выполните операцию декартова произведения графов *G = G*1X *G*2.

Результат выполнения операции выведите на экран.

**Листинг:**

**#include <iostream>**

**#include <list>**

**#include <utility>**

**#include <stdio.h>**

**#include <random>**

**#include <time.h>**

**#include <locale.h>**

**#include <windows.h>**

**#include <string.h>**

**using namespace std;**

**void dekartovo(int\*\* mass,int\*\* mass\_p2,int n,int n1) {**

**//int M[2] = { 1,2 };**

**//int J[3] = { 0,1,2 };**

**//vector<pair<int, int>> C;**

**cout << "Первая матрица";**

**cout << endl;**

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

**{**

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

**{**

**cout << mass[i][j]<<" ";**

**}**

**printf("\n");**

**}**

**cout << endl;**

**cout << "Вторая матрица";**

**cout << endl;**

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

**{**

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

**{**

**cout << mass\_p2[i][j] << " ";**

**}**

**printf("\n");**

**}**

**cout << endl;**

**int nomer1, nomer2,z,r;**

**nomer1 = 0;**

**nomer2 = 0;**

**z = 0;**

**r = 0;**

**int\*\* mass\_dek = (int\*\*)malloc(sizeof(int) \* n \* n1);**

**for (int i = 0; i < n \* n1; i++) {**

**mass\_dek[i] = (int\*)malloc(sizeof(int) \* n \* n1);**

**}**

**for (int i = 0; i < n \* n1; i++)**

**{**

**if (z == n1)**

**{**

**z = 0;**

**nomer1++;**

**}**

**r = 0;**

**nomer2 = 0;**

**for (int j = 0; j < n \* n1; j++)**

**{**

**if (r == n1)**

**{**

**r = 0;**

**nomer2++;**

**}**

**if (nomer1 == nomer2)**

**{**

**mass\_dek[i][j] = mass\_p2[z][r];**

**}**

**else**

**{**

**if (z == r)**

**{**

**mass\_dek[i][j] = mass[nomer1][nomer2];**

**}**

**else**

**{**

**mass\_dek[i][j] = 0;**

**}**

**}**

**//mass\_dek[i][j] = mass[nomer1][nomer2] \* mass\_p2[z][r];**

**r++;**

**}**

**z++;**

**}**

**for (int i = 0; i < n \* n1; i++)**

**{**

**for (int j = 0; j < n \* n1; j++)**

**{**

**cout << mass\_dek[i][j] << " ";**

**}**

**printf("\n");**

**}**

**}**

**using namespace std;**

**void cpy(int\*\* mass1, int\*\* mass2, int n) {**

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

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

**mass1[i][j] = mass2[i][j];**

**}**

**}**

**}**

**void generate(int\*\* mass, int n) { // заполнение массивов**

**int i, r, c;**

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

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

**r = rand() % 2;**

**c = rand() % 2;**

**if (r == 1) {**

**mass[i][j] = 1;**

**}**

**else**

**mass[i][j] = 0;**

**if (i == j) {**

**mass[i][j] = 0;**

**}**

**}**

**}**

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

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

**if (mass[i][j] != mass[j][i]) {**

**mass[i][j] = mass[j][i];**

**}**

**}**

**}**

**}**

**void otoz\_lst(int s, int s1, list<list<int>>\* spiski, bool print\_fl) { //отождествление**

**int i, j;**

**list<list<int>>::iterator spiski\_iter\_loc1 = spiski->begin();**

**list<list<int>>::iterator spiski\_iter\_loc2 = spiski->begin();**

**list<int> spisok\_loc1, spisok\_loc2;**

**list<int>::iterator spisok\_iter\_loc1, spisok\_iter\_loc2;**

**bool insert\_versh = false;**

**bool del\_versh = false;**

**for (i = 0; i < s; i++)**

**{**

**spiski\_iter\_loc1++;**

**}**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**for (i = 0; i < s1; i++)**

**{**

**spiski\_iter\_loc2++;**

**}**

**spisok\_iter\_loc2 = (\*spiski\_iter\_loc2).begin();**

**while (spisok\_iter\_loc2 != (\*spiski\_iter\_loc2).end())**

**{**

**insert\_versh = false;**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**if ((\*spisok\_iter\_loc1) == (\*spisok\_iter\_loc2))**

**{**

**insert\_versh = true;**

**}**

**spisok\_iter\_loc1++;**

**}**

**if (!insert\_versh)**

**(\*spiski\_iter\_loc1).push\_back((\*spisok\_iter\_loc2));**

**spisok\_iter\_loc2++;**

**}**

**spiski\_iter\_loc2 = spiski->erase(spiski\_iter\_loc2);**

**spiski\_iter\_loc1 = spiski->begin();**

**i = 0;**

**while (spiski\_iter\_loc1 != spiski->end())**

**{**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**if ((\*spisok\_iter\_loc1) == s1)**

**{**

**if (i == s)**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).erase(spisok\_iter\_loc1);**

**else**

**(\*spisok\_iter\_loc1) = s;**

**}**

**else**

**{**

**if ((\*spisok\_iter\_loc1) > s1)**

**{**

**(\*spisok\_iter\_loc1) = (\*spisok\_iter\_loc1) - 1;**

**}**

**}**

**if (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**spisok\_iter\_loc1++;**

**}**

**(\*spiski\_iter\_loc1).sort();**

**(\*spiski\_iter\_loc1).unique();**

**spiski\_iter\_loc1++;**

**i++;**

**}**

**spiski\_iter\_loc1 = spiski->begin();**

**if (print\_fl)**

**{**

**printf("Результат отождествления вершин 1ых списков смежности:\n");**

**while (spiski\_iter\_loc1 != spiski->end())**

**{**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**printf("{ ");**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**printf("%d ", \*spisok\_iter\_loc1);**

**spisok\_iter\_loc1++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc1++;**

**}**

**}**

**}**

**void otoz(int s, int s1, int\*\* mass, int n) { //отождествление**

**int i, j;**

**printf("Результат отождествления вершин 1й матрицы:\n");**

**for (i = 0; i < n; i++) { //заносим 1ю матрицу**

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

**if (mass[s][j] > mass[s1][j])**

**mass[s1][j] = 1;**

**if (mass[i][s1] > mass[s1][j])**

**mass[i][s1] = 1;**

**}**

**}**

**int l = 0, g = 0;**

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

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

**if (mass[s][j] == 1)**

**g = 1;**

**if (mass[i][s] == 1)**

**l = 1;**

**mass[s][j] = 0;**

**if (g == 1)**

**mass[s1][j] = g;**

**mass[i][s] = 0;**

**if (l == 1)**

**mass[i][s1] = l;**

**g = 0;**

**l = 0;**

**mass[s1][s1] = 0;**

**mass[s][s] = 0;**

**}**

**}**

**//выполняем отождествление и вывод матрицы**

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

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

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

**if (i == s || j == s) {**

**continue;**

**}**

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

**}**

**if (i != s) {**

**printf("\n");**

**}**

**}**

**}**

**void styag\_lst(int s, int s1, list<list<int>>\* spiski) { //стягивание**

**int i, j;**

**list<list<int>>::iterator spiski\_iter\_loc1 = spiski->begin();**

**list<list<int>>::iterator spiski\_iter\_loc2;**

**list<int> spisok\_loc1, spisok\_loc2;**

**list<int>::iterator spisok\_iter\_loc1, spisok\_iter\_loc2;**

**bool del\_versh = false;**

**bool s\_check = false;**

**bool s1\_check = false;**

**bool smezh\_versh = false;**

**for (i = 0; i < s; i++)**

**{**

**spiski\_iter\_loc1++;**

**}**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**spiski\_iter\_loc2 = spiski->begin();**

**for (i = 0; i < s1; i++)**

**{**

**spiski\_iter\_loc2++;**

**}**

**spisok\_iter\_loc2 = (\*spiski\_iter\_loc2).begin();**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**if ((\*spisok\_iter\_loc1) == s1)**

**s1\_check = true;**

**spisok\_iter\_loc1++;**

**}**

**while (spisok\_iter\_loc2 != (\*spiski\_iter\_loc2).end())**

**{**

**if ((\*spisok\_iter\_loc2) == s)**

**s\_check = true;**

**spisok\_iter\_loc2++;**

**}**

**smezh\_versh = s1\_check && s\_check;**

**if (smezh\_versh)**

**{**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**spisok\_iter\_loc2 = (\*spiski\_iter\_loc2).begin();**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**if ((\*spisok\_iter\_loc1) == s1)**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).erase(spisok\_iter\_loc1);**

**if (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**spisok\_iter\_loc1++;**

**}**

**while (spisok\_iter\_loc2 != (\*spiski\_iter\_loc2).end())**

**{**

**if ((\*spisok\_iter\_loc2) == s)**

**spisok\_iter\_loc2 = (\*spiski\_iter\_loc2).erase(spisok\_iter\_loc2);**

**if (spisok\_iter\_loc2 != (\*spiski\_iter\_loc2).end())**

**spisok\_iter\_loc2++;**

**}**

**otoz\_lst(s, s1, spiski, false);**

**printf("\nРезультат стягивания вершин 1ых списков смежности:\n");**

**spiski\_iter\_loc1 = spiski->begin();**

**while (spiski\_iter\_loc1 != spiski->end())**

**{**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**printf("{ ");**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**printf("%d ", \*spisok\_iter\_loc1);**

**spisok\_iter\_loc1++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc1++;**

**}**

**}**

**}**

**void styag(int s, int s1, int\*\* mass1, int n) { //стягивание**

**int g = 0, l = 0;**

**printf("Результат стягивания ребра 1й матрицы:\n");**

**if (mass1[s][s1]==1)**

**{**

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

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

**if (mass1[s][j] == 1)**

**g = 1;**

**if (mass1[i][s] == 1)**

**l = 1;**

**mass1[s][j] = 0;**

**if (g == 1)**

**mass1[s1][j] = g;**

**mass1[i][s] = 0;**

**if (l == 1)**

**mass1[i][s1] = l;**

**g = 0;**

**l = 0;**

**mass1[s1][s1] = 0;**

**}**

**}**

**for (int i = 0; i < n; i++) { //выполняем стягивание и вывод матрицы**

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

**if (i == s || j == s) {**

**continue;**

**}**

**printf("%d ", mass1[i][j]);**

**}**

**if (i != s) {**

**printf("\n");**

**}**

**}**

**}**

**}**

**void ras\_lst(int s, list<list<int>>\* spiski) { //расщепление**

**int i, j, size\_lst;**

**list<list<int>>::iterator spiski\_iter\_loc1 = spiski->begin();**

**list<list<int>>::iterator spiski\_iter\_loc2 = spiski->begin();**

**list<int> spisok\_insert;**

**list<int>::iterator spisok\_iter\_loc1, spisok\_iter\_loc2;**

**size\_lst = spiski->size();**

**printf("Результат расщепления вершин 1ых списков смежности:\n");**

**for (i = 0; i < s; i++)**

**{**

**spiski\_iter\_loc1++;**

**}**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**(\*spiski\_iter\_loc1).push\_back(size\_lst);**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**if ((\*spisok\_iter\_loc1) < (size\_lst / 2))**

**{**

**spisok\_insert.push\_back((\*spisok\_iter\_loc1));**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).erase(spisok\_iter\_loc1);**

**}**

**if (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**spisok\_iter\_loc1++;**

**}**

**spisok\_insert.push\_back(s);**

**spiski->push\_back(spisok\_insert);**

**spiski\_iter\_loc1 = spiski->begin();**

**while (spiski\_iter\_loc1 != spiski->end())**

**{**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**printf("{ ");**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**printf("%d ", \*spisok\_iter\_loc1);**

**spisok\_iter\_loc1++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc1++;**

**}**

**}**

**void ras(int s, int s1, int\*\* mass2, int n, int\*\* mass3) { //расщепление**

**printf("\nРезультат расщепления вершины 1й матрицы:\n");**

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

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

**mass2[i][j] = 0;**

**}**

**}**

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

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

**mass2[i][j] = mass3[i][j];**

**}**

**}**

**for (int i = 0; i < n + 1; i++) { //расщепление вершины 1й матрицы**

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

**mass2[n][s] = 1;**

**mass2[s][n] = 1;**

**if (j < n / 2 && mass2[s][j] == 1) {**

**mass2[n][j] = 1;**

**mass2[s][j] = 0;**

**}**

**if (i < n / 2 && mass2[i][s] == 1) {**

**mass2[i][n] = 1;**

**mass2[i][s] = 0;**

**}**

**printf("%d ", mass2[i][j]); //вывод результата**

**}**

**printf("\n");**

**}**

**}**

**void obe\_lst(list<list<int>>\* spiski1, list<list<int>>\* spiski2) { //объединение**

**list<list<int>>::iterator spiski\_iter\_loc1 = spiski1->begin();**

**list<list<int>>::iterator spiski\_iter\_loc2 = spiski2->begin();**

**list<int>::iterator spisok\_iter\_loc1, spisok\_iter\_loc2;**

**bool versh\_searched = false;**

**printf("Результат объединения 2ух экземпляров списков смежности:\n");**

**if (spiski1->size() >= spiski2->size())**

**{**

**while (spiski\_iter\_loc2 != spiski2->end())**

**{**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->begin();**

**while (spisok\_iter\_loc2 != (\*spiski\_iter\_loc2).end())**

**{**

**versh\_searched = false;**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**if ((\*spisok\_iter\_loc2) == (\*spisok\_iter\_loc1))**

**versh\_searched = true;**

**spisok\_iter\_loc1++;**

**}**

**if (!versh\_searched)**

**(\*spiski\_iter\_loc1).push\_back((\*spisok\_iter\_loc2));**

**spisok\_iter\_loc2++;**

**}**

**spiski\_iter\_loc2++;**

**spiski\_iter\_loc1++;**

**}**

**spiski\_iter\_loc1 = spiski1->begin();**

**while (spiski\_iter\_loc1 != spiski1->end())**

**{**

**spisok\_iter\_loc1 = (\*spiski\_iter\_loc1).begin();**

**printf("{ ");**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**printf("%d ", \*spisok\_iter\_loc1);**

**spisok\_iter\_loc1++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc1++;**

**}**

**}**

**if (spiski2->size() > spiski1->size())**

**{**

**while (spiski\_iter\_loc1 != spiski1->end())**

**{**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->begin();**

**while (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**{**

**versh\_searched = false;**

**spisok\_iter\_loc2 = (\*spiski\_iter\_loc2).begin();**

**while (spisok\_iter\_loc2 != (\*spiski\_iter\_loc2).end())**

**{**

**if ((\*spisok\_iter\_loc1) == (\*spisok\_iter\_loc2))**

**versh\_searched = true;**

**spisok\_iter\_loc2++;**

**}**

**if (!versh\_searched)**

**(\*spiski\_iter\_loc2).push\_back((\*spisok\_iter\_loc1));**

**spisok\_iter\_loc1++;**

**}**

**spiski\_iter\_loc1++;**

**spiski\_iter\_loc2++;**

**}**

**spiski\_iter\_loc2 = spiski1->begin();**

**while (spiski\_iter\_loc2 != spiski1->end())**

**{**

**spisok\_iter\_loc2 = (\*spiski\_iter\_loc2).begin();**

**printf("{ ");**

**while (spisok\_iter\_loc2 != (\*spiski\_iter\_loc2).end())**

**{**

**printf("%d ", \*spisok\_iter\_loc2);**

**spisok\_iter\_loc2++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc2++;**

**}**

**}**

**}**

**void obe(int\*\* mass\_p1, int\*\* mass\_p2, int n, int n1) { //объединение**

**int i, j;**

**printf("Матрица 1\n");**

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

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

**printf("%d ", mass\_p1[i][j]); //вывод матрицы**

**}**

**printf("\n");**

**}**

**printf("\nМатрица 2\n");**

**for (i = 0; i < n1; i++) {**

**for (j = 0; j < n1; j++) {**

**printf("%d ", mass\_p2[i][j]); //вывод матрицы**

**}**

**printf("\n");**

**}**

**//выполнение операции**

**if (n >= n1) {**

**for (i = 0; i < n1; i++) {**

**for (j = 0; j < n1; j++) {**

**if (mass\_p1[i][j] != 1 && mass\_p2[i][j] == 1) {**

**mass\_p1[i][j] = 1;**

**}**

**}**

**}**

**printf("\nРезультат объединения матриц:\n");**

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

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

**printf("%d ", mass\_p1[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**if (n1 > n) {**

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

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

**if (mass\_p1[i][j] == 1 && mass\_p2[i][j] != 1) {**

**mass\_p2[i][j] = 1;**

**}**

**}**

**}**

**printf("\nРезультат объединения матриц:\n");**

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

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

**printf("%d ", mass\_p2[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**}**

**void pere\_lst(list<list<int>>\* spiski1, list<list<int>>\* spiski2) { //пересечение**

**list<list<int>>::iterator spiski\_iter\_loc1 = spiski1->begin();**

**list<list<int>>::iterator spiski\_iter\_loc2 = spiski2->begin();**

**list<int>::iterator spisok\_iter\_loc1, spisok\_iter\_loc2;**

**bool versh\_searched = false;**

**printf("Результат пересечения 2ух экземпляров списков смежности:\n");**

**if (spiski1->size() >= spiski2->size())**

**{**

**while (spiski\_iter\_loc2 != spiski2->end())**

**{**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->begin();**

**while (spisok\_iter\_loc1 != spiski\_iter\_loc1->end())**

**{**

**versh\_searched = false;**

**if (!spiski\_iter\_loc2->empty())**

**{**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->begin();**

**while (spisok\_iter\_loc2 != spiski\_iter\_loc2->end())**

**{**

**if ((\*spisok\_iter\_loc2) == (\*spisok\_iter\_loc1))**

**{**

**versh\_searched = true;**

**break;**

**}**

**spisok\_iter\_loc2++;**

**}**

**if (!versh\_searched)**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->erase(spisok\_iter\_loc1);**

**else**

**{**

**if (spisok\_iter\_loc1 != (\*spiski\_iter\_loc1).end())**

**spisok\_iter\_loc1++;**

**}**

**}**

**}**

**spiski\_iter\_loc2++;**

**spiski\_iter\_loc1++;**

**}**

**spiski\_iter\_loc1 = spiski1->begin();**

**while (spiski\_iter\_loc1 != spiski1->end())**

**{**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->begin();**

**printf("{ ");**

**while (spisok\_iter\_loc1 != spiski\_iter\_loc1->end())**

**{**

**printf("%d ", \*spisok\_iter\_loc1);**

**spisok\_iter\_loc1++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc1++;**

**}**

**}**

**if (spiski2->size() > spiski1->size())**

**{**

**while (spiski\_iter\_loc1 != spiski1->end())**

**{**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->begin();**

**while (spisok\_iter\_loc2 != spiski\_iter\_loc2->end())**

**{**

**versh\_searched = false;**

**if (!spiski\_iter\_loc1->empty())**

**{**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->begin();**

**while (spisok\_iter\_loc1 == spiski\_iter\_loc1->end())**

**{**

**if ((\*spisok\_iter\_loc1) == (\*spisok\_iter\_loc2))**

**{**

**versh\_searched = true;**

**break;**

**}**

**spisok\_iter\_loc1++;**

**}**

**if (!versh\_searched)**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->erase(spisok\_iter\_loc2);**

**else**

**{**

**if (spisok\_iter\_loc2 != (\*spiski\_iter\_loc2).end())**

**spisok\_iter\_loc2++;**

**}**

**}**

**}**

**spiski\_iter\_loc1++;**

**spiski\_iter\_loc2++;**

**}**

**spiski\_iter\_loc2 = spiski2->begin();**

**while (spiski\_iter\_loc2 != spiski2->end())**

**{**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->begin();**

**printf("{ ");**

**while (spisok\_iter\_loc2 != spiski\_iter\_loc2->end())**

**{**

**printf("%d ", \*spisok\_iter\_loc2);**

**spisok\_iter\_loc2++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc2++;**

**}**

**}**

**}**

**void pere(int\*\* mass\_o1, int\*\* mass\_o2, int n, int n1) { //пересечение**

**int i, j;**

**printf("Матрица 1\n");**

**for (i = 0; i < n; i++) { //вывод матрицы1**

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

**printf("%d ", mass\_o1[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\nМатрица 2\n");**

**for (i = 0; i < n1; i++) { //вывод матрицы2**

**for (j = 0; j < n1; j++) {**

**printf("%d ", mass\_o2[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\n");**

**if (n >= n1) { //выполнение операции**

**for (i = 0; i < n1; i++) {**

**for (j = 0; j < n1; j++) {**

**if (mass\_o2[i][j] == 1 && mass\_o1[i][j] == 1)**

**mass\_o1[i][j] = 1;**

**else mass\_o1[i][j] = 0;**

**}**

**}**

**printf("Результат пересечения матриц:\n");**

**for (i = 0; i < n1; i++) {**

**for (j = 0; j < n1; j++) {**

**printf("%d ", mass\_o1[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**if (n1 > n) {**

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

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

**if (mass\_o2[i][j] == 1 && mass\_o1[i][j] == 1)**

**mass\_o2[i][j] = 1;**

**else mass\_o2[i][j] = 0;**

**if (n > n1) {**

**mass\_o2[i][j] = 0;**

**}**

**}**

**}**

**printf("Результат пересечения матриц:\n");**

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

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

**printf("%d ", mass\_o2[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**printf("\n");**

**}**

**void kolz\_lst(list<list<int>>\* spiski1, list<list<int>>\* spiski2) { //кольцевая сумма**

**list<list<int>>::iterator spiski\_iter\_loc1 = spiski1->begin();**

**list<list<int>>::iterator spiski\_iter\_loc2 = spiski2->begin();**

**list<int>::iterator spisok\_iter\_loc1, spisok\_iter\_loc2;**

**bool versh\_searched = false;**

**printf("Результат кольцевой суммы 2ух экземпляров списков смежности:\n");**

**if (spiski1->size() >= spiski2->size())**

**{**

**while (spiski\_iter\_loc2 != spiski2->end())**

**{**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->begin();**

**while (spisok\_iter\_loc2 != spiski\_iter\_loc2->end())**

**{**

**versh\_searched = false;**

**if (!spiski\_iter\_loc1->empty())**

**{**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->begin();**

**while (spisok\_iter\_loc1 != spiski\_iter\_loc1->end())**

**{**

**if ((\*spisok\_iter\_loc1) == (\*spisok\_iter\_loc2))**

**{**

**versh\_searched = true;**

**break;**

**}**

**spisok\_iter\_loc1++;**

**}**

**if (versh\_searched)**

**{**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->erase(spisok\_iter\_loc1);**

**}**

**else**

**{**

**spiski\_iter\_loc1->push\_back((\*spisok\_iter\_loc2));**

**}**

**spisok\_iter\_loc2++;**

**}**

**}**

**spiski\_iter\_loc2++;**

**spiski\_iter\_loc1++;**

**}**

**spiski\_iter\_loc1 = spiski1->begin();**

**while (spiski\_iter\_loc1 != spiski1->end())**

**{**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->begin();**

**printf("{ ");**

**while (spisok\_iter\_loc1 != spiski\_iter\_loc1->end())**

**{**

**printf("%d ", \*spisok\_iter\_loc1);**

**spisok\_iter\_loc1++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc1++;**

**}**

**}**

**if (spiski2->size() > spiski1->size())**

**{**

**while (spiski\_iter\_loc1 != spiski1->end())**

**{**

**spisok\_iter\_loc1 = spiski\_iter\_loc1->begin();**

**while (spisok\_iter\_loc1 != spiski\_iter\_loc1->end())**

**{**

**versh\_searched = false;**

**if (!spiski\_iter\_loc2->empty())**

**{**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->begin();**

**while (spisok\_iter\_loc2 == spiski\_iter\_loc2->end())**

**{**

**if ((\*spisok\_iter\_loc2) == (\*spisok\_iter\_loc1))**

**{**

**versh\_searched = true;**

**break;**

**}**

**spisok\_iter\_loc2++;**

**}**

**if (versh\_searched)**

**{**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->erase(spisok\_iter\_loc2);**

**}**

**else**

**{**

**spiski\_iter\_loc2->push\_back((\*spisok\_iter\_loc1));**

**}**

**spisok\_iter\_loc1++;**

**}**

**}**

**spiski\_iter\_loc1++;**

**spiski\_iter\_loc2++;**

**}**

**spiski\_iter\_loc2 = spiski2->begin();**

**while (spiski\_iter\_loc2 != spiski2->end())**

**{**

**spisok\_iter\_loc2 = spiski\_iter\_loc2->begin();**

**printf("{ ");**

**while (spisok\_iter\_loc2 != spiski\_iter\_loc2->end())**

**{**

**printf("%d ", \*spisok\_iter\_loc2);**

**spisok\_iter\_loc2++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter\_loc2++;**

**}**

**}**

**}**

**void kolz(int\*\* mass\_k1, int\*\* mass\_k2, int n, int n1) { //кольцевая сумма**

**int i, j;**

**printf("\nМатрица 1\n");**

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

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

**printf("%d ", mass\_k1[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\nМатрица 2\n");**

**for (i = 0; i < n1; i++) {**

**for (j = 0; j < n1; j++) {**

**printf("%d ", mass\_k2[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\n");**

**if (n >= n1) { //выполнение операции**

**for (i = 0; i < n1; i++) {**

**for (j = 0; j < n1; j++) {**

**if (mass\_k1[i][j] != 1 && mass\_k2[i][j] == 1 || mass\_k2[i][j] != 1 && mass\_k1[i][j] == 1) {**

**mass\_k1[i][j] = 1;**

**}**

**else mass\_k1[i][j] = 0;**

**}**

**}**

**printf("Результат кольцевой суммы матриц:\n");**

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

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

**printf("%d ", mass\_k1[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**if (n1 > n) {**

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

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

**if (mass\_k1[i][j] != 1 && mass\_k2[i][j] == 1 || mass\_k2[i][j] != 1 && mass\_k1[i][j] == 1) {**

**mass\_k2[i][j] = 1;**

**}**

**else mass\_k2[i][j] = 0;**

**}**

**}**

**printf("Результат кольцевой суммы матриц:\n");**

**for (i = 0; i < n1; i++) {**

**for (j = 0; j < n1; j++) {**

**printf("%d ", mass\_k2[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**}**

**int main() {**

**setlocale(LC\_ALL, "Rus");**

**srand(time(NULL));**

**int s, s1, n, n1;**

**printf("ЗАДАНИЕ 1\n");**

**list<list<int>>\* spiski, \* spiski1;**

**list<list<int>>\* spiski\_p2, \* spiski2;**

**list<list<int>>::iterator spiski\_iter;**

**list<int> spisok;**

**list<int>::iterator spisok\_iter;**

**spiski = new list<list<int>>;**

**spiski1 = new list<list<int>>;**

**spiski\_p2 = new list<list<int>>;**

**spiski2 = new list<list<int>>;**

**printf("Введите размеры массивов через пробел:");**

**scanf\_s("%d %d", &n, &n1);**

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

**int\*\* mass1 = (int\*\*)malloc(sizeof(int) \* n);**

**int\*\* mass\_p1 = (int\*\*)malloc(sizeof(int) \* n);**

**int\*\* mass\_o1 = (int\*\*)malloc(sizeof(int) \* n);**

**int\*\* mass\_k1 = (int\*\*)malloc(sizeof(int) \* n);**

**int\*\* mass\_k5 = (int\*\*)malloc(sizeof(int) \* n);**

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

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

**mass1[i] = (int\*)malloc(sizeof(int) \* n);**

**mass\_p1[i] = (int\*)malloc(sizeof(int) \* n);**

**mass\_o1[i] = (int\*)malloc(sizeof(int) \* n);**

**mass\_k1[i] = (int\*)malloc(sizeof(int) \* n);**

**mass\_k5[i] = (int\*)malloc(sizeof(int) \* n);**

**}**

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

**int\*\* mass\_p2 = (int\*\*)malloc(sizeof(int) \* n1);**

**int\*\* mass\_o2 = (int\*\*)malloc(sizeof(int) \* n1);**

**int\*\* mass\_k2 = (int\*\*)malloc(sizeof(int) \* n1);**

**int\*\* mass\_o5 = (int\*\*)malloc(sizeof(int) \* n1);**

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

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

**mass\_p2[i] = (int\*)malloc(sizeof(int) \* n1);**

**mass\_o2[i] = (int\*)malloc(sizeof(int) \* n1);**

**mass\_k2[i] = (int\*)malloc(sizeof(int) \* n);**

**mass\_o5[i] = (int\*)malloc(sizeof(int) \* n1);**

**}**

**int\*\* mass2 = (int\*\*)malloc(sizeof(int) \* n + 1);**

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

**mass2[i] = (int\*)malloc(sizeof(int) \* n + 1);**

**}**

**int\*\* mass3 = (int\*\*)malloc(sizeof(int) \* n);**

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

**mass3[i] = (int\*)malloc(sizeof(int) \* n);**

**}**

**int i = 0;**

**int j = 0, k = 0, m = 0;**

**int a;**

**printf("Действия с матрицами\n\n");**

**printf("\nМатрица 1:\n");**

**generate(mass, n);**

**generate(mass\_p2,**

**n1);**

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

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

**mass1[i][j] = mass[i][j];**

**mass\_o1[i][j] = mass1[i][j];**

**mass\_p1[i][j] = mass1[i][j];**

**mass\_k1[i][j] = mass1[i][j];**

**mass\_k5[i][j] = mass1[i][j];**

**}**

**}**

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

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

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

**}**

**printf("\n");**

**}**

**printf("\nМатрица 2:\n");**

**for (i = 0; i < n1; i++) {**

**for (j = 0; j < n1; j++) {**

**printf("%d ", mass\_p2[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\n");**

**printf("\nЗАДАНИЕ 2\n");**

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

**{**

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

**{**

**if (mass[i][j] == 1)**

**{**

**spisok.push\_back(j);**

**}**

**}**

**spiski->push\_back(spisok);**

**spisok.clear();**

**}**

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

**{**

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

**{**

**if (mass\_p2[i][j] == 1)**

**{**

**spisok.push\_back(j);**

**}**

**}**

**spiski\_p2->push\_back(spisok);**

**spisok.clear();**

**}**

**printf("1.Отождествление вершин\n");**

**printf("Введите номера вершин:");**

**scanf\_s("%d %d", &s, &s1);**

**printf("\n");**

**cpy(mass3, mass, n);**

**otoz(s, s1, mass3, n);**

**printf("\n");**

**printf("\n2.Стягивание ребер\n");**

**printf("\n");**

**cpy(mass3, mass, n);**

**styag(s, s1, mass3, n);**

**printf("\n");**

**printf("\n3.Расщепление вершин\n");**

**cpy(mass3, mass, n);**

**ras(s, s1, mass2, n, mass3);**

**printf("\n");**

**printf("\nЗАДАНИЕ 3\n");**

**printf("1.Объединение матриц\n");**

**cpy(mass\_k2, mass\_p2, n1);**

**cpy(mass\_o5, mass\_p2, n1);**

**printf("\n");**

**obe(mass\_p1, mass\_p2, n, n1);**

**printf("\n");**

**printf("\n2.Пересечение матриц\n");**

**cpy(mass\_o2, mass\_p2, n1);**

**printf("\n");**

**pere(mass\_o1, mass\_o2, n, n1);**

**printf("\n3.Кольцевая сумма матриц\n");**

**kolz(mass\_k1, mass\_k2, n, n1);**

**printf("\nДекартово произведение графов\n");**

**dekartovo(mass, mass\_o5,n,n1);**

**printf("\nДействия со списками смежности\n\n");**

**printf("1ые списки смежности:\n");**

**spiski\_iter = spiski->begin();**

**while (spiski\_iter != spiski->end())**

**{**

**spisok\_iter = (\*spiski\_iter).begin();**

**printf("{ ");**

**while (spisok\_iter != (\*spiski\_iter).end())**

**{**

**printf("%d ", \*spisok\_iter);**

**spisok\_iter++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter++;**

**}**

**printf("2ые списки смежности:\n");**

**spiski\_iter = spiski\_p2->begin();**

**while (spiski\_iter != spiski\_p2->end())**

**{**

**spisok\_iter = (\*spiski\_iter).begin();**

**printf("{ ");**

**while (spisok\_iter != (\*spiski\_iter).end())**

**{**

**printf("%d ", \*spisok\_iter);**

**spisok\_iter++;**

**}**

**printf("}");**

**printf("\n");**

**spiski\_iter++;**

**}**

**printf("\n1.Отождествление вершин\n");**

**printf("\n");**

**(\*spiski1) = (\*spiski);**

**otoz\_lst(s, s1, spiski1, true);**

**printf("\n");**

**printf("\n2.Стягивание ребер\n");**

**(\*spiski1) = (\*spiski);**

**styag\_lst(s, s1, spiski1);**

**printf("\n");**

**printf("\n3.Расщепление вершин\n");**

**(\*spiski1) = (\*spiski);**

**ras\_lst(s, spiski1);**

**printf("\n");**

**printf("\n4.Объединение вершин\n");**

**printf("\n");**

**(\*spiski1) = (\*spiski);**

**(\*spiski2) = (\*spiski\_p2);**

**obe\_lst(spiski1, spiski2);**

**printf("\n");**

**printf("\n5.Пересечение вершин\n");**

**(\*spiski1) = (\*spiski);**

**(\*spiski2) = (\*spiski\_p2);**

**pere\_lst(spiski1, spiski2);**

**printf("\n");**

**printf("\n6.Кольцевая сумма вершин\n");**

**(\*spiski1) = (\*spiski);**

**(\*spiski2) = (\*spiski\_p2);**

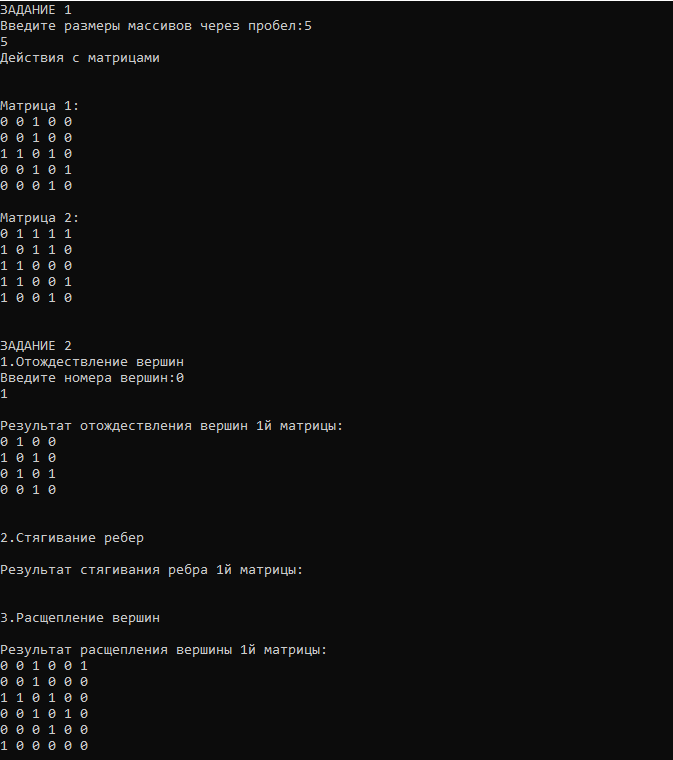
**kolz\_lst(spiski1, spiski2);**

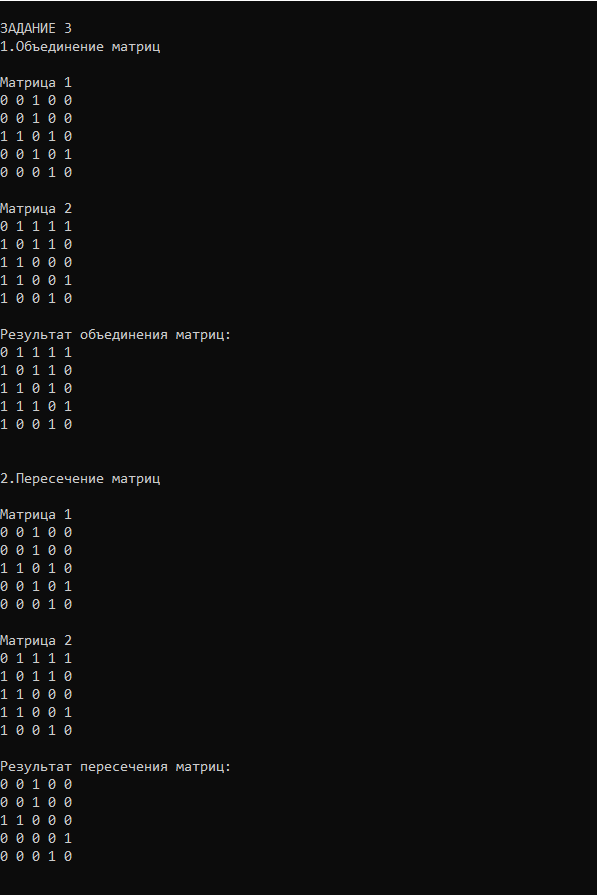
**printf("\n");**

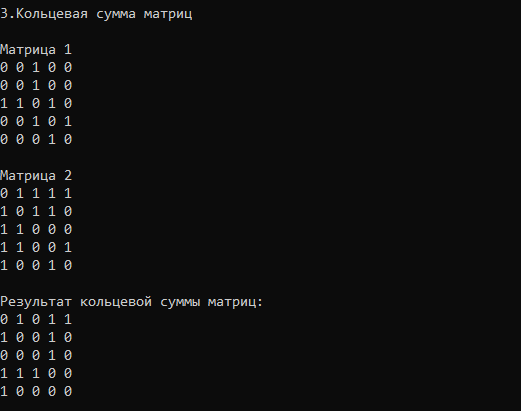
**}**

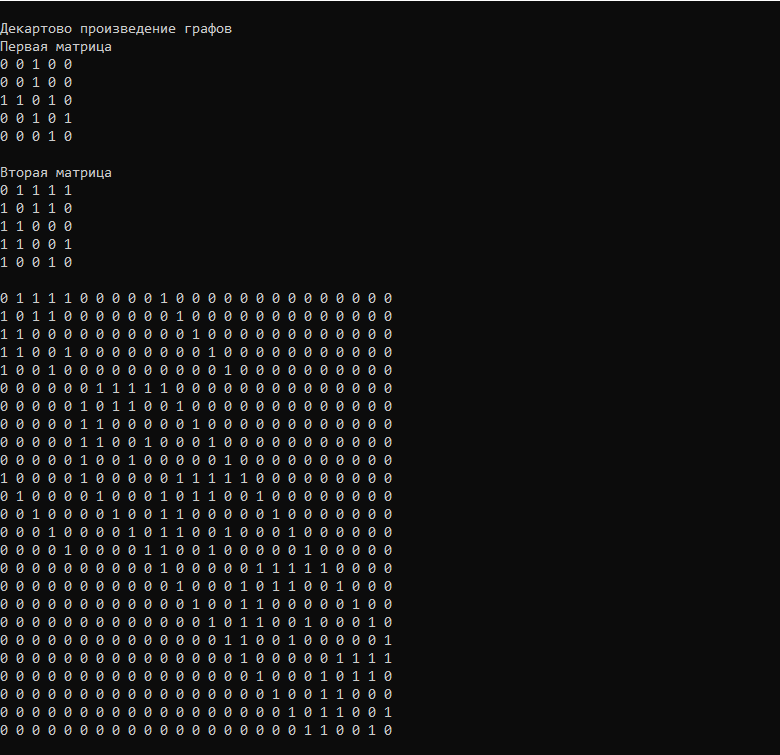
**Результат работы программы:**

С матрицами:

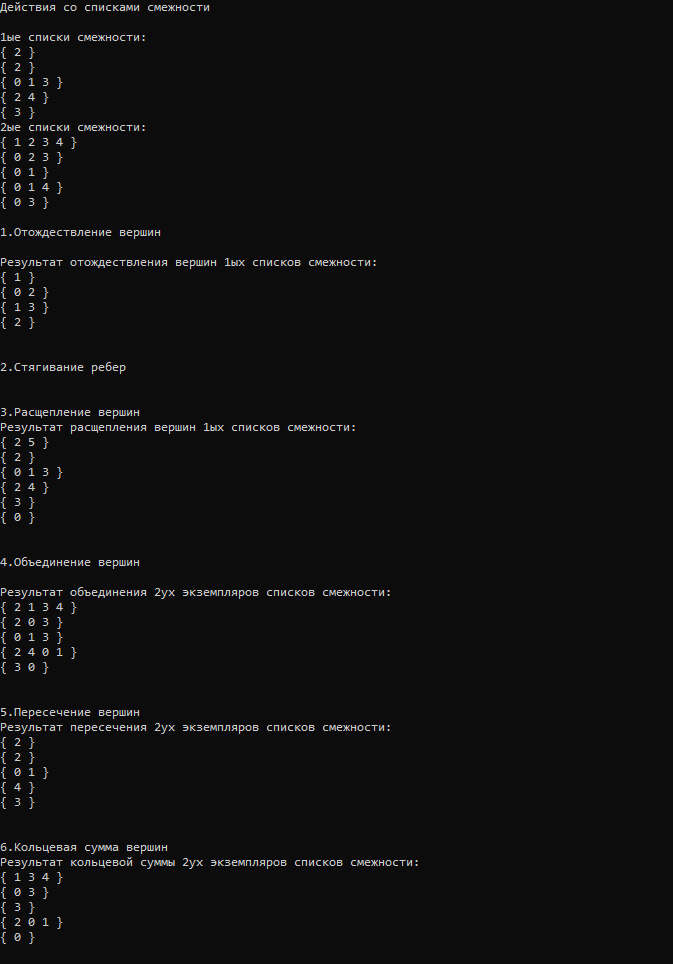








Со списками смежности:



**Вывод**:в ходе выполнения данной лабораторной работы мы получили необходимые знания о бинарных и унарных операциях над графами и отработали их на практике.Операции производились как на матрицах смежности,так и на списках смежности,что позволило нам вспомнить и улучшить наши навыки в работе со структорой списков.