Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Логика и основы алгоритмизации

в инженерных задачах»

на тему «Унарные и бинарные операции над графами»

Выполнили:

студенты группы 22ВВВ1

Дунюшкин В.А.

Ивлюшин М.Д.

Проверили:

к.т.н, доцент Юрова О.В.

к.э.н, доцент Акифьев И.В.

Пенза 2023

**Цель работы:**

Научится выполнять бинарные и унарные операции над графами

**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.

2.\* 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

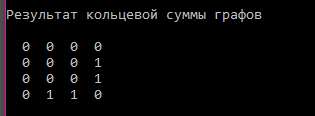
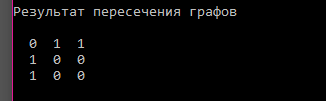
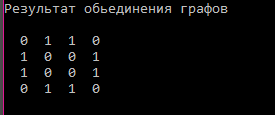
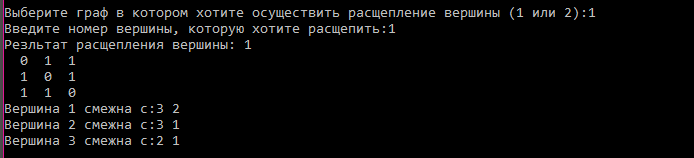
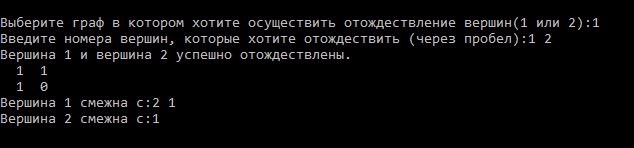
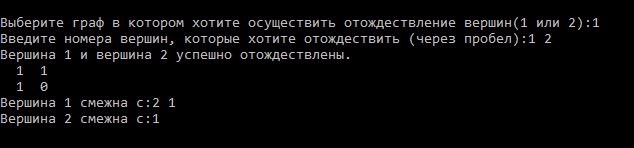
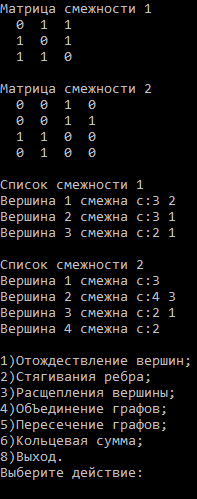
Результат выполнения операции выведите на экран.

**Задание 4 \***

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

Результат выполнения операции выведите на экран.

**Результаты работы программы:**



**Вывод**

В ходе данной лабораторной работе, мы научились выполнять унарные и бинарные операции над графами.

**Листинг**

// ConsoleApplication1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <cstdio>

#include <iomanip>

using namespace std;

//Структура для списка смежности

typedef struct Spis

{

int inf;

struct Spis\* next;

}SP;

extern int i, j;

void PrintSpisok(int ver, SP\*\* MassSpis);

void PrintMatrix(int ver, int\*\*\* smej);

void GeneratSpisok(int ver, SP\*\* MassSpis, int\*\* smej);

void CombiningGraphs(int\*\* smej1, int\*\* smej2, int ver1, int ver2);

void IntersectionGraphs(int\*\* smej1, int\*\* smej2, int ver1, int ver2);

void SummGraphs(int\*\* smej1, int\*\* smej2, int ver1, int ver2);

void IdentificationMatrix(int\*\* smej, int\* ver, int veri1, int veri2, SP\*\* MassSpis);

void RibTightening(int\*\* smej, int\* ver, int veri1, int veri2, SP\*\* MassSpis);

void SplittingAVertex(int\*\* smej, int\* ver, int veri1, SP\*\* MassSpis);

void MatrixSmej(int\*\*\* smej1, int\*\*\* smej2, int ver1, int ver2);

void SpisokSmej(int\*\*& smej1, int\*\*& smej2, int ver1, int ver2, SP\*\*& MassSpis1, SP\*\*& MassSpis2);

void PrintMatrix(int ver, int\*\*\* smej)

{

for (i = 0; i < ver; i++)

{

for (j = 0; j < ver; j++)

{

cout << " " << (\*smej)[i][j];

}

cout << endl;

}

}

void CombiningGraphs(int\*\* smej1, int\*\* smej2, int ver1, int ver2)

{

int maxver;

if (ver1 > ver2)

maxver = ver1;

else

maxver = ver2;

int\*\* smejcomb = new int\* [maxver];

for (i = 0; i < maxver; i++)

{

smejcomb[i] = new int[maxver];

for (j = 0; j < maxver; j++)

{

int val1;

if (i < ver1 && j < ver1) val1 = smej1[i][j];

else val1 = 0;

int val2;

if (i < ver2 && j < ver2) val2 = smej2[i][j];

else val2 = 0;

smejcomb[i][j] = val1 || val2;

}

}

cout << "Результат обьединения графов" << endl << endl;

PrintMatrix(maxver, &smejcomb);

cout << endl;

for (i = 0; i < maxver; i++)

{

delete[] smejcomb[i];

}

delete[] smejcomb;

getchar(); getchar();

}

void IntersectionGraphs(int\*\* smej1, int\*\* smej2, int ver1, int ver2)

{

int maxver;

if (ver1 > ver2)

maxver = ver1;

else

maxver = ver2;

int\*\* smejinter = new int\* [maxver];

for (i = 0; i < maxver; i++)

{

smejinter[i] = new int[maxver];

for (j = 0; j < maxver; j++)

{

int val1;

if (i < ver1 && j < ver1) val1 = smej1[i][j];

else val1 = 0;

int val2;

if (i < ver2 && j < ver2) val2 = smej2[i][j];

else val2 = 0;

smejinter[i][j] = val1 && val2;

}

}

cout << "Результат пересечения графов" << endl << endl;

if (ver1 == 1 && ver2 == 1) { cout << " " << 0 << endl; }

else { PrintMatrix(maxver - 1, &smejinter); }

cout << endl;

for (i = 0; i < maxver; i++)

{

delete[] smejinter[i];

}

delete[] smejinter;

getchar(); getchar();

}

void SummGraphs(int\*\* smej1, int\*\* smej2, int ver1, int ver2)

{

int maxver;

if (ver1 > ver2)

maxver = ver1;

else

maxver = ver2;

int p = abs(ver1 - ver2);

int\*\* smejsumm = new int\* [maxver];

for (i = 0; i < maxver; i++)

{

smejsumm[i] = new int[maxver];

for (j = 0; j < maxver; j++)

{

if (i < ver1 || j < ver1 || j < ver2 || i < ver2)

{

if (i == j) smejsumm[i][j] = 0;

else

{

int val1;

if (i < ver1 && j < ver1) val1 = smej1[i][j];

else val1 = 0;

int val2;

if (i < ver2 && j < ver2) val2 = smej2[i][j];

else val2 = 0;

smejsumm[i][j] = val1 || val2;

}

}

else smejsumm[i][j] = 0;

}

}

for (i = 0; i < maxver - p; i++)

{

for (j = 0; j < maxver - p; j++)

{

if (smej1[i][j] == 1 && smej2[i][j] == 1) smejsumm[i][j] = 0;

}

}

cout << "Результат кольцевой суммы графов" << endl << endl;

PrintMatrix(maxver, &smejsumm);

cout << endl;

for (i = 0; i < maxver; i++)

{

delete[] smejsumm[i];

}

delete[] smejsumm;

getchar(); getchar();

}

void DecartGraphs(int\*\* smej1, int\*\* smej2, int ver1, int ver2)

{

int decver = ver1 \* ver2;

int decveri = ver1 \* ver2;

int\*\* smejdec = new int\* [decver];

for (i = 0; i < decver; i++)

{

smejdec[i] = new int[decver];

for (j = 0; j < decver; j++)

{

smejdec[i][j] = 0;

}

}

int u = 0, v = 0, i1, j1;

for (i = 0; i < ver1; i++)

{

for (j = 0; j < ver2; j++)

{

for (i1 = 0; i1 < ver1; i1++)

{

for (j1 = 0; j1 < ver1; j1++)

{

u = i \* ver2 + j;

v = i1 \* ver2 + j1;

if (j == j1) smejdec[u][v] = smej1[i][i1];

if (i == i1) smejdec[u][v] = smej2[j][j1];

if (i != i1 && j != j1) smejdec[u][v] = 0;

}

}

}

}

cout << "Результат декартова произведения графов" << endl << endl;

for (i = 0; i < decver; i++)

{

for (j = 0; j < decveri; j++)

{

cout << setw(3) << smejdec[i][j];

}

cout << endl;

}

cout << endl;

}

void free(SP\* head)

{

SP\* current = head;

while (current != NULL)

{

SP\* temp = current;

current = current->next;

delete temp;;

}

}

void IdentificationMatrix(int\*\* smej, int\* ver, int veri1, int veri2, SP\*\* MassSpis)

{

int T;

if (veri1 > veri2)

{

T = veri1;

veri1 = veri2;

veri2 = T;

}

bool hasEdge = smej[veri1][veri2];

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

{

if (i != veri1 && i != veri2)

{

smej[veri1][i] = smej[veri1][i] || smej[veri2][i]; // Объединяем строки veri1 и veri2

smej[i][veri1] = smej[i][veri1] || smej[i][veri2]; // Объединяем столбцы veri1 и veri2

}

}

for (i = veri2; i < \*ver - 1; i++)

{

for (j = 0; j < \*ver - 1; j++)

{

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

}

}

for (i = 0; i < \*ver - 1; i++)

{

for (j = veri2; j < \*ver - 1; j++)

{

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

}

}

for (i = 0; i < \*ver - 1; i++)

{

for (j = 0; j < \*ver - 1; j++)

{

if (i == j) smej[i][j] = 0;

}

}

if (hasEdge)

{

smej[veri1][veri1] = 1;

}

cout << "Вершина " << veri1 + 1 << " и вершина " << veri2 + 1 << " успешно отождествлены." << endl;

PrintMatrix(\*ver - 1, &smej);

(\*ver)--;

for (int i = 0; i < \*ver + 1; i++)

{

free(MassSpis[i]);

MassSpis[i] = NULL;

}

GeneratSpisok(\*ver, MassSpis, smej);

PrintSpisok(\*ver, MassSpis);

cout << endl;

getchar(); getchar();

}

void RibTightening(int\*\* smej, int\* ver, int veri1, int veri2, SP\*\* MassSpis)

{

int T;

if (veri1 > veri2)

{

T = veri1;

veri1 = veri2;

veri2 = T;

}

if (smej[veri1][veri2] == 0)

{

printf("Между вершинами должно быть ребро!\n");

return;

}

bool\* hasEdge = new bool[\*ver];

memset(hasEdge, false, sizeof(bool) \* (\*ver));

for (i = 0; i < \*ver; ++i)

{

if (smej[i][i] == 1)

{

hasEdge[i] = true;

}

}

for (i = 0; i < \*ver; i++)

{

for (j = 0; j < \*ver; j++)

{

smej[i][veri1] = smej[i][veri1] || smej[i][veri2];

smej[veri1][j] = smej[veri1][j] || smej[veri2][j];

}

}

for (i = 0; i < \*ver - 1; i++)

{

for (j = veri2; j < \*ver - 1; j++)

{

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

}

}

for (i = veri2; i < \*ver - 1; i++)

{

for (j = 0; j < \*ver - 1; j++)

{

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

}

}

for (i = 0; i < \*ver - 1; ++i)

{

if (i != veri1 && i != veri2 && hasEdge[i]) smej[i][i] = 1;

else smej[i][i] = 0;

}

delete[] hasEdge;

cout << " Между вершиной " << veri1 + 1 << " и вершиной " << veri2 + 1 << " было успешно стянуто ребро." << endl;

PrintMatrix(\*ver - 1, &smej);

(\*ver)--;

for (int i = 0; i < \*ver + 1; i++)

{

free(MassSpis[i]);

MassSpis[i] = NULL;

}

GeneratSpisok(\*ver, MassSpis, smej);

PrintSpisok(\*ver, MassSpis);

cout << endl;

}

void SplittingAVertex(int\*\* smej, int\* ver, int veri1, SP\*\* MassSpis)

{

int newver = \*ver + 1;

int\*\* smejprom = new int\* [newver];

if (smejprom == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

for (i = 0; i < newver; i++)

{

smejprom[i] = new int[newver];

for (j = 0; j < newver; j++)

{

if (i < \*ver && j < \*ver)

{

smejprom[i][j] = smej[i][j]; // копируем связи исходной матрицы

}

else if (i == \*ver && j < \*ver)

{

smejprom[i][j] = smej[veri1][j]; // новая вершина соединена со смежными вершинами выбранной вершины

}

else if (i < \*ver && j == \*ver)

{

smejprom[i][j] = smej[i][veri1]; // смежные вершины соединены с новой вершиной

}

else

{

smejprom[i][j] = 0; // новая вершина не связана с остальными вершинами

}

}

}

(\*ver)++;

smej = new int\* [\*ver];

if (smej == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

for (i = 0; i < \*ver; i++)

{

smej[i] = new int[\*ver];

}

for (i = 0; i < \*ver; i++)

{

for (j = 0; j < \*ver; j++)

{

if (i == veri1 && j == \*ver - 1 || j == veri1 && i == \*ver - 1) smej[i][j] = 1;

else smej[i][j] = smejprom[i][j];

}

}

cout << "Резльтат расщепления вершины: " << veri1 + 1 << endl;

PrintMatrix(\*ver, &smej);

for (int i = 0; i < \*ver - 1; i++)

{

free(MassSpis[i]);

MassSpis[i] = NULL;

}

MassSpis = new SP \* [\*ver];

if (MassSpis == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

for (i = 0; i < \*ver; i++)

{

MassSpis[i] = NULL;

}

GeneratSpisok(\*ver, MassSpis, smej);

PrintSpisok(\*ver, MassSpis);

cout << endl;

for (i = 0; i < newver; i++)

{

delete[] smejprom[i];

}

delete[] smejprom;

getchar();; getchar();

}

int i, j;

void GeneratMatrix(int ver, int\*\*\* smej)

{

for (i = 0; i < ver; i++)

{

for (j = i; j < ver; j++)

{

if (i == j)

{

(\*smej)[i][j] = 0; // на главной диагонали нули

}

else

{

(\*smej)[i][j] = rand() % 2; // случайные значения 0 или 1

(\*smej)[j][i] = (\*smej)[i][j]; // симметрично заполнять значения для неориентированного графа

}

}

}

}

//работа с матрицами смежности

void MatrixSmej(int\*\*\* smej1, int\*\*\* smej2, int ver1, int ver2)

{

//выделение памяти под матрицу 1

\*smej1 = new int\* [ver1];

if (\*smej1 == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

for (i = 0; i < ver1; i++)

{

(\*smej1)[i] = new int[ver1];

}

//выделение памяти под матрицу 2

\*smej2 = new int\* [ver2];

if (\*smej2 == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

for (i = 0; i < ver2; i++)

{

(\*smej2)[i] = new int[ver2];

}

// создание матрицы смежности 1

GeneratMatrix(ver1, smej1);

// создание матрицы смежности 2

GeneratMatrix(ver2, smej2);

}

void GeneratSpisok(int ver, SP\*\* MassSpis, int\*\* smej)

{

for (i = 0; i < ver; i++)

{

for (j = 0; j < ver; j++)

{

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

{

SP\* newSP = new SP;

newSP->inf = j + 1;

newSP->next = MassSpis[i];

MassSpis[i] = newSP;

}

}

}

}

void PrintSpisok(int ver, SP\*\* MassSpis)

{

for (i = 0; i < ver; i++)

{

if (MassSpis[i] == NULL)

{

cout << "Вершина " << i + 1 << " не имеет смежных";

}

else

{

cout << "Вершина " << i + 1 << " смежна с:";

SP\* current = MassSpis[i];

while (current != NULL)

{

cout << current->inf << " ";

current = current->next;

}

}

cout << endl;

}

}

//работа со списками смежности

void SpisokSmej(int\*\*& smej1, int\*\*& smej2, int ver1, int ver2, SP\*\*& MassSpis1, SP\*\*& MassSpis2)

{

//Выделение памяти под массив списков 1

MassSpis1 = new SP \* [ver1];

if (MassSpis1 == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

for (i = 0; i < ver1; i++)

{

MassSpis1[i] = NULL;

}

//Выделение памяти под массив списков 2

MassSpis2 = new SP \* [ver2];

if (MassSpis2 == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

for (i = 0; i < ver2; i++)

{

MassSpis2[i] = NULL;

}

//Создание списка смежности 1

GeneratSpisok(ver1, MassSpis1, smej1);

//Создание списка смежности 2

GeneratSpisok(ver2, MassSpis2, smej2);

}

void main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int\*\* smej1 = 0; //1 матрица смежности

int\*\* smej2 = 0; //2 матрица смежности

int ver1, ver2, choice;

bool prov = 1;

SP\*\* MassSpis1 = NULL;//Массив списков смежности 1

SP\*\* MassSpis2 = NULL;//Массив списков смежности 2

do

{

cout << "Введите количество вершин 1-ого графа: ";

cin >> ver1;

cout << "Введите количество вершин 2-ого графа: ";

cin >> ver2;

if (ver1 < 0 || ver2 < 0)

cout << "Ошибка: неверный ввод номеров вершин! Повторите попытку.";

else

prov = 0;

} while (prov);

MatrixSmej(&smej1, &smej2, ver1, ver2);

SpisokSmej(smej1, smej2, ver1, ver2, MassSpis1, MassSpis2);

int graf = 0;

int veri1 = 0, veri2 = 0;

bool prov1 = true, prov2 = true, prov3 = true;

system("cls");

// вывод матрицы смежности 1 на экран

cout << "Матрица смежности 1" << endl;

PrintMatrix(ver1, &smej1);

cout << endl;

// вывод матрицы смежности 2 на экран

cout << "Матрица смежности 2" << endl;

PrintMatrix(ver2, &smej2);

cout << endl;

//Вывод списка смежности 1

cout << "Список смежности 1" << endl;

PrintSpisok(ver1, MassSpis1);

cout << endl;

//Вывод списка смежности 2

cout << "Список смежности 2" << endl;

PrintSpisok(ver2, MassSpis2);

cout << endl;

do

{

cout << "1)Отождествление вершин;" << endl;

cout << "2)Стягивания ребра;" << endl;

cout << "3)Расщепления вершины;" << endl;

cout << "4)ОбЪединение графов;" << endl;

cout << "5)Пересечение графов;" << endl;

cout << "6)Кольцевая сумма;" << endl;

cout << "8)Выход." << endl;

cout << "Выберите действие: ";

cin >> choice;

cout << endl;

switch (choice)

{

case 1:

prov1 = true;

prov2 = true;

prov3 = true;

do

{

cout << endl;

cout << "Выберите граф в котором хотите осуществить отождествление вершин(1 или 2):";

cin >> graf;

if (graf == 1 || graf == 2)

prov1 = false;

else

cout << "Ошибка: неверный ввод номера графа! Повторите попытку." << endl;

} while (prov1);

if (graf == 1)

{

if (ver1 <= 1) { cout << "Невозможно " << endl; break; }

do

{

cout << "Введите номера вершин, которые хотите отождествить (через пробел):";

cin >> veri1 >> veri2;

if (veri1 - 1 >= ver1 || veri2 - 1 >= ver1 || veri1 - 1 < 0 || veri2 - 1 < 0 || veri1 == veri2)

cout << "Ошибка: неверный ввод номеров вершин! Повторите попытку." << endl;

else

prov2 = false;

} while (prov2);

IdentificationMatrix(smej1, &ver1, veri1 - 1, veri2 - 1, MassSpis1);

}

else if (graf == 2)

{

if (ver2 <= 1) { cout << "Невозможно " << endl; break; }

do

{

cout << "Введите номера вершин, которые хотите отождествить (через пробел):";

cin >> veri1 >> veri2;

if (veri1 - 1 >= ver2 || veri2 - 1 >= ver2 || veri1 - 1 < 0 || veri2 - 1 < 0 || veri1 == veri2)

cout << "Ошибка: неверный ввод номеров вершин! Повторите попытку." << endl;

else

prov3 = false;

} while (prov3);

IdentificationMatrix(smej2, &ver2, veri1 - 1, veri2 - 1, MassSpis2);

}

break;

case 2:

prov1 = true;

prov2 = true;

prov3 = true;

veri1 = 0;

veri2 = 0;

do

{

cout << endl;

cout << "Выберите граф в котором хотите осуществить стягивание ребра (1 или 2):";

cin >> graf;

if (graf == 1 || graf == 2)

prov1 = false;

else

cout << "Ошибка: неверный ввод номера графа! Повторите попытку.";

} while (prov1);

if (graf == 1)

{

if (ver1 <= 1) { cout << "Невозможно " << endl; break; }

do

{

cout << "Введите номера вершин, между которыми надо стянуть ребро (через пробел):";

cin >> veri1 >> veri2;

if (veri1 - 1 >= ver1 || veri2 - 1 >= ver1 || veri1 - 1 < 0 || veri2 - 1 < 0 || veri1 == veri2)

cout << "Ошибка: неверный ввод номеров вершин! Повторите попытку." << endl;

else if (ver1 == 1) { cout << "Невозможно " << endl; break; }

else

prov2 = false;

} while (prov2);

RibTightening(smej1, &ver1, veri1 - 1, veri2 - 1, MassSpis1);

}

else if (graf = 2)

{

if (ver2 <= 1) { cout << "Невозможно " << endl; break; }

do

{

cout << "Введите номера вершин, между которыми надо стянуть ребро (через пробел):";

cin >> veri1 >> veri2;

if (veri1 - 1 >= ver2 || veri2 - 1 >= ver2 || veri1 - 1 < 0 || veri2 - 1 < 0 || veri1 == veri2)

cout << "Ошибка: неверный ввод номеров вершин! Повторите попытку." << endl;

else

prov3 = false;

} while (prov3);

RibTightening(smej2, &ver2, veri1 - 1, veri2 - 1, MassSpis2);

}

break;

case 3:

prov1 = true;

prov2 = true;

prov3 = true;

veri1 = 0;

do

{

cout << endl;

cout << "Выберите граф в котором хотите осуществить расщепление вершины (1 или 2):";

cin >> graf;

if (graf == 1 || graf == 2)

prov1 = false;

else

cout << "Ошибка: неверный ввод номера графа! Повторите попытку." << endl;

} while (prov1);

if (graf == 1)

{

if (ver1 < 1) { cout << "Невозможно " << endl; break; }

do

{

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

cin >> veri1;

if (veri1 - 1 >= ver1 || veri1 - 1 < 0)

cout << "Ошибка: неверный ввод номера вершины! Повторите попытку." << endl;

else

prov2 = false;

} while (prov2);

SplittingAVertex(smej1, &ver1, veri1 - 1, MassSpis1);

}

else if (graf == 2)

{

if (ver2 < 1) { cout << "Невозможно " << endl; break; }

do

{

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

cin >> veri1;

if (veri1 - 1 >= ver2 || veri1 - 1 < 0)

cout << "Ошибка: неверный ввод номера вершины! Повторите попытку." << endl;

else

prov3 = false;

} while (prov3);

SplittingAVertex(smej2, &ver2, veri1 - 1, MassSpis2);

}

break;

case 4:

CombiningGraphs(smej1, smej2, ver1, ver2);

break;

case 5:

IntersectionGraphs(smej1, smej2, ver1, ver2);

break;

case 6:

SummGraphs(smej1, smej2, ver1, ver2);

break;

case 8:

system("cls");

break;

default:

cout << "Неверный выбор! Повторите попытку.";

break;

}

} while (choice != 8);

for (i = 0; i < ver1; i++)

{

delete[] smej1[i];

}

delete[] smej1;

for (i = 0; i < ver2; i++)

{

delete[] smej2[i];

}

delete[] smej2;

delete[] MassSpis1;

delete[] MassSpis2;

}