Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Определение характеристик графа»

**Выполнили студенты группы 21вв1.1:**

Кузнецов А.С.

Гришин С.В.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

**Задание 1**: Сгенерировать матрицу смежности для неориентированного графа G. Вывести матрицу на экран. Определить размер графа, используя матрицу смежности , найти изолированные , концевые и доминирующие вершины.

**Задание 2**: Построить для графа G матрицу инцидентности .Определить размер графа , найти изолированные , концевые и доминирующие вершины.

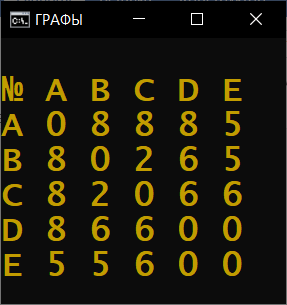
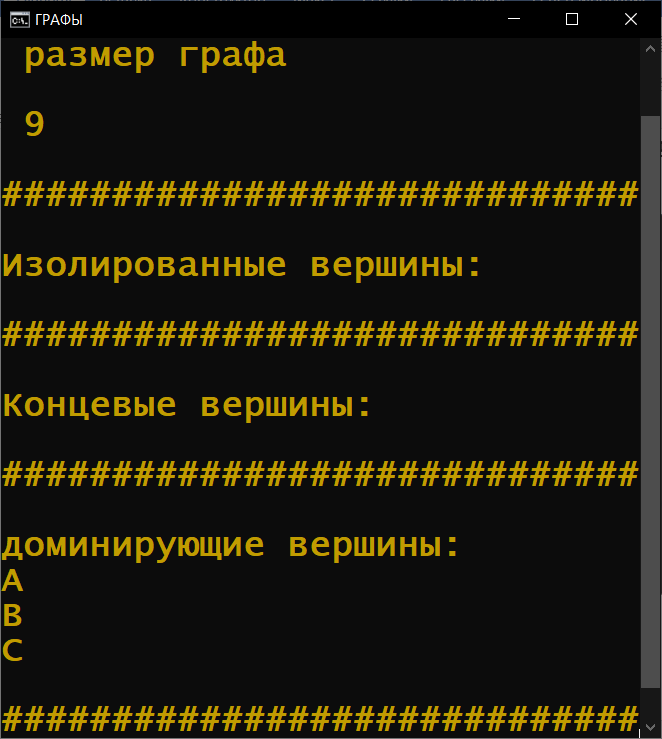
**Описание метода решения задачи:**

**Задание 1**: Создаем массив и заполняем его как матрицу смежности , определяем размер графа, находим степень вершины для каждого ребра и определяем , чем они являются : изолированными , концевыми или доминирующими.

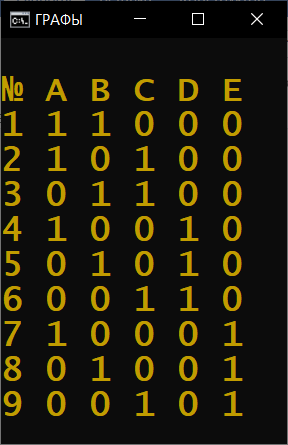
**Задание 2**:По матрице смежности создаем матрицу инцидентности , определяем размер графа , проверяем каждую вершину , изолирована она или нет (нет путей ни в одну другую вершину графа) , доминирующая или нет (есть пути во все вершины графа) , концевая она или нет ( есть путь только в одну вершину графа).

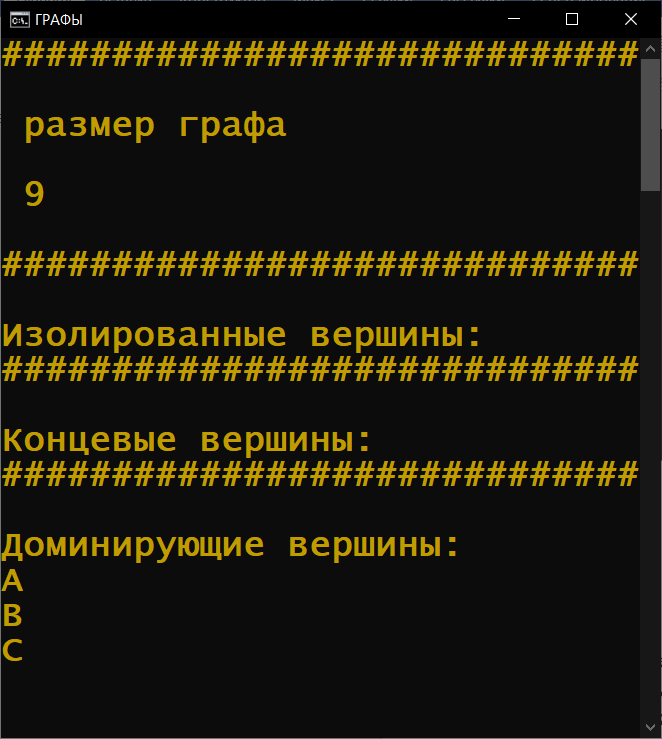
**Результаты работы программы:**

**Задание 1:**

**Задание 2:**





**Вывод:** в ходе выполнения лабораторной работы была реализована программа, которая строит матрицу смежности и определяет ее размер , наличие изолированных , концевых , доминирующих вершин , а на основе матрицы смежности строит матрицу инцидентности и определяет ее размер , наличие изолированных , концевых , доминирующих вершин.

**Листинг:**

using System;

namespace lab\_5\_6

{

internal class Program

{

const byte ww = 30;

const byte wh = 9;

const byte ww2 = 30;

const byte wh2 = 8;

struct graf

{

public int[,] matrIN, matrSM;

}

//СОЗДАНИЕ ГРАФА СМЕЖ \/ \/ \/

static int matinsm(graf matr, in int ras)

{

Random rand = new Random();

int col\_reb = 0;

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

{

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

{

if (i == j)

{

matr.matrSM[i, j] = 0;

}

else

{

matr.matrSM[i, j] = rand.Next(9);

if (matr.matrSM[i, j] != 0) { col\_reb++; }

matr.matrSM[j, i] = matr.matrSM[i, j];

}

}

}

return (col\_reb);

}

//СОЗДАНИЕ ГРАФА ИНТ \/ \/ \/

static void matinin(ref graf matr, in int col\_reb)

{

matr.matrIN = new int[col\_reb, Convert.ToInt32(Math.Sqrt(matr.matrSM.Length))];

int col\_reb\_int = 0;

for (int i = 0; i < Math.Sqrt(matr.matrSM.Length); i++)

{

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

{

if (matr.matrSM[i, j] != 0)

{

matr.matrIN[col\_reb\_int, i] = 1;

matr.matrIN[col\_reb\_int, j] = 1;

col\_reb\_int++;

}

}

}

}

//ИЗМЕНЕНИЕ РАЗМЕРА КОНСОЛИ \/ \/ \/

static void consize(int weigh, int hight)

{

Console.SetWindowSize(weigh - 1, hight - 1);

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

Console.SetWindowSize(weigh, hight);

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

}

//ВЫВОД МАТРИЦЫ СМЕЖНОСТИ \/ \/ \/

static void matprintsm(in int[,] matr)

{

Console.CursorVisible = false;

if (Math.Sqrt(matr.Length) < 5)

{

Console.WindowWidth = 11;

Console.WindowHeight = 11;

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

}

else

{

Console.WindowWidth = Convert.ToInt32(Math.Sqrt(matr.Length)) \* 2 + 2;

Console.WindowHeight = Convert.ToInt32(Math.Sqrt(matr.Length)) + 2;

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

}

for (int i = 0; i < Math.Sqrt(matr.Length) + 1; i++)

{

Console.WriteLine();

for (int j = 0; j < Math.Sqrt(matr.Length) + 1; j++)

{

if (j != 0 && i != 0)

{

Console.Write(" " + matr[i - 1, j - 1]);

continue;

}

if (i == 0 && j != 0)

{

Console.Write(" " + Convert.ToChar(64 + j));

continue;

}

if (j == 0 && i != 0)

{

Console.Write(Convert.ToChar(64 + i));

continue;

}

if (i == 0 && j == 0)

{

Console.Write("№");

continue;

}

}

}

}

//ВЫВОД ИНФОРМАЦИИ И О ГРАФЕ(см) \/ \/ \/

static void grafinf(graf matr)

{

Console.Clear();

consize(ww, wh + 11);

int rasm = 0;

for (int i = 1; i < Math.Sqrt(matr.matrSM.Length); i++)

{

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

{

if (matr.matrSM[i, j] != 0) { rasm++; }

}

}

Console.WriteLine("##############################");

Console.WriteLine(" размер графа\n");

Console.WriteLine(" " + rasm + "\n");

Console.WriteLine("##############################");

Console.Write("Изолированные вершины:\n");

for (int i = 0; i < Math.Sqrt(matr.matrSM.Length); i++)

{

bool ch = false;

for (int j = 0; j < Math.Sqrt(matr.matrSM.Length); j++)

{

if (matr.matrSM[i, j] != 0)

{

if (i == Math.Sqrt(matr.matrSM.Length) - 1)

{

goto ec1;

}

else

{

ch = true;

break;

}

}

}

if (ch == false)

{

Console.WriteLine(Convert.ToChar(65 + i));

Console.BufferHeight++;

}

}

ec1:

Console.Write("\n##############################\n");

Console.Write("Концевые вершины:\n");

for (int i = 0; i < Math.Sqrt(matr.matrSM.Length); i++)

{

rasm = 0;

for (int j = 0; j < Math.Sqrt(matr.matrSM.Length); j++)

{

if (matr.matrSM[i, j] != 0)

{

rasm++;

}

}

if (rasm == 1)

{

Console.WriteLine(Convert.ToChar(65 + i));

Console.BufferHeight++;

}

}

Console.Write("\n##############################\n");

Console.Write("доминирующие вершины:\n");

for (int i = 0; i < Math.Sqrt(matr.matrSM.Length); i++)

{

rasm = 0;

for (int j = 0; j < Math.Sqrt(matr.matrSM.Length); j++)

{

if (i != j)

{

if (matr.matrSM[i, j] != 0)

{

rasm++;

}

}

}

if (rasm == Math.Sqrt(matr.matrSM.Length) - 1)

{

Console.WriteLine(Convert.ToChar(65 + i));

Console.BufferHeight++;

}

}

Console.Write("\n#############################");

Console.ReadKey();

}

//ВЫВОД ИНФОРМАЦИИ И О ГРАФЕ(инт) \/ \/ \/

static void grafinfint(graf matr)

{

Console.Clear();

consize(ww, wh + 11);

Console.BufferHeight = 100;

int rasm = Convert.ToInt32(matr.matrIN.Length / Math.Sqrt(matr.matrSM.Length));

Console.WriteLine("##############################");

Console.WriteLine(" размер графа\n");

Console.WriteLine(" " + rasm + "\n");

Console.WriteLine("##############################");

Console.Write("Изолированные вершины:\n");

for (int i = 0; i < Math.Sqrt(matr.matrSM.Length); i++)

{

int ch = 0;

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

{

if (matr.matrIN[j, i] != 0)

{

ch++;

}

}

if (ch == 0)

{

Console.WriteLine(Convert.ToChar(65 + i));

}

}

Console.WriteLine("##############################");

Console.Write("Концевые вершины:\n");

for (int i = 0; i < Math.Sqrt(matr.matrSM.Length); i++)

{

int ch = 0;

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

{

if (matr.matrIN[j, i] != 0)

{

ch++;

}

}

if (ch == 1)

{

Console.WriteLine(Convert.ToChar(65 + i));

}

}

Console.WriteLine("##############################");

Console.Write("Доминирующие вершины:\n");

for (int i = 0; i < Math.Sqrt(matr.matrSM.Length); i++)

{

int ch = 0;

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

{

if (matr.matrIN[j, i] == 1)

{

ch++;

}

}

if (ch == Math.Sqrt(matr.matrSM.Length) - 1)

{

Console.WriteLine(Convert.ToChar(65 + i));

}

}

Console.ReadKey();

}

//ВЫВОД МАТРИЦЫ ИНЦИДЕНТНОСТИ \/ \/ \/

static void matprintin(graf matr)

{

Console.CursorVisible = false;

if (Math.Sqrt(matr.matrSM.Length) < 5)

{

Console.WindowWidth = 11;

Console.WindowHeight = 14;

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

}

else

{

Console.WindowWidth = Convert.ToInt32(Math.Sqrt(matr.matrSM.Length)) \* 2 + 3;

if (matr.matrIN.Length / Math.Sqrt(matr.matrSM.Length) \* 2 + 2 < 30)

{

Console.WindowHeight = Convert.ToInt32(matr.matrIN.Length / Math.Sqrt(matr.matrSM.Length)) + 2;

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

}

else

{

Console.WindowHeight = 30;

Console.SetBufferSize(Console.WindowWidth, Convert.ToInt32(matr.matrIN.Length / Math.Sqrt(matr.matrSM.Length)) + 2);

}

}

for (int i = 0; i < matr.matrIN.Length / Math.Sqrt(matr.matrSM.Length) + 1; i++)

{

Console.WriteLine();

for (int j = 0; j < Math.Sqrt(matr.matrSM.Length) + 1; j++)

{

if (j != 0 && i != 0)

{

Console.Write(" " + matr.matrIN[i - 1, j - 1]);

continue;

}

if (i == 0 && j != 0)

{

Console.Write(" " + Convert.ToChar(64 + j));

continue;

}

if (j == 0 && i != 0)

{

if (matr.matrIN.Length / Math.Sqrt(matr.matrSM.Length) >= 10)

{

if (i < 10) { Console.Write(" " + i); }

else { Console.Write(i); }

}

else { Console.Write(i); }

continue;

}

if (i == 0 && j == 0)

{

if (matr.matrIN.Length / Math.Sqrt(matr.matrSM.Length) < 10) { Console.Write("№"); }

else { Console.Write(" №"); }

continue;

}

}

}

}

//ИЗМЕНЕНИЕ ЦВЕТА КОНСОЛИ \/ \/ \/ (0 - жёлтый; 1 - зелённый; 2 - красный)

static void sc(byte mod)

{

switch (mod)

{

case 0:

Console.ForegroundColor = ConsoleColor.DarkYellow;

break;

case 1:

Console.ForegroundColor = ConsoleColor.DarkGreen;

break;

case 2:

Console.ForegroundColor = ConsoleColor.DarkRed;

break;

}

}

static void Main(string[] args)

{

//НАСТРОЙКА КОНСОЛИ начало

Console.Title = "ГРАФЫ";

sc(0);

consize(ww, wh);

//Console.CursorVisible = false; //невидимый курсор

//НАСТРОЙКА КОНСОЛИ конец

//ПЕРЕМЕННЫЕ начало

graf A = new graf(); //ГРАФ А

ref graf mat = ref A;

int mod = 0; //ВЫБОР ДЕЙСТВИЯ

//ПЕРЕМЕННЫЕ конец

MenuMain:

Console.WriteLine("##############################");

Console.WriteLine(" 1) Создать граф");

Console.WriteLine(" 2) Вывести матрицу графа");

Console.WriteLine(" 3) Редактирование графа");

Console.WriteLine(" 4) Информация о графе(см)");

Console.WriteLine(" 5) Информация о граф(инт)\n");

Console.Write("#############################");

mod = Convert.ToInt32(Console.ReadKey().KeyChar);

if (mod < 49 || mod > 53)

{

consize(ww, wh);

Console.Clear();

goto MenuMain;

}

switch (mod)

{

case 49://СОЗДАНИЕ ГРАФА

{

int ras;

Console.Clear();

consize(ww, 8);

ERROR1:

Console.WriteLine("##############################");

Console.WriteLine(" Введите размер графа");

Console.WriteLine(" (от 0 до 14)\n\n\n");

Console.Write("#############################");

Console.SetCursorPosition(15, Console.WindowHeight - 3);

if (int.TryParse(Console.ReadLine(), out ras) == false)

{

Console.Clear();

consize(ww, 12);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" введено некоректное число\n");

sc(0);

goto ERROR1;

}

if (ras < 0)

{

Console.Clear();

consize(ww, 13);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" Размер матрицы не может");

Console.WriteLine(" быть отрицательным!\n");

sc(0);

goto ERROR1;

}

if (ras > 14)

{

Console.Clear();

consize(ww, 12);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" превышен максимальный размер\n");

sc(0);

goto ERROR1;

}

int col\_reb;

mat.matrSM = new int[ras, ras];

col\_reb = matinsm(mat, ras);

mat.matrIN = new int[col\_reb, ras];

matinin(ref mat, col\_reb);

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("##############################");

sc(1);

Console.WriteLine(" !граф успешно создан!\n");

sc(0);

goto MenuMain;

}

case 50://ВЫВОД ГРАФА

{

if (mat.matrSM == null)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" !граф пуст!\n");

sc(0);

goto MenuMain;

}

Console.Clear();

consize(ww, 10);

ERROR2:

Console.WriteLine("##############################");

Console.WriteLine(" выберите тип матрицы\n");

Console.WriteLine("##############################");

Console.WriteLine(" 1) матрица смежности");

Console.WriteLine(" 2) матрица инцидентности\n");

Console.Write("#############################");

mod = Convert.ToInt32(Console.ReadKey().KeyChar);

if (mod < 49 || mod > 50)

{

Console.Clear();

goto ERROR2;

}

Console.Clear();

switch (mod)

{

case 49: //МАТРИЦА СМЕЖНОСТИ

{

matprintsm(mat.matrSM);

break;

}

case 50: //МАТРИЦА ИНЦИДЕНТНОСТИ

{

matprintin(mat);

break;

}

}

Console.ReadKey();

Console.CursorVisible = true;

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

case 51://РЕДАКТИРОВАНИЕ ГРАФА

{

if (mat.matrSM == null)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" !граф пуст!\n");

sc(0);

goto MenuMain;

}

Console.Clear();

consize(ww2, wh2);

MenuRed:

Console.WriteLine("##############################");

Console.WriteLine(" 1) изолировать вершину");

Console.WriteLine(" 2) удалить ребро");

Console.WriteLine(" 3) добавить ребро");

Console.WriteLine(" 4) назад\n");

Console.Write("#############################");

mod = Convert.ToInt32(Console.ReadKey().KeyChar);

if (mod < 49 || mod > 52)

{

Console.Clear();

goto MenuRed;

}

switch (mod)

{

case 49://ИЗОЛЯЦИЯ ВЕРШИНЫ

{

bool ch = true;

int cp;

er2:

Console.Clear();

consize(ww2, Convert.ToInt32(Math.Sqrt(mat.matrSM.Length)) + 3);

Console.WriteLine("Выберите вершину: ");

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

Console.WriteLine(i + 1 + ") " + Convert.ToChar(i + 65));

}

ch = int.TryParse(Console.ReadLine(), out cp);

if ((ch == false) || (cp < 0 || cp > Math.Sqrt(mat.matrSM.Length)))

{

Console.Clear();

goto er2;

}

if (cp < 1 || cp > Math.Sqrt(mat.matrSM.Length))

{

goto er2;

}

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

mat.matrSM[i, cp - 1] = 0;

mat.matrSM[cp - 1, i] = 0;

}

int col\_reb = 0;

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

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

{

if

(mat.matrSM[i, j] != 0) { col\_reb++; }

}

}

mat.matrIN = new int[col\_reb, Convert.ToInt32(Math.Sqrt(mat.matrSM.Length))];

matinin(ref mat, col\_reb);

Console.Clear();

consize(ww2, wh2 + 4);

Console.WriteLine("##############################");

sc(1);

Console.WriteLine(" !вершина изолирована!\n");

sc(0);

goto MenuRed;

}

case 50://УДАЛЕНИЕ РЕБРА

{

int v1, v2;

bool ch = true;

cc50a:

consize(ww2, Convert.ToInt32(Math.Sqrt(mat.matrSM.Length)) + 8);

Console.Clear();

Console.WriteLine("##############################");

Console.WriteLine(" выберите ребро\n");

Console.WriteLine("##############################");

Console.WriteLine("Вершина 1:");

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

Console.WriteLine(i + 1 + ") " + Convert.ToChar(i + 65));

}

ch = int.TryParse(Console.ReadLine(), out v1);

if ((ch == false) || (v1 < 0 || v1 > Math.Sqrt(mat.matrSM.Length)))

{

Console.Clear();

goto cc50a;

}

v1--;

cc50b:

consize(ww2, Convert.ToInt32(Math.Sqrt(mat.matrSM.Length)) + 7);

Console.Clear();

Console.WriteLine("##############################");

Console.WriteLine(" выберите ребро\n");

Console.WriteLine("##############################");

Console.WriteLine("Вершина 2:");

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

Console.WriteLine(i + 1 + ") " + Convert.ToChar(i + 65));

}

ch = int.TryParse(Console.ReadLine(), out v2);

if ((ch == false) || (v2 < 0 || v2 > Math.Sqrt(mat.matrSM.Length)))

{

Console.Clear();

goto cc50b;

}

v2--;

if (v1 == v2)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" выбрана 1 вершина\n");

sc(0);

goto MenuRed;

}

if (mat.matrSM[v1, v2] == 0)

{

Console.Clear();

consize(ww, wh + 5);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" выбранное ребро");

Console.WriteLine(" не существует\n");

sc(0);

goto MenuRed;

}

mat.matrSM[v1, v2] = 0;

mat.matrSM[v2, v1] = 0;

int col\_reb = 0;

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

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

{

if (mat.matrSM[i, j] != 0) { col\_reb++; }

}

}

mat.matrIN = new int[col\_reb, Convert.ToInt32(Math.Sqrt(mat.matrSM.Length))];

matinin(ref mat, col\_reb);

Console.Clear();

consize(ww2, wh2 + 4);

Console.WriteLine("##############################");

sc(1);

Console.WriteLine(" ребро успешно удалено\n");

sc(0);

goto MenuRed;

}

case 51://ДОБАВЛЕНИЕ РЕБРА

{

int v1, v2;

bool ch = true;

cc51a:

consize(ww2, Convert.ToInt32(Math.Sqrt(mat.matrSM.Length)) + 8);

Console.Clear();

Console.WriteLine("##############################");

Console.WriteLine(" выберите ребро\n");

Console.WriteLine("##############################");

Console.WriteLine("Вершина 1:");

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

Console.WriteLine(i + 1 + ") " + Convert.ToChar(i + 65));

}

ch = int.TryParse(Console.ReadLine(), out v1);

if ((ch == false) || (v1 < 0 || v1 > Math.Sqrt(mat.matrSM.Length)))

{

Console.Clear();

goto cc51a;

}

v1--;

cc51b:

consize(ww2, Convert.ToInt32(Math.Sqrt(mat.matrSM.Length)) + 7);

Console.Clear();

Console.WriteLine("##############################");

Console.WriteLine(" выберите ребро\n");

Console.WriteLine("##############################");

Console.WriteLine("Вершина 2:");

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

Console.WriteLine(i + 1 + ") " + Convert.ToChar(i + 65));

}

ch = int.TryParse(Console.ReadLine(), out v2);

if ((ch == false) || (v2 < 0 || v2 > Math.Sqrt(mat.matrSM.Length - 1)))

{

Console.Clear();

goto cc51b;

}

v2--;

if (v1 == v2)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" выбрана 1 вершина\n");

sc(0);

goto MenuRed;

}

if (mat.matrSM[v1, v2] != 0)

{

Console.Clear();

consize(ww, wh + 5);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" выбранное ребро");

Console.WriteLine(" уже существует\n");

sc(0);

goto MenuRed;

}

Console.Clear();

int ras;

ERROR4:

Console.WriteLine("##############################");

Console.WriteLine(" введите вес ребра");

Console.WriteLine(" (от 0 до 9)\n\n\n");

Console.Write("#############################");

Console.SetCursorPosition(15, Console.WindowHeight - 3);

if (int.TryParse(Console.ReadLine(), out ras) == false)

{

Console.Clear();

consize(ww2, 12);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" введено некоректное число\n");

sc(0);

goto ERROR4;

}

if (ras < 0)

{

Console.Clear();

consize(ww2, 13);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" вес ребра не может");

Console.WriteLine(" быть отрицательным\n");

sc(0);

goto ERROR4;

}

if (ras > 9)

{

Console.Clear();

consize(ww2, 12);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" превышен максимальный размер\n");

sc(0);

goto ERROR4;

}

mat.matrSM[v1, v2] = ras;

mat.matrSM[v2, v1] = ras;

int col\_reb = 0;

for (int i = 0; i < Math.Sqrt(mat.matrSM.Length); i++)

{

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

{

if (mat.matrSM[i, j] != 0) { col\_reb++; }

}

}

mat.matrIN = new int[col\_reb, Convert.ToInt32(Math.Sqrt(mat.matrSM.Length))];

matinin(ref mat, col\_reb);

Console.Clear();

consize(ww2, wh2 + 4);

Console.WriteLine("##############################");

sc(1);

Console.WriteLine(" ребро успешно добавленно\n");

sc(0);

goto MenuRed;

}

case 52://ВОЗВРАЩЕНИЕ В ГЛАВНОЕ МЕНЮ

{

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

}

break;

}

case 52://ВЫВОД ИНФОРМАЦИИ О ГРАФЕ

{

if (mat.matrSM == null)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" !граф пуст!\n");

sc(0);

goto MenuMain;

}

grafinf(mat);

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

case 53:

{

if (mat.matrSM == null)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("##############################");

sc(2);

Console.WriteLine(" !граф пуст!\n");

sc(0);

goto MenuMain;

}

grafinfint(mat);

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

}

}

}

}