Министерство 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 21вв1.1:**

Кузнецов А.С.

Гришин С.В.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Цель:** научиться работать с графами с помощью унарных и бинарный операций.

**Методические указания.**

**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 https://lh4.googleusercontent.com/yk7U1RRB9DcYI6EYXMZ69GzFlx3H28m-NSRrwzRmophzZGbuifNDqyUv8ySDo0_JFTEm5PW_J4yXo4q7cLnV6fPjY7TASec22ttwN7--w6XKu8uK7Pn_oW2t26fi3rBOP3bhlyYSd879q8AWdylmF3Hry_59MBS5HdRGdktkTUWhEgjO5tZabojDRmGJrPt_cj62c1rZlg *G*2

б) пересечения  *G* = *G*1 https://lh4.googleusercontent.com/2cPI0AQimsydRtulIQ4kq2VYJICR-pbdgK4cBafR8bgVhZnftdkbpva8Jz2dq3SvPmA0vfddRVGOxNh_NA_x8sI-kaMclGyNAbZzbivIDBehv0GQCq6UQtV8jW4c9ez9TxnBhDmaOVK27q4BAAiMKYCgupwacSIp-641X7osu8uRPeBgVaj7iL4akBvF_GhTXIvkfXOdBA *G*2

в) кольцевой суммы *G* = *G*1 https://lh4.googleusercontent.com/Qw3MAfpI592XdQb4wKJKOQQzp0qNXhBW5B31uxG0HXRd47QzswXl2tINm-X0T0-Sc0loxAIHomj-5HBRHxpGxSnta1M8RGRCSTM8cf2YksUxptw1UlvybAh9B4059z5UBfsJ12Povi7fgYR13zobBR_vPAneTHvzhoWWTP3YbBpGdbGoEAFXr_2MFYL2I75ane1wDYPEjA *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.

**Описание метода решения задачи:**

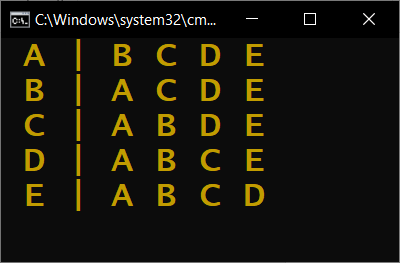
1)Сгенерировать две матрицы смежности и на их основе задать список смежности.

2)Реализовать операции над вершинами для матричной формы представление и для представления в виде списка смежности.

3)Реализовать операции над графами и вывести результаты.

**Результаты работы программы:**

1)Вывели преобразованные матрицы в матрицу смежности и список смежности.



2) Реализованы операции с графами для матрицы смежности :

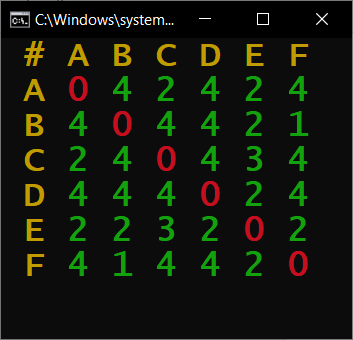
а) отождествления вершин

б) стягивания ребра

в) расщепления вершины



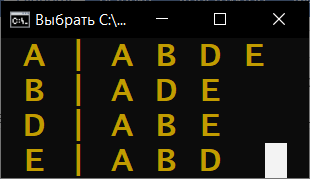


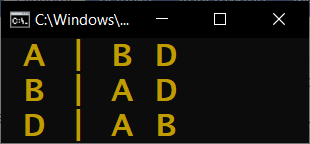
  
3)Реализованы операции с графами для списка смежности :

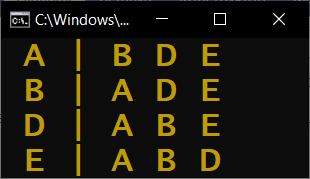
а) отождествления вершин

б) стягивания ребра

в) расщепления вершины







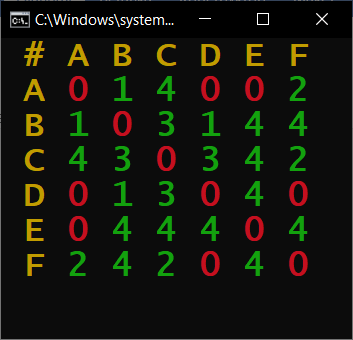
4) Для матричной формы представления графов выполнили операции:

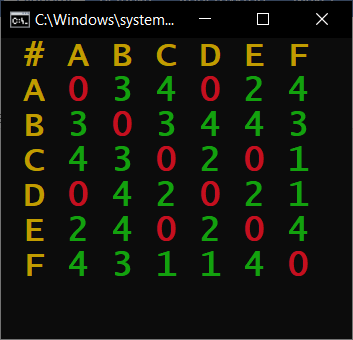
а) объединения *G* = *G*1 https://lh4.googleusercontent.com/yk7U1RRB9DcYI6EYXMZ69GzFlx3H28m-NSRrwzRmophzZGbuifNDqyUv8ySDo0_JFTEm5PW_J4yXo4q7cLnV6fPjY7TASec22ttwN7--w6XKu8uK7Pn_oW2t26fi3rBOP3bhlyYSd879q8AWdylmF3Hry_59MBS5HdRGdktkTUWhEgjO5tZabojDRmGJrPt_cj62c1rZlg *G*2

б) пересечения  *G* = *G*1 https://lh4.googleusercontent.com/2cPI0AQimsydRtulIQ4kq2VYJICR-pbdgK4cBafR8bgVhZnftdkbpva8Jz2dq3SvPmA0vfddRVGOxNh_NA_x8sI-kaMclGyNAbZzbivIDBehv0GQCq6UQtV8jW4c9ez9TxnBhDmaOVK27q4BAAiMKYCgupwacSIp-641X7osu8uRPeBgVaj7iL4akBvF_GhTXIvkfXOdBA *G*2

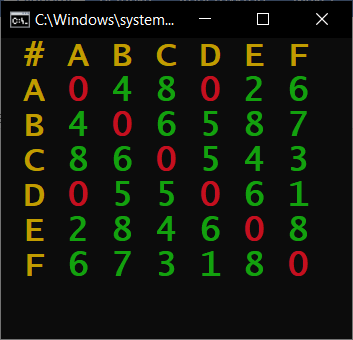
в) кольцевой суммы *G* = *G*1 https://lh4.googleusercontent.com/Qw3MAfpI592XdQb4wKJKOQQzp0qNXhBW5B31uxG0HXRd47QzswXl2tINm-X0T0-Sc0loxAIHomj-5HBRHxpGxSnta1M8RGRCSTM8cf2YksUxptw1UlvybAh9B4059z5UBfsJ12Povi7fgYR13zobBR_vPAneTHvzhoWWTP3YbBpGdbGoEAFXr_2MFYL2I75ane1wDYPEjA *G*2

Сгенерированные матрицы:

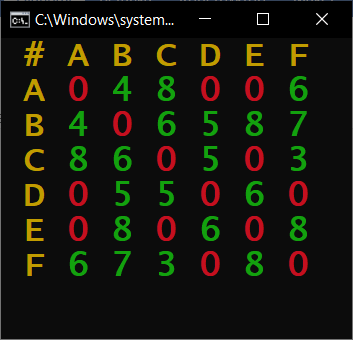




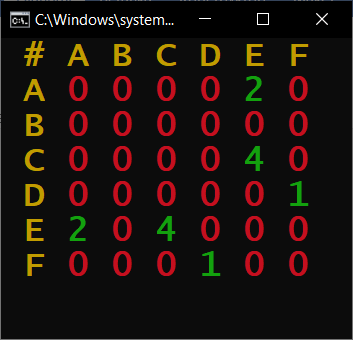
объединение *G* = *G*1 https://lh4.googleusercontent.com/yk7U1RRB9DcYI6EYXMZ69GzFlx3H28m-NSRrwzRmophzZGbuifNDqyUv8ySDo0_JFTEm5PW_J4yXo4q7cLnV6fPjY7TASec22ttwN7--w6XKu8uK7Pn_oW2t26fi3rBOP3bhlyYSd879q8AWdylmF3Hry_59MBS5HdRGdktkTUWhEgjO5tZabojDRmGJrPt_cj62c1rZlg *G*2



пересечение  *G* = *G*1 https://lh4.googleusercontent.com/2cPI0AQimsydRtulIQ4kq2VYJICR-pbdgK4cBafR8bgVhZnftdkbpva8Jz2dq3SvPmA0vfddRVGOxNh_NA_x8sI-kaMclGyNAbZzbivIDBehv0GQCq6UQtV8jW4c9ez9TxnBhDmaOVK27q4BAAiMKYCgupwacSIp-641X7osu8uRPeBgVaj7iL4akBvF_GhTXIvkfXOdBA *G*2



кольцевая сумма *G* = *G*1 https://lh4.googleusercontent.com/Qw3MAfpI592XdQb4wKJKOQQzp0qNXhBW5B31uxG0HXRd47QzswXl2tINm-X0T0-Sc0loxAIHomj-5HBRHxpGxSnta1M8RGRCSTM8cf2YksUxptw1UlvybAh9B4059z5UBfsJ12Povi7fgYR13zobBR_vPAneTHvzhoWWTP3YbBpGdbGoEAFXr_2MFYL2I75ane1wDYPEjA *G*2



**Вывод:** в ходе выполнения лабораторной работы, мы научились реализовывать операции над вершинами графов и операции с графами. Также научились реализовывать представление операций над вершинам списком смежности.

**Листинг:**

using System;

using System.Collections.Generic;

using System.Runtime.InteropServices;

namespace lb6

{

internal class Program

{

const byte ww = 30;

const byte wh = 10;

const byte ww2 = 30;

const byte wh2 = 10;

struct graf

{

public int[,] matrSM, matrIN;

public int[][] listSM;

}//граф

//ОБХОДВШ \/ \/ \/

static void obhodvsh(int[,] matr, ref bool[] be, ref Queue<int> q, ref bool nach, int pos = 1)

{

if (be[pos - 1] == false)

{

be[pos - 1] = true;

if (pos != 1) { Console.Write(">"); }

Console.Write(Convert.ToChar(matr[pos, 0] + 64));

}

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

{

if (matr[pos, i] != 0 && be[i - 1] == false)

{

be[i - 1] = true;

q.Enqueue(i);

Console.Write(">" + Convert.ToChar(matr[0, i] + 64));

}

}

}

//ОБХОДВГрекурсия,список \/ \/ \/

static void obhodreclist(int[][] list, ref bool[] be, int pos = 1)

{

if (be[pos - 1] == false)

{

be[pos - 1] = true;

if (pos != 1) { Console.Write(">"); }

Console.Write(Convert.ToChar(list[pos - 1][0] + 64));

}

for (int i = 1; i < list[pos - 1].Length; i++)

{

int j = 0;

while (list[pos - 1][i] != list[j][0] || pos - 1 == j)

{

j++;

}

if (be[j] == false)

{

obhodreclist(list, ref be, j + 1);

}

}

}

//ОБХОДВГрекурсия,матрица \/ \/ \/

static void obxodrecm(in int[,] matr, ref bool[] be, int pos = 1)

{

if (be[pos - 1] == false)

{

be[pos - 1] = true;

if (pos != 1) { Console.Write(">"); }

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

}

for (int i = 1; i <= be.Length; i++)

{

if (matr[pos, i] != 0 && be[i - 1] == false)

{

obxodrecm(matr, ref be, i);

}

}

}

//ГЕНЕРАЦИЯ МАТРИЦЫ СМЕЖНОСТИ \/ \/ \/

static void matrSMgen(graf gr, int rand\_key)

{

Random rand = new Random(rand\_key);

int ras = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length));

for (int i = 1; i < ras; i++)

{

gr.matrSM[i, 0] = i;

gr.matrSM[0, i] = i;

}

gr.matrSM[0, 0] = -29;

for (int i = 1; i < ras; i++)

{

for (int j = 1; j < i; j++)

{

gr.matrSM[i, j] = rand.Next(5);

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

}

}

}

//ГЕНЕРАЦИЯ МАТРИЦЫ ИНЦИДЕНТНОСТИ \/ \/ \/

static int[,] matrINgen(graf grp, int v = 0)

{

graf gr = new graf();

gr.matrSM = new int[Convert.ToInt32(Math.Sqrt(grp.matrSM.Length)), Convert.ToInt32(Math.Sqrt(grp.matrSM.Length))];

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

{

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

{

gr.matrSM[i, j] = grp.matrSM[i, j];

}

}

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

{

gr.matrSM[i, 0] = i;

gr.matrSM[0, i] = i;

}

int ras = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length));

int col\_reb = 0;

for (int i = 1; i < ras; i++)

{

for (int j = 1; j <= i; j++)

{

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

}

}

gr.matrIN = new int[col\_reb + 1, ras];

for (int i = 1; i < ras; i++) { gr.matrIN[0, i] = gr.matrSM[0, i]; }

for (int i = 1; i < col\_reb + 1; i++) { gr.matrIN[i, 0] = i; }

gr.matrIN[0, 0] = -29;

int col\_reb\_int = 1;

for (int i = 1; i < ras; i++)

{

for (int j = 1; j <= i; j++)

{

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

{

if (v != 0 && i >= v)

{

gr.matrIN[col\_reb\_int, gr.matrSM[i, 0] - 1] = 1;

}

else

{

gr.matrIN[col\_reb\_int, gr.matrSM[i, 0]] = 1;

}

if (v != 0 && j >= v)

{

gr.matrIN[col\_reb\_int, gr.matrSM[0, j] - 1] = 1;

}

else

{

gr.matrIN[col\_reb\_int, gr.matrSM[0, j]] = 1;

}

col\_reb\_int++;

}

}

}

return (gr.matrIN);

}

//ГЕНЕРАЦИЯ СПИСКА СМЕЖНОСТИ \/ \/ \/

static void listSMgen(ref graf gr)

{

gr.listSM = new int[Convert.ToInt32(Math.Sqrt(gr.matrSM.Length)) - 1][];

int ch = 0, chl = 0;

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

{

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

{

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

{

ch++;

}

}

gr.listSM[chl] = new int[ch + 1];

gr.listSM[chl][0] = gr.matrSM[i, 0];

int chll = 1;

chl++;

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

{

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

{

gr.listSM[i - 1][chll] = gr.matrSM[0, j];

chll++;

}

}

ch = 0;

}

}

//ВЫВОД МАТРИЦЫ СМЕЖНОСТИ \/ \/ \/

static void matrSMprint(graf gr)

{

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

{

Console.WindowWidth = 11;

Console.WindowHeight = 11;

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

}

else

{

Console.WindowWidth = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length)) \* 2 + 1;

Console.WindowHeight = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length)) + 1;

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

}

int ras = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length));

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

{

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

{

if (i == 0 || j == 0)

{

Console.Write(" " + Convert.ToChar(gr.matrSM[i, j] + 64));

}

else

{

if (true) { if (gr.matrSM[i, j] == 0) { sc(2); } else { sc(1); } }//ЦВЕТНАЯ ТАБЛИЦА

Console.Write(" " + gr.matrSM[i, j]);

sc(0);

}

}

Console.WriteLine();

}

}

//ВЫВОД МАТРИЦЫ ИНЦИДЕНТНОСТИ \/ \/ \/

static void matrINprint(graf gr)

{

int ras = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length));

int col\_reb = Convert.ToInt32(gr.matrIN.Length / ras);

if (ras < 5)

{

Console.WindowWidth = 11;

Console.WindowHeight = 7;

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

}

else

{

Console.WindowWidth = ras \* 2 + 3;

if (col\_reb \* 2 + 2 < 30)

{

Console.WindowHeight = col\_reb + 2;

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

}

else

{

Console.WindowHeight = 25;

Console.SetBufferSize(Console.WindowWidth, col\_reb + 2);

}

}

for (int i = 0; i < col\_reb; i++)

{

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

{

if (col\_reb > 9 && i < 10 && j == 0) { Console.Write(" "); }

if (i == 0)

{

Console.Write(" " + Convert.ToChar(gr.matrIN[i, j] + 64));

}

else

{

Console.Write(" " + gr.matrIN[i, j]);

}

}

Console.WriteLine();

}

}

//ВЫВОД СПИСКА СМЕЖНОСТИ \/ \/ \/

static void listSMprint(graf gr)

{

Console.CursorVisible = false;

int maxw = 0;

for (int i = 0; i < gr.listSM.Length; i++)

{

if (maxw < gr.listSM[i].Length) { maxw = gr.listSM[i].Length; }

}

if (maxw < 5) { maxw = 5; }

consize(maxw \* 2 + 4, gr.listSM.Length);

for (int i = 0; i < gr.listSM.Length; i++)

{

if (i != 0) { Console.WriteLine(); }

for (int j = 0; j < gr.listSM[i].Length; j++)

{

if (j == 0) { Console.Write(" " + Convert.ToChar(gr.listSM[i][j] + 64) + " |"); }

else { Console.Write(" " + Convert.ToChar(gr.listSM[i][j] + 64)); }

}

}

}

//УДАЛЕНИЕ ТОЧКИ \/ \/ \/

static graf delpoint(ref graf matr, int v)

{

graf buf = matr;

graf newmat = matr;

newmat.matrSM = new int[Convert.ToInt32(Math.Sqrt(buf.matrSM.Length)) - 1, Convert.ToInt32(Math.Sqrt(buf.matrSM.Length)) - 1];

int mx = 0, my = 0;

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

{

if (buf.matrSM[i, 0] != v)

{

mx = 0;

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

{

if (buf.matrSM[0, j] != v)

{

newmat.matrSM[my, mx] = buf.matrSM[i, j];

mx++;

}

}

my++;

}

}

return (newmat);

}

//СОЗДАНИЕ ТОЧКИ \/ \/ \/

static graf addpoint(ref graf matr)

{

graf buf = matr;

matr.matrSM = new int[Convert.ToInt32(Math.Sqrt(matr.matrSM.Length)) + 1, Convert.ToInt32(Math.Sqrt(matr.matrSM.Length)) + 1];

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

{

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

{

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

}

}

matr.matrSM[0, Convert.ToInt32(Math.Sqrt(matr.matrSM.Length)) - 1] = matr.matrSM[0, Convert.ToInt32(Math.Sqrt(matr.matrSM.Length)) - 2] + 1;

matr.matrSM[Convert.ToInt32(Math.Sqrt(matr.matrSM.Length)) - 1, 0] = matr.matrSM[0, Convert.ToInt32(Math.Sqrt(matr.matrSM.Length)) - 1];

return (matr);

}

//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;

}

}

//ОТОЖДЕСТВЛЕНИЕ ВЕРШИН

static graf otver(ref graf grp, int v1, int v2, bool reb = false)

{

graf gr = new graf();

int[] key = new int[Convert.ToInt32(Math.Sqrt(grp.matrSM.Length)) - 1];

int nom = 0;

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

{

if (grp.matrSM[0, i] == v2) { v2 = i; }

if (grp.matrSM[0, i] == v1) { v1 = i; }

if (grp.matrSM[0, i] != v2)

{

key[nom] = grp.matrSM[0, i];

nom++;

}

}

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

{

grp.matrSM[0, i] = i;

}

gr.matrSM = new int[Convert.ToInt32(Math.Sqrt(grp.matrSM.Length)), Convert.ToInt32(Math.Sqrt(grp.matrSM.Length))];

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

{

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

{

gr.matrSM[j, i] = grp.matrSM[i, j];

}

}

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

{

grp.matrSM[v1, i] = gr.matrSM[v2, i] + gr.matrSM[v1, i];

gr.matrSM[i, v1] = gr.matrSM[v1, i];

}

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

{

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

{

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

}

}

grp = delpoint(ref grp, v2);

for (int i = 0; i < key.Length - 1; i++)

{

grp.matrSM[0, i + 1] = key[i];

grp.matrSM[i + 1, 0] = key[i];

}

if (reb == true)

{

for (int i = 1; i < Math.Sqrt(grp.matrSM.Length); i++) { grp.matrSM[i, i] = 0; }

}

return (grp);

}

//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);

}

static void Main(string[] args)

{

graf A = new graf();

graf B = new graf();

graf AB = new graf();

Console.CursorVisible = false;

int mod = 0;

sc(0);

consize(ww, wh);

MenuMain:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" 1) Сгенерировать графы");

Console.WriteLine(" 2) Вывести матрицы графов");

Console.WriteLine(" 3) Операции с вершинами");

Console.WriteLine(" 4) Операции с графами");

Console.WriteLine(" 5) Обход в глубину");

Console.WriteLine(" 6) Обход в ширину\n");

Console.Write("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

if ((mod < 49 || mod > 54) && mod != 32)

{

Console.Clear();

consize(ww, wh);

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;

}

ras++;

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

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

matrSMgen(A, ras);

matrSMgen(B, ras + 10);

A.matrIN = matrINgen(A);

B.matrIN = matrINgen(B);

listSMgen(ref A);

listSMgen(ref B);

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

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

sc(0);

goto MenuMain;

}

case 50://ВЫВОД ГРАФОВ

{

if (A.matrSM == null)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(2);

Console.WriteLine(" графы не сгенерированы\n");

sc(0);

goto MenuMain;

}

Console.Clear();

if (AB.matrSM == null)

{

ER\_501:

consize(ww, 10);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" ВЫБЕРИТЕ ГРАФ\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" 1) A");

Console.WriteLine(" 2) B\n");

Console.Write("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

if (mod < 49 || mod > 50)

{

Console.Clear();

consize(ww, 10);

goto ER\_501;

}

}

else

{

ER\_502:

consize(ww, 11);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" ВЫБЕРИТЕ ГРАФ\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" 1) A");

Console.WriteLine(" 2) B");

Console.WriteLine(" 3) AB\n");

Console.Write(" ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

if (mod < 49 || mod > 51)

{

Console.Clear();

consize(ww, 11);

goto ER\_502;

}

}

Console.Clear();

graf buf = new graf();

if (mod == 49) { buf = A; }

if (mod == 50) { buf = B; }

if (mod == 51) { buf = AB; }

ER\_503:

consize(ww, 11);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" ВЫБЕРИТЕ ТИП МАТРИЦЫ\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

Console.WriteLine(" 2) Матрица инцидентности");

Console.WriteLine(" 3) Список смежности\n");

Console.Write("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

if (mod < 49 || mod > 51)

{

Console.Clear();

consize(ww, 10);

goto ER\_503;

}

Console.Clear();

switch (mod)

{

case 49:

{

matrSMprint(buf);

Console.ReadKey();

break;

}

case 50:

{

matrINprint(buf);

Console.ReadKey();

break;

}

case 51:

{

listSMprint(buf);

Console.ReadKey();

break;

}

}

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

case 51://ОПЕРАЦИИ С ВЕРШИНАМИ

{

if (A.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);

MenuRed2:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" 1) отождествеление вершин");

Console.WriteLine(" 2) стягивание ребра");

Console.WriteLine(" 3) расщепление вершины");

Console.WriteLine(" 4) Изоляция вершины");

Console.WriteLine(" 5) Удаление ребра");

Console.WriteLine(" 6) назад\n");

Console.Write(" ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

if (mod < 49 || mod > 54)

{

Console.Clear();

consize(ww2, wh2);

goto MenuRed2;

}

switch (mod)

{

case 49://ОТОЖДЕСТВЛЕНИЕ ВЕРШИН

{

int v1, v2;

Console.Clear();

consize(ww2, 6 + Convert.ToInt32(Math.Sqrt(A.matrSM.Length)));

ove1:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" выберите первую вершину\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

{

Console.WriteLine(" " + i + ") " + Convert.ToChar(A.matrSM[i, 0] + 64));

}

if (int.TryParse(Console.ReadLine(), out v1) == false) { Console.Clear(); goto ove1; }

if (v1 < 1 || v1 > Math.Sqrt(A.matrSM.Length) - 1)

{

Console.Clear();

goto ove1;

}

Console.Clear();

consize(ww2, 6 + Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1);

ove2:

int[,] choos = new int[2, Convert.ToInt32(Math.Sqrt(A.matrSM.Length) - 1)];

int nom = 0;

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

{

if (A.matrSM[0, i] != v1)

{

choos[0, nom] = nom + 1;

choos[1, nom] = A.matrSM[0, i];

nom++;

}

}

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" выберите вторую вершину\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

{

Console.WriteLine(" " + choos[0, i] + ")" + Convert.ToChar(choos[1, i] + 64));

}

if (int.TryParse(Console.ReadLine(), out v2) == false)

{ Console.Clear(); goto ove2; }

if (v2 < 1 || v2 > choos.Length / 2)

{

Console.Clear();

goto ove2;

}

v2 = choos[1, v2 - 1];

if (v1 > v2) { int b = v1; v1 = v2; v2 = b; }

A = otver(ref A, v1, v2);

A.matrIN = matrINgen(A, v2);

listSMgen(ref A);

B = otver(ref B, v1, v2);

B.matrIN = matrINgen(B, v2);

listSMgen(ref B);

Console.Clear();

consize(ww2, wh2 + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" операция успешно завершена\n");

sc(0);

goto MenuRed2;

}

case 50://СТЯГИВАНИЕ РЕБРА

{

int v1, v2, col = 0;

Console.Clear();

consize(ww2, 6 + Convert.ToInt32(Math.Sqrt(A.matrSM.Length)));

ove1:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" выберите первую вершину\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

{

Console.WriteLine(" " + i + ") " + Convert.ToChar(A.matrSM[i, 0] + 64));

}

if (int.TryParse(Console.ReadLine(), out v1) == false) { Console.Clear(); goto ove1; }

if (v1 < 1 || v1 > Math.Sqrt(A.matrSM.Length) - 1)

{

Console.Clear();

goto ove1;

}

Console.Clear();

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

{

if (v1 != i)

{

if (A.matrSM[v1, i] != 0) { col++; }

}

}

if (col == 0)

{

Console.Clear();

goto ove1;

}

consize(ww2, 6 + Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1);

ove2:

int[,] choos = new int[2, col];

int nom = 0;

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

{

if (A.matrSM[0, i] != v1 && A.matrSM[v1, i] != 0)

{

choos[0, nom] = nom + 1;

choos[1, nom] = A.matrSM[0, i];

nom++;

}

}

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" выберите вторую вершину\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

{

Console.WriteLine(" " + choos[0, i] + ")" + Convert.ToChar(choos[1, i] + 64));

}

if (int.TryParse(Console.ReadLine(), out v2) == false) { Console.Clear(); goto ove2; }

if (v2 < 1 || v2 > choos.Length / 2)

{

Console.Clear();

goto ove2;

}

v2 = choos[1, v2 - 1];

if (v1 > v2) { int b = v1; v1 = v2; v2 = b; }

A = otver(ref A, v1, v2, true);

A.matrIN = matrINgen(A, v2);

listSMgen(ref A);

B = otver(ref B, v1, v2);

B.matrIN = matrINgen(B, v2);

listSMgen(ref B);

Console.Clear();

consize(ww2, wh2 + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" операция успешно завершена\n");

sc(0);

goto MenuRed2;

}

case 51://РАСЩЕПЛЕНИЕ ВЕРШИН

{

int v;

Console.Clear();

consize(ww2, 6 + Convert.ToInt32(Math.Sqrt(A.matrSM.Length)));

ove1:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

{

Console.WriteLine(" " + i + ") " + Convert.ToChar(A.matrSM[i, 0] + 64));

}

if (int.TryParse(Console.ReadLine(), out v) == false) { Console.Clear(); goto ove1; }

if (v < 1 || v > Math.Sqrt(A.matrSM.Length) - 1)

{

Console.Clear();

goto ove1;

}

Console.Clear();

A = addpoint(ref A);

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

{

if (i != v)

{

A.matrSM[i, Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1] = A.matrSM[v, i];

A.matrSM[Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1, i] = A.matrSM[i, Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1];

B.matrSM[i, Convert.ToInt32(Math.Sqrt(B.matrSM.Length)) - 1] = B.matrSM[v, i];

B.matrSM[Convert.ToInt32(Math.Sqrt(B.matrSM.Length)) - 1, i] = B.matrSM[i, Convert.ToInt32(Math.Sqrt(B.matrSM.Length)) - 1];

}

}

A.matrSM[v, Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1] = 1;

A.matrSM[Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1, v] = 1;

B.matrSM[v, Convert.ToInt32(Math.Sqrt(B.matrSM.Length)) - 1] = 1;

B.matrSM[Convert.ToInt32(Math.Sqrt(B.matrSM.Length)) - 1, v] = 1;

A.matrIN = matrINgen(A, v);

listSMgen(ref A);

B.matrIN = matrINgen(B, v);

listSMgen(ref B);

Console.Clear();

consize(ww2, wh2 + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" операция успешно завершена\n");

sc(0);

goto MenuRed2;

}

case 52://ИЗОЛЯЦИЯ ВЕРШИНЫ

{

int v;

Console.Clear();

consize(ww2, 6 + Convert.ToInt32(Math.Sqrt(A.matrSM.Length)));

ove11:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

{

Console.WriteLine(" " + i + ") " + Convert.ToChar(A.matrSM[i, 0] + 64));

}

if (int.TryParse(Console.ReadLine(), out v) == false) { Console.Clear(); goto ove11; }

if (v < 1 || v > Math.Sqrt(A.matrSM.Length) - 1)

{

Console.Clear();

goto ove11;

}

Console.Clear();

int g = Convert.ToInt32(Math.Sqrt(A.matrSM.Length));

for (int i = 1; i < g; i++)

{

A.matrSM[v, i] = 0;

A.matrSM[i, v] = 0;

B.matrSM[v, i] = 0;

B.matrSM[i, v] = 0;

}

A.matrIN = matrINgen(A, v);

listSMgen(ref A);

B.matrIN = matrINgen(B, v);

listSMgen(ref B);

Console.Clear();

consize(ww2, wh2 + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" операция успешно завершена\n");

sc(0);

goto MenuRed2;

}

case 53://УДАЛЕНИЕ РЕБРА

{

int v1, v2;

Console.Clear();

consize(ww2, 6 + Convert.ToInt32(Math.Sqrt(A.matrSM.Length)));

ove1:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" выберите первую вершину\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

{

Console.WriteLine(" " + i + ") " + Convert.ToChar(A.matrSM[i, 0] + 64));

}

if (int.TryParse(Console.ReadLine(), out v1) == false) { Console.Clear(); goto ove1; }

if (v1 < 1 || v1 > Math.Sqrt(A.matrSM.Length) - 1)

{

Console.Clear();

goto ove1;

}

Console.Clear();

consize(ww2, 6 + Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1);

ove2:

int[,] choos = new int[2, Convert.ToInt32(Math.Sqrt(A.matrSM.Length) - 1)];

int nom = 0;

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

{

if (A.matrSM[0, i] != v1)

{

choos[0, nom] = nom + 1;

choos[1, nom] = A.matrSM[0, i];

nom++;

}

}

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" выберите вторую вершину\n");

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

{

Console.WriteLine(" " + choos[0, i] + ")" + Convert.ToChar(choos[1, i] + 64));

}

if (int.TryParse(Console.ReadLine(), out v2) == false) { Console.Clear(); goto ove2; }

if (v2 < 1 || v2 > choos.Length / 2)

{

Console.Clear();

goto ove2;

}

v2 = choos[1, v2 - 1];

if (v1 > v2) { int b = v1; v1 = v2; v2 = b; }

A.matrSM[v1, v2] = 0;

A.matrSM[v2, v1] = 0;

B.matrSM[v1, v2] = 0;

B.matrSM[v2, v1] = 0;

A.matrIN = matrINgen(A);

listSMgen(ref A);

B.matrIN = matrINgen(B);

listSMgen(ref B);

Console.Clear();

consize(ww2, wh2 + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" операция успешно завершена\n");

sc(0);

goto MenuRed2;

}

case 54://ВОЗВРАЩЕНИЕ В ГЛАВНОЕ МЕНЮ

{

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

}

break;

}

case 52://ОПЕРАЦИИ С ГРАФАМИ

{

if (A.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 - 2);

MenuRed3:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" 1) Объединение");

Console.WriteLine(" 2) пересечения");

Console.WriteLine(" 3) кольцевая суммая");

Console.WriteLine(" 4) назад\n");

Console.Write(" ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

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

if (mod < 49 || mod > 52)

{

Console.Clear();

consize(ww2, wh2 - 2);

goto MenuRed3;

}

switch (mod)

{

case 49://ОБЪЕДИНЕНИЕ

{

AB.matrSM = new int[Convert.ToInt32(Math.Sqrt(A.matrSM.Length)),

Convert.ToInt32(Math.Sqrt(A.matrSM.Length))];

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

{

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

{

if (i == 0 || j == 0)

{

AB.matrSM[i, j] = A.matrSM[i, j];

}

else

{

AB.matrSM[i, j] = A.matrSM[i, j] + B.matrSM[i, j];

}

}

}

AB.matrIN = matrINgen(AB);

listSMgen(ref AB);

Console.Clear();

consize(ww2, wh2 - 2 + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" операция успешно завершена\n");

sc(0);

goto MenuRed3;

}

case 50://ПЕРЕСЕЧЕНИЕ

{

AB.matrSM = new int[Convert.ToInt32(Math.Sqrt(A.matrSM.Length)), Convert.ToInt32(Math.Sqrt(A.matrSM.Length))];

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

{

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

{

if (i == 0 || j == 0)

{

AB.matrSM[i, j] = A.matrSM[i, j];

}

else

{

if (A.matrSM[i, j] != 0 && B.matrSM[i, j] != 0)

AB.matrSM[i, j] = A.matrSM[i, j] + B.matrSM[i, j];

}

}

}

AB.matrIN = matrINgen(AB);

listSMgen(ref AB);

Console.Clear();

consize(ww2, wh2 - 2 + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" операция успешно завершена\n");

sc(0);

goto MenuRed3;

}

case 51://КОЛЬЦЕВАЯ СУММА

{

AB.matrSM = new int[Convert.ToInt32(Math.Sqrt(A.matrSM.Length)), Convert.ToInt32(Math.Sqrt(A.matrSM.Length))];

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

{

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

{

if (i == 0 || j == 0)

{

AB.matrSM[i, j] = A.matrSM[i, j];

}

else

{

if (A.matrSM[i, j] == 0 || B.matrSM[i, j] == 0)

AB.matrSM[i, j] = A.matrSM[i, j] + B.matrSM[i, j];

}

}

}

AB.matrIN = matrINgen(AB);

listSMgen(ref AB);

Console.Clear();

consize(ww2, wh2 - 2 + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" операция успешно завершена\n");

sc(0);

goto MenuRed3;

}

case 52://ВОЗВРАЩЕНИЕ В ГЛАВНОЕ МЕНЮ

{

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

}

break;

}

case 53://ОБХОД В ГЛУБИНУ

{

if (A.matrSM == null)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(2);

Console.WriteLine(" граф пуст\n");

sc(0);

goto MenuMain;

}

Console.Clear();

consize(40, wh);

bool[] be = new bool[Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1];

Console.WriteLine("рекурсивный обход по матрице:");

for (int i = 0; i < be.Length; i++)

{

obxodrecm(A.matrSM, ref be, i + 1);

}

Console.WriteLine("\n");

Console.WriteLine("рекурсивный обход по списку:");

for (int i = 0; i < be.Length; i++)

{

be[i] = false;

}

for (int i = 0; i < be.Length; i++)

{

obhodreclist(A.listSM, ref be, i + 1);

}

Console.ReadKey();

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

case 54://ОБХОД В ШИРИНУ

{

if (A.matrSM == null)

{

Console.Clear();

consize(ww, wh + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(2);

Console.WriteLine(" граф пуст\n");

sc(0);

goto MenuMain;

}

Console.Clear();

bool[] be = new bool[Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1];

Queue<int> q = new Queue<int>(1);

q.Enqueue(1);

bool nach = true;

while (q.Count != 0)

{

obhodvsh(A.matrSM, ref be, ref q, ref nach, q.Dequeue());

}

Console.ReadKey();

Console.Clear();

consize(ww, wh);

goto MenuMain;

}

}

}

}

}