Оглавление

[Введение 4](#_Toc7236)

[Постановка задачи 5](#_Toc7119)

[Теоретическая часть задания 6](#_Toc7562)

[Описание алгоритма программы 8](#_Toc29299)

[Ручной расчет 11](#_Toc5085)

[Описание программы 13](#_Toc990)

[Тест программы и результаты работы 15](#_Toc22796)

[Список литературы 20](#_Toc17441)

[Приложение А. Листинг программы 21](#_Toc29816)

# Введение

Хроматическое число - одна из характеристик графа, определяется как минимальное число цветов, в которые можно раскрасить вершины графа так, чтобы две смежные вершины имели разные цвета (концы любого ребра были разного цвета). Хроматическое число находится в процессе решения задачи о раскраске графа. Решение данной задачи необходимо в разных приложениях, например, распределение радиочастот (необходимо эффективно распределять спектр радиочастот между станциями сотовой связи, чтобы минимизировать помехи), составление расписаний, распределение регистров в микропроцессорах.

Цель курсового проекта - разработать программу, которая способна вычислять хроматическое число неориентированного графа. В качестве среды разработки использовалась Microsoft Visual Studio 2019, выбранный язык программирования - C.

# Постановка задачи

Требуется разработать программу, которая рассчитает хроматическое число графа.

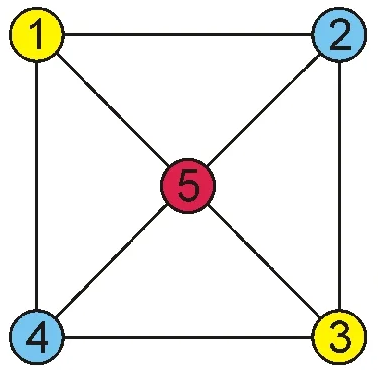
Программа должна работать с невзвешенным неориентированным графом, который задается матрицей смежности. Программа должна предусматривать возможность генерации случайной матрицы заданного размера, ввода матрицы с клавиатуры, ввода матрицы из текстового файла, а также сохранение матрицы и её хроматического числа в файл с возможностью последующего вывода в окно консоли.

Программа должна работать в текстовом режиме, устройство ввода - клавиатура. Необходимо реализовать текстовый интерфейс, с помощью которого пользователь сможет комфортно взаимодействовать с программой (выбирать нужное действие, завершать работу).

# Теоретическая часть задания

Граф G называется k-раскрашиваемым, если каждой его вершине можно приписать один из k цветов таким образом, чтобы никакие две смежные вершины не были одного цвета. Если граф G является k-раскрашиваемым, но не является k − 1-раскрашиваемым, то граф называется k-хроматическим, а k – его хроматическим числом.

На данном рисунке приведен неориентированный невзвешенный граф:



Вершины графа окрашены, причем таким образом, что две любые смежные вершины имеют разный цвет, при этом количество цветов минимально. Число цветов (в данном случае три) и является хроматическим числом графа.

Существуют несколько алгоритмов находжения хроматического числа графа. Мной был реализован последовательный алгоритм. Он состоит в следующем:

1. Составляется список вершин, упорядоченных по неубыванию степени.
2. Первая вершина в списке (имеющая наибольшую степень), окрашивается в цвет 1.
3. Пока не окрашены все вершины:
   1. Окрасить в выбранный цвет всякую вершину, несмежную с другой, уже окрашенной в этот цвет
   2. Для первой в списке неокрашенной вершины выбрать следующий цвет.

# Описание алгоритма программы

Нахождение хроматического числа выполняет функция colorize(). В качестве аргументов она принимает указатель на матрицу int\*\* matrix, указатель на список вершин sorted\_mas, отсортированных по неубыванию степеней, и число вершин rasm. Массив sorted\_mas формируется функцией create\_p(). Она создает массив структур с двумя полями: номер вершины и её степень. Функция использует алгоритм пузырьковой сортировки по степеням вершин, и формирует массив номеров вершин, отсортированных в соответствии с их степенями.

Функция colorize() создает массив color\_mas, в который будут записаны цвета вершин. Изначально все элементы массива - нули. В элемент, имеющий индекс sorted\_mas[i] (то есть вершина наибольшей степень), записывается цвет 1. Переменная no\_colorize хранит количество непосещенных вершин.

Пока no\_colorize не равна нулю, программа проверяет строку матрицы смежности под номером sorted\_mas[i]. Если обнаруживается несмежная вершина (matrix[sorted\_mas[i]][j] == 0), и при этом данная вершина не окрашена, проверяется, не является ли найденная вершина той, которая сейчас рассматривается (так как программа работает с графами без петель, главная диагональ матрицы всегда будет заполнена нулями). Если это действительно новая несмежная и незакрашенная вершина, в массив color\_mas записывается номер цвета 1, переменная no\_colorize декриментируется.

После завершения работы вложенного цикла, программа инкрементирует переменную i, которая является счетчиком массива sorted\_mas, и проверяет, закрашена ли эта вершина. Если она не закрашена, то переменная color (содержит номер текущего цвета) увеличивается, и текущая вершина окрашивается в новый цвет. Далее цикл повторяется, пока не кончатся незакрашенные вершины.

Функция возвращает значение color - номер последнего выбранного цвета т.е хроматическое число.

Функция save\_matrix() записывает матрицу, которая находится в памяти в данный момент, в файл data.txt . Печать происходит построчно, файл открывается в режиме продолжения записи.

Функция read\_file() построчно считывает данные из файла data.txt и печатает их в окно консоли.

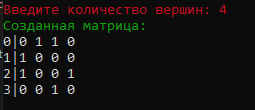
Функция read\_file\_data применяется для получения матрицы из файла и последующего нахождения хроматического числа. Она использует вспомогательную функцию get\_size\_file() для определения количества вершин матрицы, создает матрицу необходимого размера, записывает в нее данные, и возвращает указатель на созданную матрицу.

Оператор switch, находящийся в бесконечном цикле, запускает функции в зависимости от выбора пользователя. После выполнения действий над матрицей консоль отчищается, и программа запрашивает завершение работы. Если работу необходимо продолжить, цикл повторяется.

В программе применены макросы font\_g, font\_w, font\_r, которые окрашивают печатаемый текст в зеленый, белый и красный цвета. Окрашивание частей текста в консоли делает программу более легкой для восприятия.

# Ручной расчет

Проведем проверку алгоритма посредством ручного расчета хроматического числа графа, сгенерированного программой:

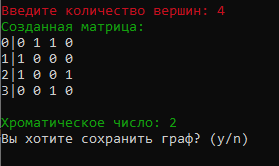


Выбираем цвет 1, окрашиваем в него вершину наибольшей степени. В данном случае таких вершин две (0 и 2), их порядок не имеет значения. Допустим, мы окрасили вершину 0 в цвет 1. Проверяем строку 0 матрицы смежности, и окрашивает вершину 3 в цвет 1 (так как она неокрашена и несмежна вершине 0). Далее, проверяем, окрашена ли следуюшая по степени вершина. В нашем случае это вершина 2, и она неокрашена. Её закрашиваем в цвет 2. Проходим строку матрицы 2: вершина 1 несмежна текущей вершине и неокрашена, значит, окрашиваем её в цвет 2.

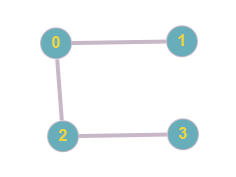
Проверяем следующую по невозрастанию степеней вершину, допустим, вершину 1. Она окрашена в цвет 2. Проверяем вершины на смежность с ней: вершины 2 и 3 несмежны вершине 1, но уже окрашены, поэтому будут пропущены.

Последняя оставшаяся в списке вершина - вершина 3, но она закрашена. Алгоритм закончился, для покраски было использовано 2 цвета. Таким образом - хроматическое число равно 2.

Результат работы программы:



Результат работы программы совпал с результатом ручного расчета. Из этого можно сделать выбор, что алгоритм работает верно. Правильность работы можно также оценить, изобразив данный граф:



По рисунку видно, что граф имеет 2 независимых множества: {1,2}, {0,3}. Решение задачи о раскраске графа можно свести к нахождению максимальных независимых множеств и их раскраске. Таким образом, для раскраски данного графа необходимо минимум 2 цвета.

# Описание программы

Программа написана на языке С в среде разработки Microsoft Visual Studio 2019. Программа является консольным приложением. В программе применялись функции WinAPI для работы с консолью.

После запуска программы на консоль выводится заставка, далее у пользователя запрашивается действие: сгенерировать случайную матрицу, ввести матрицу вручную, считать матрицу из файла, или вывести в консоль сохраненные данные.

При выборе варианта «случайная матрица» программа запрашивает количество вершин матрицы, и печатает созданную матрицу. Хроматическое число высчитывается и печатается автоматически. Далее программа запрашивает, нужно ли сохранить граф. После ответа консоль отчищается, и программа запрашивает, нужно ли продолжать работу. В случае утвердительного отвела вновь появляется запрос действия.

Пользователь может ввести матрицу вручную по строкам. Для этого он должен указать размер матрицы, и ввести строки без пробелов, нажимая клавишу Enter после каждой строки. После ввода матрицы программа работает как в предыдущем случае.

По запросу пользователя программа может напечатать данные, ранящиеся в файле data.txt. Матрицы и их хроматические числа печатаются в консоль в том порядке, в котором они были сохранены. Для перехода к запросу действий необходимо ввести E в консоль.

Программа может посчитать хроматическое число матрицы, сохраненной в файле matrix.txt. Пользователь должен заранее ввести в файл матрицу (не имеет значения, разделены ли числа пробелами ли нет). После подсчета хроматического числа матрицу можно будет сохранить в файл data.txt.

Работа программы продолжается до тех пор. Пока пользователь не подтвердит завершение работы.

# Тест программы и результаты работы

Среда разработки Microsoft Visual Studio 2019 предоставляет все

средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с алгоритмом вычисления, считыванием и сохранением данных. Ниже приведены результаты работы всех функций программы:

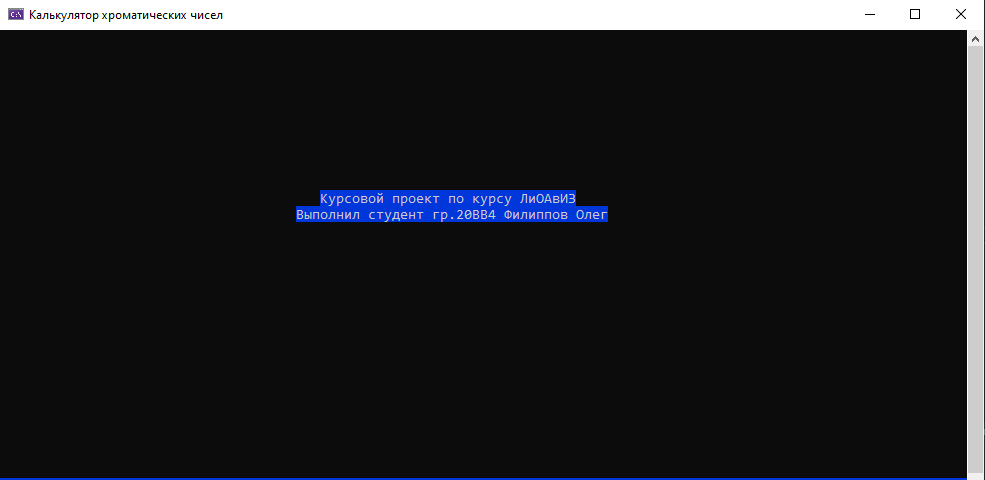


Рисунок 1. Заставка

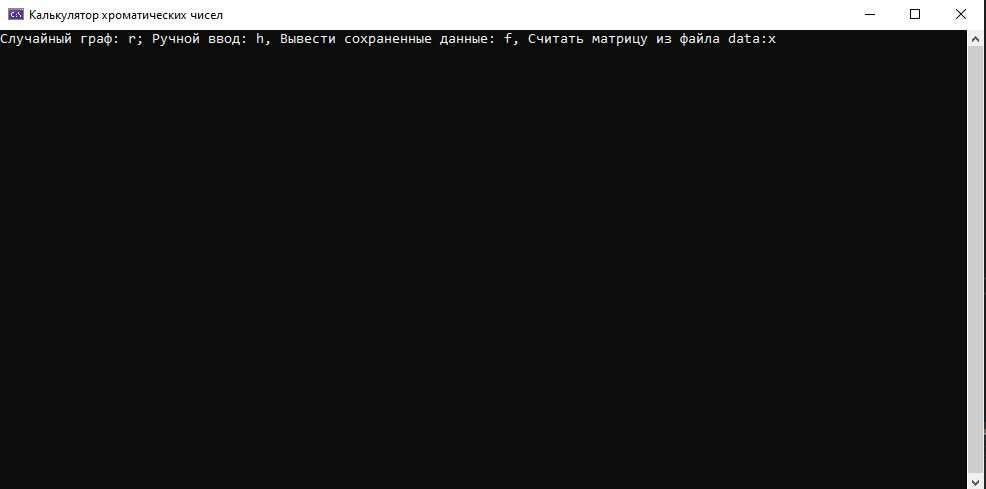


Рисунок 2. Основное окно

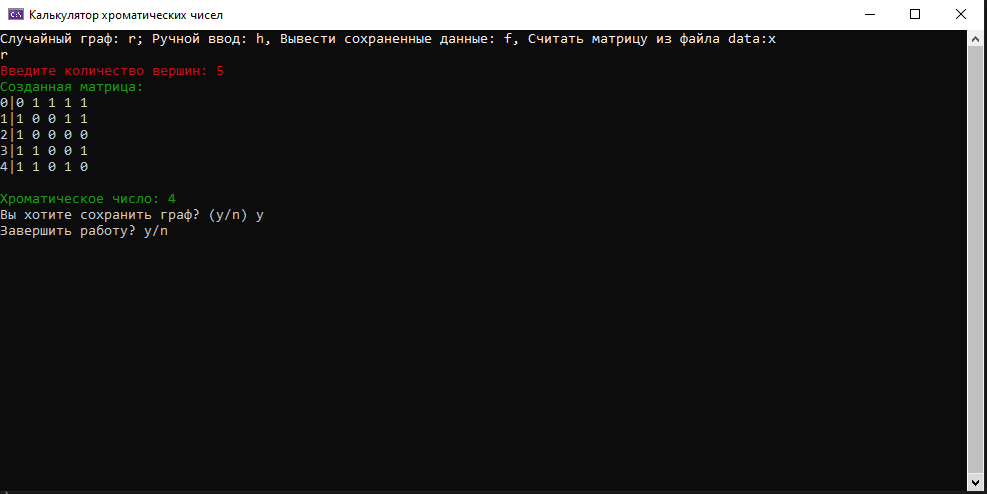


Рисунок 3. Граф создан программой и сохранен.

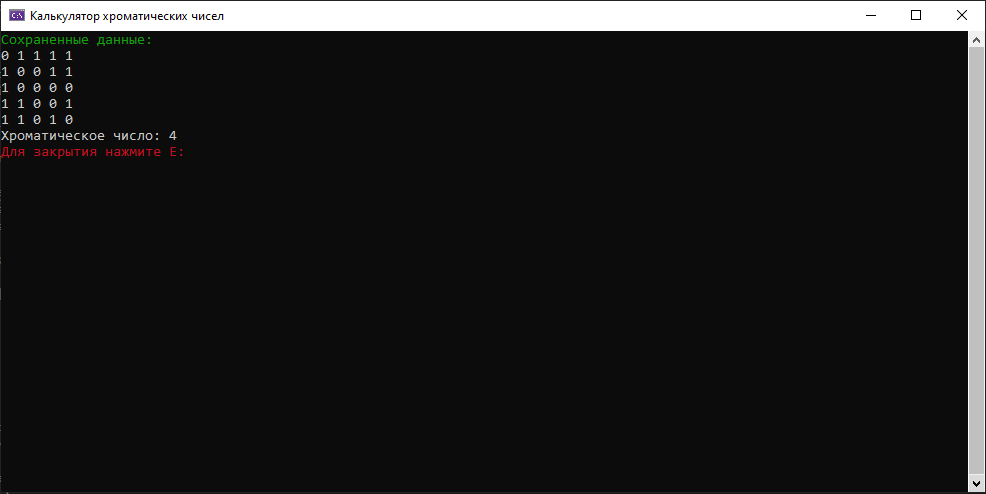


Рисунок 4. Вывод сохраненного графа.

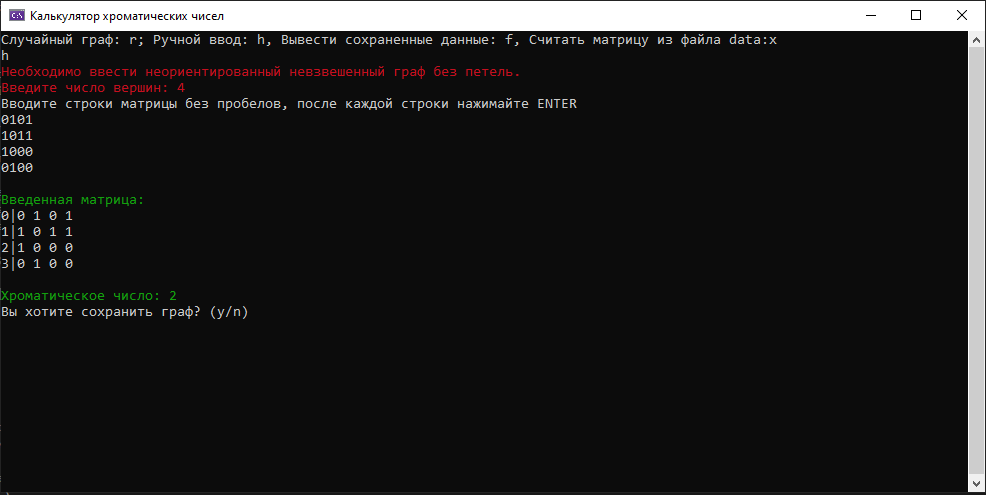


Рисунок 5. Матрица смежности введена пользователем в консоль.

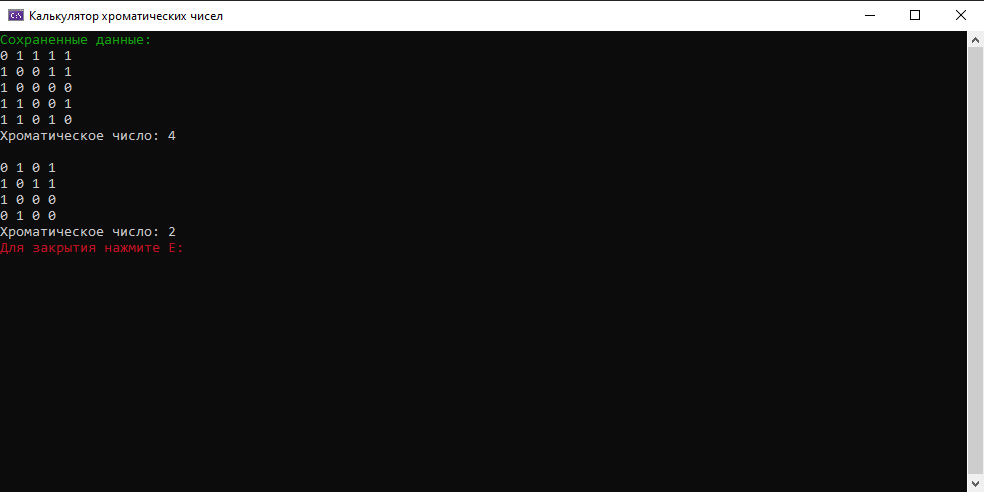
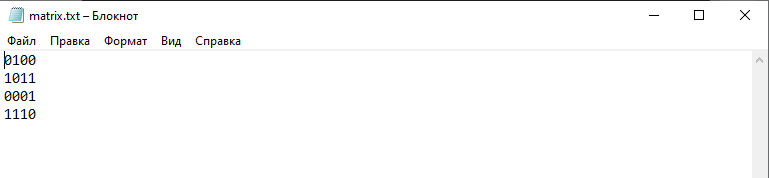


Рисунок 6. Вторая матрица сохранена в файл.



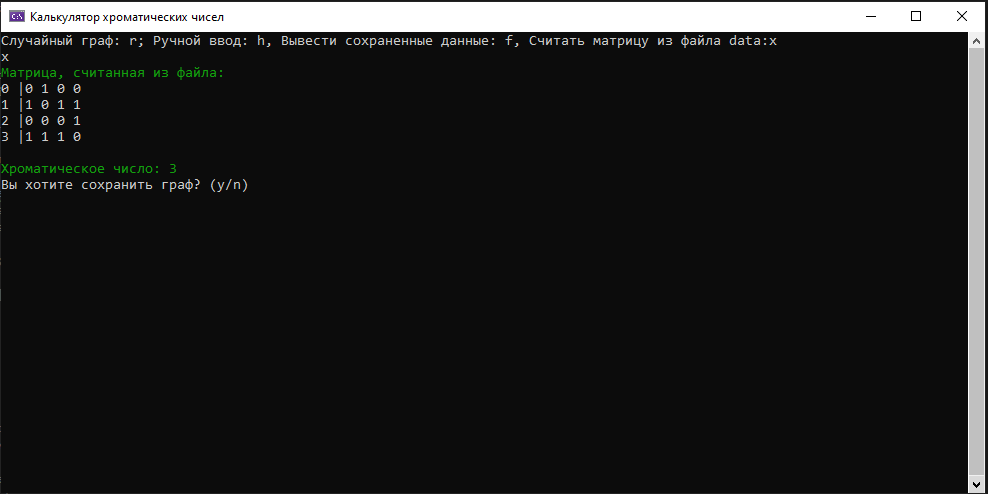


Рисунок 6. Матрица была введена через файл.

Программа корректно выполняет все свои функции.

# Список литературы

1. Доля П.Г «Введение в С/С++ программирование консоли»
2. Сайт «Библиотека алгоритмов на графах» <http://urban-sanjoo.narod.ru/colours.html>
3. Кольцов Д.М. «СИ на примерах. Практика, практика и только практика»

# Приложение А. Листинг программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#define font\_g SetConsoleTextAttribute(hstdout, FOREGROUND\_GREEN);

#define font\_w SetConsoleTextAttribute(hstdout, FOREGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN);

#define font\_r SetConsoleTextAttribute(hstdout,FOREGROUND\_RED);

#include<stdio.h>

#include<locale.h>

#include<time.h>

#include<stdlib.h>

#include<Windows.h>

#include<time.h>

int comp(const int\*, const int\*);

int\* create\_p(int\*\* matrix, int rasm);

int colorize(int\*\* matrix, int\* sorted\_mas, int rasm);

void save\_matrix(int\*\* matrix, int hrome, int size);

void read\_file();

int\*\* read\_file\_data(int size);

int get\_size\_file();

HANDLE hstdout;

void main()

{

setlocale(LC\_ALL, "Rus");

int\*\* matrix = NULL;

int i = 0;

int j = 0;

int sum = 0;

int a = 0;

int\* sorted\_mas = NULL;

char type;

char\* str\_matr;

char buf[2];

char save;

wchar\_t name\_console[] = L"Калькулятор хроматических чисел";

SetConsoleTitle(name\_console);

Sleep(40);

HWND console = FindWindow(NULL, name\_console);

MoveWindow(console, 400, 250, 1000, 500, TRUE);

hstdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD position;

position.X = 40;

position.Y = 10;

//system("color 02");

SetConsoleCursorPosition(hstdout, position);

SetConsoleTextAttribute(hstdout, 23);

printf("Курсовой проект по курсу ЛиОАвИЗ");

position.Y = 11;

position.X = 37;

SetConsoleCursorPosition(hstdout, position);

printf("Выполнил студент гр.20ВВ4 Филиппов Олег");

Sleep(6000);

SetConsoleTextAttribute(hstdout, 0|15);

system("cls");

while (1)

{

printf\_s("Случайный граф: r; Ручной ввод: h, Вывести сохраненные данные: f, Считать матрицу из файла data:x\n");

scanf(" %c", &type, sizeof(char));

switch (type)

{

case 'r':

{

font\_r

printf\_s("Введите количество вершин: ");

scanf("%d", &a);

srand(time(NULL));

matrix = (int\*\*)malloc(a \* sizeof(int));

if (matrix == NULL)

{

printf\_s("Ошибка");

}

font\_w

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

{

matrix[i] = (int\*)malloc(a \* sizeof(int));

}

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

{

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

{

if (i == j)

{

matrix[i][j] = 0;

continue;

}

if (rand() % 21 > 5) matrix[i][j] = 1;

else matrix[i][j] = 0;

matrix[j][i] = matrix[i][j];

}

}

font\_g

printf("Созданная матрица:\n");

font\_w

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

{

printf("%d|", i);

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

{

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

}

printf("\n");

}

sorted\_mas = create\_p(matrix, a);//По данному указателю находится отсортированный массив вершин

int color = colorize(matrix, sorted\_mas, a);

font\_g

printf("\nХроматическое число: %d", color);

font\_w

printf("\nВы хотите сохранить граф? (y/n) ");

getchar();

scanf(" %c", &save);

if (save == 'y') save\_matrix(matrix, color, a);

if (save == 'n') system("cls");

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

{

free(matrix[i]);

}

free(matrix);

}

break;

case 'h':

{

font\_r

printf("Необходимо ввести неориентированный невзвешенный граф без петель.\n");

printf("Введите число вершин: ");

scanf("%d", &a);

matrix = (int\*\*)malloc(a \* sizeof(int));

if (matrix == NULL)

{

printf("Ошибка");

}

font\_w

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

{

matrix[i] = (int\*)malloc(a \* sizeof(int));

}

str\_matr = (char\*)malloc(a \* sizeof(char) + 1);

printf("Вводите строки матрицы без пробелов, после каждой строки нажимайте ENTER\n");

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

{

scanf("%s", str\_matr);

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

{

switch (str\_matr[j])

{

case '0': matrix[i][j] = 0;

break;

case '1':matrix[i][j] = 1;

}

}

}

//

font\_g

printf("\nВведенная матрица:\n");

font\_w

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

{

printf("%d|", i);

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

{

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

}

printf("\n");

}

sorted\_mas = create\_p(matrix, a);//По данному указателю находится отсортированный массив вершин

int color = colorize(matrix, sorted\_mas, a);

font\_g

printf\_s("\nХроматическое число: %d", color);

font\_w

printf("\nВы хотите сохранить граф? (y/n) ");

getchar();

scanf(" %c", &save);

if (save == 'y') save\_matrix(matrix, color, a);

if (save == 'n') system("cls");

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

{

free(matrix[i]);

}

free(matrix);

}

break;

case 'f':

{

char end = NULL;

system("cls");

read\_file();

font\_r

printf("Для закрытия нажмите E: ");

font\_w

scanf(" %c", &end);

if (end != NULL) system("cls");

}

break;

case 'x':

{

int size = get\_size\_file();

int\*\* matrix\_d = read\_file\_data(size);

sorted\_mas = create\_p(matrix\_d, size);//По данному указателю находится отсортированный массив вершин

int color = colorize(matrix\_d, sorted\_mas, size);

font\_g

printf("\nХроматическое число: %d", color);

font\_w

printf("\nВы хотите сохранить граф? (y/n) ");

getchar();

scanf(" %c", &save);

if (save == 'y') save\_matrix(matrix\_d, color, size);

if (save == 'n') system("cls");

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

{

free(matrix\_d[i]);

}

free(matrix\_d);

}

}

printf("Завершить работу? y/n ");

getchar();

scanf(" %c", &save);

if (save == 'y')

{

break;

system("cls");

}

else

{

system("cls");

}

}

}

int colorize(int\*\* matrix, int\* sorted\_mas, int rasm)

{

int color = 1;

int no\_colorize = rasm;

int i = 0, j = 0, k = 0;

int\* color\_mas = NULL;

color\_mas = (int\*)malloc(rasm \* sizeof(int));

if (color\_mas == NULL) printf\_s("error");

for (i=0; i < rasm; i++) color\_mas[i] = 0;

i = 0;

color\_mas[sorted\_mas[i]] = color;

no\_colorize = --no\_colorize;

while (no\_colorize != 0)

{

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

{

if ((matrix[sorted\_mas[i]][j] == 0) && (color\_mas[j] == 0))

{

if (sorted\_mas[i] == j) continue;

color\_mas[j] = color;

no\_colorize = --no\_colorize;

}

}

i = i++;

if (color\_mas[sorted\_mas[i]] == 0)

{

color = ++color;

color\_mas[sorted\_mas[i]] = color;

no\_colorize = --no\_colorize;

}

}

free(color\_mas);

free(sorted\_mas);

return color;

}

int\* create\_p(int\*\* matrix, int rasm)//Эта функция сортирует номера вершин по убыванию порядка вершины

{

struct sp

{

int num=0;

int por=0;

};

sp hold;

int i = 0;

int j = 0;

int sum = 0;

sp\* mas\_struct = NULL;

mas\_struct = (sp\*)malloc(rasm \* sizeof(sp));

int\* mas = 0;

mas = (int\*)malloc(rasm \* sizeof(int));

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

{

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

{

sum = sum + matrix[i][j];

}

mas\_struct[i].num = i;

mas\_struct[i].por = sum;

sum = 0;

}

//Сортировка:

int switched = 0;

for (i = rasm - 1; i > 0; i--)

{

switched = 0;

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

{

if (mas\_struct[j].por < mas\_struct[j + 1].por)

{

switched = 1;

hold = mas\_struct[j];

mas\_struct[j] = mas\_struct[j + 1];

mas\_struct[j + 1] = hold;

switched = 1;

}

}

}

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

{

mas[i] = mas\_struct[i].num;

}

return mas;

}

void save\_matrix(int\*\* matrix, int hrome, int size)

{

FILE\* file = NULL;

file = fopen("data.txt", "a");

fprintf(file, "\n");

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

{

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

{

fprintf(file, "%d ", matrix[i][j]);

}

fprintf(file, "\n");

}

fprintf(file, "Хроматическое число: %d\n", hrome);

fclose(file);

}

void read\_file()

{

FILE\* file = NULL;

file = fopen("data.txt", "r");

char str[100];

font\_g

printf("Сохраненные данные:");

font\_w

while (fgets(str, 100, file) != NULL)

printf("%s", str);

fclose(file);

}

int\*\* read\_file\_data(int size)

{

FILE\* file = NULL;

file = fopen("matrix.txt", "r+");

char c;

int\*\* matrix = NULL;

int i = 0, j = 0;

matrix = (int\*\*)malloc(size \* sizeof(int));

if (matrix == NULL)

{

printf\_s("Ошибка");

}

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

{

matrix[i] = (int\*)malloc(size \* sizeof(int));

}

i = 0;

do

{

c = fgetc(file);

switch (c)

{

case '1':

{

matrix[i][j] = 1;

j = ++j;

}

break;

case '0':

{

matrix[i][j] = 0;

j = ++j;

}

break;

case'\n':

{

i = ++i;

j = 0;

}

}

} while (c != EOF);

fclose(file);

font\_g

printf("Матрица, считанная из файла:\n");

font\_w

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

{

printf("%d |", i);

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

{

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

}

printf("\n");

}

return matrix;

}

int get\_size\_file()

{

FILE\* file = fopen("matrix.txt", "r+");

int size = 0;

char c;

do

{

c = fgetc(file);

if (c == '1' || c == '0') size = ++size;

} while (c != '\n');

fclose(file);

return size;

}