Министерство образования Российской Федерации

Пензенский государственный университета

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовому проектированию

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Реализация алгоритма раскрашивания графа»

Выполнил:

Студент группы 22ВВВ1

Николаев А.А.

Приняли:

К.т.н, доцент Акифьев И. В.

К.т.н, доцент Юрова О. В.

Пенза 2023

**Содержание**

[Реферат 3](#_Toc153478642)

[Введение 4](#_Toc153478643)

[1. Постановка задачи 5](#_Toc153478644)

[2. Теоретическая часть задания 6](#_Toc153478645)

[3. Описание алгоритма программы 7](#_Toc153478646)

[4. Описание программы 1](#_Toc153478647)0

[5. Отладка и тестирование 1](#_Toc153478648)2

[6. Ручной расчет 1](#_Toc153478649)5

[Заключение 1](#_Toc153478652)6

[Список литературы 1](#_Toc153478653)7

[Приложение А. Листинги программы](#_Toc153478654) 18

Реферат

Отчет 24 стр, 9 рисунков.

ГРАФ, МАТРИЦА СМЕЖНОСТИ, НЕОРИЕНТИРОВАННЫЙ ГРАФ, ПЕРЕБОРНЫЙ АЛГОРИТМ, СПИСОК СМЕЖНОСТИ, РАСКРАШИВАНИЕ ГРАФОВ.

Цель исследования – разработка программного решения для генерации графа указанного размера и раскраски его вершин.

В работе осуществляется анализ и реализация алгоритма раскрашивания графа. Исследование направлено на изучение способов раскрашивания графа указанного размера.

# Введение

Раскрашивание графов – это задача, которая находит свое применение в различных областях, начиная от компьютерных наук и заканчивая реальными прикладными задачами. Одной из основных задач раскрашивания графа является присвоение цветов вершинам графа таким образом, чтобы никакие две смежные вершины не имели одинаковый цвет. Данная задача имеет свои важные теоретические и практические применения, такие как оптимизация расписания, планирование ресурсов, улучшение процессов планирования в сетях связи и др.

Был разработан алгоритм раскрашивания графа, основанный на методе перебора. Он использует жадный подход, пытаясь окрасить каждую вершину в минимально возможный цвет, учитывая при этом цвета ее смежных вершин.

Для реализации данного алгоритма была выбрана среда разработки Microsoft Visual Studio 2022, а язык программирования C++ использован в качестве основного инструмента. Этот выбор обусловлен широким распространением языка C++ и его применимостью в различных областях программирования.

# Постановка задачи

Необходимо разработать программу для раскрашивания графа.

Программа должна отвечать следующим требованиям:

1.Текстовое или графическое меню для удобства использования программы.

2. Возможность задания пользователем размера графа (множества).

3. Возможность выбора автоматического (случайного) или ручного (с клавиатуры или из файла) задания графа (элементов множества).

4. Возможность сохранения результатов работы программы.

5. Устройство ввода информации: клавиатура/мышь.

# Теоретическая часть задания

Граф — это абстрактный тип данных, предназначенный для реализации концепций неориентированного графа и ориентированного графа из области теории графов в математике.

Структура данных графа состоит из конечного (и, возможно, изменяемого) набора вершин (также называемых узлами или точками) вместе с набором неупорядоченных пар этих вершин для неориентированного графа или набором упорядоченных пар для ориентированного графа. Эти пары известны как ребра (также называемые связями или линиями), а для ориентированного графа также известны как ребра, но также иногда как стрелки или дуги. Вершины могут быть частью структуры графа или могут быть внешними объектами, представленными целочисленными индексами или ссылками.

Раскрашивание вершин - процесс присвоения цветов каждой вершине графа так, чтобы никакие две смежные вершины не имели одинаковый цвет.

Жадный алгоритм - один из наиболее распространенных методов. Он последовательно просматривает вершины графа и назначает каждой вершине наименьший доступный цвет, который не используется у её смежных вершин.

# Описание алгоритма программы

Перед началом разработки программы был проведен анализ требований и определены основные функциональности, которые она должна включать. Основные компоненты программы включают в себя генерацию и отображение графа, обработка его вершин для окрашивания, сохранение графа и цветов его вершин в файл.

На вход подаётся кол-во вершин для генерации графа. После окончания генерации или копирования из файла, появляется меню, с помощью которого можно выбрать раскраску графа.

Была реализована функция paintGraph, которая отвечает за раскраску графа и занесение результата в файл rez.txt.

**Функция paintGraph:**

void paintGraph(vector<vector<int>>& graf, vector<int> colors, int ver, vector<SP\*>& masSP)

{

ofstream outputFile;

vector<pair<int, int>> degrees;//вектор пар(степень, номер)

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

{

int degree = 0;

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

{

degree += graf[i][j];

}

degrees.push\_back(make\_pair(degree, i));//заносим степень и номер

}

sort(degrees.rbegin(), degrees.rend());//сортируем вектор по степени(от наибольшей до минимальной)

colors.assign(ver, 0);//заполняем вектор цветов нулями

int r = 1; //начальный цвет

for(int k = 0; k<ver; k++)

{

int u = degrees[k].second;//заносим индекс вершины

//проверка цветов смежных вершин

vector<bool> usedCol(r + 1, false);

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

{

if (graf[u][j] && colors[j] > 0)

{

usedCol[colors[j]] = true;

}

}

//находим минимальный допустимый цвет

int minCol;

for (minCol = 1; minCol <= r; ++minCol)

{

if (!usedCol[minCol])

{

break;

}

}

if (minCol > r)

{

r++;

colors[u] = r;

}

else colors[u] = minCol;

}

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

{

cout << "Вершина " << i + 1 << " окрашена в цвет: " << colors[i] << endl;

}

//открытие файла

outputFile.open("rez.txt" , ios::out);

if (!outputFile.is\_open())

{

cout << "Ошибка открытия файла!Данные не были записаны в файл rez.txt. Повторите попытку,исправив файл. Нажмите любую клавишу..." << endl;

\_getch();

return;

}

//запись графа в файл

for (const auto& znach1 : graf)

{

for (int znach2 : znach1)

{

outputFile << setw(3) << znach2;

}

outputFile << endl;

}

outputFile << endl;

//запись списка смежности в файл

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

{

if (masSP[i] == NULL)

{

outputFile << "Вершина " << i + 1 << " не имеет смежных" << endl;

}

else {

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

SP\* current = masSP[i];

while (current != NULL)

{

outputFile << current->inf << " ";

current = current->next;

}

outputFile << endl;

}

}

outputFile << endl;

//запись окраски графа в файл

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

{

outputFile << "Вершина " << i + 1 << " окрашена в цвет: " << colors[i] << endl;

}

outputFile.close();

cout << endl << "Данные успешно записаны в файл rez.txt.";}

# Описание программы

Программа состоит из одного модуля: main.cpp, где реализуется вся программа.

При запуске программы открывается консольное окно с меню создания графа:



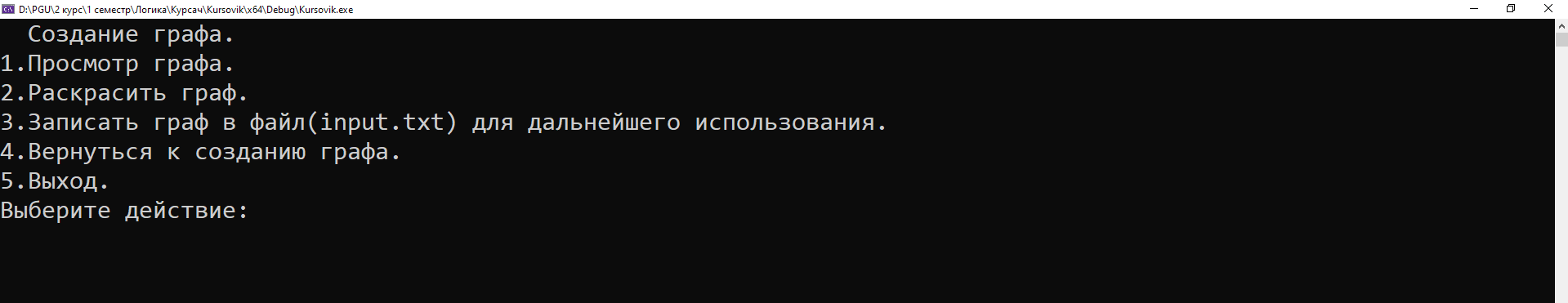
**Рисунок 1 – Меню создания графа**

После выбора способа создания графа, выведется запрос на кол-во вершин в графе и после ввода кол-ва выведется матрица смежности для графа и список смежности. Либо(в случае копирования) сразу выведется матрица смежности для графа и список смежности:



**Рисунок 2 – Создание графа**

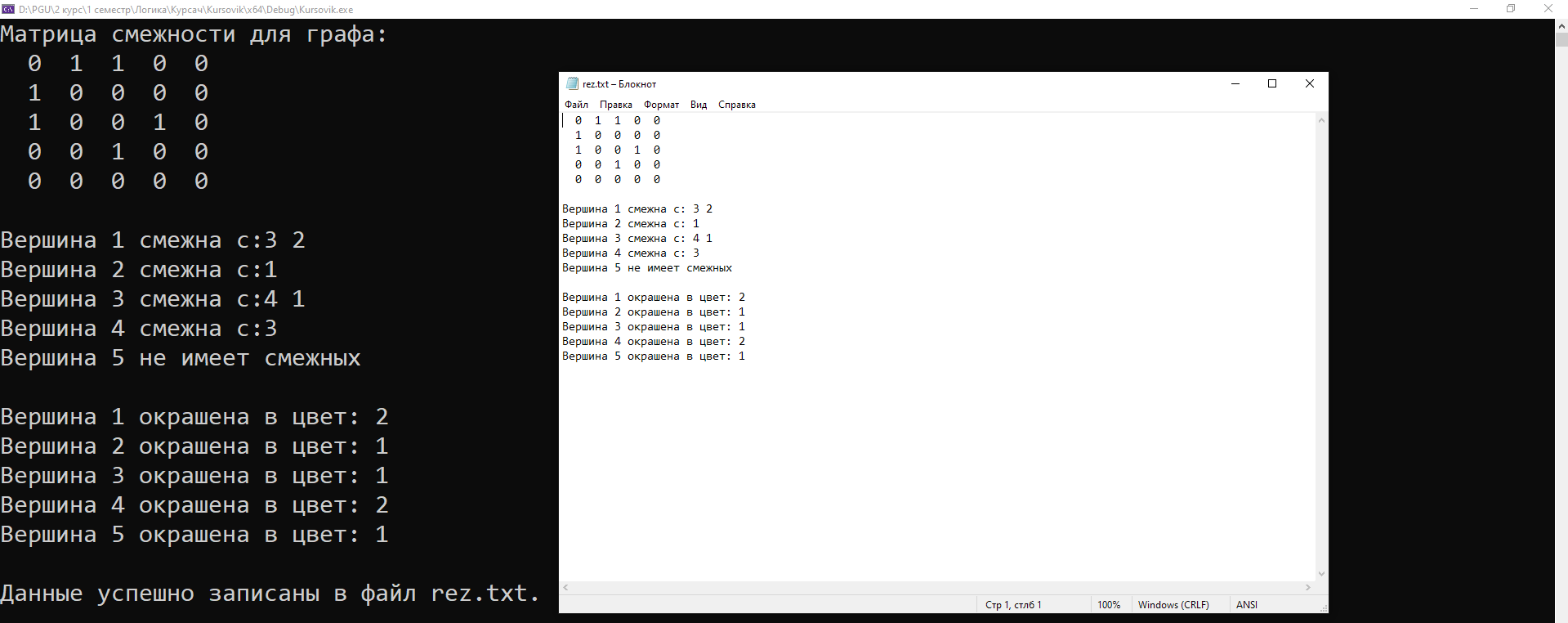
После нажатия любой клавиши выведется основное меню для работы с графом:



**Рисунок 3 – Основное меню**

При выборе первого пункта, выведется матрица смежности для данного графа и список смежности.

При выборе второго пункта граф раскраситься и данные запишутся в файл rez.txt.



**Рисунок 4 – Раскраска и запись графа в файл.**

При выборе третьего пункта граф, который был сгенерирован, запишется в файл input.txt для дальнейшего использования.

При выборе четвертого пункта пользователь вернется к выбору способа создания графа.

При выборе пятого пункта программа завергится.

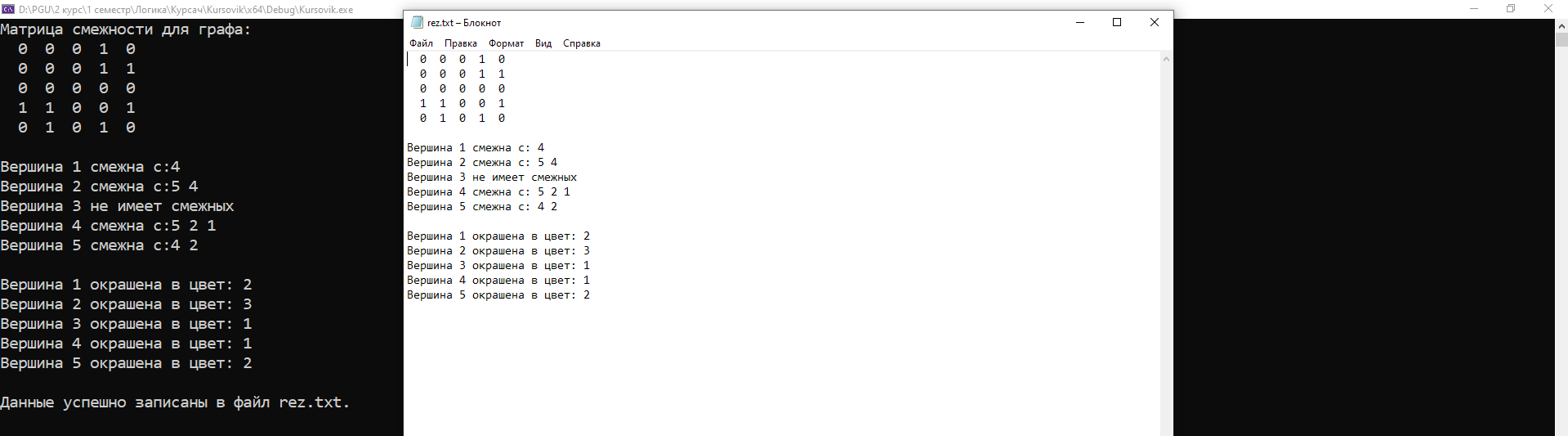
# Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей Visual Studio: точка останова, трассировка, анализ содержимого переменных.

Тестирование проводилось во время разработки и также после завершения разработки. В ходе нее было выявлено огромное количество проблем, связанных с работой с файлами, работой с памятью, работой с векторами и т. д.

Тестирование программы:



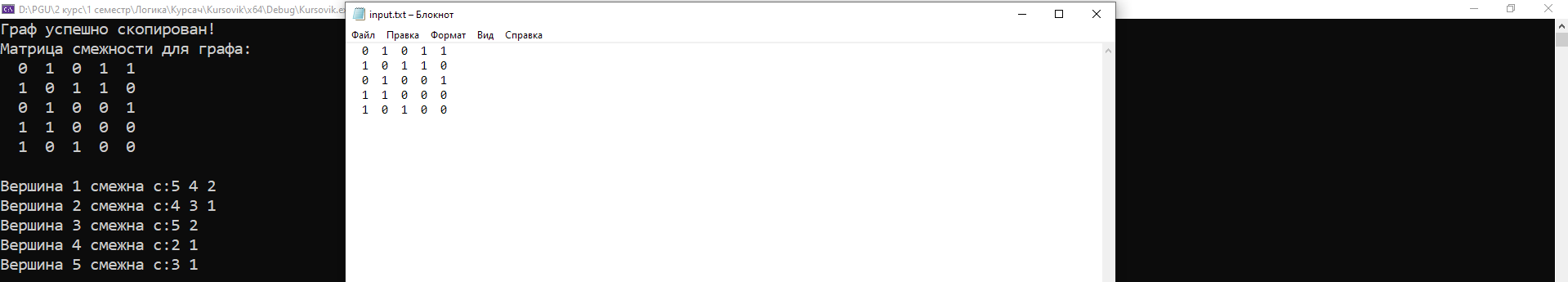


**Рисунок 5 – создание графа случайным образом**





**Рисунок 6 – создание графа вручную**





**Рисунок 7 – копирование графа из файла**

***Таблица 1 – Описание поведения программы при тестировании***

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Ввод кол-ва вершин | Правильное считывание размера, генерация графа нужного размера и вывод графа | Верно |
| Раскрашивание графа | Правильный раскрас, правильное использованные цвета, использовано минимальное количество цветов. | Верно |
| Сохранение в файл/чтение из файла | Граф верно считывается из файла и верно записывается результат. | Верно |

# Ручной расчет

# https://sun9-21.userapi.com/impf/gZA-1deiK4EO_uBAvD9xShnLVW1CCIXngpgAcA/sMgfBUV6nPo.jpg?size=960x1280&quality=95&sign=11d3a761b7246392abf990c1af23754f&type=album

**Рисунок 8 – ручной расчет**

# 

**Рисунок 9 – расчет программы**

# Заключение

Успешно выполнена цель: разработана программа для генерации графа и раскрашивания его вершин. Выполнены поставленные задачи, программа отвечает требованиям к функциональности.

Важной особенностью программы является реализация текстового пользовательского интерфейса, что обеспечивает удобство использования и повышает интуитивность взаимодействия пользователя с программой.

Код программы структурирован, что улучшает его читаемость и возможность доработки.

В процессе разработки были улучшены навыки программирования на C++, в области работы с векторами, файловым вводом/выводом.

Возможны дополнительные улучшения, такие как создание графического интерфейса, расширение функциональности этого интерфейса и оптимизация алгоритмов для более быстрого окраса, но, несмотря на это, все поставленные задачи были выполнены.

# Список литературы

1. learn.microsoft.com, URL: https://learn.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170
2. portal.tpu.ru,URL:https://portal.tpu.ru/SHARED/t/TRACEY/Courses/Graph\_Theory/Tab1/graph\_lec\_10.pdf
3. Лекция, Деревья\_потоки\_раскрашивания, Митрохин М.А.

# Приложение А. Листинги программы

**Main.cpp**

#include <vector>

#include <iostream>

#include <ctime>

#include <locale>

#include <iomanip>

#include <conio.h>

#include <fstream>

#include <sstream>

#include <algorithm>

using namespace std;

typedef struct Spis

{

int inf;

struct Spis\* next;

}SP;

//создание списка смежности

void GeneratSpisok(int ver, vector<SP\*>& masSP, vector<vector<int>>& graf)

{

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

{

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

{

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

{

SP\* newSP = new SP;

newSP->inf = j + 1;

newSP->next = masSP[i];

masSP[i] = newSP;

}

}

}

}

//вывод списка смежности

void PrintSpisok(int ver, vector<SP\*>& masSP)

{

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

{

if (masSP[i] == NULL)

{

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

}

else

{

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

SP\* current = masSP[i];

while (current != NULL)

{

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

current = current->next;

}

}

cout << endl;

}

}

//вывод матрицы смежности для графа

void print(const vector<vector<int>>& graf)

{

for (const auto& znach1 : graf)//auto-автомотически определяет тип

{

for (int znach2 : znach1)

{

cout << setw(3) << znach2;

}

cout << endl;

}

}

//функция раскрашивания

void paintGraph(vector<vector<int>>& graf, vector<int> colors, int ver, vector<SP\*>& masSP)

{

ofstream outputFile;

vector<pair<int, int>> degrees;//вектор пар(степень, номер)

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

{

int degree = 0;

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

{

degree += graf[i][j];

}

degrees.push\_back(make\_pair(degree, i));//заносим степень и номер

}

sort(degrees.rbegin(), degrees.rend());//сортируем вектор по степени(от наибольшей до минимальной)

colors.assign(ver, 0);//заполняем вектор цветов нулями

int r = 1; //начальный цвет

for(int k = 0; k<ver; k++)

{

int u = degrees[k].second;//заносим индекс вершины

//проверка цветов смежных вершин

vector<bool> usedCol(r + 1, false);

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

{

if (graf[u][j] && colors[j] > 0)

{

usedCol[colors[j]] = true;

}

}

//находим минимальный допустимый цвет

int minCol;

for (minCol = 1; minCol <= r; ++minCol)

{

if (!usedCol[minCol])

{

break;

}

}

if (minCol > r)

{

r++;

colors[u] = r;

}

else colors[u] = minCol;

}

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

{

cout << "Вершина " << i + 1 << " окрашена в цвет: " << colors[i] << endl;

}

//открытие файла

outputFile.open("rez.txt" , ios::out);

if (!outputFile.is\_open())

{

cout << "Ошибка открытия файла!Данные не были записаны в файл rez.txt. Повторите попытку,исправив файл. Нажмите любую клавишу..." << endl;

\_getch();

return;

}

//запись графа в файл

for (const auto& znach1 : graf)

{

for (int znach2 : znach1)

{

outputFile << setw(3) << znach2;

}

outputFile << endl;

}

outputFile << endl;

//запись списка смежности в файл

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

{

if (masSP[i] == NULL)

{

outputFile << "Вершина " << i + 1 << " не имеет смежных" << endl;

}

else {

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

SP\* current = masSP[i];

while (current != NULL)

{

outputFile << current->inf << " ";

current = current->next;

}

outputFile << endl;

}

}

outputFile << endl;

//запись окраски графа в файл

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

{

outputFile << "Вершина " << i + 1 << " окрашена в цвет: " << colors[i] << endl;

}

outputFile.close();

cout << endl << "Данные успешно записаны в файл rez.txt.";

}

void menu(vector<vector<int>>& graf)

{

int choice;

int ver = graf.size();

vector<int> colors(ver);

vector<SP\*> masSP(ver);

ofstream outputFile;

do

{

system("cls");

cout << setw(10) << "Меню." << endl;

cout << "1.Просмотр графа." << endl;

cout << "2.Раскрасить граф." << endl;

cout << "3.Записать граф в файл(input.txt) для дальнейшего использования." << endl;

cout << "4.Вернуться к созданию графа." << endl;

cout << "5.Выход." << endl;

cout << "Выберите действие:";

cin >> choice;

switch (choice)

{

case 1:

system("cls");

cout << "Матрица смежности для графа:" << endl;

print(graf);

cout << endl;

masSP.resize(ver);

GeneratSpisok(ver, masSP, graf);

PrintSpisok(ver, masSP);

\_getch();

break;

case 2:

system("cls");

cout << "Матрица смежности для графа:" << endl;

print(graf);

cout << endl;

masSP.resize(ver);

GeneratSpisok(ver, masSP, graf);

PrintSpisok(ver, masSP);

cout << endl;

paintGraph(graf, colors, ver, masSP);

\_getch();

break;

case 3:

system("cls");

outputFile.open("input.txt");

if (!outputFile.is\_open())

{

cout << "Ошибка открытия файла! Повторите попытку, исправив файл. Нажмите любую клавишу..." << endl;

\_getch();

break;

}

for (const auto& znach1 : graf)

{

for (int znach2: znach1)

{

outputFile << setw(3) << znach2;

}

outputFile << endl;

}

outputFile.close();

cout << "Граф успешно записан в файл." << endl << "Нажмите любую клавишу...";

\_getch();

break;

case 4:

return;

case 5:

system("cls");

cout << "До свидания!";

exit(0);

default:

cout << endl;

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

cout << endl;

break;

}

} while (choice != 5);

}

void main()

{

srand(time(NULL));

setlocale(LC\_ALL, "RUS");

cout << "\n" << setw(90) << "Курсовая работа" << endl << setw(105) << "Тема: реализация алгоритма раскрашивания графа" << setw(90) << endl << "\n\n\n" << setw(105) << "Выполнил: ст. гр. 22ВВВ1" << endl << setw(104) << "Николаев А.А.";

int choice, ver;

bool prov = true, prov1 = true;

string str;

vector<vector<int>> graf;

vector<SP\*> masSP;

ifstream inputFile;

\_getch();

do

{

system("cls");

cout << setw(17) << "Создание графа." << endl;

cout << "1.Создать граф случайным образом." << endl;

cout << "2.Задать граф в ручную." << endl;

cout << "3.Скопировать граф из файла(input.txt)." << endl;

cout << "4.Выход" << endl;

cout << "Выберите действие:";

cin >> choice;

switch (choice)

{

case 1:

system("cls");

graf.clear();

masSP.clear();

do

{

cout << "Ведите количество вершин в графе:";

cin >> ver;

if (ver < 1) cout << "Некорректный номер вершины! Повторите попытку." << endl << endl;

else prov1 = false;

} while (prov1);

graf.resize(ver);

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

{

graf[i].resize(ver);

}

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

{

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

{

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

else

{

graf[i][j] = rand() % 2;

graf[j][i] = graf[i][j];

}

}

}

cout << "Матрица смежности для графа:" << endl;

print(graf);

cout << endl;

masSP.resize(ver);

GeneratSpisok(ver, masSP, graf);

PrintSpisok(ver,masSP);

\_getch();

menu(graf);

break;

case 2:

system("cls");

graf.clear();

masSP.clear();

do

{

cout << "Ведите количество вершин в графе:";

cin >> ver;

if (ver < 1) cout << "Некорректный номер вершины! Повторите попытку." << endl << endl;

else prov1 = false;

} while (prov1);

graf.resize(ver);

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

{

graf[i].resize(ver);

}

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

{

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

{

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

else

{

do

{

cout << "Введите значение(1 или 0) для элемента " << i << "X" << j << ":";

cin >> graf[i][j];

if (graf[i][j] == 1 || graf[i][j] == 0) prov = false;

else cout << "Неверный ввод! Повторите попытку." << endl;

} while (prov);

graf[j][i] = graf[i][j];

}

}

}

cout << "Матрица смежности для графа:" << endl;

print(graf);

cout << endl;

masSP.resize(ver);

GeneratSpisok(ver, masSP, graf);

PrintSpisok(ver, masSP);

\_getch();

menu(graf);

break;

case 3:

system("cls");

graf.clear();

masSP.clear();

inputFile.open("input.txt");

if (!inputFile.is\_open())

{

cout << "Ошибка открытия файла! Повторите попытку, исправив файл. Нажмите любую клавишу..." << endl;

\_getch();

break;

}

//проверка файла на наличие данных

if (inputFile.peek() == EOF)

{

cout << "Файл пуст! Повторите попытку, добавив данные в файл. Нажмите любую клавишу..." << endl;

\_getch();

break;

}

while (getline(inputFile, str)) //пока есть строки. с помощью getline построчно читаем файл

{

vector<int> znach1;

for (char znach2 : str)

{

if (znach2 == '0' || znach2 == '1' || znach2 == ' ')

{

if (znach2 == '0' || znach2 == '1')

znach1.push\_back(znach2 - '0');//запись связи для одной вершины

}

else

{

cout << "Ошибка чтения из файла! Повторите попытку, исправив данные в файле." << endl << "Учтите, что граф неориентированный и невзвешенный." << endl << "Нажмите любую клавишу...";

\_getch();

goto EndCase3;

}

}

graf.push\_back(znach1);//связи вершины записываются в строку матрицы

}

inputFile.close();

cout << "Граф успешно скопирован!" << endl;

cout << "Матрица смежности для графа:" << endl;

print(graf);

ver = graf.size();

cout << endl;

masSP.resize(ver);

GeneratSpisok(ver, masSP, graf);

PrintSpisok(ver, masSP);

\_getch();

menu(graf);

break;

EndCase3:

inputFile.close();

break;

case 4:

system("cls");

cout << "До свидания!";

break;

default:

cout << endl;

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

cout << endl;

break;

}

} while (choice != 4);}