Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра “Вычислительная техника”

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовому проектированию

по курсу “Логика и основы алгоритмизации в инженерных задачах”

на тему «Разработка алгоритма нахождения хроматического числа графа”

Выполнил:

Студент группы 23ВВВ1

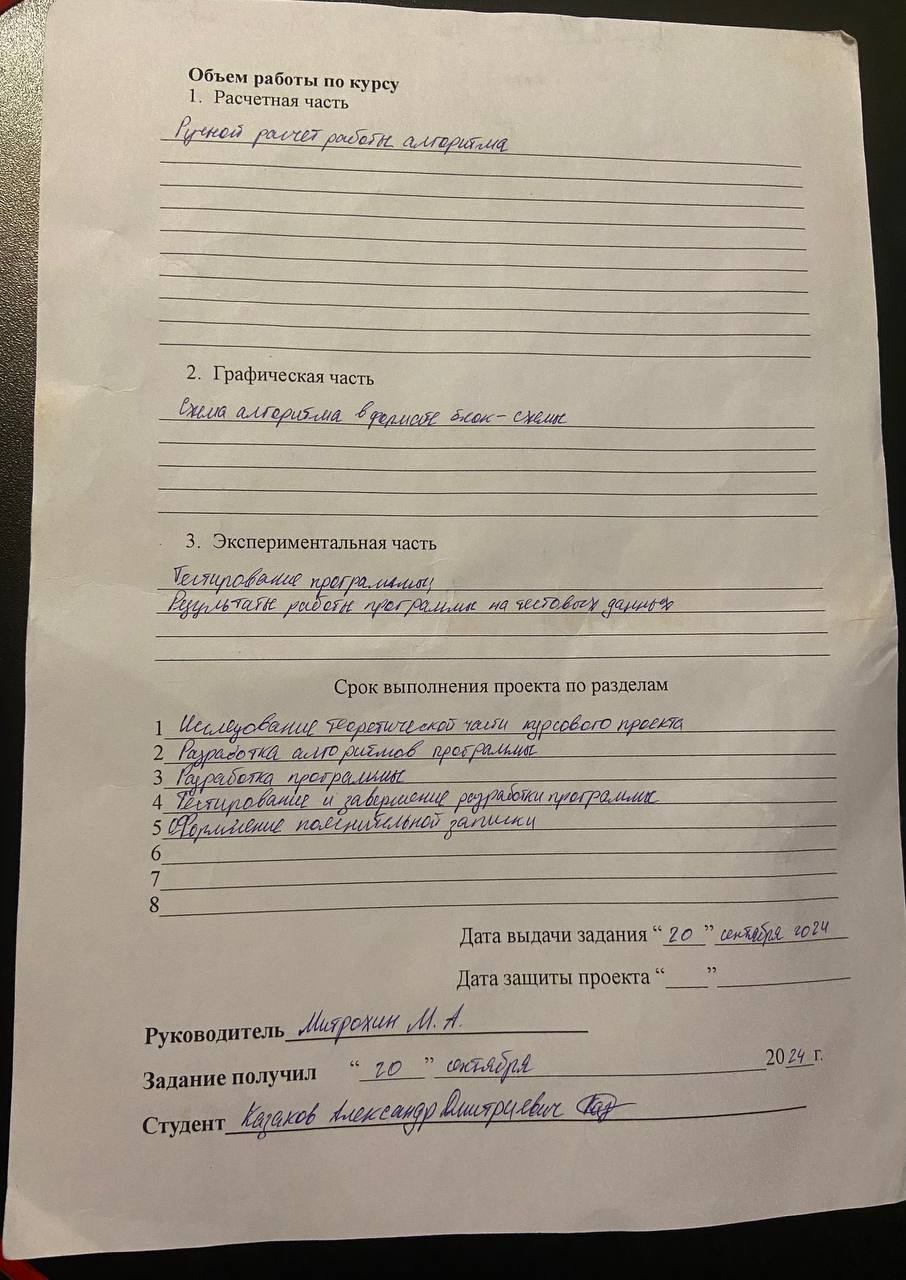
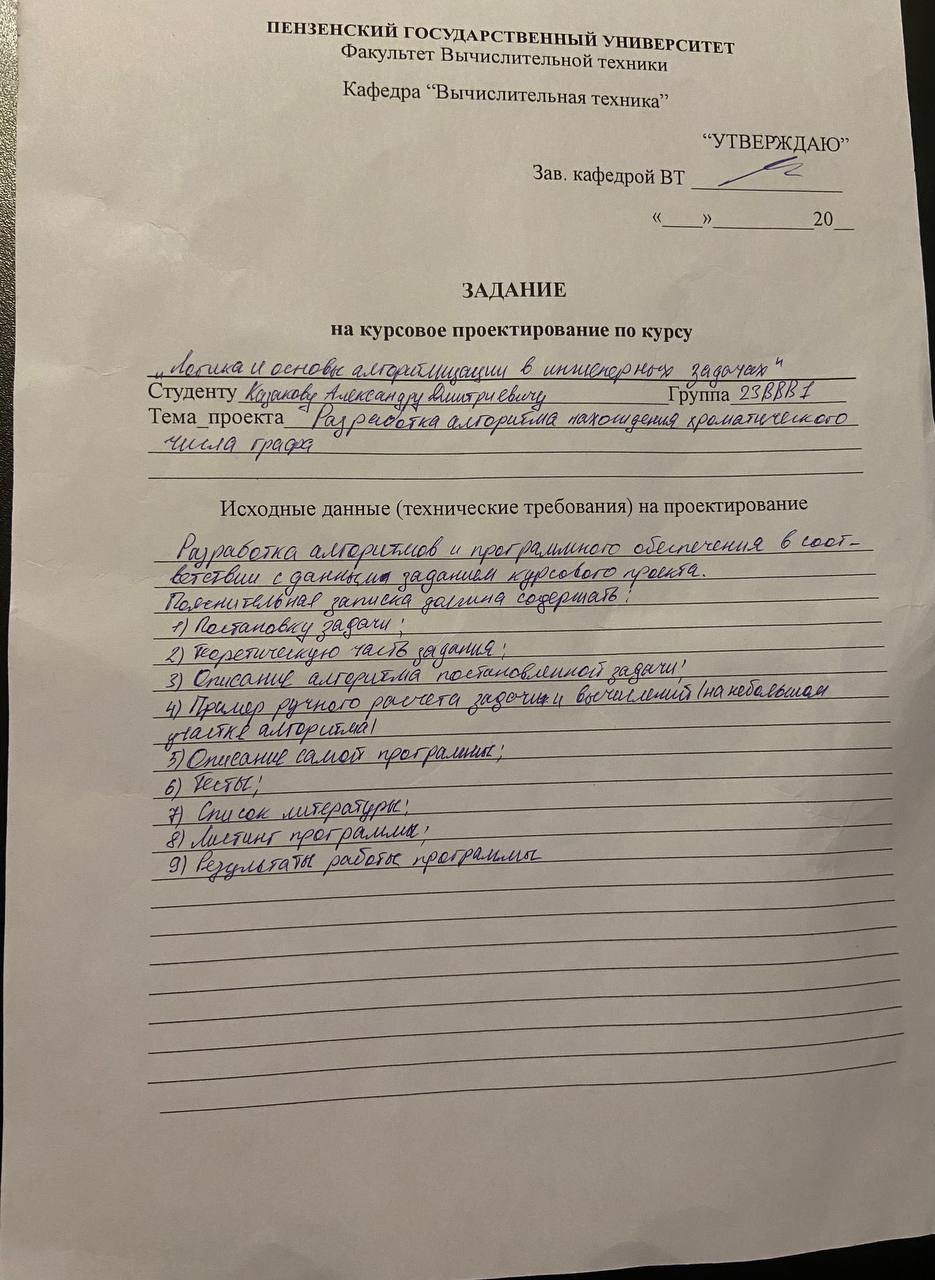
Казаков А.Д.

Приняли

К.т.н, доцент Юрова О.В

Д.т.н, профессор Митрохин М.А

Пенза 2024



**Содержание**

[Реферат 5](#_Toc184295986)

[Введение 6](#_Toc184295987)

[1. Постановка задачи 7](#_Toc184295988)

[2. Теоретическая часть задания 8](#_Toc184295989)

[3. Описание алгоритма программы 9](#_Toc184295990)

[4.Описание программы 10](#_Toc184295991)

[5. Отладка и тестирование 12](#_Toc184295992)

[6. Ручной расчет 14](#_Toc184295993)

[Заключение 16](#_Toc184295994)

[Список литературы 17](#_Toc184295995)

[Приложение А. Листинги программы 18](#_Toc184295996)

# Реферат

Отчет 22 стр, 9 рисунков.

ГРАФ, МАТРИЦА СМЕЖНОСТИ, НЕОРИЕНТИРОВАННЫЙ ГРАФ, ПЕРЕБОРНЫЙ АЛГОРИТМ, ХРОМАТИЧЕСКОЕ ЧИСЛО ГРАФОВ.

Цель исследования – разработка программного решения для генерации графа указанного размера и нахождения хроматического числа.

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

# Введение

Хроматическое число графа – это задача, которая находит свое применение в различных областях, начиная от компьютерных наук и заканчивая реальными прикладными задачами. Одной из основных задач нахождения хроматического числа графа является нахождения минимального количества цветов, которыми можно разукрасить вершины 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 нахождения хроматического числа графа, основанный на методе перебора. Он использует жадный подход, пытаясь подсчитать каждую вершину и минимально возможный цвет, в который её можно раскрасить, учитывая при этом цвета ее смежных вершин.

Для реализации данного алгоритма была выбрана среда разработки Microsoft Visual Studio 2019, а язык программирования C использован в качестве основного инструмента. Этот выбор обусловлен широким распространением языка C и его применимостью в различных областях программирования.

# 1. Постановка задачи

Необходимо разработать программу для нахождения хроматического числа 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.

# 2. 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.

Хроматическое число – это число, которое хранит в себе минимально возможное количество присвоенных цветов каждой вершине графа так, чтобы никакие две смежные вершины не имели одинаковых цветов.

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.

# 3. 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, подсчёт хроматического числа, сохранение графа 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 подсчёт хроматического числа графа.

Была реализована функция chromatic\_number, которая отвечает за подсчёт количества чисел.

**Функция chromatic\_number:**

int chromatic\_number(int \*\*graph, int n) {

int color[MAX\_VERTICES];

for (int i = 0; i < n; i++) {

color[i] = -1;

}

color[0] = 0;

for (int v = 1; v < n; v++) {

int available[MAX\_VERTICES] = { 0 };

for (int u = 0; u < n; u++) {

if (graph[v][u] == 1 && color[u] != -1) {

available[color[u]] = 1;

}

}

for (int c = 0; c < n; c++) {

if (available[c] == 0) {

color[v] = c;

break;

}

}

}

int max\_color = -1;

for (int i = 0; i < n; i++) {

if (color[i] > max\_color) {

max\_color = color[i];

}

}

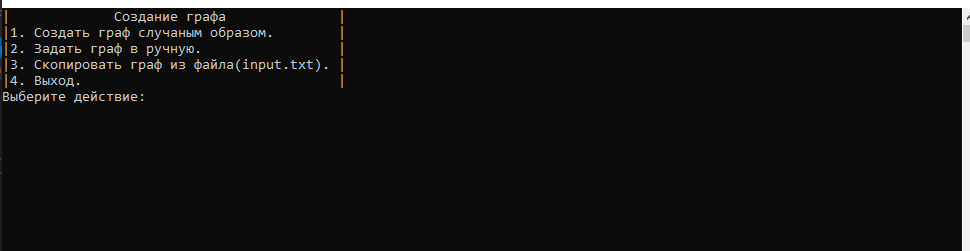
return max\_color + 1;

}

# 4.Описание программы

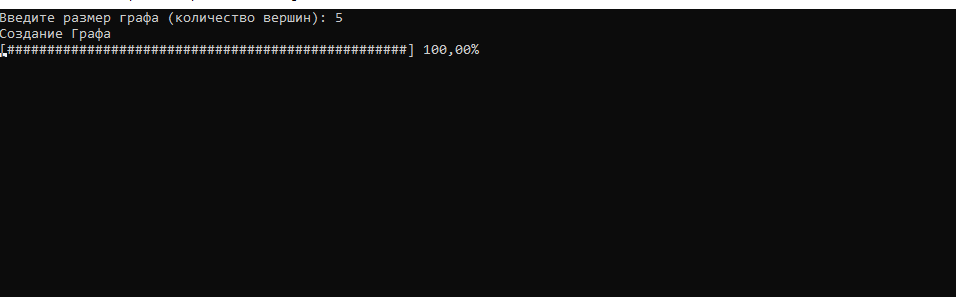
Программа состоит из одного модуля: main.cpp, где реализуется вся программа.

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 количества пройдёт анимация загрузки и появится новое меню(рисунок 3). Либо(в случае копирования) сразу переход на следующее меню:



***Рисунок 2 – Создание графа***

После загрузки выведется основное меню для работы с графом:



***Рисунок 3 – Основное меню***

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

При выборе второго пункта граф проведутся подсчёты и выведется количество цветов, в которые можно разукрасить граф.



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

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

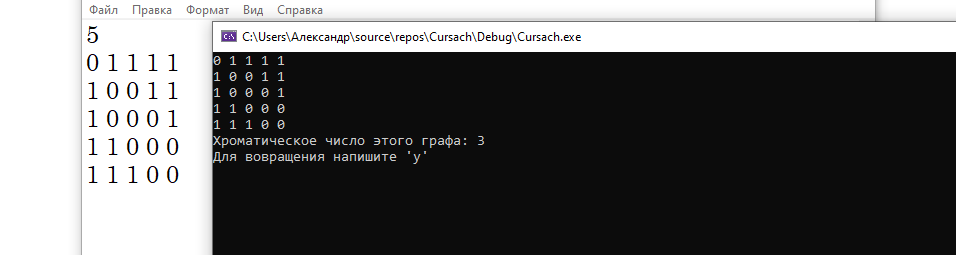
При выборе четвертого пункта пользователь вернется к выбору способа создания графа.

# 5. Отладка и тестирование

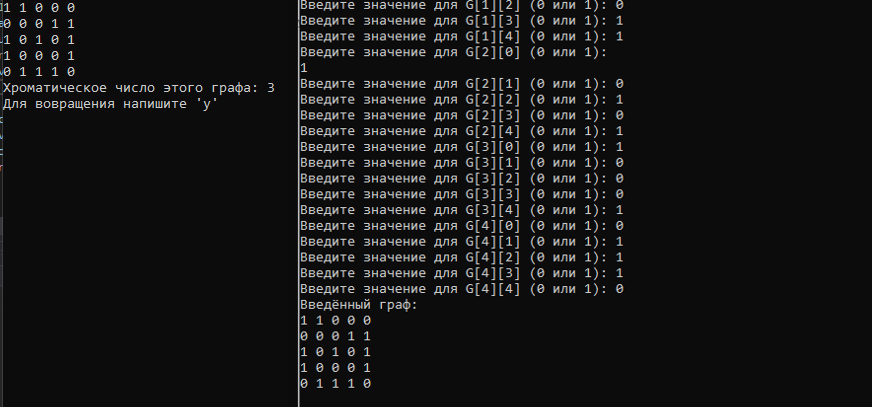
В качестве среды разработки была выбрана программа Microsoft Visual Studio 2019. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей Visual Studio: точка останова, трассировка, анализ содержимого переменных.

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 ни т. д.

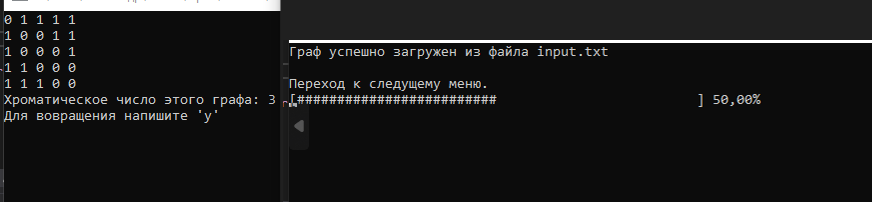
Тестирование программы:



***Рисунок 5 – создание графа случайным образом***



***Рисунок 6 – создание графа вручную***



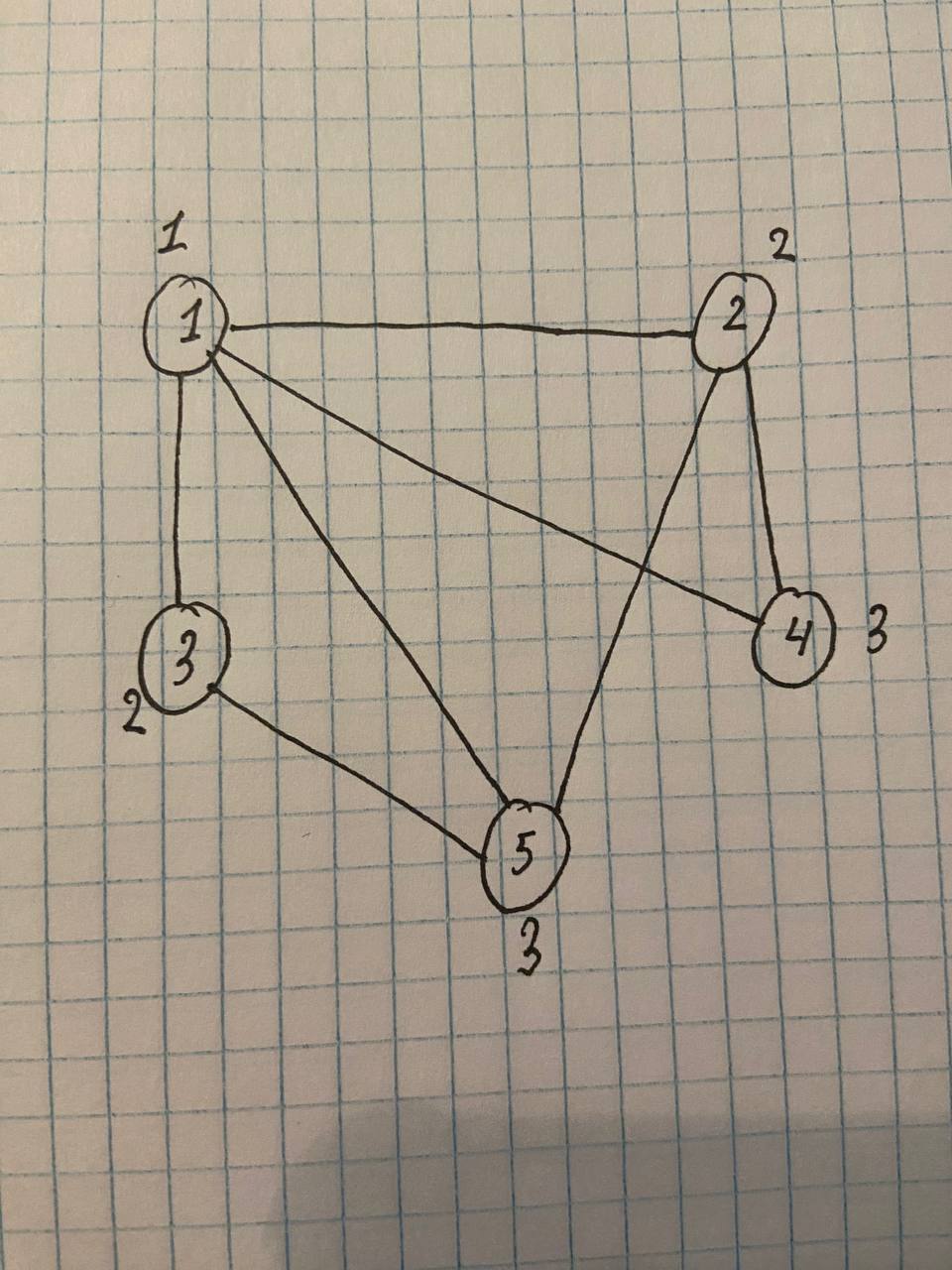
***Рисунок 7 – копирование графа из файла***

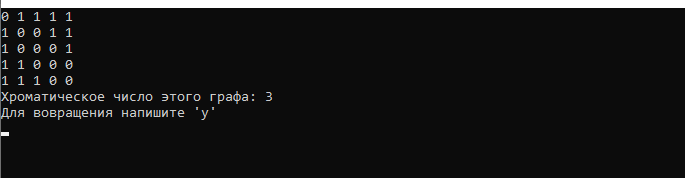
***Таблица 1 – Описание поведения программы при тестировании***

|  |  |  |
| --- | --- | --- |
| **Описание теста** | **Ожидаемый результат** | **Полученный результат** |
| Ввод кол-ва вершин | Правильное считывание размера, генерация графа нужного размера и вывод графа | Верно |
| Подсчёт Хроматического числа | Правильное количество, использовано минимальное количество цветов. | Верно |
| Сохранение в файл/чтение из файла | Граф верно считывается из файла и верно записывается результат. | Верно |

# 6. Ручной расчет

***Рисунок 8 – ручной расчет***





***Рисунок 9 – расчет программы***

# Заключение

Успешно выполнена цель: разработана программа для генерации графа и расчёта хроматического числа графа. 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 памятью, файловым вводом/выводом.

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

# Список литературы

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

### Приложение А. Листинги программы

**Main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <time.h>

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

#define MAX\_VERTICES 100

int\*\* createG(int size) {

int\*\* G;

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

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

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

}

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

for (int j = i; j < size; j++) {

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

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

G[j][i] = G[i][j];

}

}

return G;

}

int\*\* writeCreateG(int \*\*G, int size) {

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

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

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

}

printf("Введите матрицу смежности графа (%d x %d):\n", size, size);

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

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

while (1) {

printf("Введите значение для G[%d][%d] (0 или 1): ", i, j);

int input;

scanf("%d", &input);

if (input == 0 || input == 1) {

G[i][j] = input;

break;

}

else {

printf("Ошибка: введите 0 или 1.\n");

}

}

}

}

return G;

}

void printG(int\*\* G, int size) {

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

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

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

}

printf("\n");

}

}

int chromatic\_number(int \*\*graph, int n) {

int color[MAX\_VERTICES];

for (int i = 0; i < n; i++) {

color[i] = -1;

}

color[0] = 0;

for (int v = 1; v < n; v++) {

int available[MAX\_VERTICES] = { 0 };

for (int u = 0; u < n; u++) {

if (graph[v][u] == 1 && color[u] != -1) {

available[color[u]] = 1;

}

}

for (int c = 0; c < n; c++) {

if (available[c] == 0) {

color[v] = c;

break;

}

}

}

int max\_color = -1;

for (int i = 0; i < n; i++) {

if (color[i] > max\_color) {

max\_color = color[i];

}

}

return max\_color + 1;

}

void saveGraphToFile(int\*\* G, int size) {

FILE\* file = fopen("input.txt", "w");

if (file == NULL) {

printf("Ошибка при открытии файла для записи.\n");

return;

}

fprintf(file, "%d\n", size);

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

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

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

}

fprintf(file, "\n");

}

fclose(file);

printf("Граф успешно сохранен в файл input.txt\n");

}

void firstMenu() {

printf("| %12sСоздание графа %12s |\n","","");

printf("|1. Создать граф случаным образом. %7s|\n", "");

printf("|2. Задать граф в ручную. %16s|\n", "");

printf("|3. Скопировать граф из файла(input.txt). |\n");

printf("|4. Выход.%31s |\n", "");

printf("Выберите действие: ");

}

void nameMenu() {

printf("\%40s Курсовая работа\n", "");

printf("\%20s Тема: Разработка алгоритма нахождения хроматического числа графа\n\n\n\n\n\n\n\n\n", "");

printf("\%90s Выполнил: ст. гр. 23ВВВ1\n", "");

printf("\%103s Казаков А.Д.\n", "");

printf("Для продолжения нажмите клавишу Enter...");

}

void secondText() {

printf("| %12sРабота с графом %12s |\n", "", "");

printf("|1. Просмотр графа. %23s|\n", "");

printf("|2. Подсчёт Хроматического числа графа. %3s|\n", "");

printf("|3. Запись графа в файл (input.txt). %6s|\n", "");

printf("|4. Вернуться к созданию графа. %11s|\n", "");

printf("Выберите действие: ");

}

int\*\* loadGraphFromFile(int \*size) {

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

if (file == NULL) {

printf("Ошибка при открытии файла для чтения.\n");

return NULL;

}

fscanf(file, "%d", size);

int size1 = \*size;

int\*\* G = (int\*\*)malloc(size1 \* sizeof(int\*));

for (int i = 0; i < size1; i++) {

G[i] = (int\*)malloc( size1 \* sizeof(int));

}

for (int i = 0; i < size1; i++) {

for (int j = 0; j < size1; j++) {

fscanf(file, "%d ", &G[i][j]);

}

}

fclose(file);

printf("Граф успешно загружен из файла input.txt\n");

return G;

}

int secondMenu(int \*\*G, int size, char ch) {

int secondChoice = 0;

do {

system("cls");

secondText();

scanf("%d", &secondChoice);

switch (secondChoice)

{

case 1: {

system("cls");

printf("Матрица смежности для графа\n");

printG(G,size);

printf("Для вовращения напишите 'y'\n");

scanf("%s", &ch);

break;

}

case 2: {

system("cls");

int chromaticNum = chromatic\_number(G, size);

printG(G,size);

printf("Хроматическое число этого графа: %d\n", chromaticNum);

printf("Для вовращения напишите 'y'\n");

scanf("%s", &ch);

break;

}

case 3: {

system("cls");

saveGraphToFile(G, size);

Sleep(2000);

break;

}

case 4: {

system("cls");

break;

}

default:

printf("\nНеверный выбор! Повтори попытку.\n\nЧерез ");

for (int i = 2; i > 0; i--) {

printf("%d.. ", i);

Sleep(1000);

}

break;

}

} while (secondChoice != 4);

return 1;

}

void displayProgressBar(int progress, int total) {

int barWidth = 50;

float ratio = (float)progress / total;

int pos = barWidth \* ratio;

printf("[");

for (int i = 0; i < barWidth; i++) {

if (i < pos) {

printf("#");

}

else printf(" ");

}

printf("] %.2f%%\r", ratio \* 100);

fflush(stdout);

}

int main() {

srand(time(NULL));

setlocale(LC\_ALL, "Rus");

int\*\* G = NULL, count = 0;

int\* size1 = 0;

int size = 0;

int choice = 0, secondChoice = 0;

char wait = '0';

nameMenu();

getchar();

do {

system("cls");

firstMenu();

scanf("%d", &choice);

switch (choice) {

case 1: {

system("cls");

printf("Введите размер графа (количество вершин): ");

scanf("%d", &size);

G = createG(size);

printf("Создание Графа\n");

for (int i = 0; i <= 3; i++) {

displayProgressBar(i, 3);

Sleep(1000);

}

secondMenu(G,size, wait);

break;

}

case 2: {

system("cls");

printf("Введите размер графа (количество вершин): ");

scanf("%d", &size);

G= writeCreateG(G,size);

printf("Введённый граф:\n");

printG(G, size);

printf("\nПереход к следущему меню.\n");

for (int i = 0; i <= 2; i++) {

displayProgressBar(i, 2);

Sleep(1000);

}

system("cls");

secondText();

scanf("%d", &secondChoice);

secondMenu(G, size, wait);

break;

}

case 3:

system("cls");

G = loadGraphFromFile(&size);

printf("\nПереход к следущему меню.\n");

for (int i = 0; i <= 2; i++) {

displayProgressBar(i, 2);

Sleep(1000);

}

secondMenu(G, size, wait);

break;

case 4:

break;

default:

printf("\nНеверный выбор! Повтори попытку.\n\nЧерез ");

for (int i = 2; i >0; i--) {

printf("%d.. ", i);

Sleep(1000);

}

break;

}

} while (choice != 4);

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

free(G[i]);

}

free(G);

return 0;

}