Министерство образования Российской Федерации

Пензенский государственный университета

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ АЗАПИСКА**

К курсовому проектированию

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Реализация алгоритма раскрашивания графа»

Выполнил:

Студент группы 22ВВВ1

Курушин Я.С.

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 25 стр, 12 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, "жадный алгоритм раскраски графа" (Greedy Graph Coloring Algorithm). 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, можно выполнить раскраску графа или сохранить его в файл.

Была реализована функция colorGraph, которая отвечает за раскраску графа.

**Функция colorGraph:**

void colorGraph(int\*\* arr, int rows, int\* colors) {

// Инициализация массива цветов

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

colors[i] = -1;

}

// Проход по всем вершинам графа

for (int node = 0; node < rows; node++) {

// Проверка, свободен ли цвет для текущей вершины

bool\* available = (bool\*)malloc(rows \* sizeof(bool));

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

available[i] = true;

}

for (int neighbor = 0; neighbor < rows; neighbor++) {

if (arr[node][neighbor] && colors[neighbor] != -1) {

available[colors[neighbor]] = false;

}

}

// Находим минимальный доступный цвет

for (int color = 0; color < rows; color++) {

if (available[color]) {

colors[node] = color;

break;

}

}

}

}

Ниже представлен псевдокод функции colorGraph:

1. void colorGraph(int\*\* arr, int rows, int\* colors):

a. для i от 0 до rows:

I. colors[i] = -1

II. конец цикла

b. для node от 0 до rows:

I. bool\* available = создатьМассивИстин(rows)

II. для neighbor от 0 до rows:

1. если arr[node][neighbor] и colors[neighbor] != -1:

a. available[colors[neighbor]] = false

2. конец условия и конец цикла

III. для color от 0 до rows:

1. если available[color]:

a. colors[node] = color

b. выйти из цикла

2. конец условия и конец цикла

IV. освободить память, выделенную для available

V. конец цикла

c. конец функции

# Описание программы

Программа состоит из нескольких модулей (многомодульная): Source.cpp, где реализуется вся программа, Header.h – заголовочный файл, Functions.cpp – файл с функциями.

Разработанная программа состоит из нескольких функций модуля, которые реализовывались в следующем порядке:

1. main()– Меню программы.
2. Header.h – библиотека с константами.
3. addSp()– функция для добавления вершины в список смежности.
4. printSp() – функция для вывода на экран списка смежности.
5. inputNumbers() – функция для проверки ввода чисел пользователем.
6. colorGraph() – функция для раскрашивания графа.

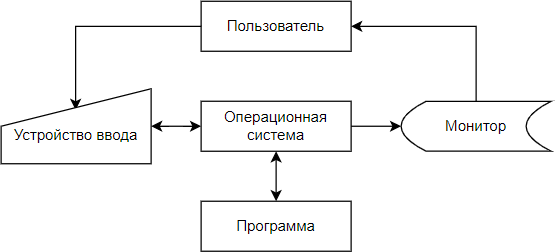
По порядку реализации модулей можно понять, что использовался нисходящей подход реализации программы. Этот метод основан на том, что 5 функций сначала проектируются основные компоненты программы, а затем уже дорабатываются ее мелкие детали. Это позволяет упростить ее тестирование, так как каждая функция (модуль) тестируется сразу же после его реализации, что помогает в создании более структурированных программ.

### При запуске программы происходит вывод заставки-приветствия, после чего пользователю необходимо выбрать пункт меню для дальнейшего использования программы. Выход из меню осуществляется после выбора пользователем нужного пункта. Описание состояний программы выполнено в таблице ниже.

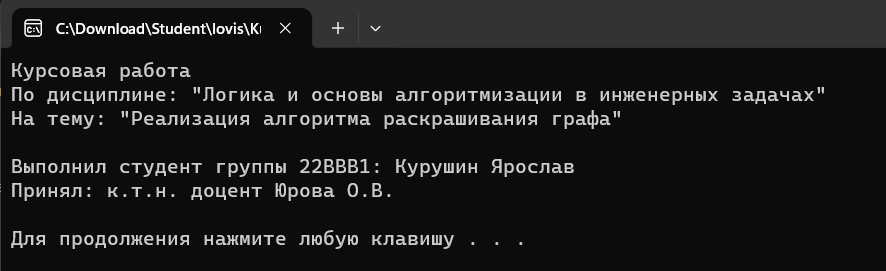
#### Таблица 1 – Главная функция

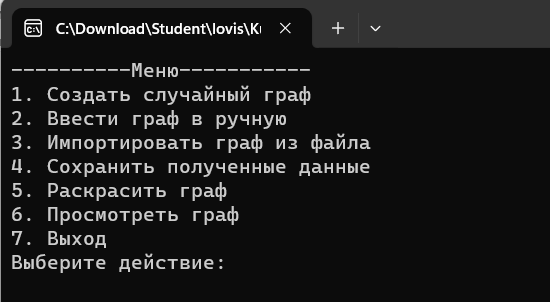
|  |  |  |
| --- | --- | --- |
| Клавиши,  вызывающие событие | Действие  пользователя | Действие программы |
| 1, Enter | Выбран пункт  «Создать случайный граф» | Запускается диалог создания случайного графа |
| 2, Enter | Выбран пункт  «Ввести граф в ручную» | Запускается диалог  Создания графа |
| 3, Enter | Выбран пункт  «Импортировать граф из файла» | Запускает диалог импортирования графа из файла |
| 4, Enter | Выбран пункт  «Сохранить полученные данные» | Запускается диалог сохранения полученных данных в файл |
| 5, Enter | Выбран пункт  «Раскрасить граф» | Запускается алгоритм раскраски графа |
| 6, Enter | Выбран пункт  « Просмотреть граф» | Выводится матрица смежности и список смежности графа, если он существует |
| 7, Enter | Выбран пункт  «Выход» | Программа завершает свою работу |

### Далее на рисунке 1 представлена схема данных.

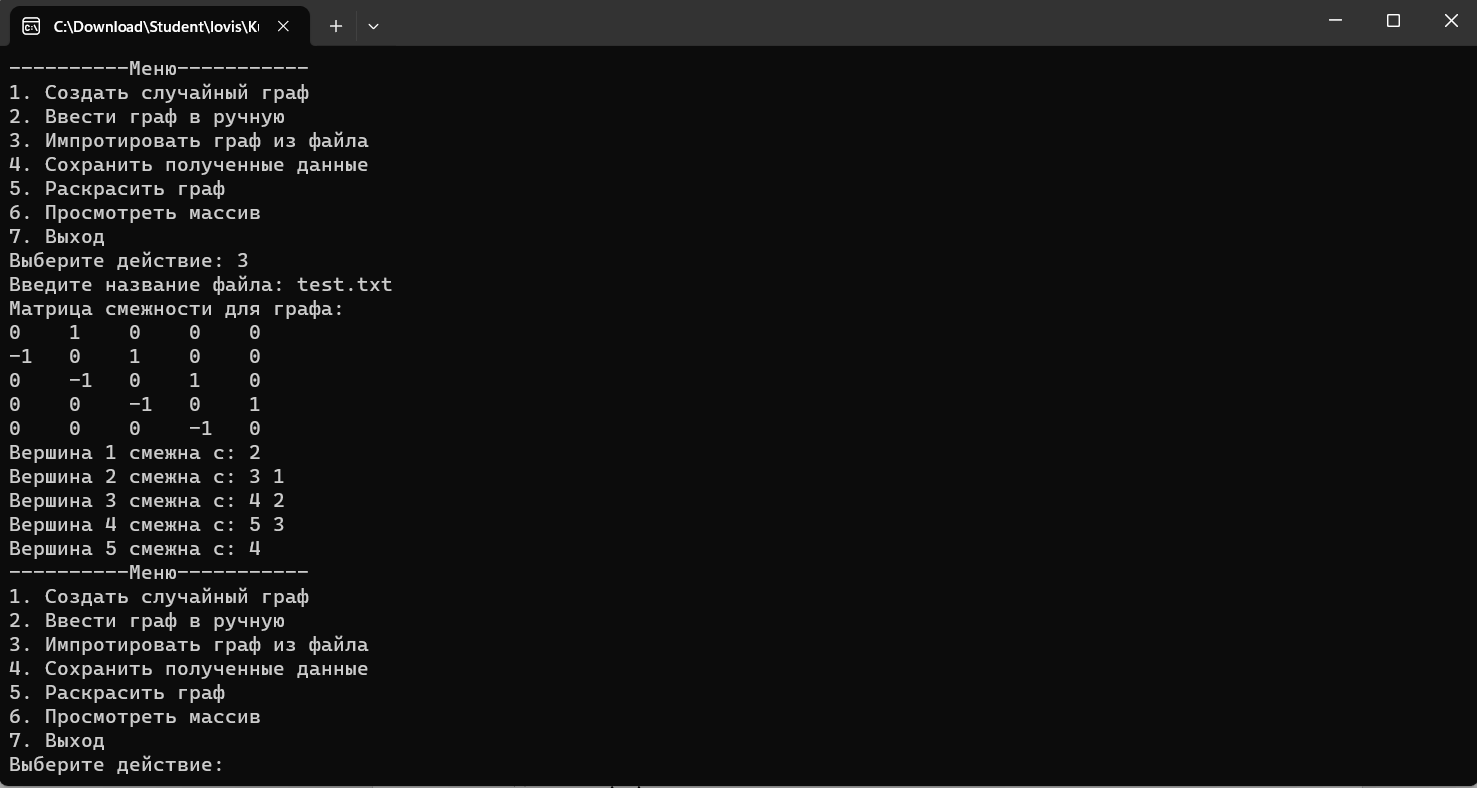


***Рисунок 1 - Схема данных***

При запуске программы открывается консольное окно с меню:



**Рисунок 2 – Главное меню**

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:

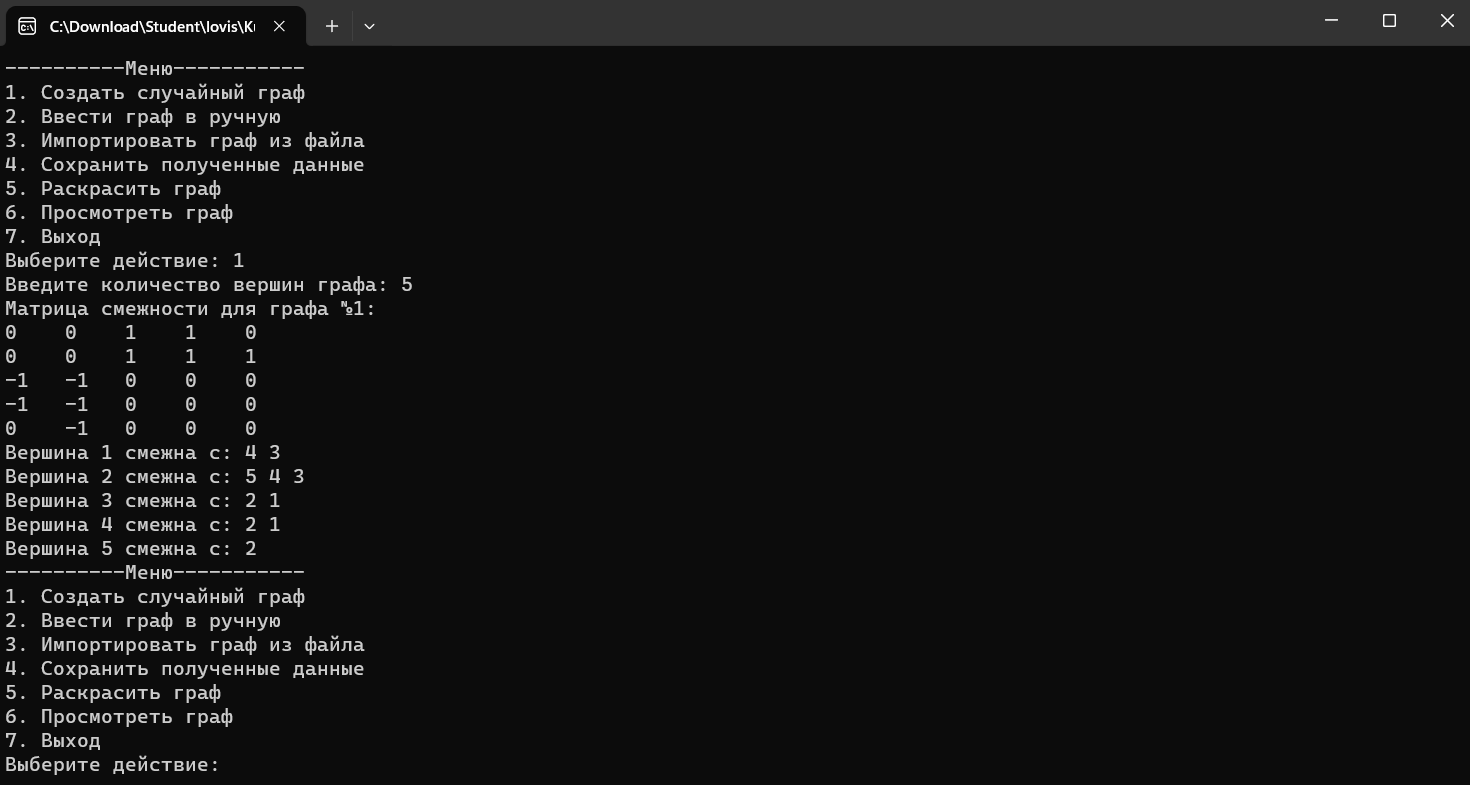
**Рисунок 3 – Создание графа**

# Отладка и тестирование

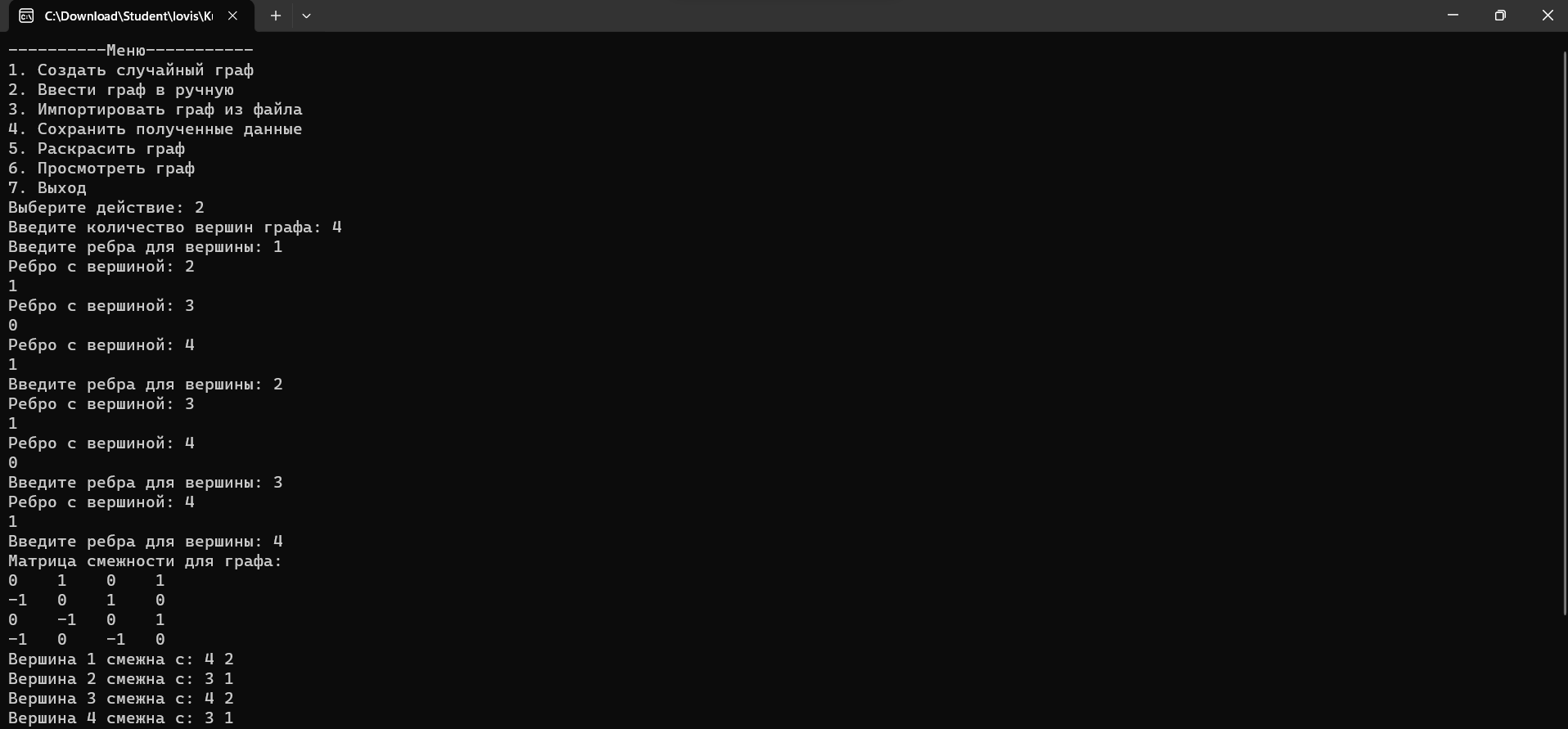
В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей Visual Studio: точка останова, трассировка, анализ содержимого переменных.

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 полями структуры, с размерами самих массивов и т. д.

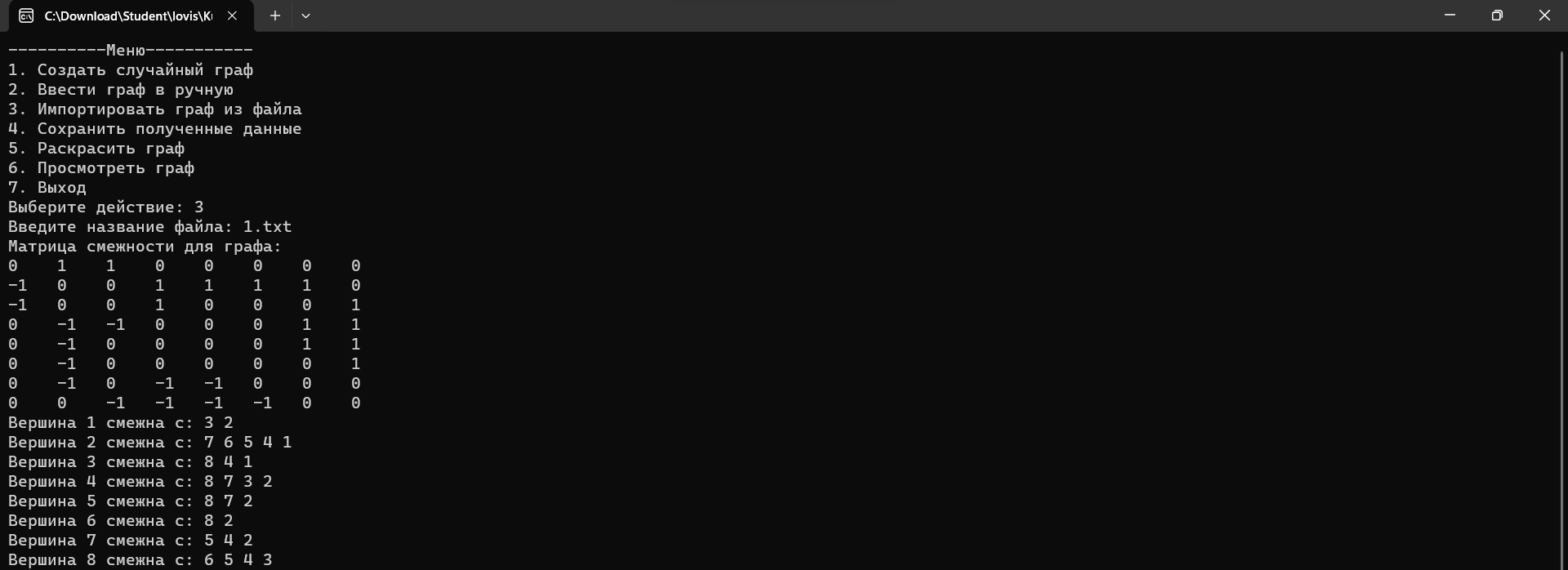
Тестирование программы:

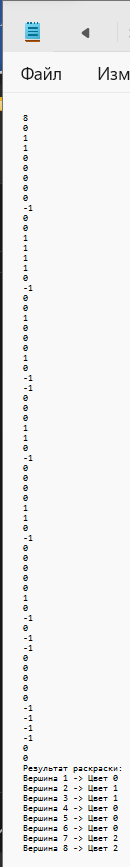


**Рисунок 4 – создание графа случайным образом**

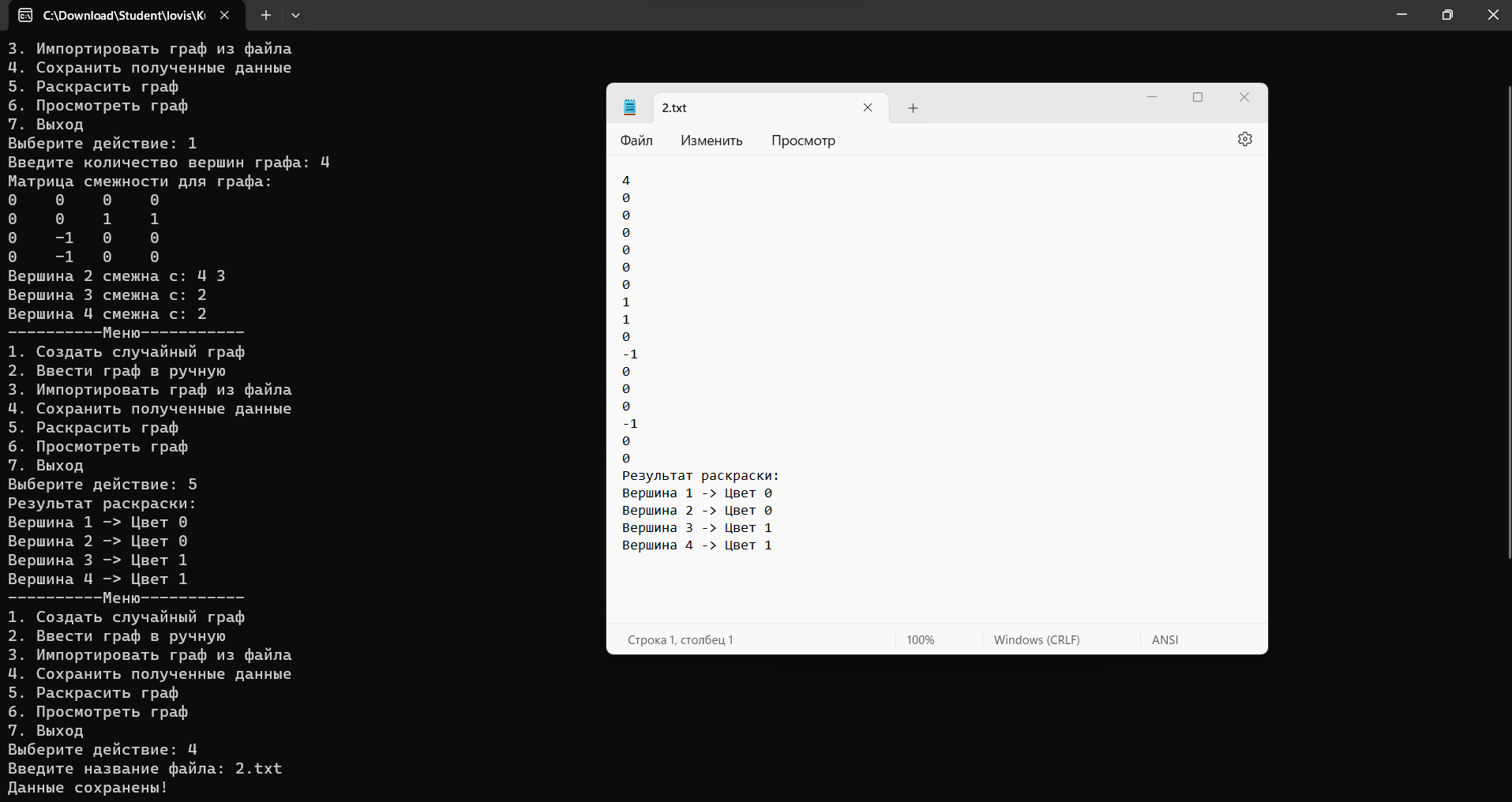


**Рисунок 5 – создание графа вручную**

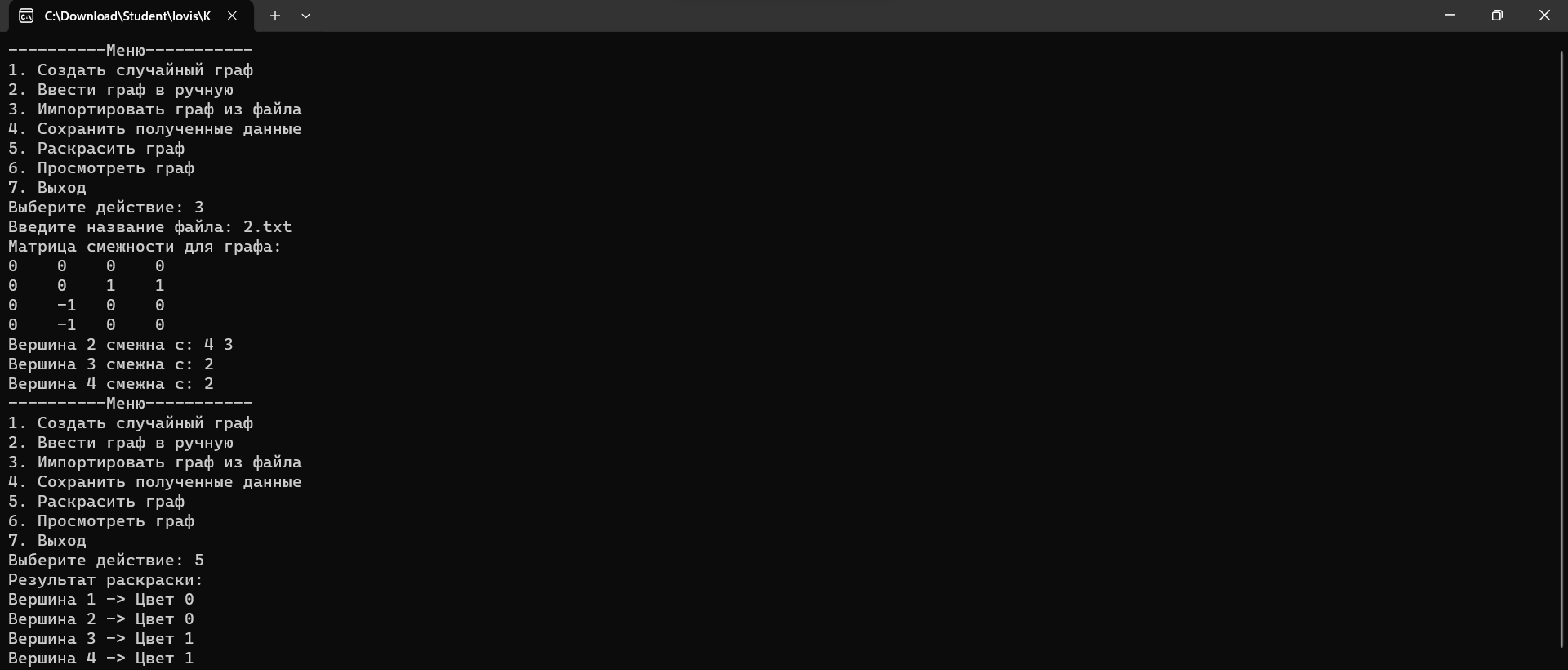




**Рисунок 6 – копирование графа из файла**



**Рисунок 7 – сохранение графа в файл**



**Рисунок 8 – раскраска графа**



**Рисунок 9 – просмотр графа**



**Рисунок 10 – выход из программы**

***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 | Правильное раскрашивание, 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

# 

**Рисунок 11 – ручной расчет**

# 

**Рисунок 12 – расчет программы**

# Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Была освоена работа с файлами, изучены функции работы с консолью и оперативной памятью. Также были получены основные навыки отладки и тестирования программ и программирования в среде Visual Studio 2022 на языке Си.

В рамках курсовой работы была написана программа для раскраски графа. Она имеет все необходимые функции для работы с графом.

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 дальнейшем программу можно улучшить, путём добавления графического интерфейса, что позволит улучшить навигацию по меню программы, путём добавления возможности использования мыши.

# Список литературы

### Э. Таненбаум, Т. Остин. Архитектура компьютера (6-е издание). 2016 г.

### Шилдт Герберт. Полный справочник по С. 2002 г.

### MSDN.

### Дэвид Гриффитс, Дон Гриффитс. Изучаем программирование на С. 2013 г.

### Брайан Керниган. Деннис Ритчи. Язык программирования С. 2009 г.

1. Лекция, Деревья\_потоки\_раскрашивания, Митрохин М.А.

# Приложение А. Листинги программы

**Source.cpp**

#include "Header.h"

void main() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int\*\* arr = 0; // матрица смежности

int\* colors = 0; // массив для цветов

int i, j, rows;

FILE\* file;

Node\*\* arr\_sp = NULL;

char name[255];

int choice, rows1, rows2, what;

printf("Курсовая работа\nПо дисциплине: \"Логика и основы алгоритмизации в инженерных задачах\"\nНа тему: \"Реализация алгоритма раскрашивания графа\"\n\nВыполнил студент группы 22ВВВ1: Курушин Ярослав\nПринял: к.т.н. доцент Юрова О.В.\n\n");

system("PAUSE");

system("cls");

do {

printf("----------Меню-----------\n");

printf("1. Создать случайный граф\n");

printf("2. Ввести граф в ручную\n");

printf("3. Импортировать граф из файла\n");

printf("4. Сохранить полученные данные\n");

printf("5. Раскрасить граф\n");

printf("6. Просмотреть граф\n");

printf("7. Выход\n");

printf("Выберите действие: ");

choice = inputNumber();

switch (choice) {

case 1: // Создать случайный граф

printf("Введите количество вершин графа: ");

rows = inputNumber();

arr = (int\*\*)malloc(rows \* sizeof(int\*));

if (arr == NULL) {

printf("Не удалось выделить память!\n");

break;;

}

for (i = 0; i < rows; i++) {

arr[i] = (int\*)malloc(rows \* sizeof(int));

}

// генерировать случайные значения для матрицы смежности

for (i = 0; i < rows; i++) {

for (j = i; j < rows; j++) {

if (i == j) {

arr[i][j] = 0; // на главной диагонали нули

}

else {

arr[i][j] = rand() % 2; //(-10) + rand() % 21; // случайные значения 0 или 1

arr[j][i] = -arr[i][j]; // симметрично заполнять значения для неориентированного графа

}

}

}

// выводить матрицу смежности на экран

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

for (i = 0; i < rows; i++) {

for (j = 0; j < rows; j++) {

printf("%-4d ", arr[i][j]);

}

printf("\n");

}

// Выделение памяти для массива списков смежности

arr\_sp = (Node\*\*)malloc(rows \* sizeof(Node\*));

if (arr\_sp == NULL) {

printf("Не удалось выделить память!\n");

return;

}

for (i = 0; i < rows; i++) {

arr\_sp[i] = NULL;

}

addSp(arr\_sp, arr, rows);

printSp(arr\_sp, rows);

break;

case 2: // Ввести граф в ручную

printf("Введите количество вершин графа: ");

rows = inputNumber();

arr = (int\*\*)malloc(rows \* sizeof(int\*));

if (arr == NULL) {

printf("Не удалось выделить память!\n");

break;

}

for (i = 0; i < rows; i++) {

arr[i] = (int\*)malloc(rows \* sizeof(int));

}

for (i = 0; i < rows; i++) {

printf("Введите ребра для вершины: %d\n", i + 1);

for (j = i; j < rows; j++) {

if (i == j) {

arr[i][j] = 0;

}

else {

printf("Ребро с вершиной: %d\n", j+1);

scanf("%d", &arr[i][j]);

arr[j][i] = -arr[i][j];

}

}

}

// выводить матрицу смежности на экран

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

for (i = 0; i < rows; i++) {

for (j = 0; j < rows; j++) {

printf("%-4d ", arr[i][j]);

}

printf("\n");

}

// Выделение памяти для массива списков смежности

arr\_sp = (Node\*\*)malloc(rows \* sizeof(Node\*));

if (arr\_sp == NULL) {

printf("Не удалось выделить память!\n");

return;

}

for (i = 0; i < rows; i++) {

arr\_sp[i] = NULL;

}

addSp(arr\_sp, arr, rows);

printSp(arr\_sp, rows);

break;

case 3: // Импротировать граф из файла

printf("Введите название файла: ");

scanf("%s", &name);

if ((file = fopen(name, "r")) == NULL)

{

printf("Не удалось открыть файл");

getchar();

break;

}

fclose(file);

file = fopen(name, "r");

fscanf(file, "%d", &rows);

arr = (int\*\*)malloc(rows \* sizeof(int\*));

if (arr == NULL) {

printf("Не удалось выделить память!\n");

break;

}

for (i = 0; i < rows; i++) {

arr[i] = (int\*)malloc(rows \* sizeof(int));

}

for (i = 0; i < rows; i++) {

for (j = 0; j < rows; j++) {

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

}

}

fclose(file);

// выводить матрицу смежности на экран

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

for (i = 0; i < rows; i++) {

for (j = 0; j < rows; j++) {

printf("%-4d ", arr[i][j]);

}

printf("\n");

}

// Выделение памяти для массива списков смежности

arr\_sp = (Node\*\*)malloc(rows \* sizeof(Node\*));

if (arr\_sp == NULL) {

printf("Не удалось выделить память!\n");

return;

}

for (i = 0; i < rows; i++) {

arr\_sp[i] = NULL;

}

addSp(arr\_sp, arr, rows);

printSp(arr\_sp, rows);

break;

case 4: // Сохранить полученные данные

if (arr == NULL) {

printf("Сначала создайте граф!\n");

break;

}

printf("Введите название файла: ");

scanf("%s", &name);

if ((file = fopen(name, "w")) == NULL)

{

printf("Не удалось открыть файл");

getchar();

break;

}

fclose(file);

file = fopen(name, "w");

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

for (i = 0; i < rows; i++) {

for (j = 0; j < rows; j++) {

fprintf(file, "%d\n", arr[i][j]);

}

}

fclose(file);

if (colors == NULL) {

break;

}

file = fopen(name, "a");

fprintf(file, "Результат раскраски:\n");

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

fprintf(file, "Вершина %d -> Цвет %d\n", i + 1, colors[i]);

}

fclose(file);

printf("Данные сохранены!\n");

break;

case 5: // Раскрасить граф

if (arr == NULL) {

printf("Сначала создайте граф!\n");

break;

}

colors = (int\*)malloc(rows \* sizeof(int));

if (colors == NULL) {

printf("Не удалось выделить память!\n");

break;

}

colorGraph(arr, rows, colors);

printf("Результат раскраски:\n");

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

printf("Вершина %d -> Цвет %d\n", i + 1, colors[i]);

}

break;

case 6: // Просмотреть массив

if (arr == NULL) {

printf("Сначала создайте граф!\n");

break;

}

// выводить матрицу смежности на экран

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

for (i = 0; i < rows; i++) {

for (j = 0; j < rows; j++) {

printf("%-4d ", arr[i][j]);

}

printf("\n");

}

printSp(arr\_sp, rows);

break;

case 7: // Выход

system("cls");

printf("До свидания!\n");

break;

default:

printf("Некорректный выбор!\n");

break;

}

} while (choice != 7);

}

**Header.h**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include <string.h>

#include <ctype.h>

typedef struct Node {

int inf;

struct Node\* next;

}Node;

int inputNumber();

void colorGraph(int\*\* arr, int rows, int\* colors);

void addSp(Node\*\* arr\_sp, int\*\* arr, int rows);

void printSp(Node\*\* arr\_sp, int rows);

**Functions.cpp**

#include "Header.h"

void addSp(Node\*\* arr\_sp, int\*\* arr, int rows) {

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

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

if (arr[i][j] != 0) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->inf = j + 1;

newNode->next = arr\_sp[i];

arr\_sp[i] = newNode;

}

}

}

}

void printSp(Node\*\* arr\_sp, int rows) {

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

Node\* current = arr\_sp[i];

if (current == NULL) {

}

else {

printf("Вершина %d смежна с: ", i + 1);

while (current != NULL) {

printf("%d ", current->inf);

current = current->next;

}

printf("\n");

}

}

}

// Функция для ввода цифр с проверкой

int inputNumber() {

char input[255];

int number;

int isValidInput = 0;

do {

isValidInput = 1; // Предполагаем, что ввод корректен

scanf("%s", input);

// Проверяем каждый символ в строке

for (int i = 0; input[i] != '\0'; i++) {

if (!isdigit(input[i])) {

isValidInput = 0; // Некорректный ввод, если символ не цифра

printf("Ошибка! Введите только цифры.\n");

break;

}

}

// Очищаем буфер

while (getchar() != '\n');

} while (!isValidInput);

// Преобразуем строку в число

sscanf(input, "%d", &number);

return number;

}

void colorGraph(int\*\* arr, int rows, int\* colors) {

// Инициализация массива цветов

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

colors[i] = -1;

}

// Проход по всем вершинам графа

for (int node = 0; node < rows; node++) {

// Проверка, свободен ли цвет для текущей вершины

bool\* available = (bool\*)malloc(rows \* sizeof(bool));

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

available[i] = true;

}

for (int neighbor = 0; neighbor < rows; neighbor++) {

if (arr[node][neighbor] && colors[neighbor] != -1) {

available[colors[neighbor]] = false;

}

}

// Находим минимальный доступный цвет

for (int color = 0; color < rows; color++) {

if (available[color]) {

colors[node] = color;

break;

}

}

}

}