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 «Реализация алгоритма нахождения хроматического числа графа»

Выполнила:

Студентка группы 19ВВ1:

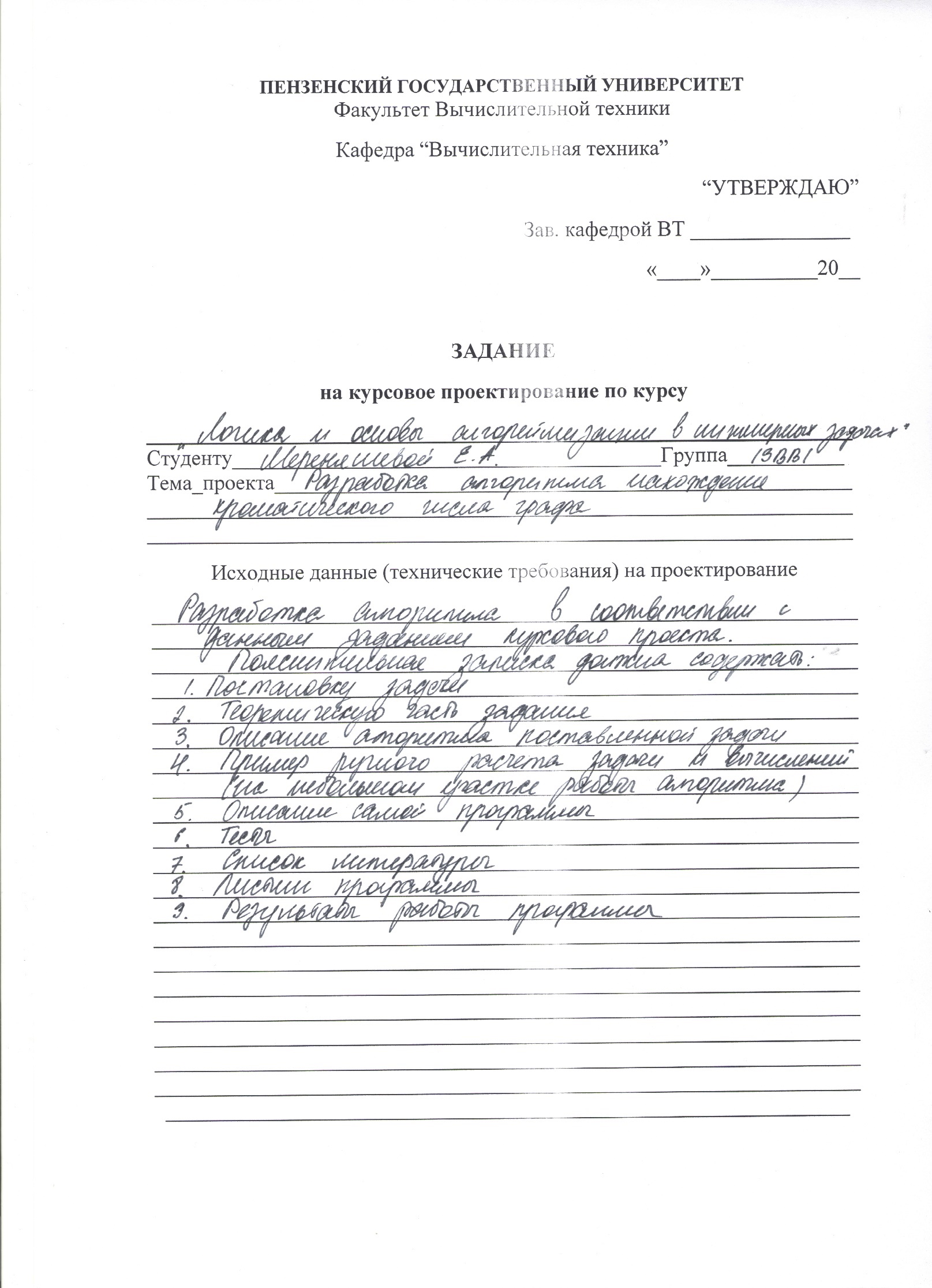
Мереняшева Е.А.

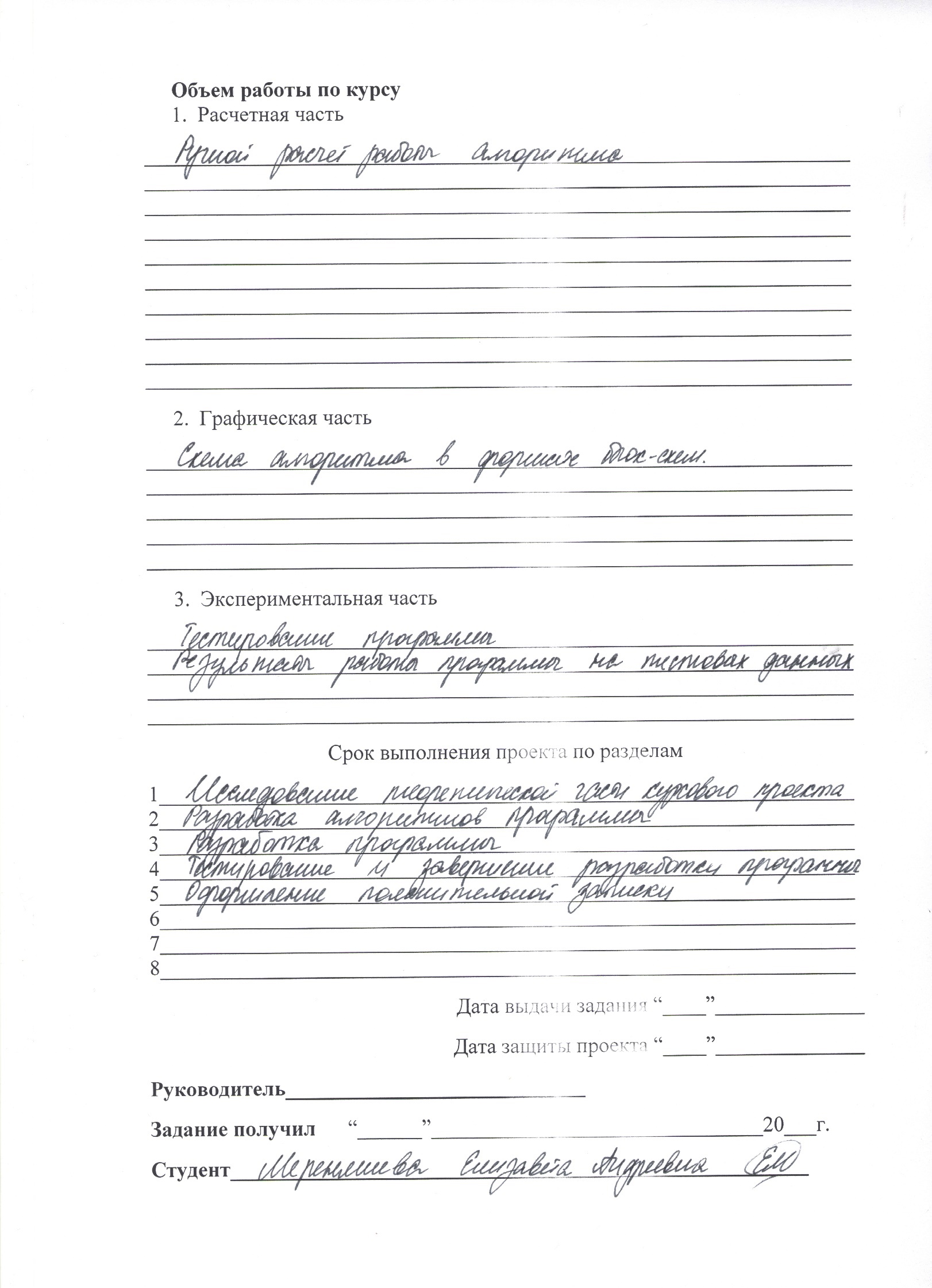
Приняли:

Митрохин М.А.

Юрова О.В.

Пенза 2020

****

****

**Содержание**

[Введение 6](#_Toc56629276)

[Постановка задачи 7](#_Toc56629277)

[Теоретическая часть задания 8](#_Toc56629278)

[Описание алгоритма программы 9](#_Toc56629279)

[Описание программы 12](#_Toc56629280)

[Тестирование 16](#_Toc56629281)

[Заключение 24](#_Toc56629282)

[Список литературы 24](#_Toc56629283)

[Приложение А. 25](#_Toc56629284)

# Введение

Граф в математике – это набор точек (вершин графа), соединенных отрезками (ребрами графа).

Раскраска вершин — это присвоение цветов вершинам графа так, что никакие две смежные вершины не имеют одинакового цвета. Проще говоря, две вершины ребра не должны быть одного цвета.

Главная задача курсовой работы - в какое наименьшее число цветов можно раскрасить плоскость так, чтобы любые две точки на расстоянии 1 были покрашены в различные цвета?  
  
 Минимальное количество цветов, необходимое для раскраски вершин графа, называется хроматическим числом.

Первоначально раскраски графов были нужны для составления географических карт. Сегодня же они используются, например, для составления расписаний, распределения регистров в микропроцессорах, распараллеливания численных методов.

В качестве среды разработки мною была выбрана среда Microsoft Visual Studio 2019, а язык программирования – С#.

# Постановка задачи

Цель исследования – разработка программы, способная находить хроматическое число графа.

Требуется разработать программу, которая должна находить хроматическое число графа.

Пользователь должен иметь возможность сам выбирать размер графа. В программе должна присутствовать возможность выбора автоматического или ручного задания графа. Программа должна иметь либо графическое, либо текстовое меню. Необходимо реализовать сохранение результата.

Устройство ввода – клавиатура и мышь.

Задания выполняются в соответствии с вариантом №25 (разработка алгоритма нахождения хроматического числа графа).

# Теоретическая часть задания

Задачу раскраски графов сформулировали Хуго Хадвигер и Пал Эрдёш в сороковых годах XX века. Независимо от них примерно в то же самое время этой задачей занимались Эдуард Нелсон и Дж. Р. Исбелл. После работы Хадвигера 1961 года об открытых на тот момент проблемах, хроматическое число плоскости стало активно изучаться.

Граф G (рисунок 1) задается множеством вершин X1, X2, ..., Xn. и множеством ребер, соединяющих между собой определенные вершины.

Каждой вершине соответствует цвет, такой, что бы никакие две смежные вершины не имели одинакового цвета.

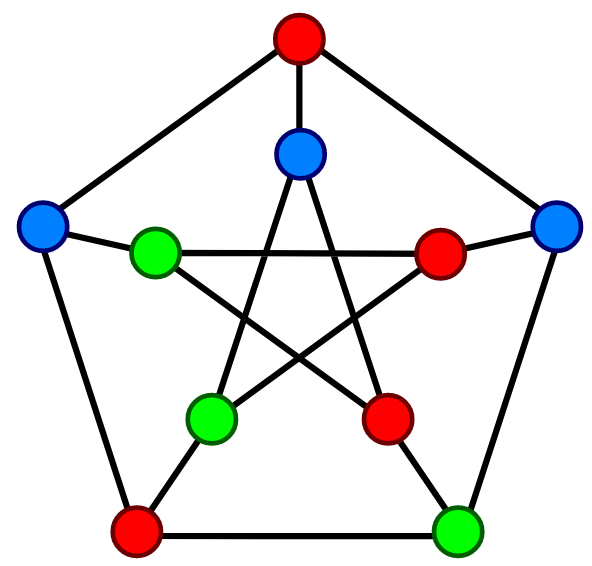


Рисунок 1 — Пример раскраски графа

# Описание алгоритма программы

Для реализации программы был применен данный алгоритм:

1. Упорядочить вершины по невозрастанию степени.

2. Окрасить первую вершину в цвет 1.

3. Выбрать цвет окраски 1.

4. Пока не окрашены все вершины, повторять п.4.1.-4.2.:

4.1. Окрасить в выбранный цвет всякую вершину, которая не смежна

с другой, уже окрашенной в этот цвет.

4.2. Выбрать следующий цвет.

Это последовательный метод, основанный на упорядочивании множества вершин.

В этом алгоритме вершины вначале располагаются в порядке невозрастания их степеней. Для этого используется словарь razr (помещаются вершина-степень) и очередь vertex (отсортированная по невозрастанию степеней вершин очередь). Первая вершина в очереди окрашивается в цвет 1; затем матрица по вершинам просматривается и в цвет 1 окрашивается всякая вершина, которая не смежна с другой, уже окрашенной в этот цвет. Потом возвращаемся к первой в очереди неокрашенной вершине, окрашиваем ее в цвет 2 и снова просматриваем матрицу по вершинам вниз, окрашивая в цвет 2 любую неокрашенную вершину, которая не соединена ребром с другой, уже окрашенной в цвет 2 вершиной. Аналогично действуем с цветами 3, 4 и т. д., пока не будут окрашены все вершины. Число использованных цветов будет равно хроматическому числу графа.

Ниже представлен псевдокод функции vertex\_queue() и Colour().

Алгорит vertex\_queue

Вход: size - размер матрицы, M[i, j] – матрица смежности

Выход: очередь vertex с отсортированными вершинами

1. создаем очередь Queue<int> vertex = new Queue<int>()
2. создаем словарь Dictionary<int, int> razr = new Dictionary<int, int>(size)
3. sr = 0;
4. для i = 0; пока i < size; делать i=i+1
   1. для j = 0; пока j < size; делать j=j+1
      1. если M[i, j] == 1
         1. sr=sr+1;
   2. Добавить в словарь razr (i, sr)
   3. sr = 0
5. переводим словарь в список
6. сортируем список по невозрастанию степеней вершин
7. для i = 0; пока i < size; делать i=i+1
   1. добавляем в очередь vertex вершину
8. вызываем функцию Colour

Алгоритм Colour

Вход: M[i, j] – матрица смежности, size - размер матрицы, vertex- отсортированная очередь.

Выход: вектор exit с цветами.

1. создаем вектор int[] exit = new int[size];
2. берем первую вершину из очереди и удаляем ее(w = vertex.Dequeue())
3. задаем первый цвет color = 1;
4. окрашиваем первую вершину в цвет exit[w] = color;
5. e = 0;
6. пока количество элементов в очереди не равно нулю
   1. для i = 0; пока i < size; делать i=i+1
      1. для j = 0; пока j < size; делать j=j+1
         1. если M[w, j] == 0 и exit[j] == 0
            1. присваеваем вершине цвет exit[j] = color;
            2. e = exit[j];
   2. берем первую вершину из очереди и удаляем ее (w = vertex.Dequeue())
   3. изменяем цвет color++
7. выводим результат на экран
8. вызываем функцию chrom

Полный код программы можно увидеть в Приложении А.

# Описание программы

Для написания данной программы использован язык программирования С#. C# - это универсальный язык программирования, задуманный для того, чтобы сделать программирование более приятным для серьезного програм­миста.

Язык C# довольно простой, он типобезопасен и объектно-ориентирован. Благодаря огромному количеству внедрений, C# обеспечивает возможность разработки приложений с минимальной затратой времени, но при этом, он сохраняет выразительность и элегантность, которая присуща языкам C.

Проект был создан в виде приложения WindowsForm (Visual C#).

Работа программы начинается с открытия формы 1 (рисунок 2).

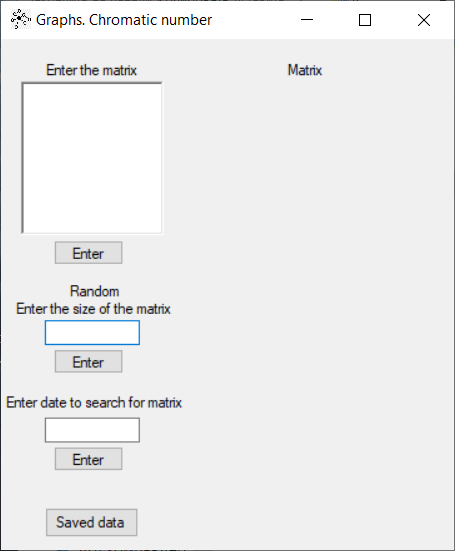


Рисунок 2 — Форма 1

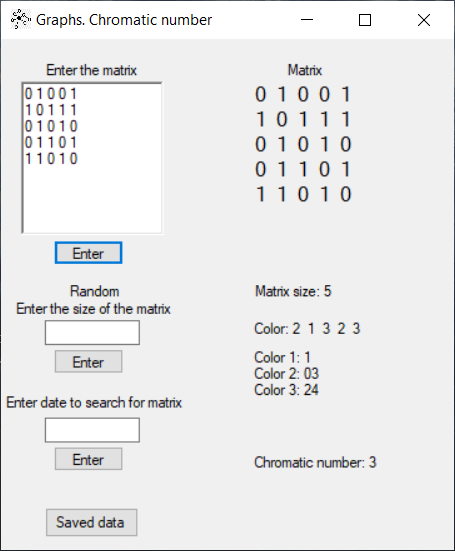
Пользователь может сам ввести матрицу (через пробел), сгенерировать рандомно или вывести сохраненную. После ввода матрицы нужно нажать на кнопку “Enter”. Выведется сгенерированная матрица, ее размер, цвета по порядку вершин и хроматическое число. Пример ручного ввода матрицы на Рисунке 3.

Рисунок 3— Пример ручного ввода матрицы

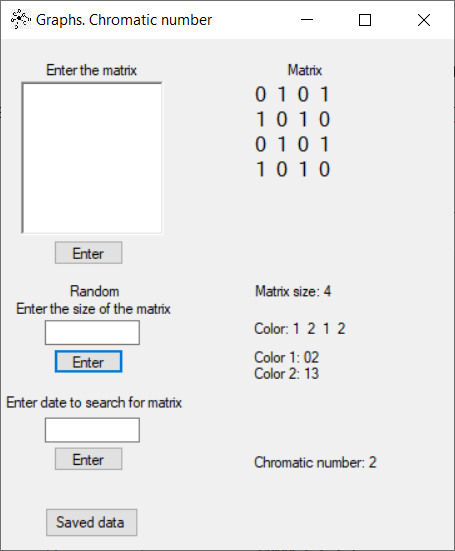
Если пользователь хочет сгенерировать матрицу рандомно, то нужно ввести ее размерность в окошко и нажать “Enter”. Матрица будет сгенерирована и выведена справа. Пример рандомного ввода приведен на Рисунке 4.

Рисунок 4 — Пример рандомного ввода матрицы

В программе имеется функция автоматического сохранения данных. Сохраненные данные можно просмотреть, нажав на кнопку “Saved date”.

При нажатии на эту кнопку открывается новое окно с сохранениями.

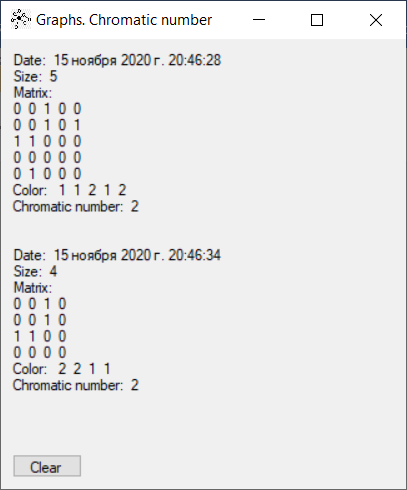
Сохраняются дата, размер матрицы, матрица, цвета и хроматическое число (Рисунок 5).

Рисунок 5 – Пример сохранения

Сохраненные данные можно удалить, нажав на кнопку “Clear”. При следующем открытии этого окна, оно будет пустым.

Программа имеет функцию ввода сохраненной матрицы, для этого надо скопировать дату нужной матрицы и вставать в окошко “Enter date to search for matrix” и нажать кнопку “Enter”. Выбранная матрица появится в окошке для ввода. (Рисунок 6,7).

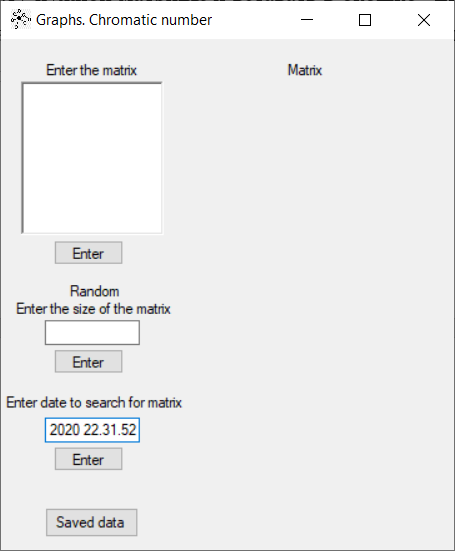
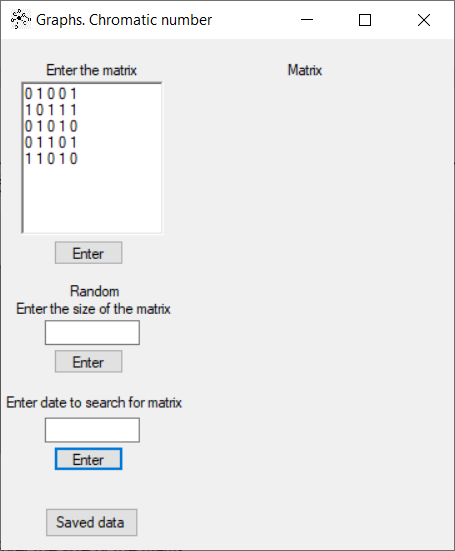


Рисунок 7 – Вывод сохраненной матрицы в окошко для ввода

Рисунок 6 – Ввод даты для вывода нужной матрицы

# Тестирование

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| **Описание теста** | **Ожидаемый результат** | **Полученный результат** |
| Запуск программы | Открытие формы | Верно |
| Ручной ввод матрицы | Вывод матрицы и последующие действия над ней | Верно |
| Рандомный ввод матрицы | Вывод рандомной матрицы и последующие действия над ними | Верно |
| Ввод сохраненной матрицы | Вывод сохраненной ранее матрицы в окошко для ввода | Верно |
| Результат работы программы | Верный вывод хроматического числа графа | Верно |
| Сохранение данных | Сохранение данных и вывод данных, открытие нового окна с сохранениями, очистка окна с сохранениями | Верно |
| Ручной расчет | Соответствие результату работы алгоритма | Верно |

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, работой алгоритма, выводом данных, взаимодействием функций.

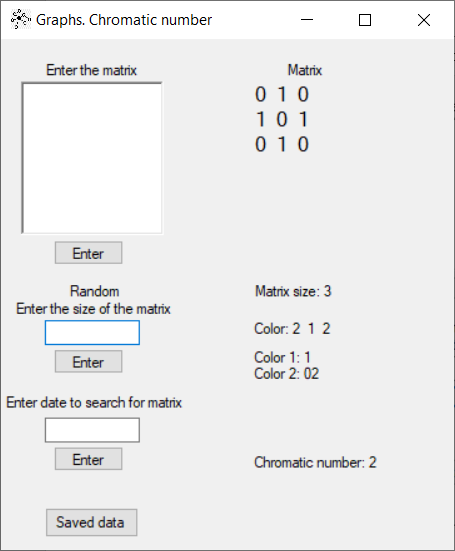
Ниже продемонстрирован результат тестирования программы при рандомном вводе пользовтелем матриц различной размерностью.

Рисунок 8 – Тестирование при размерности матрицы 3

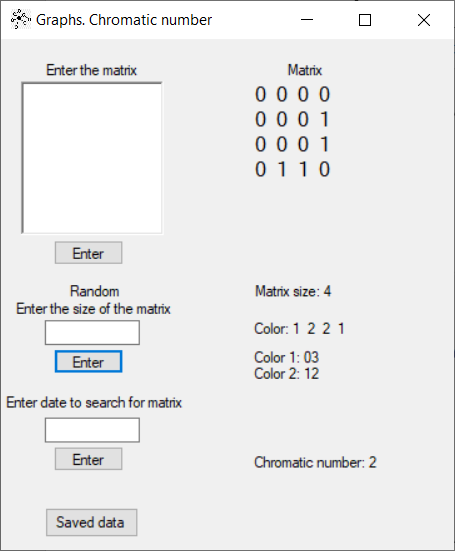
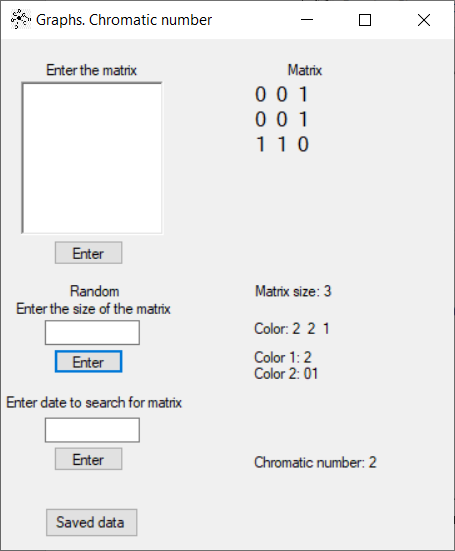


Рисунок 9 – Тестирование при размерности матрицы 3

Рисунок 10 – Тестирование при размерности матрицы 4

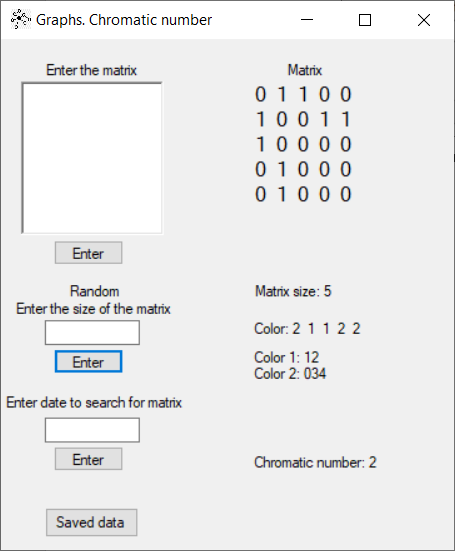
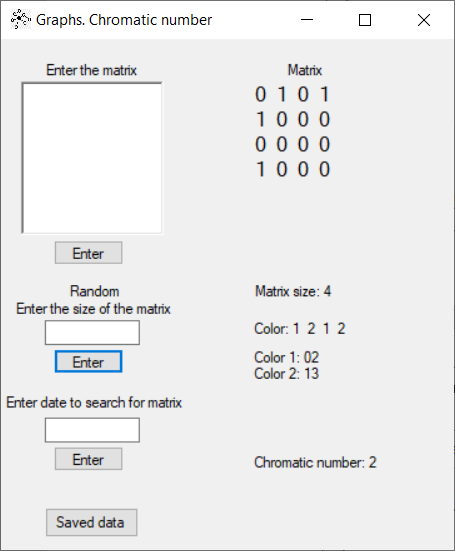


Рисунок 11 – Тестирование при размерности матрицы 4

Рисунок 12 – Тестирование при размерности матрицы 5

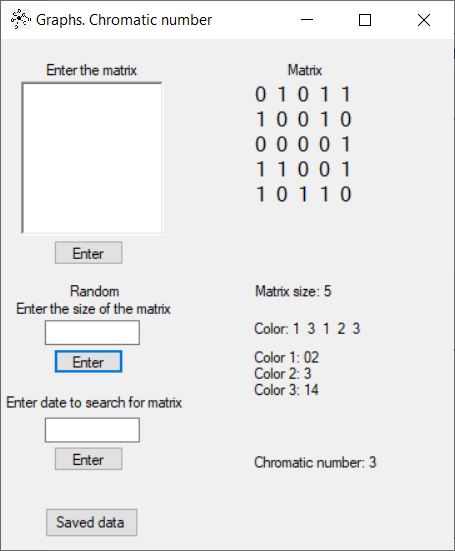


Рисунок 13 – Тестирование при размерности матрицы 5

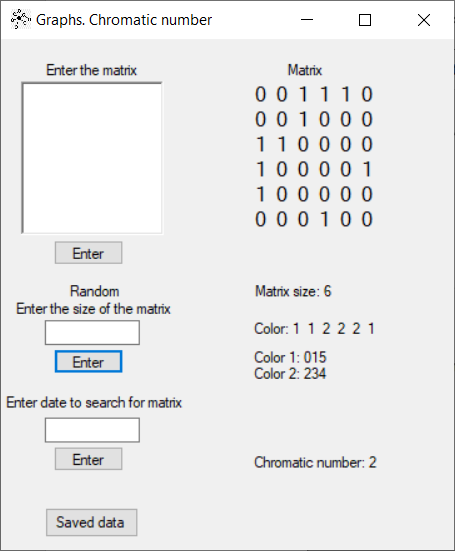


Рисунок 14 – Тестирование при размерности матрицы 6

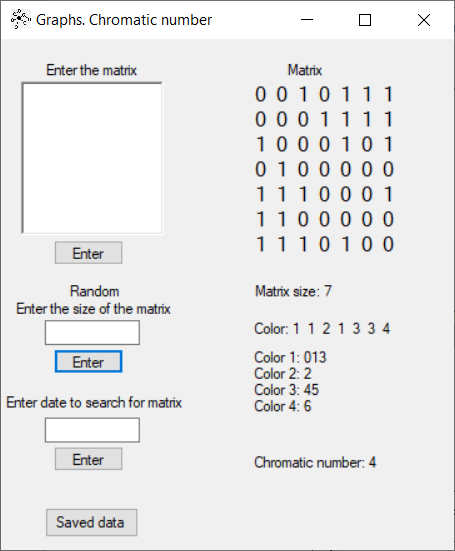
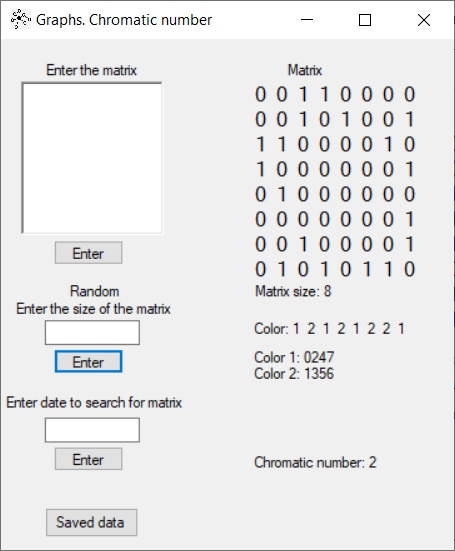


Рисунок 16 – Тестирование при размерности матрицы 8

Рисунок 15 – Тестирование при размерности матрицы 7

В результате тестирования было выявлено, что программа успешно обрабатывает данные и выдает верный результат.

**Ручной расчёт задачи**

Проведем проверку программы посредством ручных вычислений на примере графа с 5 вершинами (рисунок 14).

Матрица смежности: 0 1 0 0 0

1 0 1 1 1

0 1 0 1 1

0 1 1 0 0

0 1 1 0 0

Визуализация графа:

1

2

4

3

0

Фигура 1 – Визуализация графа

Сначала создается словарь (вершина, степень). Запись начинается с нулевой вершины. Проходим все вершины и получаем {[0,1][1,4][2,3][3,2][4,2]}.

Далее программа сортирует вершины по невозрастанию степеней и заносит номера вершин в очередь.

Очередь выглядит следующим образом: {1,2,3,4,0}

Далее переходим к алгоритму раскраски. Начинаем с цвета 1. Первая вершина очереди 1, ее мы сразу окрашиваем в цвет 1.

Т.к у вершины 1 больше нет не смежных ей вершин, то меняем цвет на 2. Раскраска графа:{0,1,0,0,0}

2

4

1

3

0

Фигура 2 – Визуализация раскраски графа

Вторая вершина в очереди – 2. У второй вершины имеется две не смежных ей и не посещенных, это вершины 0 и 2. Красим их цветом 2 и меняем цвет на 3. Раскраска графа:{2,1,2,0,0}.

2

4

1

3

0

Фигура 3 – Визуализация раскраски графа

Следующая вершина в очереди – 3. У нее не смежных ей вершин три, это 0, 3 и 4. Все эти три вершины не посещены, поэтому окрашиваем их в цвет 3. Раскраска графа: {2,1,2,3,3}.

2

4

1

3

0

Фигура 4 – Визуализация раскраски графа

Алгоритм раскраски графа закончен. Далее считаем хроматическое число. Проходим по всему вектору раскраски и находит самое большое число. В нашем случае это 3. Хроматическое число графа = 3.

Таким образом, можно сделать вывод, что программа работает верно.

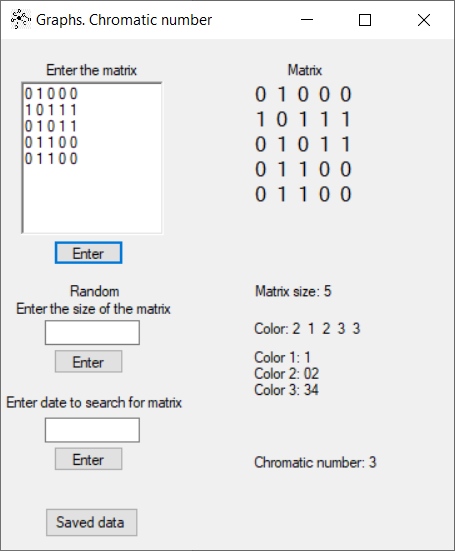


Рисунок 17 – Ручное тестирование программы

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм нахождения хроматического числа графа в Microsoft Visual Studio 2019.

При выполнении данной работы были получены навыки работы с матрицей смежности и был изучен алгоритм раскраски графа. Так же были улучшены навыки разработки графического приложения, создании интерфейса с помощью Windows Form. Углублены знания языка программирования C#.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

# Список литературы

* 1. А.М. Райгородский. Хроматические числа. 2003. 44 с.
  2. Н.А. Осипов. Разработка Windows приложений на C#. Учебное пособие. 2012. 74 с.
  3. О.Г. Савихин. Основы разработки приложений в Microsoft Visual Studio.NET. 2007. 203 с.
  4. С. Э. Мурадханов. Разработка на языке C# приложений с графическим интерфейсом. 2019. 396 с.

## Приложение А.

**Листинг программы.**

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace CrsGr

{

public partial class Form1 : Form

{

int size;

public Form1()

{

InitializeComponent();

}

public void Button1\_Click(object sender, EventArgs e)

{

label1.Text = "";

size = int.Parse(textBox1.Text);

label1.Text = "Matrix size: " + size;

Random random = new Random();

int[,] M = new int[size, size];

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

{

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

{

M[i, j] = random.Next(2);

M[j, i] = M[i, j];

}

}

Inic(size, M);

}

public void Inic(int size, int[,]M)

{

label2.Text = "";

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

{

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

{

label2.Text += M[i, j];

label2.Text += " ";

}

label2.Text += "\n";

}

vertex\_queue(size, M);

}

public void vertex\_queue(int size, int[,] M)

{

Queue<int> vertex = new Queue<int>();

Dictionary<int, int> razr = new Dictionary<int, int>(size);

int sr = 0;

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

{

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

{

if (M[i, j] == 1)

{

sr++;

}

}

razr.Add(i, sr);

sr = 0;

}

var list = razr.ToList();

list.Sort((p1, p2) => p2.Value.CompareTo(p1.Value));

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

{

vertex.Enqueue(list.ElementAt(i).Key);

}

Colour(M, size, vertex);

}

public void Colour(int [,]M,int size, Queue<int> vertex)

{

int[] exit = new int[size];

int w = vertex.Dequeue();

int color = 1;

exit[w] = color;

int e = 0;

while (vertex.Count != 0)

{

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

{

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

{

if (M[w, j] == 0 && exit[j] == 0)

{

exit[j] = color;

e = exit[j];

}

}

}

w = vertex.Dequeue();

color++;

}

label3.Text = "";

label3.Text += "Color: ";

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

{

label3.Text +=+exit[i];

label3.Text += " ";

}

chrom\_(exit,M);

}

public void Button2\_Click(object sender, EventArgs e)

{

label1.Text = "";

int m = richTextBox1.Lines.Length;

int n = richTextBox1.Lines[0].Split(' ').Length;

int[,] M = new int[m, n];

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

{

string[] s = richTextBox1.Lines[i].Split(' ');

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

{

M[i, j] = Int32.Parse(s[j]);

}

}

size = n;

Inic(size, M);

label1.Text = "Matrix size: " + n;

}

private void Button3\_Click(object sender, EventArgs e)

{

Form2 form2 = new Form2();

string records = File.ReadAllText("D:/file.txt");

form2.setTextToRich(records);

form2.Show();

}

public void saveRecords(int[,] M, int[] exit, int chrom)

{

DateTime now = DateTime.Now;

StreamWriter file = new StreamWriter("D:/file.txt", true);

try

{

file.Write(("Date: " + now.ToString("dd MMMM yyyy HH.mm.ss")) + "\n");

file.Write("Size: " + size + " \n");

file.Write("Matrix: \n");

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

{

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

{

file.Write(M[i, j] + " ");

}

file.Write("\n");

}

file.Write("Color: ");

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

{

file.Write(exit[i] + " ");

}

file.Write("\n");

file.Write("Сhromatic number: " + chrom);

file.Write("\n\n\n");

}

finally

{

file.Flush();

file.Close();

}

string s = DateTime.Now.ToString("dd MMMM yyyy HH.mm.ss");

StreamWriter save = new StreamWriter(s+".txt", true);

try

{

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

{

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

{

save.Write(M[i, j] );

if (j == size - 1) {

continue;

}

else

{

save.Write(" ");

}

}

if (i == size - 1)

{

continue;

}

else

{

save.Write("\n");

}

}

}

finally

{

save.Flush();

save.Close();

}

}

public void chrom\_(int[]exit,int [,]M) {

int chrom = 0;

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

{

if (exit[i] > chrom) {

chrom = exit[i];

}

}

label8.Text = "";

label8.Text = "Сhromatic number: " + chrom;

label9.Text = "";

int cl = 0;

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

{

cl++;

label9.Text += "Color "+cl+": ";

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

{

if (exit[j] == cl)

{

label9.Text += j ;

}

}

label9.Text += "\n";

}

saveRecords(M, exit, chrom);

}

private void Button4\_Click\_1(object sender, EventArgs e)

{

string name = textBox2.Text+".txt";

if (!(File.Exists(name)))

{

MessageBox.Show("Input Error.\nTry again");

}

else

{

string soxr = File.ReadAllText(name);

richTextBox1.Text = soxr;

}

textBox2.Clear();

}

}

}

Form2.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace CrsGr

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

public void setTextToRich(String text)

{

richTextBox1.Text = text;

}

private void Button1\_Click(object sender, EventArgs e)

{

File.WriteAllText("D:/file.txt", string.Empty);

DirectoryInfo Dr = new DirectoryInfo(@"C:\Users\Лиза\source\repos\CrsGr\bin\Debug");

FileInfo[] files = Dr.GetFiles("\*.txt").Where(p => p.Extension == ".txt").ToArray();

foreach (FileInfo file in files)

try

{

file.Attributes = FileAttributes.Normal;

File.Delete(file.FullName);

}

catch

{

}

}

}

}