Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 13

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Классы и объекты»

Выполнил: ст. гр. ТУУ-111

Волобуев Е.Д.

Вариант №13

03.06.2024

Проверил: к.т.н., доц. Сафронов А.И.

Москва – 2024 г.

1. **Цель работы**

В среде *Microsoft* *Visual Studio* на языке *Visual C#* в консольном режиме составить программное обеспечение для решения типовых задач программирования по тематике «Классы и объекты» («*Classes and Objects*»), заданных по варианту.

1. **Формулировка задачи**

**Вариант №13**  
Создайте класс «Текст» (возможное имя «clsText»).

**Поля (fields):**

* идентификатор (int);
* содержимое (string);
* семейство шрифта (enum);
* размер шрифта (int);
* цвет шрифта (enum);
* регистр шрифта (sbyte: -1 – нижний, 1 – верхний, 0 – нормальный);
* признак всех заглавных букв (bool).

**Конструктор (instance constructor):**

* пустой конструктор для инициализации экземпляра ненулевыми значениями по умолчанию.

**Методы (methods):**

* полная реинициализация экземпляра по указанным входным параметрам (void);
* реинициализация экземпляра по семейству и размеру шрифта (void);
* реинициализация цвета (void);
* изменение размера шрифта по указанному значению шага (int).

**Операторы (operators):**

* переключение признака всех заглавных букв (!);
* увеличение шрифта на шаг (++);
* уменьшение шрифта на шаг (--).  
  Шаги изменения размеров шрифта выполнить по аналогии с тем, как реализовано в офисных приложениях (Word, Excel, PowerPoint)

**Хранение элементов:**

* предусмотреть возможность чтения и записи текстовых надписей и их параметров при использовании текстового файла.

**Сопоставление (Equals()):**  
Тексты считать равными, если все их параметры, кроме идентификатора и содержимого равны.

**Вывод (ToString()):**  
«1.«Новый Текст»[Times New Roman][14][Красный][нормальный]»  
«2.«ДРУГОЙ ТЕКСТ»[COMIC SANS MS][10][СИНИЙ][НИЖНИЙ]»

**Хэширование (GetHashCode()):**  
Сумма порядковых номеров буквенных символов в содержимом текста. Латинские символы участвуют в расчёте хэш-функции с отрицательным знаком, кириллические – с положительным. Сначала считаются строчные символы в последовательности, затем заглавные, то есть необходим учёт 52 позиций для латинских символов и 66 позиций для кириллических символов. При этом признак всех заглавных букв влияет только на оформление надписи и не влияет на порядковый номер исходного символа, который может быть строчным.

1. **Блок-схема алгоритма**

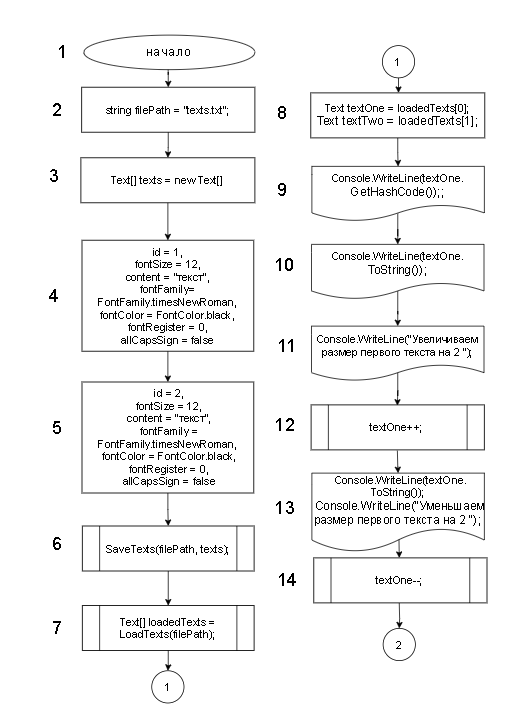
****

Рисунок 1 – Метод Main класса Program (часть 1)

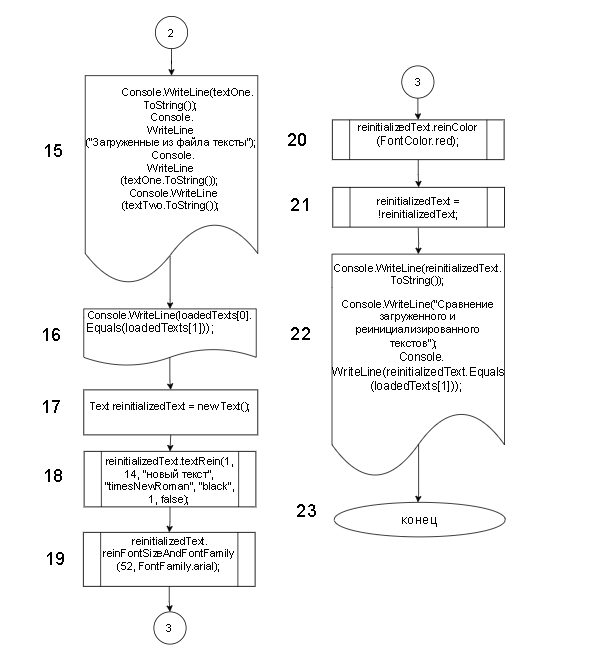


Рисунок 2 – Метод Main класса программ (часть 2)

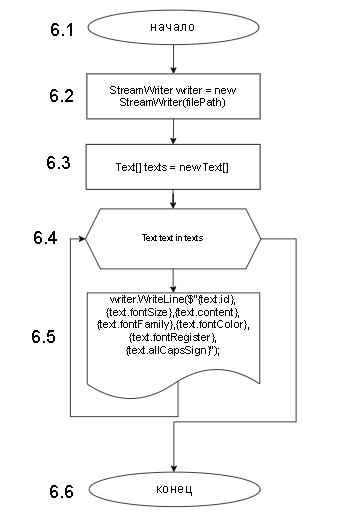


Рисунок 3 – Метод SaveTexts класса Program

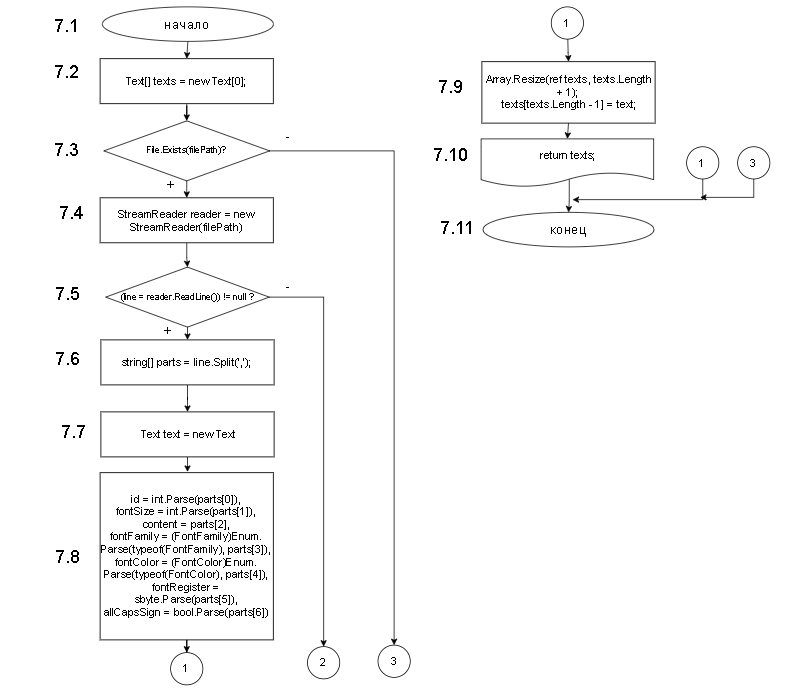


Рисунок 4 – Метод loadTexts класса Program

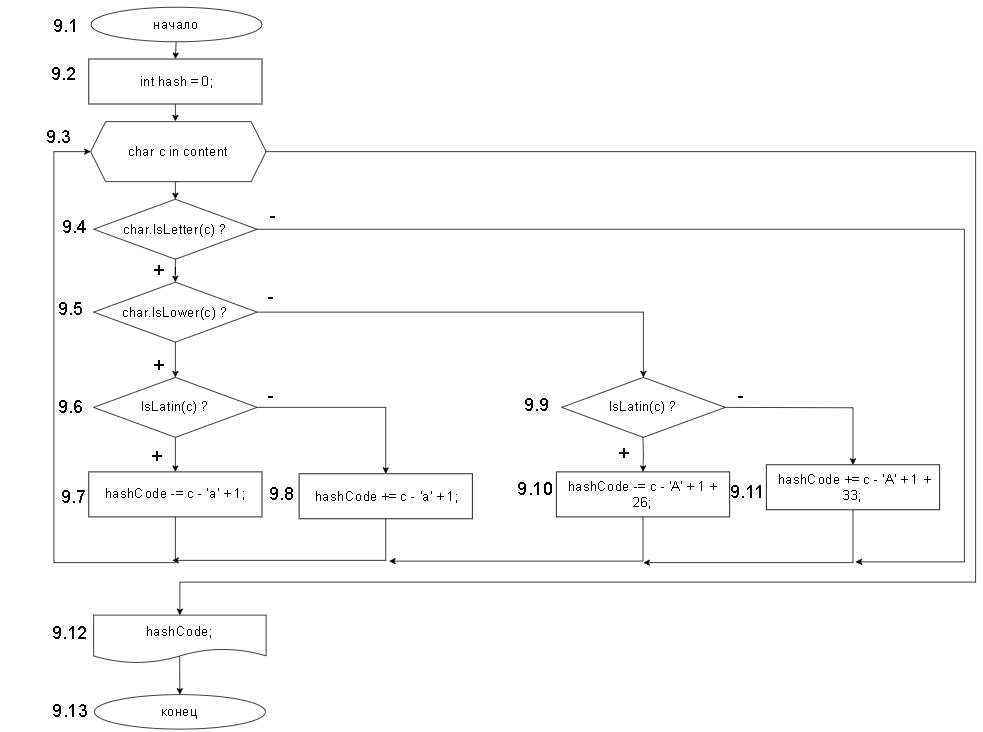


Рисунок 5 – Метод GetHashCode класса Text

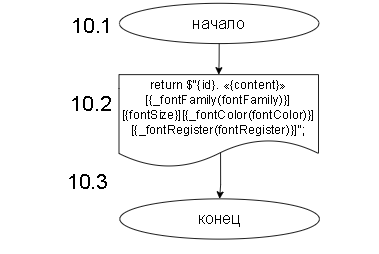


Рисунок 6 – Метод ToString класса Text

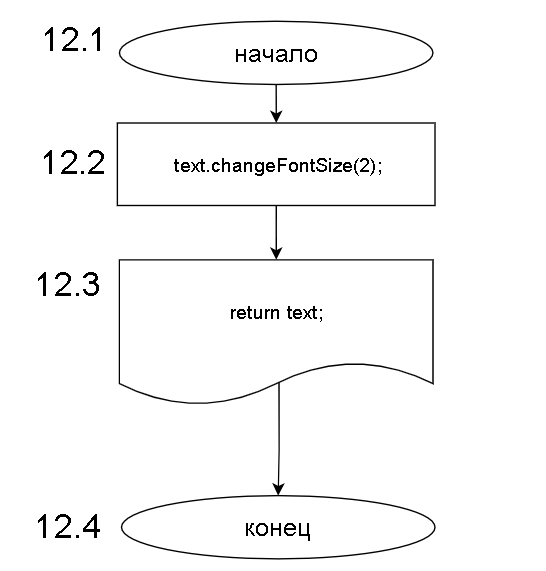


Рисунок 7 – Оператор увеличения шрифта на шаг класса Text

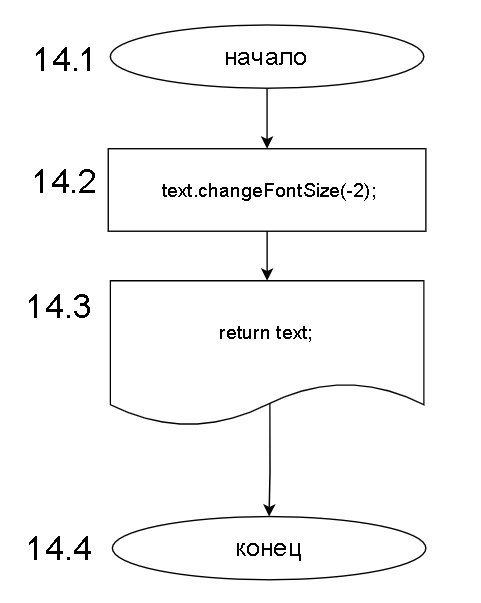


Рисунок 8 – Оператор уменьшения шрифта на шаг класса Text

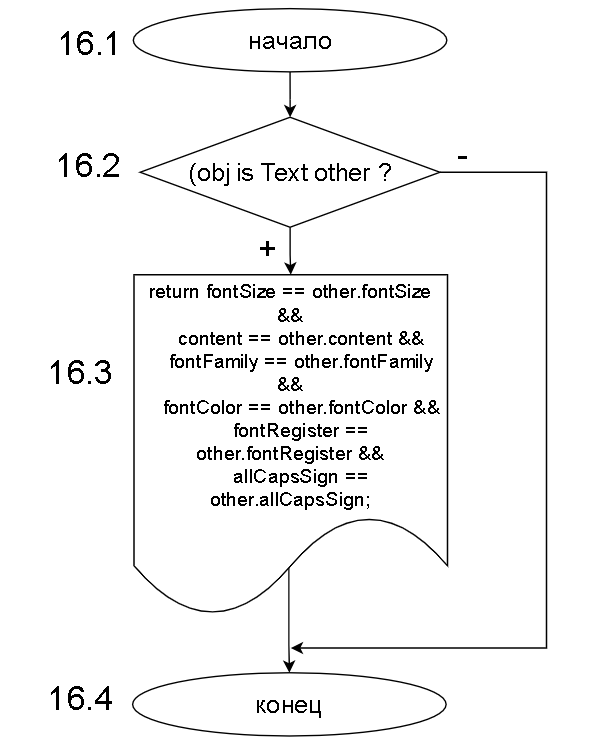


Рисунок 9 – Метод Equals класса Text

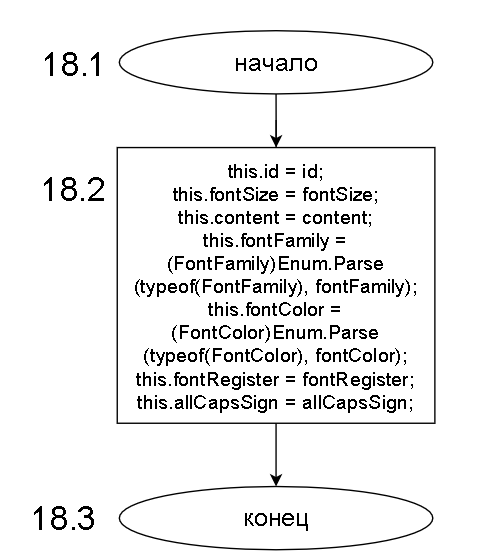


Рисунок 10 – Метод textRein класса Text

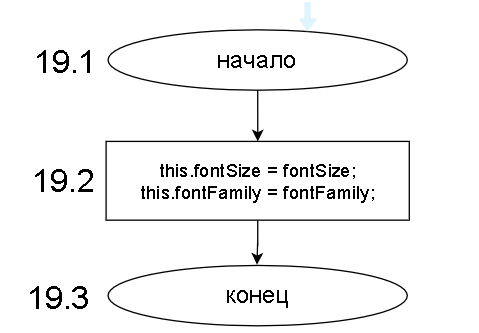


Рисунок 11 – Метод reinFontSizeAndFontFamily класса Text

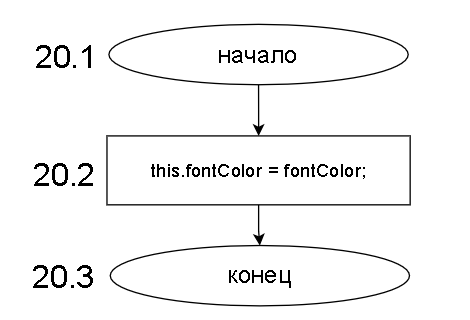


Рисунок 12 – Метод reinColor класса Text

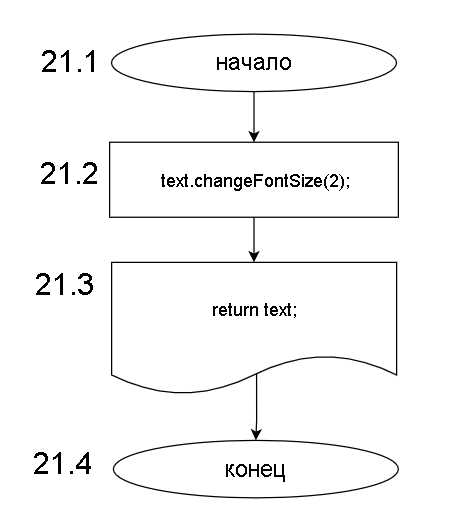


Рисунок 13 – Оператор переключения признака всех заглавных букв класса Text

1. **Подбор тестовых примеров**

Создаём два экземпляра класса со следующими данными:

1) id = 1

fontSize = 12

content = "текст"

fontFamily = FontFamily.timesNewRoman

fontColor = FontColor.black

fontRegister = 0

allCapsSign = false

2) id = 2

fontSize = 12

content = "текст"

fontFamily = FontFamily.timesNewRoman

fontColor = FontColor.black

fontRegister = 0

allCapsSign = false

* 1. **Метод SaveTexts**

Сериализация двух экземпляров класса текст, чтобы предусмотреть возможность записи текстовых надписей и их параметров при использовании текстового файла

* 1. **Метод LoadTexts**

Десериализация двух экземпляров класса текст, чтобы предусмотреть возможность чтения текстовых надписей и их параметров при использовании текстового файла

* 1. **Метод GetHashCode**

При заданных параметрах хеширования и входных данных, мы должны получить число 73

* 1. **Метод toString**

С помощью данного метода выводим получившиеся строки

1. <текст> [Times New Roman][12][Черный][нормальный]

2. <текст> [Times New Roman][12][Черный][нормальный]

* 1. **Оператор ++**

С помощью данного оператора у нас увеличивается размер шрифта:

1. <текст> [Times New Roman][14][Черный][нормальный]

* 1. **Оператор –**

С помощью данного оператора у нас уменьшается размер шрифта:

1. <текст> [Times New Roman][12][Черный][нормальный]

* 1. **Методы textRein, reinFontSizeAndFontFamily, reinColor**

Реиницилизация экземпляра по входным параметрам, по семейству и размеру шрифта, цвету соответственно:

1. <новый текст> [Arial][52][Красный][верхний]

* 1. **Метод Equals**

Сравнение двух начальных экземпляров класса и начального c реиницилихированным:

1. <текст> [Times New Roman][12][Черный][нормальный]

2. <текст> [Times New Roman][12][Черный][нормальный]

True

1. <новый текст> [Arial][52][Красный][верхний]

False

1. **Листинг**

using System;

using System.IO;

using static rabota\_13.Text;

using static System.Net.Mime.MediaTypeNames;

namespace rabota\_13

{

internal class Program

{

static void Main(string[] args)

{

string filePath = "texts.txt";

Text[] texts = new Text[]

{

new Text

{

id = 1,

fontSize = 12,

content = "текст",

fontFamily = FontFamily.timesNewRoman,

fontColor = FontColor.black,

fontRegister = 0,

allCapsSign = false

},

new Text

{

id = 2,

fontSize = 12,

content = "текст",

fontFamily = FontFamily.timesNewRoman,

fontColor = FontColor.black,

fontRegister = 0,

allCapsSign = false

}

};

// сохранение текстов в файл

SaveTexts(filePath, texts);

// загрузка текстов из файла

Text[] loadedTexts = LoadTexts(filePath);

Text textOne = loadedTexts[0];

Text textTwo = loadedTexts[1];

Console.WriteLine(textOne.GetHashCode());

Console.WriteLine(textOne.ToString());

Console.WriteLine("Увеличиваем размер первого текста на 2 ");

textOne++;

Console.WriteLine(textOne.ToString());

Console.WriteLine("Уменьшаем размер первого текста на 2 ");

textOne--;

Console.WriteLine(textOne.ToString());

Console.WriteLine("Загруженные из файла тексты");

Console.WriteLine(textOne.ToString());

Console.WriteLine(textTwo.ToString());

Console.WriteLine(loadedTexts[0].Equals(loadedTexts[1]));

Console.WriteLine("Реинициализируем текст");

Text reinitializedText = new Text();

reinitializedText.textRein(1, 14, "новый текст", "timesNewRoman", "black", 1, false);

reinitializedText.reinFontSizeAndFontFamily(52, FontFamily.arial);

reinitializedText.reinColor(FontColor.red);

reinitializedText = !reinitializedText;

Console.WriteLine(reinitializedText.ToString());

Console.WriteLine("Cравнение загруженного и реинициализированного текстов");

Console.WriteLine(reinitializedText.Equals(loadedTexts[1]));

Console.ReadKey();

}

static void SaveTexts(string filePath, Text[] texts)

{

using (StreamWriter writer = new StreamWriter(filePath))

{

foreach (Text text in texts)

{

writer.WriteLine($"{text.id},{text.fontSize},{text.content},{text.fontFamily},{text.fontColor},{text.fontRegister},{text.allCapsSign}");

}

}

}

static Text[] LoadTexts(string filePath)

{

Text[] texts = new Text[0];

if (File.Exists(filePath))

{

using (StreamReader reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

string[] parts = line.Split(',');

Text text = new Text

{

id = int.Parse(parts[0]),

fontSize = int.Parse(parts[1]),

content = parts[2],

fontFamily = (FontFamily)Enum.Parse(typeof(FontFamily), parts[3]),

fontColor = (FontColor)Enum.Parse(typeof(FontColor), parts[4]),

fontRegister = sbyte.Parse(parts[5]),

allCapsSign = bool.Parse(parts[6])

};

Array.Resize(ref texts, texts.Length + 1);

texts[texts.Length - 1] = text;

}

}

}

return texts;

}

}

public class Text

{

public enum FontFamily

{

timesNewRoman,

arial,

comicSans

}

public enum FontColor

{

black,

white,

red

}

public int id, fontSize;

public string content;

public FontFamily fontFamily;

public FontColor fontColor;

public sbyte fontRegister;

public bool allCapsSign;

public Text()

{

id = 0;

fontSize = 14;

content = "";

fontFamily = FontFamily.arial;

fontColor = FontColor.black;

fontRegister = 0;

allCapsSign = false;

}

public void reinFontSizeAndFontFamily(int fontSize, FontFamily fontFamily)

{

this.fontSize = fontSize;

this.fontFamily = fontFamily;

}

public void reinColor(FontColor fontColor)

{

this.fontColor = fontColor;

}

public int changeFontSize(int step)

{

fontSize += step;

return fontSize;

}

public static Text operator !(Text text)

{

text.allCapsSign = !text.allCapsSign;

return text;

}

public static Text operator ++(Text text)

{

text.changeFontSize(2);

return text;

}

public static Text operator --(Text text)

{

text.changeFontSize(-2);

return text;

}

public void textRein(int id, int fontSize, string content, string fontFamily, string fontColor, sbyte fontRegister, bool allCapsSign)

{

this.id = id;

this.fontSize = fontSize;

this.content = content;

this.fontFamily = (FontFamily)Enum.Parse(typeof(FontFamily), fontFamily);

this.fontColor = (FontColor)Enum.Parse(typeof(FontColor), fontColor);

this.fontRegister = fontRegister;

this.allCapsSign = allCapsSign;

}

public override bool Equals(object obj)

{

if (obj is Text other)

{

return fontSize == other.fontSize &&

content == other.content &&

fontFamily == other.fontFamily &&

fontColor == other.fontColor &&

fontRegister == other.fontRegister &&

allCapsSign == other.allCapsSign;

}

return false;

}

public string \_fontFamily(Enum \_enum)

{

switch (\_enum)

{

case FontFamily.timesNewRoman:

return "Times New Roman";

break;

case FontFamily.arial:

return "Arial";

break;

case FontFamily.comicSans:

return "Comic Saans";

break;

default:

break;

}

return null;

}

public string \_fontColor(Enum \_enum)

{

switch (\_enum)

{

case FontColor.red:

return "Красный";

break;

case FontColor.white:

return "Белый";

break;

case FontColor.black:

return "Черный";

break;

default:

break;

}

return null;

}

public string \_fontRegister(sbyte fontRegister)

{

switch (fontRegister)

{

case -1:

return "нижний";

break;

case 1:

return "верхний";

break;

default:

return "нормальный";

break;

}

}

public override int GetHashCode()

{

int hashCode = 0;

foreach (char c in content)

{

if (char.IsLetter(c))

{

if (char.IsLower(c))

{

if (IsLatin(c))

{

hashCode -= c - 'a' + 1;

}

else

{

hashCode += c - 'а' + 1;

}

}

else

{

if (IsLatin(c))

{

hashCode -= c - 'A' + 1 ;

}

else

{

hashCode += c - 'А' + 1 ;

}

}

}

}

return hashCode;

}

public bool IsLatin(char c)

{

return (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');

}

public override string ToString()

{

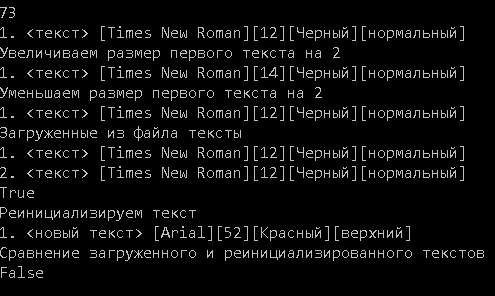
return $"{id}. «{content}» [{\_fontFamily(fontFamily)}][{fontSize}][{\_fontColor(fontColor)}][{\_fontRegister(fontRegister)}]";

}

}

}

1. **Расчет тестовых примеров на ПК**



1. **Вывод по работе**

Научился работать с классами и объектами, а также создавать методы классов, взаимодействовать с объектами класса, использовать метод GetHashCode.