МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ
Кировское областное государственное профессиональное образовательное бюджетное учреждение
«Слободской колледж педагогики и социальных отношений»

КУРСОВОЙ ПРОЕКТ

по ПМ 01 «Разработка программных модулей» на тему:

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ УЧЕТА ПРОПУСКОВ УЧЕБНЫХ ЗАНЯТИЙ

Выполнила: Кротова Ксения Николаевна

Специальность 09.02.07 Информационные системы и программирование

Группа 21П-1
Форма обучения: очная

Руководитель: Пентин Николай Сергеевич

Дата защиты курсовой работы:
Председатель ПЦК:
Оценка за защиту курсовой работы:

Слободской
2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном образовательном процессе одной из ключевых задач является обеспечение академической успеваемости студентов, а также эффективный контроль за их посещаемостью. Учебные заведения, будь то школы, колледжи или университеты, сталкиваются с необходимостью систематически фиксировать и отслеживать посещаемость занятий. Проблема учета посещаемости учебных занятий актуальна как для студентов, так и для педагогов и администрации учебных заведений, поскольку стабильное посещение занятий влияет на качество образовательного процесса и общую успеваемость учащихся.

Важность учета посещаемости студентов в учебном процессе не может быть недооценена. Пропуски занятий напрямую влияют на уровень знаний студентов и их успеваемость, а также могут служить индикатором ряда проблем, включая отсутствие мотивации, личные трудности или даже проблемы со здоровьем. Получая своевременную информацию о пропусках, преподаватели и администрация могут принимать меры, направленные на улучшение ситуации.

Исторически ведение учета посещаемости занимало значительное количество ресурсов, включая как человеческие, так и временные. Традиционные методы, такие как журнал или тетрадь для записи посещаемости, имеют свои ограничения: они подвержены ошибкам, могут быть неправильно заполнены и не обеспечивают необходимого уровня удобства и эффективности. Век цифровых технологий диктует новые требования к процессам, связанным с администрированием образовательных учреждений. На сегодняшний день многие учебные заведения переходят на автоматизированные системы учета, что позволяет значительно упростить и улучшить ведение документации.

Таким образом, программный модуль для учета посещаемости учебных занятий может стать инструментом, который поможет в реализации проактивного подхода к поддержке студентов, позволяя выявлять значимые закономерности и тенденции.

Цель курсового проекта – разработка программного обеспечения для учета посещаемости учебных занятий студентами.

Задачи исследования:

- Описать предметную область.
- Разработать технического задание на создание программного продукта.
- Описать архитектуру программы.
- Описать алгоритмы и функционирование программы.
- Провести тестирование и опытную эксплуатацию.
- Разработать руководство оператора

Объект исследования – процесс учета и мониторинга посещаемости студентов в образовательной среде.

Предмет исследования – разработка программной системы для учета и мониторинга посещаемости учащихся.

Методы исследования: системный анализ и функциональное моделирование.

Информационную систему исследования составили официальные нормативно-правовые источники, данные об использовании современных информационных систем. Структура работы состоит из введения, трех глав, заключения, списка используемой литературы и приложений.

АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В процессе анализа предметной области было проведено исследование, в ходе которого была разработана диаграмма вариантов использования, отражающая пользователей и выполняемые ими функции (рисунок 1).

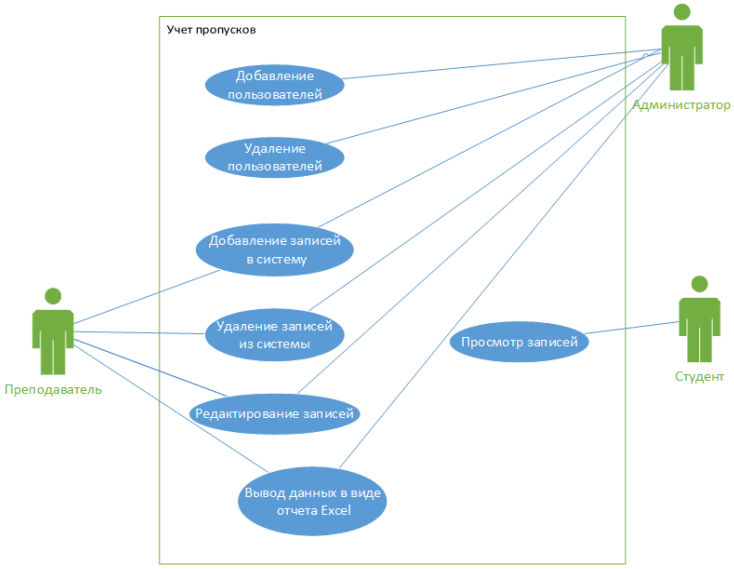

Рисунок 1 - Диаграмма вариантов использования

Обзор аналогов

Учебный учет – программа для учета посещаемости и успеваемости в помощь классным руководителям и заведующим отделения. Программа позволит подготовить электронные журналы, сформировать отчеты посещаемости занятий и текущей успеваемости.

Плюсами данного программного продукта является ввод из журнала оценок и пропусков студентов (учеников), составление ведомостей, автоматическое составление отчетов по успеваемости и посещаемости, а также наглядное представление качественной успеваемости по группам (классам) и специальностям в виде графиков. Есть возможность настройки создания шаблонов печатных форм в Word. Минусом данного программного продукта является ее старый интерфейс, а также высокая стоимость – за лицензию для одного компьютера вам придется заплатить около 8000 рублей, а за сетевую лицензию около 13000 рублей.

UUStud - программа предназначена для учета успеваемости студентов в высших учебных заведениях. Плюсами данного программного продукта является простой интерфейс, а также вывод на печать отчета. Минусом данного программного продукта является очень медленная работа.

После изучения аналогов разрабатываемого программного модуля были выделены следующие сущности и их атрибуты:

User_type:

- Id_types (int);
- Type_Name (varchar);
- Description (varchar).

User:

- Id_user (int);
- Login (varchar);
- Password (varchar);
- Id_student (int);
- Type_User (int).

Student:

- Id_student (int);
- Surname (varchar);
- Name (varchar);
- Middle_Name (varchar);
- Group (int).

Group:

- Id_group (int);
- Group_Name (varchar);
- Specialization (int).

Specialization:

- Id_specialization (int);
- Number_Specialization (varchar);
- Specialization_Name (varchar);
- Description (varchar).

Attendance:

- Id_attendence (int);
- Id_student (int);
- Subject (int);
- Data (datetime);
- Reason (varchar);
- Description (varchar).

Subject:

- id_subject (int);
- Name (varchar).

На основании выделенных сущностей была создана модель базы данных в СУБД Microsoft SQL Server 2019 Express (Рисунок 2):

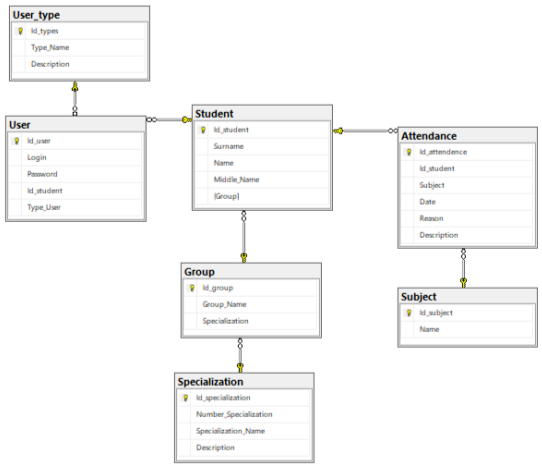

Рисунок 2 – Модель базы данных в системе

РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

Наименование программы – «Учет посещаемости». Программа предназначена для ведения и автоматизации учёта посещаемости учебных занятий студентами [ГОСТ 19.201-78].

Разработка программы ведется на основании учебного плана и перечня тем утвержденных на заседании предметно цикловой комиссии информатики и программирования.

Функциональным назначением программы является ведение и автоматизация учёта посещаемости учебных занятий студентами.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- Просмотр данных о группах, студентах, предметах, специальностях и учете посещаемости.
- Ведение учёта посредством добавления информации о пропусках занятий обучающимися.
- Возможность удаления, редактирования данных.
- Возможность сохранения данных в виде отчета.

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнение заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

- организация бесперебойного питания технических средств;
- использование лицензионного программного обеспечения;
- отсутствие вредоносного программного обеспечения, наличие антивирусной программы;
- соблюдение правил и требований по эксплуатации технических средств.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не

крахом) операционной системы, не должно превышать 5 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу пользователя без предоставления ему административных привилегий.

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий себя:

- процессор с тактовой частотой, 1 ГГц, не менее;
- оперативную память объемом 512 Мб, не менее;
- жесткий диск со свободным местом 500 Мб, не менее;
- монитор, с разрешением экрана 1024*768, не менее;
- оптический привод;
- компьютерная мышь;
- клавиатура;

Исходные коды программы должны быть реализованы на языке C#. В качестве интегрированной среды разработки программы должна быть использована среда программирования Microsoft Visual Studio 2022, в качестве СУБД для разработки базы данных должен быть использован Microsoft SQL Server 2019 Express.

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7/8/10/11.

Программное обеспечение поставляется в виде изделия на CD диске.

Упаковка программного изделия должна осуществляться в упаковочную тару предприятия-изготовителя компакт диска

Требования к транспортировке и хранению должны соответствовать условиям эксплуатации носителей, на которых находится программный продукт.

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса.

Предварительный состав программной документации включает в себя следующие документы:

- техническое задание;
- руководство оператора.

Разработка должна быть проведена в следующие стадии и этапы:

1. Анализ требований:

На стадии анализ требований формулируются цели и задачи проекта. Создается основа для дальнейшего проектирования

2. Проектирование:

На стадии проектирование должны быть выполнены перечисленные ниже этапы работ:

- разработка программной документации;

На этапе разработка программной документации должна быть выполнена разработка технического задания.

При разработке технического задания должны быть выполнены перечисленные работы: постановка задачи, определение и уточнение требований к техническим средствам, определение требований к программе, определение стадий, этапов и сроков разработки программы и документации на нее, выбор языков программирования.

- разработка алгоритма программы;

На этапе разработки алгоритма программы должен быть разработан алгоритм работы программы.

- кодирование;

На стадии кодирования происходит реализация алгоритмов в среде программирования.

- тестирование и отладка.

На стадии тестирование и отладка происходит проверка алгоритмов, реализованных в программе на работоспособность в различных ситуациях. Исправление выявленных ошибок, повторное тестирование.

Приемо-сдаточные испытания должны проводиться при использовании технических средств. Приемка программы заключается в проверке работоспособности программы путем ввода реальных или демонстрационных данных.

Во время приемки работы разработчик предоставляет программу и документацию, которая к ней прилагается. Проводятся испытания программы, при успешных испытаниях программа вводится в эксплуатацию. При ошибках, недопустимых для успешной работы программного продукта – отправляется на доработку.

Было описано техническое задание, содержащее в себе информацию о программном продукте, его функциях, эксплуатации и требования, которые должны учитываться при создании программы и документации к ней.

ОПИСАНИЕ АЛГОРИТМОВ И ФУНКЦИОНИРОВАНИЯ ПРОГРАММЫ

Наименование программы – «Цифровой водяной знак». Программа предназначена для создания цифрового водяного знака в цифровом изображении.

Функциональным назначением программы является создание файлов изображения с цифровой подписью.

Алгоритм выполнения программы приведен схематично на рисунке 1 в нем отражается вся функциональная составляющая программы и ее основные функции в упрощенном виде.

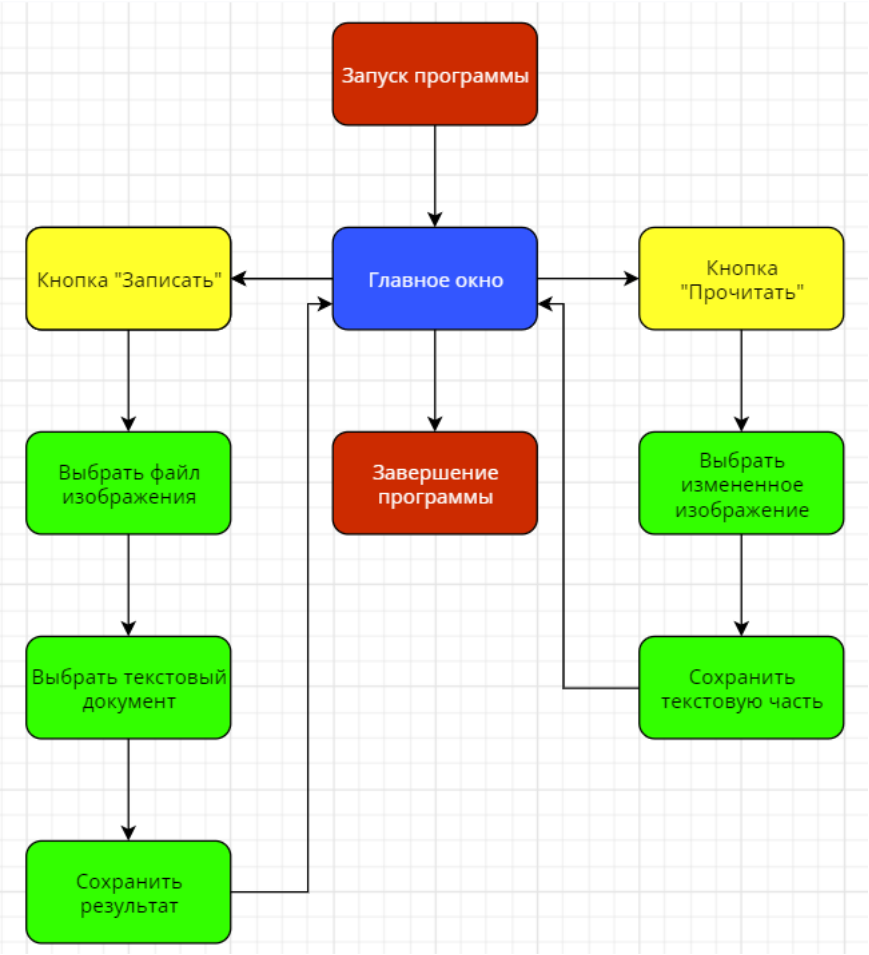

Рисунок 1 – Алгоритм выполнения программы

При запуске программы происходит отображение главной формы (рисунок 2) на которой пользователю предлагается выбрать зашифровать или прочитать зашифрованное изображение.

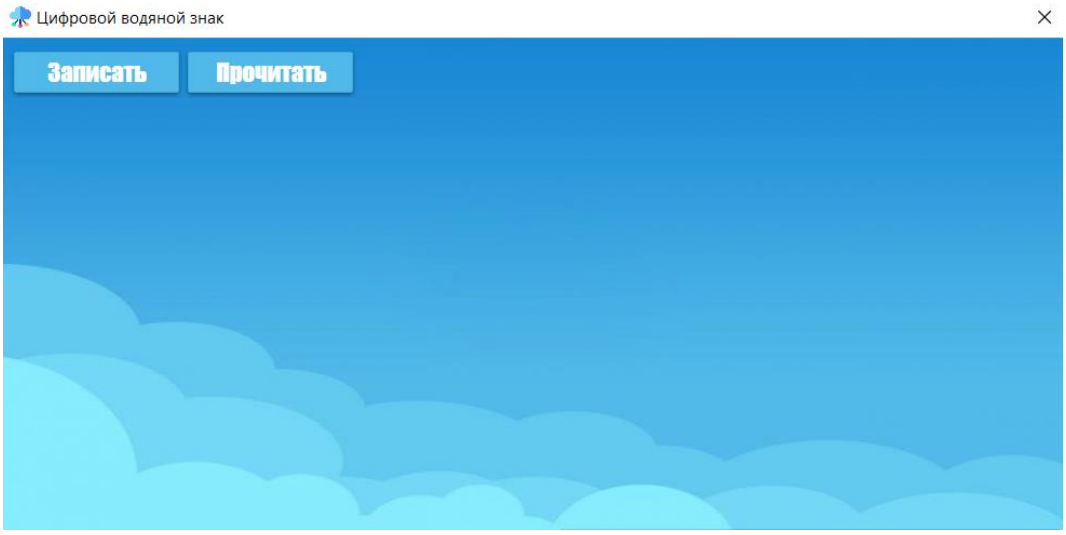

Рисунок 2 – Главное окно

Кнопка «Записать» запускает выбор файлов изображения, текстового документа, а после чего сохраняет получившийся результат в виде изображения, выгружая результаты на форму (рисунок 3).

Рисунок 3 – Результаты преобразования

Кнопка «Прочитать» запускает процесс выбора зашифрованного изображения, после чего расшифровывает его и записывает результат в текстовый файл, выгружая результаты на форму.

Для стилизации программы были использованы пакеты NuGet MaterialDesignColors и MaterialDesignThemes.

Для разбития пикселя и других взаимодействий с изображением были использованы библиотеки: System.Drawing; System.Windows.Media.Imaging.

Для вызова различных взаимодействий и преобразований в операционной системе были задействованы следующие библиотеки: System.IO; Microsoft.Win32.

В программе присутствуют методы, используемые для перевода из byte to bit для разбития пикселя и пересборки его (рисунок 4).

```
private BitArray ByteToBit(byte src)
        {
            BitArray bitArray = new BitArray(8);
            bool st = false;
            for (int i = 0; i < 8; i++)
            {
                if ((src >> i & 1) == 1) {st = true}
                else st = false;
                bitArray[i] = st;
            }
            return bitArray;
        }
```

Рисунок 4 – Метод перевода из byte to bit

Для нормализации количества символов шифрования используется метод NormalizeWriteCount с определенной константой, чтобы символы всегда занимали определённое количество байт (рисунок 5).

```
private byte[] NormalizeWriteCount(byte[] CountSymbols)
        {
            int PaddingByte = ENCRYP_TEXT_SIZE - CountSymbols.Length;
            byte[] WriteCount = new byte[ENCRYP_TEXT_SIZE];
            for (int j = 0; j < PaddingByte; j++)
            {WriteCount[j] = 0x30;}
            for (int j = PaddingByte; j < ENCRYP_TEXT_SIZE; j++)
            {
                WriteCount[j] = CountSymbols[j - PaddingByte];
            }
            return WriteCount;
        }
```

Рисунок 5 – Нормализация количества символов

Для проверки зашифрован ли файл используется метод isEncryptionc (рисунок 6).

```
private bool isEncryption(Bitmap scr)
      {
         byte[] rez = new byte[1];
         System.Drawing.Color color = scr.GetPixel(0, 0);
         BitArray colorArray = ByteToBit(color.R);
         BitArray messageArray = ByteToBit(color.R);

         messageArray[0] = colorArray[0];
         messageArray[1] = colorArray[1];
         colorArray = ByteToBit(color.G);

         messageArray[2] = colorArray[0];
         messageArray[3] = colorArray[1];
         messageArray[4] = colorArray[2];
         colorArray = ByteToBit(color.B);

         messageArray[5] = colorArray[0];
         messageArray[6] = colorArray[1];
         messageArray[7] = colorArray[2];
         rez[0] = BitToByte(messageArray);
         string m = Encoding.GetEncoding(1251).GetString(rez);
         if (m == "/")
         {
            return true;
         }
         else return false;
      }
```

Рисунок 6 – Проверка шифрации

Методы ByteToBit (перевода из byte to bit), NormalizeWriteCount (Нормализация количества символов) и isEncryptionc (Проверка шифрации) необходимы для упрощенного функционирования программы и уменьшения общего кода программы.

Основной способ шифрования используя выше приведенные методы представлен в ниже стоящем коде.

Первая часть кода представляет собой запись первого символа в первый пиксель для дальнейшего определения нахождения символов (рисунок 7).

```
byte[] Symbol = Encoding.GetEncoding(1251).GetBytes("/");
BitArray ArrBeginSymbol = ByteToBit(Symbol[0]);
Color curColor = bPic.GetPixel(0, 0);

BitArray tempArray = ByteToBit(curColor.R);
tempArray[0] = ArrBeginSymbol[0];
tempArray[1] = ArrBeginSymbol[1];
byte nR = BitToByte(tempArray);

tempArray = ByteToBit(curColor.G);
tempArray[0] = ArrBeginSymbol[2];
tempArray[1] = ArrBeginSymbol[3];
tempArray[2] = ArrBeginSymbol[4];
byte nG = BitToByte(tempArray);

tempArray = ByteToBit(curColor.B);
tempArray[0] = ArrBeginSymbol[5];
tempArray[1] = ArrBeginSymbol[6];
tempArray[2] = ArrBeginSymbol[7];
byte nB = BitToByte(tempArray);

Color nColor = Color.FromArgb(nR, nG, nB);
bPic.SetPixel(0, 0, nColor);
```

Рисунок 7 – Первая часть шифрования

Вторая часть кода используется непосредственно для представления всех символов в пикселях изображения (рисунок 8).

```
        WriteCountText(CountText, bPic);
        int index = 0;
        bool st = false;
        for (int i = ENCRYP_TEXT_SIZE + 1; i < bPic.Width; i++)
        {
           for (int j = 0; j < bPic.Height; j++)
           {
              Color pixelColor = bPic.GetPixel(i, j);
              if (index == bList.Count)
              {
                 st = true;
                 break;
              }
              BitArray colorArray = ByteToBit(pixelColor.R);
              BitArray messageArray = ByteToBit(bList[index]);
              colorArray[0] = messageArray[0];
              colorArray[1] = messageArray[1];
              byte newR = BitToByte(colorArray);
              colorArray = ByteToBit(pixelColor.G);
              colorArray[0] = messageArray[2];
              colorArray[1] = messageArray[3];
              colorArray[2] = messageArray[4];
              byte newG = BitToByte(colorArray);
              colorArray = ByteToBit(pixelColor.B);
              colorArray[0] = messageArray[5];
              colorArray[1] = messageArray[6];
              colorArray[2] = messageArray[7];
              byte newB = BitToByte(colorArray);
              Color newColor = Color.FromArgb(newR, newG,
newB);
              bPic.SetPixel(i, j, newColor);
              index++;
           }
           if (st)
           {
              break;
           }
        }
```

Рисунок 8 – Вторая часть шифрования

Для чтения символов из картинки используется похожий алгоритм, как приведенный выше (рисунок 9).

```
int countSymbol = ReadCountText(bPic);
byte[] message = new byte[countSymbol];
int index = 0;
bool st = false;
for (int i = ENCRYP_TEXT_SIZE + 1; i < bPic.Width; i++)
{
   for (int j = 0; j < bPic.Height; j++)
   {
      Color pixelColor = bPic.GetPixel(i, j);
      if (index == message.Length)
      {
         st = true;
         break;
      }
      BitArray colorArray = ByteToBit(pixelColor.R);
      BitArray messageArray = ByteToBit(pixelColor.R); ;
      messageArray[0] = colorArray[0];
      messageArray[1] = colorArray[1];
      colorArray = ByteToBit(pixelColor.G);
      messageArray[2] = colorArray[0];
      messageArray[3] = colorArray[1];
      messageArray[4] = colorArray[2];
      colorArray = ByteToBit(pixelColor.B);
      messageArray[5] = colorArray[0];
      messageArray[6] = colorArray[1];
      messageArray[7] = colorArray[2];
      message[index] = BitToByte(messageArray);
      index++;
   }
   if (st){break;}
}
string strMessage = Encoding.GetEncoding(1251).GetString(message);
```

Рисунок 9 – Чтение символов

В этом параграфе были описаны основные методы и алгоритмы функционирования программы и библиотеки, используемые в программе.

ТЕСТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ

Для проведения тестирования программы мною было произведено базовое тестирование во время разработки программы. При тестировании был выявлен ряд ошибок, которые возникли в ходе выполнения программы.

- Попытка выбрать файл некорректного формата

Выбор некорректного формата файла (.zip).

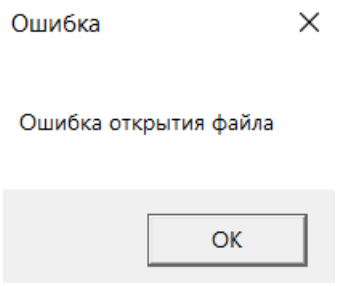

Рисунок 1 – Ошибка открытия файла

Ожидаемый результат: Ошибка о некорректных данных.

Полученный результат: Ошибка открытия файла (рисунок 1).

Решение проблемы: Для выбора корректных значений были установлены фильтры в диалоге открытия файла, а также написаны форматы допустимых файлов в руководстве оператора.

- Попытка выбора уже зашифрованного изображения

Выбор зашифрованного изображения.

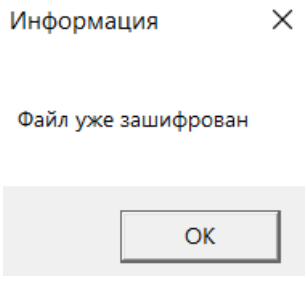

Рисунок 2 – Файл зашифрован

Ожидаемый результат: Сообщение о том, что файл зашифрован.

Полученный результат: Сообщение о том, что файл зашифрован (рисунок 2).

Решение проблемы: Для предотвращения повторной записи сообщения на изображение был установлен алгоритм проверяющий зашифровано ли изображения.

- Попытка выбора маленького для записи изображения

Выбор маленького для записи изображения (50 px на 50 px).

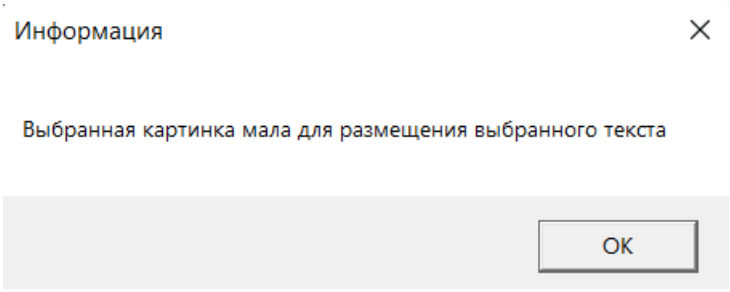

Рисунок 3 – Выбор маленького изображения

Ожидаемый результат: Сообщение о том, что изображение слишком мало.

Полученный результат: Сообщение о том, что изображения слишком мало (рисунок 3).

Решение проблемы: Для предотвращения данной проблемы было установлено условие, при котором если размер символов больше чем разрешение изображения программа возвращалась к исходному состоянию.

- Попытка выбора текста размер, которого превышает исполняемый алгоритм.

Выбор большого текстового файла (3000 символов).

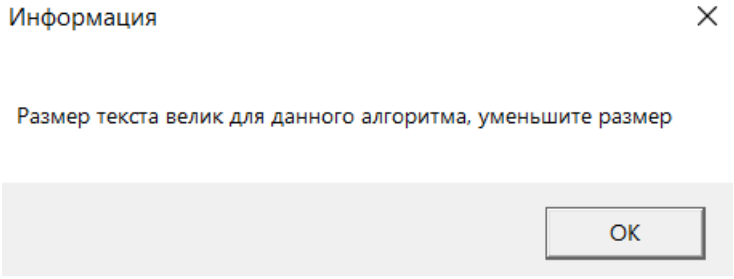

Рисунок 4 – Большой размер текста

Ожидаемый результат: Сообщение о том, что текст слишком большой.

Полученный результат: Сообщение о том, что текст слишком большой (рисунок 4).

Решение проблемы: В алгоритме создания цифрового водяного знака присутствует константа, согласно которой текстовый файл не может превышать 1999 символов, так как алгоритм не рассчитан на больший набор символов.

- Попытка выбора файла без зашифрованной информации при дешифрации

Выбор не зашифрованного изображения при дешифрации

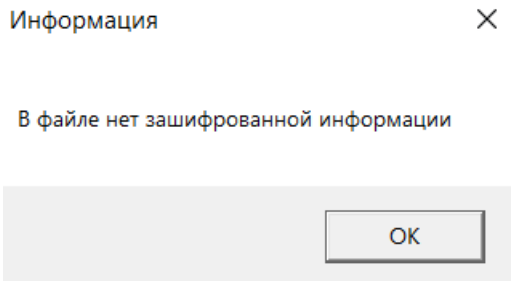

Рисунок 5 – Файл не зашифрован

Ожидаемый результат: Сообщение о том, что в файле нет зашифрованной информации.

Полученный результат: Сообщение о том, что в файле нет зашифрованной информации (рисунок 5).

Решение проблемы: Для предотвращения ошибок чтения зашифрованного файла в приложении присутствует алгоритм нахождения зашифрованной информации в изображении.

Тестирование программы осуществлялось на персональном компьютере со следующими техническими характеристиками:

- Процессор – Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz
- Оперативная память – DDR3 8 ГБ
- Видеокарта – AMD Radeon 530 2 ГБ
- Операционная система – Windows 11

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Функциональным назначением программы является создание файлов изображения с цифровой подписью.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- Открывать изображения и текстовые файлы.
- Шифровать в изображение текстовую информацию.
- Расшифровывать изображения, зашифрованные программой.
- Сохранять изображения и текстовые файлы.

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий себя:

- процессор с тактовой частотой, 1 ГГц, не менее;
- оперативную память объемом 512 Мб, не менее;
- жесткий диск со свободным местом 500 Мб, не менее;
- монитор, с разрешением экрана 1024*768, не менее;
- оптический привод;
- компьютерная мышь;
- клавиатура;

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7/8/10/11.

Все пользователи должны обладать навыками работы с графическим пользовательским интерфейсом операционной системы.

Выполнение программы. Для запуска программного продукта необходимо запустить «DW» с расширением exe.

Для выбора файлов записи цифрового водяного знака следует нажать кнопку «Записать», а для расшифровки изображения с цифровым водяным знаком нажать кнопку «Прочитать» (рисунок 1).

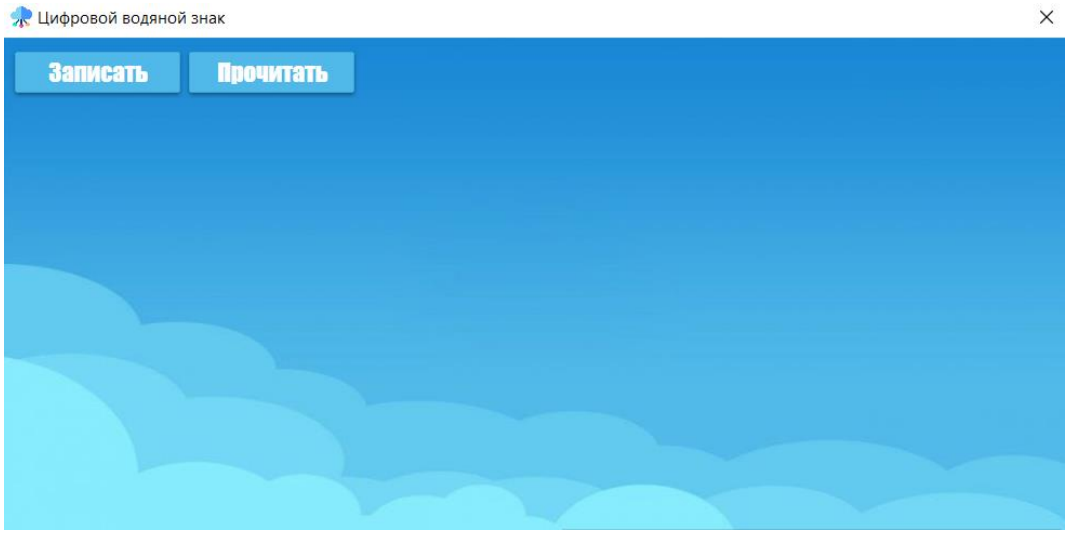

Рисунок 1 – Главное окно

После выбора изображения .png, .jpg, .bmp форматов и текстового файла с кодировкой ANSI, пользователю предлагается сохранить получившееся зашифрованное изображения, выгружая результаты в окно пользователя (рисунок 2).

Рисунок 2 – Результаты преобразования

При выборе уже зашифрованного изображения выходит сообщение об этом (рисунок 3).

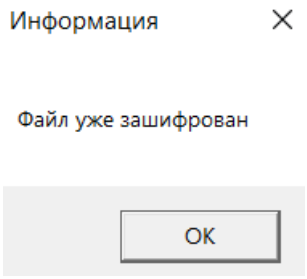

Рисунок 3 – Файл зашифрован

При расшифровке изображения предлагает выбрать файл изображения с зашифрованной информацией в нем, если файл не зашифрован пользователь получит сообщение об этом (рисунок 4).

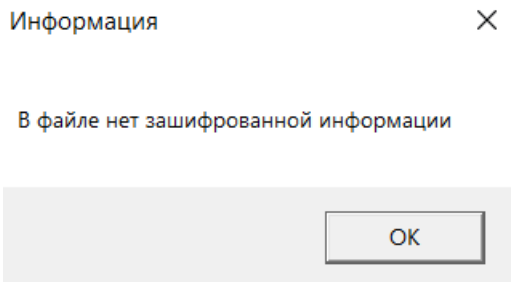

Рисунок 4 – Файл не зашифрован

При выборе других форматов файлов, не подходящих под нужное разрешение, выходит сообщение об ошибке открытия (рисунок 5), после чего можно выбрать другие файлы.

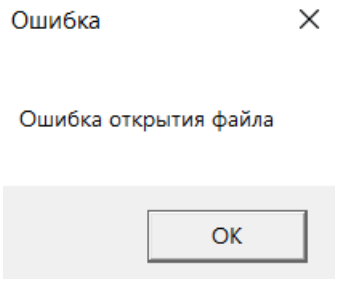

Рисунок 5 – Ошибка открытия файла

Были проверены все функциональные возможности программы при работе с реальными данными.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была создана программы для защиты цифрового изображения с помощью цифрового водяного знака. Основные задачи – предоставление пользователю понятного интерфейса для создания цифрового водяного знака на цифровом изображении – были осуществлены в ходе разработки приложения. Были выполнены следующие поставленные задачи:

1. Создание и сохранение цифрового водяного знака в цифровом изображении.
2. Расшифровка цифрового водяного знака в цифровом изображении.
3. Удобный интерфейс, интуитивно понятный пользователю.

Программа обладает функциями:

1. Открытие изображения и текстового файла.
2. Сохранение зашифрованного изображения и текстового файла.
3. Вывод результатов шифрования на экран пользователя.

Было осуществлено тестирование программы:

1. Ошибок не обнаружено.
2. Программа работает корректно.
3. Устойчивый вычислительный процесс.

В данном приложении пользователь может с легкостью защитить свое изображение и текстовую информацию от вредоносного использования злоумышленников. Пользователи, которые ищут способ защитить свои изображения могут воспользоваться этим приложением для защиты данных.

СПИСОК ЛИТЕРАТУРЫ

1. Аграновский А.В., Балакин А.В., Грибунин В.Г., Сапожников С.А. Стеганография, цифровые водяные знаки и стеганоанализ. – М.: Вузовская книга, 2009. – 220 с.

2. Зарыпов А. Распараллеливание алгоритма шифрования AES с использованием функциональной парадигмы программирования. – Челябинск: ЮУрГУ, 2016. – 25 с.

3. Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и практика – М.: МК-Пресс, 2006. – 288 с.

4. AES-Based Authenticated Encryption Modes in Parallel HighPerfomance Software, 2014. [Электронный ресурс] URL: http://eprint.iacr.org/2014/186.pdf (дата обращения: 10.07.2022).

5. Daemen J., Rijmen V. AES Proposal: Rijndael, 1999. [Электронный ресурс] URL: http://www.eng.tau.ac.il/~yash/crypto-netsec/Rijndael.pdf (дата обращения: 10.07.2022).

6. Lu, C.-S. Multimedia security: Steganography and digital watermarking techniques for protection of intellectual property – Hershey: Idea Group Publishing, 2005. – 255 p.

7. Microsoft Portable Executable and Common Object File Format Specification, 2017. [Электронный ресурс] URL: http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx (дата обращения: 10.07.2022).

8. National Policy on the Use of the Advanced Encryption Standard (AES). [Электронный ресурс] URL: http://cryptome.org/aes-natsec.htm (дата обращения: 10.07.2022).

9. Parallel AES Encryption Engines for Many-Core Processors Arrays, 2014. [Электронный ресурс] URL: http://www.rroij.com/open-access/parallelaes-encryption-engines-for-manycoreprocessor-arrays.pdf (дата обращения: 10.07.2022).

10. Pietrek M. An In-Depth Look into the Win32 Portable Executable File Format, 2002. [Электронный ресурс] URL: https://msdn.microsoft.com/ruru/magazine/bb985992(en-us).aspx (дата обращения: 10.07.2022).

11. Pietrek M. An In-Depth Look into the Win32 Portable Executable File Format, Part 2, 2002. [Электронный ресурс] URL: https://msdn.microsoft.com/ruru/magazine/bb985994(en-us).aspx (дата обращения: 10.07.2022).

12. Pietrek M. Peering Inside the PE: A Tour of the Win32 Portable Executable File Format. 1994. [Электронный ресурс] URL: https://msdn.microsoft.com/en-us/library/ms809762.aspx (дата обращения: 10.07.2022).

13. Recommendation for Block Chiper Mods of Operation, 2001. [Электронный ресурс] URL: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf (дата обращения: 10.07.2022).

14. Shin D., Kim Y., Byun K., Lee S. Data Hiding in Windows Executable Files, 2008. [Электронный ресурс] URL: http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1050&context=adf (дата обращения: 10.07.2022).

15. Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001. [Электронный ресурс] URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (дата обращения: 10.07.2022).

ПРИЛОЖЕНИЕ

Приложение 1

КОД ПРОГРАММЫ

Файл MainWindow.xaml.cs

```
using System;
using Microsoft.Win32;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.IO;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Collections;
using Color = System.Drawing.Color;

namespace DW
{
   public partial class MainWindow : Window
   {
      const int ENCRYP_PESENT_SIZE = 1;
      const int ENCRYP_TEXT_SIZE = 3;
      const int ENCRYP_TEXT_MAX_SIZE = 1999;
      private BitArray ByteToBit(byte src)
      {
         BitArray bitArray = new BitArray(8);
         bool st = false;
         for (int i = 0; i < 8; i++)
         {
            if ((src >> i & 1) == 1)
            {
               st = true;
            }
            else st = false;
```

```
            bitArray[i] = st;
        }
        return bitArray;
    }
    private byte BitToByte(BitArray scr) // перевод из bit in byte
    {
        byte num = 0;
        for (int i = 0; i < scr.Count; i++)
            if (scr[i] == true)
                num += (byte)Math.Pow(2, i);
        return num;
    }
    private bool isEncryption(Bitmap scr)
    {
        byte[] rez = new byte[1];
        System.Drawing.Color color = scr.GetPixel(0, 0);
        BitArray colorArray = ByteToBit(color.R);
        BitArray messageArray = ByteToBit(color.R);
        messageArray[0] = colorArray[0];
        messageArray[1] = colorArray[1];
        colorArray = ByteToBit(color.G);
        messageArray[2] = colorArray[0];
        messageArray[3] = colorArray[1];
        messageArray[4] = colorArray[2];
        colorArray = ByteToBit(color.B);
        messageArray[5] = colorArray[0];
        messageArray[6] = colorArray[1];
        messageArray[7] = colorArray[2];
        rez[0] = BitToByte(messageArray);
        string m = Encoding.GetEncoding(1251).GetString(rez);
        if (m == "/")
        {
            return true;
        }
        else return false;
    }
    private byte[] NormalizeWriteCount(byte[] CountSymbols)
    {
        int PaddingByte = ENCRYP_TEXT_SIZE - CountSymbols.Length;
        byte[] WriteCount = new byte[ENCRYP_TEXT_SIZE];
        for (int j = 0; j < PaddingByte; j++)
        {
```

```
        WriteCount[j] = 0x30;
    }
    for (int j = PaddingByte; j < ENCRYP_TEXT_SIZE; j++)
    {
        WriteCount[j] = CountSymbols[j - PaddingByte];
    }
    return WriteCount;
}
        private void WriteCountText(int count, Bitmap src)
{
    byte[] CountSymbols = Encoding.GetEncoding(1251).GetBytes(count.ToString());
    if (CountSymbols.Length < ENCRYP_TEXT_SIZE)
    {
        CountSymbols = NormalizeWriteCount(CountSymbols);
    }
    for (int i = 0; i < ENCRYP_TEXT_SIZE; i++)
    {
        BitArray bitCount = ByteToBit(CountSymbols[i]);
        Color pColor = src.GetPixel(0, i + 1);
        BitArray bitsCurColor = ByteToBit(pColor.R);
        bitsCurColor[0] = bitCount[0];
        bitsCurColor[1] = bitCount[1];
        byte nR = BitToByte(bitsCurColor);
        bitsCurColor = ByteToBit(pColor.G);
        bitsCurColor[0] = bitCount[2];
        bitsCurColor[1] = bitCount[3];
        bitsCurColor[2] = bitCount[4];
        byte nG = BitToByte(bitsCurColor);
        bitsCurColor = ByteToBit(pColor.B);
        bitsCurColor[0] = bitCount[5];
        bitsCurColor[1] = bitCount[6];
        bitsCurColor[2] = bitCount[7];
        byte nB = BitToByte(bitsCurColor);
        Color nColor = Color.FromArgb(nR, nG, nB);
        src.SetPixel(0, i + 1, nColor);
    }
}
private int ReadCountText(Bitmap src)
{
    byte[] rez = new byte[ENCRYP_TEXT_SIZE];
    for (int i = 0; i < ENCRYP_TEXT_SIZE; i++)
    {
```

```
            Color color = src.GetPixel(0, i + 1);
            BitArray colorArray = ByteToBit(color.R);
            BitArray bitCount = ByteToBit(color.R); ;
            bitCount[0] = colorArray[0];
            bitCount[1] = colorArray[1];
            colorArray = ByteToBit(color.G);
            bitCount[2] = colorArray[0];
            bitCount[3] = colorArray[1];
            bitCount[4] = colorArray[2];
            colorArray = ByteToBit(color.B);
            bitCount[5] = colorArray[0];
            bitCount[6] = colorArray[1];
            bitCount[7] = colorArray[2];
            rez[i] = BitToByte(bitCount);
        }
        string m = Encoding.GetEncoding(1251).GetString(rez);
        return Convert.ToInt32(m, 10);
    }
    public MainWindow()
    {
        InitializeComponent();
    }
    private void write_Click(object sender, RoutedEventArgs e)
    {
        string FilePic;
        string FileText;
        OpenFileDialog dPic = new OpenFileDialog();
        dPic.Filter = "Файлы изображения|*.bmp;*.jpg;*.png|Все файлы|*.*";
        if (dPic.ShowDialog() == true)
        {
            FilePic = dPic.FileName;
        }
        else
        {
            FilePic = "";
            return;
        }
        FileStream rFile;
        Bitmap bPic;
        try
        {
            rFile = new FileStream(FilePic, FileMode.Open);
```

```
    bPic = new Bitmap(rFile);
}
catch (Exception)
{
    MessageBox.Show("Ошибка открытия файла", "Ошибка");
    return;
}
OpenFileDialog dText = new OpenFileDialog();
dText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
if (dText.ShowDialog() == true)
{
    FileText = dText.FileName;
}
else
{
    FileText = "";
    return;
}
try
{
    tb.Text = File.ReadAllText(FileText, Encoding.Default);
}
catch (Exception)
{
    rFile.Close();
    MessageBox.Show("Ошибка открытия файла", "Ошибка");
    return;
}
FileStream rText;
BinaryReader bText;
try
{
    rText = new FileStream(FileText, FileMode.Open);
    bText = new BinaryReader(rText, Encoding.ASCII);
}
catch (Exception)
{
    rFile.Close();
    MessageBox.Show("Ошибка открытия файла", "Ошибка");
    return;
}
List<byte> bList = new List<byte>();
```

```
while (bText.PeekChar() != -1)
{
   bList.Add(bText.ReadByte());
}
int CountText = bList.Count;
bText.Close();
rFile.Close();
if (CountText > (ENCRYP_TEXT_MAX_SIZE - ENCRYP_PESENT_SIZE - ENCRYP_TEXT_SIZE))
{
   MessageBox.Show("Размер текста велик для данного алгоритма, уменьшите размер", "Информация");
   return;
}
if (CountText > (bPic.Width * bPic.Height))
{
   MessageBox.Show("Выбранная картинка мала для размещения выбранного текста", "Информация");
   return;
}
if (isEncryption(bPic))
{
   MessageBox.Show("Файл уже зашифрован", "Информация");
   return;
}
byte[] Symbol = Encoding.GetEncoding(1251).GetBytes("/");
BitArray ArrBeginSymbol = ByteToBit(Symbol[0]);
Color curColor = bPic.GetPixel(0, 0);
BitArray tempArray = ByteToBit(curColor.R);
tempArray[0] = ArrBeginSymbol[0];
tempArray[1] = ArrBeginSymbol[1];
byte nR = BitToByte(tempArray);
tempArray = ByteToBit(curColor.G);
tempArray[0] = ArrBeginSymbol[2];
tempArray[1] = ArrBeginSymbol[3];
tempArray[2] = ArrBeginSymbol[4];
byte nG = BitToByte(tempArray);
tempArray = ByteToBit(curColor.B);
tempArray[0] = ArrBeginSymbol[5];
tempArray[1] = ArrBeginSymbol[6];
tempArray[2] = ArrBeginSymbol[7];
byte nB = BitToByte(tempArray);
Color nColor = Color.FromArgb(nR, nG, nB);
bPic.SetPixel(0, 0, nColor);
WriteCountText(CountText, bPic);
```

```
int index = 0;
bool st = false;
for (int i = ENCRYP_TEXT_SIZE + 1; i < bPic.Width; i++)
{
   for (int j = 0; j < bPic.Height; j++)
   {
      Color pixelColor = bPic.GetPixel(i, j);
      if (index == bList.Count)
      {
         st = true;
         break;
      }
      BitArray colorArray = ByteToBit(pixelColor.R);
      BitArray messageArray = ByteToBit(bList[index]);
      colorArray[0] = messageArray[0];
      colorArray[1] = messageArray[1];
      byte newR = BitToByte(colorArray);
      colorArray = ByteToBit(pixelColor.G);
      colorArray[0] = messageArray[2];
      colorArray[1] = messageArray[3];
      colorArray[2] = messageArray[4];
      byte newG = BitToByte(colorArray);
      colorArray = ByteToBit(pixelColor.B);
      colorArray[0] = messageArray[5];
      colorArray[1] = messageArray[6];
      colorArray[2] = messageArray[7];
      byte newB = BitToByte(colorArray);
      Color newColor = Color.FromArgb(newR, newG, newB);
      bPic.SetPixel(i, j, newColor);
      index++;
   }
   if (st)
   {
      break;
   }
}
MemoryStream ms = new MemoryStream();
((System.Drawing.Bitmap)bPic).Save(ms, System.Drawing.Imaging.ImageFormat.Bmp);
BitmapImage image = new BitmapImage();
image.BeginInit();
ms.Seek(0, SeekOrigin.Begin);
image.StreamSource = ms;
```

```
    image.EndInit();
    imagebox.Source = image;
    String sFilePic;
    SaveFileDialog dSavePic = new SaveFileDialog();
    dSavePic.Filter = "Файлы изображений (*.bmp)|*.bmp;*.jpg;*.png|Все файлы (*.*)|*.*";
    if (dSavePic.ShowDialog() == true)
    {
        sFilePic = dSavePic.FileName;
    }
    else
    {
        sFilePic = "";
        return;
    };
    FileStream wFile;
    try
    {
        wFile = new FileStream(sFilePic, FileMode.Create);
    }
    catch (Exception)
    {
        MessageBox.Show("Ошибка открытия файла на запись", "Ошибка");
        return;
    }
    bPic.Save(wFile, System.Drawing.Imaging.ImageFormat.Bmp);
    wFile.Close();
}
private void read_Click(object sender, RoutedEventArgs e)
{
    string FilePic;
    OpenFileDialog dPic = new OpenFileDialog();
    dPic.Filter = "Файлы изображений (*.bmp)|*.bmp;*.jpg;*.png|Все файлы (*.*)|*.*";
    if (dPic.ShowDialog() == true)
    {
        FilePic = dPic.FileName;
    }
    else
    {
        FilePic = "";
        return;
    }
    FileStream rFile;
```

```
Bitmap bPic;
try
{
   rFile = new FileStream(FilePic, FileMode.Open);
   bPic = new Bitmap(rFile);
}
catch (Exception)
{
   MessageBox.Show("Ошибка открытия файла", "Ошибка");
   return;
}
if (!isEncryption(bPic))
{
   MessageBox.Show("В файле нет зашифрованной информации", "Информация");
   rFile.Close();
   return;
}

int countSymbol = ReadCountText(bPic);          byte[] message = new byte[countSymbol];
int index = 0;
bool st = false;
for (int i = ENCRYP_TEXT_SIZE + 1; i < bPic.Width; i++)
{
   for (int j = 0; j < bPic.Height; j++)
   {
      Color pixelColor = bPic.GetPixel(i, j);
      if (index == message.Length)
      {
         st = true;
         break;
      }
      BitArray colorArray = ByteToBit(pixelColor.R);
      BitArray messageArray = ByteToBit(pixelColor.R); ;
      messageArray[0] = colorArray[0];
      messageArray[1] = colorArray[1];
      colorArray = ByteToBit(pixelColor.G);
      messageArray[2] = colorArray[0];
      messageArray[3] = colorArray[1];
      messageArray[4] = colorArray[2];
      colorArray = ByteToBit(pixelColor.B);
      messageArray[5] = colorArray[0];
      messageArray[6] = colorArray[1];
```

```
        messageArray[7] = colorArray[2];
        message[index] = BitToByte(messageArray);
        index++;
    }
    if (st)
    {
        break;
    }
}
string strMessage = Encoding.GetEncoding(1251).GetString(message);
string sFileText;
SaveFileDialog dSaveText = new SaveFileDialog();
dSaveText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
if (dSaveText.ShowDialog() == true)
{
    sFileText = dSaveText.FileName;
}
else
{
    sFileText = "";
    rFile.Close();
    return;
};
FileStream wFile;
try
{
    wFile = new FileStream(sFileText, FileMode.Create);          }
catch (Exception)
{
    MessageBox.Show("Ошибка открытия файла на запись", "Ошибка");
    rFile.Close();
    return;
}
MemoryStream ms = new MemoryStream();
((System.Drawing.Bitmap)bPic).Save(ms, System.Drawing.Imaging.ImageFormat.Bmp);
BitmapImage image = new BitmapImage();
image.BeginInit();
ms.Seek(0, SeekOrigin.Begin);
image.StreamSource = ms;
image.EndInit();
imagebox.Source = image;
StreamWriter wText = new StreamWriter(wFile, Encoding.Default);
```

```
            wText.Write(strMessage);
            wText.Close();
            wFile.Close();
            rFile.Close();
            try
            {
                tb.Text = File.ReadAllText(sFileText, Encoding.Default);
            }
            catch (Exception)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка");
                return;
            }
            MessageBox.Show("Текст записан в файл", "Информация");
        }

        private void Window_Closed(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Файл MainWindow.xaml

```
<Window x:Class="Cursun.MainWindow"      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:local="clr-namespace:Cursun"
mc:Ignorable="d"
xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
Title="Цифровой водяной знак" Height="405" Width="805" WindowStartupLocation="CenterScreen"
Closed="Window_Closed" ResizeMode="NoResize" Icon="free-icon-cloud-network-8637953.ico">
    <Grid>
        <Grid.Background>
            <ImageBrush ImageSource="/63f5a466aae9ad295925078ab628af80.jpg"/>
        </Grid.Background>
        <Button x:Name="read" Content="Прочитать" HorizontalAlignment="Left" Margin="139,10,0,0"
VerticalAlignment="Top" Width="124" FontFamily="Impact" FontSize="18" BorderBrush="{x:Null}"
Foreground="White" Background="#FF50B9E9" Click="read_Click"/>
        <Button x:Name="write" Content="Записать" HorizontalAlignment="Left" Margin="10,10,0,0"
VerticalAlignment="Top" Width="124" FontFamily="Impact" FontSize="18" BorderBrush="{x:Null}"
Foreground="White" Background="#FF50B9E9" Click="write_Click"/>
```

```
    <Image x:Name="imagebox" HorizontalAlignment="Left" Height="320" Margin="10,50,0,0"
VerticalAlignment="Top" Width="400" OpacityMask="Black" Stretch="Uniform"/>
    <TextBox x:Name="tb" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto"
HorizontalAlignment="Left" Margin="415,45,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="320"
Width="370" FontFamily="Impact" FontSize="18" Foreground="White" IsReadOnly="True"></TextBox>
  </Grid>
</Window>
```

Приложение 2

Компакт-диск с материалами проекта

На диске располагается:

- Установщик программы
- Проект программы
- Файл курсового проекта в формате MS Word
- Файл с презентацией курсового проекта