**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»  
ИНСТИТУТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

|  |  |
| --- | --- |
|  | **Допустить к защите** Заместитель директора  по УМР \_\_\_\_\_\_\_\_\_\_\_\_ Е.Г. Конакина\_  (Подпись) (ФИО)  «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2024 г. |
|  |

**курсовой проект**Тема: «Программа для взлома шифра Цезаря»

специальность 09.02.07 группа 329191/1

Студент (ка) Шабанов Д.В.  
 (подпись) (ФИО)  
Преподаватель Ильин Ю.П.

(подпись) (ФИО)

Санкт-Петербург  
2024

Содержание

[ВВЕДЕНИЕ 3](#_Toc157501182)

[1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ 4](#_Toc157501183)

[1.1. Описание предметной области 4](#_Toc157501184)

[1.2. Анализ методов решения 6](#_Toc157501185)

[1.3. Обзор средств программирования 7](#_Toc157501186)

[1.4. Описание языка C# 7](#_Toc157501187)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_Toc157501188)

[2.1. Постановка задачи 11](#_Toc157501189)

[2.1.1. Основания для разработки 11](#_Toc157501190)

[2.1.2. Назначение программы 11](#_Toc157501191)

[2.2. Проектирование приложения 11](#_Toc157501192)

[2.2.1. Диаграмма прецедентов 11](#_Toc157501193)

[2.2.2. Диаграмма последовательностей 12](#_Toc157501194)

[2.2.3. Диаграмма классов 14](#_Toc157501195)

[2.2.4. Диаграмма активности 14](#_Toc157501196)

[2.3. Текст программы 16](#_Toc157501197)

[2.4. Описание программы 16](#_Toc157501198)

[2.4.1. Общие сведения 16](#_Toc157501199)

[2.4.2. Функциональное назначение 16](#_Toc157501200)

[2.4.3. Описание логической структуры системы 16](#_Toc157501201)

[2.4.4. Используемые технические и программные средства 18](#_Toc157501202)

[2.4.5. Вызов и загрузка 19](#_Toc157501203)

[2.5. Руководство оператора 19](#_Toc157501204)

[2.5.1. Назначение программы 19](#_Toc157501205)

[2.5.2. Выполнение программы и сообщения оператору 19](#_Toc157501206)

[2.6. Программа и методика испытаний 20](#_Toc157501207)

[2.6.1. Объект испытаний 20](#_Toc157501208)

[2.6.2. Цель испытаний 20](#_Toc157501209)

[2.6.3. Требования к программе 20](#_Toc157501210)

[2.6.4. Требования к программной документации 22](#_Toc157501211)

[2.6.5. Средства и порядок испытаний 22](#_Toc157501212)

[2.6.6. Методы испытаний 23](#_Toc157501213)

[2.7. Протокол испытаний 25](#_Toc157501214)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc157501215)

[ПРИЛОЖЕНИЕ А 2](#_Toc157501216)

[ПРИЛОЖЕНИЕ Б 8](#_Toc157501220)

[ПРИЛОЖЕНИЕ В 12](#_Toc157501221)

# ВВЕДЕНИЕ

В эпоху информационных технологий защита конфиденциальной информации становится все более важной задачей. Одним из простых, но все еще широко используемых методов шифрования является шифр Цезаря. Этот метод основан на сдвиге букв в алфавите на определенное количество позиций. Несмотря на свою простоту, шифр Цезаря до сих пор применяется в различных сферах, и поэтому существует потребность в инструментах для тестирования его надежности.

Разработанный проект представляет собой программу для взлома шифра Цезаря. С помощью этой программы пользователь может расшифровывать зашифрованные сообщения, также и наоборот. Программа предоставляет гибкий интерфейс для загрузки и работы с шифром, а также возможности для взлома шифра Цезаря с минимальным участием человека.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ

## 1.1. Описание предметной области

Программы по взлому шифров предназначены для анализа, расшифровки и восстановлению зашифрованных данных. Они используются в различных областях, включая криптографию и кибербезопасность. В основе их функционирования лежит анализ зашифрованных данных и применения к ним различных методов и алгоритмов для восстановления исходной информации.

**Криптография и шифрование –** передовая область кибербезопасности, занимающаяся шифрованием и защитой данных. Если говорить простыми словами, то она использует сложные алгоритмы для преобразования информации в зашифрованный вид, обеспечивая конфиденциальность и целостность данных в цифровом мире. В криптографии существует два главных типа алгоритмов шифрования: симметричный и асимметричный.

**Анализ исходных данных –** проводят анализ зашифрованных данных с целью выявления закономерностей, структуры и характеристик исходной информации. Этот анализ может включать в себя частотный анализ, и т.д.

**Методы взлома –** программы применяют различные методы и алгоритмы для взлома шифров, включая переборные методы, атаки по словарю, аналитические и статистические методы, машинное обучение и искусственный интеллект.

**Интерфейс пользователя –** программы обладают пользовательским интерфейсом, который позволяет загружать зашифрованные данные, просматривать результаты анализа и выбирать оптимальные методы взлома. Они также предоставляют возможности для настройки параметров анализа и взлома и вывода результатов в удобном формате.

## 1.2. Анализ методов решения

Алфавит размещается на круге по часовой стрелке (при этом в русском алфавите, после А идет Б, а после Я - А). Для зашифровки буквы текста заменяются буквами, отстоящими по кругу на заданное число букв дальше по часовой стрелке. Если, скажем, сдвиг на 3, то вместо i-й используется (i+3)-я буква, например, вместо А пишется Г а вместо Я пишется В. При расшифровке наоборот берут букву на заданное число букв ближе, т. е. двигаясь против часовой стрелки.

Шифр Цезаря расшифровать легко. Известны вероятности букв p[i],i =1,2,...,n, в языке сообщения ( n - число букв в алфавите). подсчитаем частоты букв f[i] в зашифрованном сообщении. Если оно не очень короткое, то f[i] должны сравнительно хорошо согласовываться с p[i]: f[i] = p[i-s] для некоторого сдвига s. Затем начнем делать перебор по сдвигам. Когда сдвиг не угадан, общее различие между p[i] и f[i+s] , равное D(s) = S | p[i] - f[i+s] | (суммирование берется по всем i от 1 до n), будет велико, а когда сдвиг угадан – мало. Минимизация D(s) по всем s = 1,2,...,n дает ключ к расшифровке кода Цезаря.   
 В моём проекте я буду использовать такие методы взлома, как:

* **Частотный анализ:**

Для русского языка наиболее часто встречающейся буквой является буква "о", за которой обычно следуют буквы "а", "е", "и", "н" и так далее в порядке убывания частотности. Однако точные частоты появления символов могут немного различаться в зависимости от текста и стиля письма.

Однако при малом объеме информации в зашифрованном сообщении, шанс его взлома очень низок, поэтому я использую второй метод для взлома зашифрованных сообщений:

* **Атаки по словарю:**

В моем случае словарем является файл со множеством русских слов в разных склонениях, по которому и сравниваются слова из зашифрованного сообщения.

## 1.3. Обзор средств программирования

**Анализ средств программирования**

|  |  |  |
| --- | --- | --- |
| **Язык** | **Скорость работы** | **Простота графического интерфейса** |
| Python | Высокая | Python известен своей простотой и читаемостью кода, что делает его идеальным выбором для разработки приложений. |
| С++ | Высокая | C++ предоставляет разработчикам широкий спектр инструментов для создания графического интерфейса, но требует более высокого уровня экспертизы и усилий по сравнению с другими языками. |
| С | Средняя | C обеспечивает среднюю скорость работы. Однако, создание графического интерфейса на C может быть более сложным и требует использования дополнительных библиотек и инструментов. |
| С# | Высокая | C# обеспечивает высокую простоту создания графического интерфейса благодаря платформе .NET и инструментам, таким как Windows Forms и WPF. Эти инструменты позволяют разработчикам легко создавать пользовательские интерфейсы. |

Таким образом, средством программирования данного курсового проекта был выбран язык С#, среда разработки Visual Studio 2017.

## 1.4. Описание языка C#

Язык C#, разработанный компанией Майкрософт, один из самых популярных современных языков программирования. Он востребован на рынке разработки в различных странах, C# применяют при работе с программами для ПК, создании сложных веб-сервисов или мобильных приложений. Появившийся как язык для собственных нужд платформы Microsoft .NET, постепенно этот язык стал очень популярным. А потому мы решили сделать небольшой обзор для тех, кто выбирает, какие инструменты разработки стоит освоить в ближайшее время.

Итак, разработка языка началась в 1998 году, а первая версия увидела свет в 2001. Группой разработчиков руководил известный в профессиональных кругах специалист Андерс Хейлсберг. Новые версии C# выходят сравнительно часто, а текущие доработки, исправление багов и расширение библиотек ведется практически на постоянной основе.

В результате язык получился крайне гибкий, мощный и универсальный. На нем пишут практически все, что угодно, от небольших веб-приложений до мощных программных систем, объединяющих в себе веб-структуры, приложения для десктопов и мобильных устройств. Все это стало возможным благодаря удобному Си-подобному синтаксису, строгому структурированию, огромному количеству фреймворков и библиотек (их число достигает нескольких сотен).

Долгое время платформа .NET поставлялась с закрытым ядром, что создавало определенные сложности в разработке и снижало популярность C# в профессиональной среде. Но в ноябре 2014 Майкрософт радикально изменила подход и стала выдавать бесплатные лицензии для Visual Studio уже с открытым исходным кодом для всех наборов инструментов. [4]

**Ключевые слова C#:**

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | extern | null | struct |
| as | false | object | switch |
| base | finally | operator | this |
| bool | fixed | out | throw |
| break | float | override | true |
| byte | for | params | try |
| case | foreach | private | typeof |
| catch | goto | protected | uint |
| const | if | public | ulong |
| continue | implicit | readonly | unchecked |
| decimal | in | ref | unsafe |
| default | int | return | ushort |
| delegate | interface | sbyte | using |
| do | internal | sealed | virtual |
| double | is | short | void |
| else | lock | sizeof | while |
| enum | long | stackalloc |  |
| event | namespace | static |  |
| explicit | new | string |  |

**Контекстные ключевые слова C#:**

|  |  |  |  |
| --- | --- | --- | --- |
| add | equals | join | set |
| ascending | from | let | value |
| async | get | on | var |
| await | global | orderby | where |
| by | group | partial | yield |
| descending | in | remove |  |
| dynamic | into | select |  |

**Основные алгоритмические конструкции C#:**

|  |  |
| --- | --- |
| Оператор присваивания | string hello = "hello " + "world"; //результат равен "hello world"  int x1 = 2 + 4; // результат равен 6  int x2 = 10 - 6; //результат равен 4  int x3 = 10 \* 6; //результат равен 60  double x4 = 10.0 / 4.0; //результат равен 2.5  double x5 = 10.0 % 4.0; //результат равен 2  int y1 = 5;  int z1 = ++y1; // z1=6; y1=6  int y2 = 5;  int z2 = y2++; // z2=5; y2=6  int y3 = 5;  int z3 = --y3; // z3=4; y3=4  int y4 = 5;  int z4 = y4--; // z4=5; y4=4 |
| Условный оператор | *if (условие) {(действие) } else {(альтернатива)} ;*  int num1 = 8;  int num2 = 6;  if(num1 > num2)  {      Console.WriteLine("Число {0} больше числа {1}", num1, num2);  }  else  {      Console.WriteLine("Число {0} меньше числа {1}", num1, num2);  } |
| Оператор множественного ветвления | *Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий:*  Console.WriteLine("Нажмите Y или N");  string selection = Console.ReadLine();  switch (selection)  {      case "Y":          Console.WriteLine("Вы нажали букву Y");          break;      case "N":          Console.WriteLine("Вы нажали букву N");          break;      default:          Console.WriteLine("Вы нажали неизвестную букву");          break;  }  *После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора сase. И если совпадение будет найдено, то будет выполняться определенный блок сase.*  *В конце блока сase ставится оператор break, чтобы избежать выполнения других блоков.*  *Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок default,.* |
| Арифметический цикл  **(**применяется, когда известно количество повторений цикла) | for *([инициализация счетчика]; [условие]; [изменение счетчика])*  for (int i = 0; i < 10; i++)  {      Console.WriteLine("Квадрат числа {0} равен {1}", i, i \* i);  } |
| Цикл с предусловием  **(**применяется, когда неизвестно количество повторений цикла) | *while (условие) { тело цикла };*  Этот цикл будет выполняться до тех пор, пока истинно *условие* (логическое выражение, возвращающее значение типа **Boolean**). При этом если это выражение сразу равно **false**, *тело цикла* не будет выполнено ни разу. Нужно очень внимательно следить за написанием *условия* и контролем завершения цикла, так как в результате ошибки цикл **while** будет повторяться бесконечное количество раз, что приведёт к "зацикливанию" и "зависанию" программы. |
| Цикл с постусловием  **(**применяется, когда неизвестно количество повторений цикла) | *do { тело цикла } while (условие);*  Повторения сначала выполняет *тело цикла*, а затем уже проверяет выполнение *условия*. Таким образом, этот вариант цикла гарантирует, что *тело цикла* будет выполнено по крайней мере один раз. И будет выполняться до тех пор, пока *условие* не станет истинным (**true**). |

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Постановка задачи

### 2.1.1. Основания для разработки

Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.01 «Разработка программных модулей» и утверждена Институтом среднего профессионального образования.

### 2.1.2. Назначение программы

Программа для взлома шифра Цезаря: человек вводит с клавиатуры либо загружает из файла исходный текст, далее он может его зашифровать, сдвинув все буквы текста на k позиций. Также зная ключ, может его расшифровать. На второй вкладке можно загрузить зашифрованный текст из файла и взломать его, программа сама подберет ключ и преобразует текст в нормальную форму.

Полный текст технического задания приведен в приложении А.

## 2.2. Проектирование приложения

На этапе проектирования были разработаны диаграмма прецедентов, диаграмма последовательностей и диаграмма классов.

### 2.2.1. Диаграмма прецедентов

Диаграммы прецедентов частично описывает use-case – прецедент использования проектируемой системы, давая частичное описание частичного применения системы с точки зрения условного внешнего обозревателя. При этом описание фокусируется на том, что должна делать система по отношению к своему внешнему окружению (периферии), а не то на том, как она эта делает, то есть диаграмма прецедентов есть частичная спецификация (рисунок 1).



Рисунок 1 Диаграмма прецедентов

### 2.2.2. Диаграмма последовательностей

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Это линия жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.



Рисунок 2 Диаграмма последовательностей

### 2.2.3. Диаграмма классов

На рисунке 3 показана диаграмма классов проекта. Программа содержит 2 основных класса: Form1-класс формы игры, класс Login.cs содержит методы работы с бинарным файлом логинов и рекордами игроков. Этот класс использует модуль Form1 во время авторизации и при отображении таблицы рекордов.



Рисунок 3 Диаграмма классов проекта

## 2.3. Текст программы

Текст программы в соответствии с ГОСТ 19.101-77 (СТ СЭВ 1626-79) и ГОСТ 19.401-79 (СТ СЭВ 3746-82) представляет собой запись программы на исходном языке программирования с необходимыми комментариями. Текст программы представляет собой документ, выполненный машинным способом, и приведен в приложении В.

## 2.4. Описание программы

### 2.4.1. Общие сведения

Программа для взлома шифра Цезаря: человек вводит с клавиатуры либо загружает из файла исходный текст, далее он может его зашифровать, сдвинув все буквы текста на k позиций. Также зная ключ, может его расшифровать. На второй вкладке можно загрузить зашифрованный текст из файла и взломать его, программа сама подберет ключ и преобразует текст в нормальную форму.

### 2.4.2. Функциональное назначение

Основное назначение программного продукта заключается во взломе шифра Цезаря.

### 2.4.3. Описание логической структуры системы

Программа содержит 3 основных модуля:

Form1.cs — модуль формы работы с шифром Цезаря (Зашифровка и Дешифровка вручную), модуль Form2.cs — модуль формы для взлома шифра Цезаря, модуль Cezar.cs — содержит методы для работы с Шифром Цезаря. Этот класс используют модули Form1 и Form2 во время зашифровки / дешифровки, и взлома соответственно.

Программа использует функции следующих библиотек среды С#:

System, System.Collections.Generic, System.Linq, System.Text, System.Windows.Forms, System.IO.

Исполняемый файл программы создан средствами среды C#, имеет имя   
CezarHack.exe и размер 617 000 байт.

#### 2.4.3.1. Описание методов класса Login.cs

Класс Cezar является статическим и предназначен для организации работы с шифром Цезаря. Связность модуля функциональная (СС = 10), так как он направлен на решение одной проблемы – взлом шифра Цезаря.

Модуль содержит 5 полей: dictionary, alphabet, HeadAlphabet, pun и num. Также 5 методов:

public static string Coding(string Text,byte step) предназначен для шифрования текста;

public static string Decoding(string Coded\_text,byte step) предназначен для дешифрования текста;

public static string Vzlom\_Shifra\_Chastotnim(string Coded\_text) предназначен для взлома шифра частотным анализом;

public static void LoadDictionary() используется для загрузки текстового файла (словаря) в dictionary;

public static string Alphabet\_Attack(string Coded\_text) предназначен для взлома шифра при помощи сравнения слов из зашифрованного сообщения со словарем.

#### 2.4.3.2. Описание методов класса Form1.cs

Класс формы Form1.cs предназначен для мануальной работы с шифром цезаря, данная форма является главной. Связность модуля информационная (СС=9), так как действия внутри модуля связаны с данными.

Модуль содержит 9 методов:

private void взломToolStripMenuItem\_Click(object sender, EventArgs e) предназначен для перехода на Form2;

private void Kluch\_KeyPress(object sender, KeyPressEventArgs e) метод обработки события при вводе данных в поле Ключ;

private void Coding\_Click(object sender, EventArgs e) метод, для зашифровки сообщения шифром Цезаря на указанное значение поля Ключ;

private void Decoding\_Click(object sender, EventArgs e) метод для дешифровки сообщения шифром Цезаря на указанное значение поля Ключ;

private void ИсходныйТекстОткрыть\_Click(object sender, EventArgs e) метод для открытия не зашифрованного текстового файла;

private void ЗашифрованныйТекстОткрыть\_Click(object sender, EventArgs e) метод для открытия зашифрованного текстового файла;

private void ИсходныйТекстСохранить\_Click(object sender, EventArgs e) метод для сохранения расшифрованного текста;

private void ЗашифрованныйТекстСохранить\_Click(object sender, EventArgs e) метод для сохранения зашифрованного текста;

private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e) предназначен для открытия справки о программе.

#### 2.4.3.3. Описание методов класса Form2.cs

Класс формы Form2.cs предназначен для автоматизированной работы с шифром цезаря, данная форма является побочной. Связность модуля информационная (СС=9), так как действия внутри модуля связаны с данными.

Модуль содержит 7 методов:

private void Form2\_FormClosing(object sender, FormClosingEventArgs e) метод предназначенный для отображения Form1 при закрытии данной;

private void ВернутьсяНаГлавную\_Click(object sender, EventArgs e) предназначен для перехода на Form1;

private void ЗашифрованныйТекстОткрыть\_Click(object sender, EventArgs e) метод для открытия зашифрованного текстового файла;

private void ИсходныйТекстСохранить\_Click(object sender, EventArgs e) метод для сохранения расшифрованного текста;

private void ЗашифрованныйТекстСохранить\_Click(object sender, EventArgs e) метод для сохранения зашифрованного текста;

private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e) предназначен для открытия справки о программе;

private void Hack\_Click(object sender, EventArgs e) метод для взлома сообщения, зашифрованного шифром Цезаря.

### 2.4.4. Используемые технические и программные средства

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~130 Мб;
* объем необходимой памяти на жестком диске ~3Мб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь».

### 2.4.5. Вызов и загрузка

Программа может быть загружена как с диска, так и с жесткого диска. В последнем случае требуется предварительно переписать программу с диска на жесткий диск.

Исполняемым файлом программы является файл HackCezar.exe. Для его запуска необходимо дважды щелкнуть по исполняемому файлу левой кнопкой мышки.

## 2.5. Руководство оператора

### 2.5.1. Назначение программы

Основное назначение программного продукта заключается во взломе шифра Цезаря: человек вводит с клавиатуры либо загружает из файла исходный текст, далее он может его зашифровать, сдвинув все буквы текста на k позиций. Также зная ключ, может его расшифровать. На второй вкладке можно загрузить зашифрованный текст из файла и взломать его, программа сама подберет ключ и преобразует текст в нормальную форму

### 2.5.2. Выполнение программы и сообщения оператору

Для запуска программы дважды щелкните левой кнопкой мыши по исполняемому файлу HackCezar.exe.

Процесс работы приложения показаны в виде иллюстраций в приложении Б.

При запуске приложения загружается главная форма, на которой расположены поля для загрузки или ввода текста, также кнопки для шифрования и дешифрования текста. В верхней панели присутствуют кнопки для сохранения и загрузки текста, а также для перехода на форму для взлома зашифрованного текста.

На побочной форме для взлома текста также присутствуют поля для текста и кнопка взлома.

## 2.6. Программа и методика испытаний

### 2.6.1. Объект испытаний

Объектом испытаний является игровая программа HackCezar.exe. Человек вводит с клавиатуры либо загружает из файла исходный текст, далее он может его зашифровать, сдвинув все буквы текста на k позиций. Также зная ключ, может его расшифровать. На второй вкладке можно загрузить зашифрованный текст из файла и взломать его, программа сама подберет ключ и преобразует текст в нормальную форму

2.6.2. Цель испытаний

Целью испытаний является проверка соответствия программы требованиям Технического Задания.

### 2.6.3. Требования к программе

В процессе испытаний подлежат проверке следующие требования к программе:

#### 2.6.3.1. Требования к функциональным характеристикам

**1 версия:**

* программа должна поддерживать загрузку текста из файла;
* программа должна поддерживать ввод текста с клавиатуры
* программа должна обеспечивать возможность корректировки текста и повторной записи его в файл.
* программа должна шифровать текст, заданным ключом
* программа должна дешифровать текст, введенный человеком, с заданным ключом.

**2 версия:**

* программа должна взламывать текст, зашифрованный Шифром Цезаря с неизвестным ключом.

#### 2.6.3.2. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows 7 или совместимой. Язык интерфейса – русский.

#### 2.6.3.3. Требования к маркировке и упаковке

Программа должна поставляться на системном диске H: в виде исполняемого (еxе) файла, документации и проекта. Структура папок:

Корневой каталог: «Курсовой проект. Взлом Шифра Цезаря» с подкаталогами:

EXE (исполняемый файл);

Source (исходные файлы по каждой версии);

Vers 1;

Vers 2;

DOC (ТЗ и Пояснительная записка).

#### 2.6.3.4. Требования к транспортировке и хранению

Программа распространяется в электронном виде. Требования к транспортировке и хранению не предъявляются.

### 2.6.4. Требования к программной документации

На испытания должны быть представлены следующие программные документы:

- «Техническое задание»;

- разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии;

- разрабатываемое программное обеспечение должно включать справочную систему.

### 2.6.5. Средства и порядок испытаний

Для проведения испытаний необходимы следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~130 Мб;
* объем необходимой памяти на жестком диске ~3Мб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь»;

Для проведения испытаний необходимы следующие программные средства:

Операционная система Windows 7.

Испытания проводятся в следующем порядке:

1) проверяется наличие и комплектность программной документации (п.2.6.4);

2) проверяется соответствие требованиям к маркировке и упаковке (п.2.6.3.3);

3) проверяется соответствие требованиям к функциональным характеристикам (п.2.6.3.1);

4) проверяется соответствие требованиям к информационной и программной совместимости (п.2.6.3.2).

### 2.6.6. Методы испытаний

#### 2.6.6.1. Для проверки способности программы загружать не зашифрованный текст необходимо:

* запустить программу;
* нажать в верхней панели кнопку «Файл»;
* нажать кнопку «Открыть»;
* нажать кнопку «Исходный текст»;
* выбрать необходимый файл;
* нажать «Открыть»;
* убедиться, что файл загрузился в поле «Исходный текст».

#### 2.6.6.2. Для проверки способности программы загружать зашифрованный текст необходимо:

* запустить программу;
* нажать в верхней панели кнопку «Файл»;
* нажать кнопку «Открыть»;
* нажать кнопку «Зашифрованный текст»;
* выбрать необходимый файл;
* нажать «Открыть»;
* убедиться, что файл загрузился в поле «Зашифрованный текст».

#### 2.6.6.3. Для проверки способности программы сохранять зашифрованный текст необходимо:

* запустить программу;
* нажать в верхней панели кнопку «Файл»;
* нажать кнопку «Сохранить»;
* нажать кнопку «Зашифрованный текст»;
* выбрать необходимый файл;
* нажать «Сохранить»;
* убедиться, что в файл загрузился текст из поля «Зашифрованный текст».

#### 2.6.6.4. Для проверки способности программы сохранять не зашифрованный текст необходимо:

* запустить программу;
* нажать в верхней панели кнопку «Файл»;
* нажать кнопку «Сохранить»;
* нажать кнопку «Исходный текст»;
* выбрать необходимый файл;
* нажать «Сохранить»;
* убедиться, что в файл загрузился текст из поля «Исходный текст».

#### 2.6.6.5. Для проверки способности программы зашифровать текст, необходимо:

* запустить программу;
* ввести или загрузить в поле «Исходный текст» необходимый текст;
* ввести значение в поле Ключ (от 0 до 33);
* нажать кнопку «Зашифровать»;
* убедиться, что текст изменился на не понятный для человека и отобразился в поле «Зашифрованный текст».

#### 2.6.6.6. Для проверки способности программы расшифровать текст, необходимо:

* запустить программу;
* ввести или загрузить в поле «Зашифрованный текст» необходимый текст;
* ввести значение в поле Ключ (от 0 до 33);
* нажать кнопку «Расшифровать»;
* убедиться, что текст изменился на понимаемый для человека и отобразился в поле «Исходный текст».

#### 2.6.6.7. Для проверки способности программы взламывать текст не зная ключ, необходимо:

* запустить программу;
* нажать в верхней панели кнопку «Взлом»;
* ввести или загрузить в поле «Текст для взлома» необходимый текст;
* нажать кнопку «Взломать»;
* убедиться, что в поле «Результат взлома» отобразился понимаемый для человека текст;

## 2.7. Протокол испытаний

Результаты испытаний программы представлены в таблице 1, рисунки приведены в приложении Б.

**Результаты испытаний программы**

*Таблица 1*

| **№** | **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Результат** |
| --- | --- | --- | --- | --- |
| 1 | Способность программы загружать не зашифрованный текст | * запустить программу; * нажать в верхней панели кнопку «Файл»; * нажать кнопку «Открыть»; * нажать кнопку «Исходный текст»; * выбрать необходимый файл; * нажать «Открыть». | Файл загрузился в поле «Исходный текст». | Пройден  Рис. Б1 |
| 2 | Способность программы загружать зашифрованный текст | * запустить программу; * нажать в верхней панели кнопку «Файл»; * нажать кнопку «Открыть»; * нажать кнопку «Зашифрованный текст»; * выбрать необходимый файл; * нажать «Открыть». | файл загрузился в поле «Зашифрованный текст». | Пройден  Рис. Б2 |
| 3 | Способность программы сохранять зашифрованный текст | * запустить программу; * нажать в верхней панели кнопку «Файл»; * нажать кнопку «Сохранить»; * нажать кнопку «Зашифрованный текст»; * выбрать необходимый файл; * нажать «Сохранить». | В файл загрузился текст из поля «Зашифрованный текст». | Пройден  Рис. Б3 |
| 4 | Способность программы сохранять не зашифрованный текст | * запустить программу; * нажать в верхней панели кнопку «Файл»; * нажать кнопку «Сохранить»; * нажать кнопку «Исходный текст»; * выбрать необходимый файл; * нажать «Сохранить». | В файл загрузился текст из поля «Исходный текст». | Пройден  Рис. Б4 |
| 5 | Способность программы зашифровать текст | * запустить программу; * ввести или загрузить в поле «Исходный текст» необходимый текст; * ввести значение в поле Ключ (от 0 до 33); * нажать кнопку «Зашифровать». | Текст изменился на не понятный для человека и отобразился в поле «Зашифрованный текст». | Пройден  Рис. Б5– Б6 |
| 6 | Способность программы расшифровать текст | * запустить программу; * ввести или загрузить в поле «Зашифрованный текст» необходимый текст; * ввести значение в поле Ключ (от 0 до 33); * нажать кнопку «Расшифровать». | текст изменился на понимаемый для человека и отобразился в поле «Исходный текст». | Пройден  Рис. Б7– Б8 |
| 7 | Способность программы взламывать текст не зная ключ | * запустить программу; * нажать в верхней панели кнопку «Взлом»; * ввести или загрузить в поле «Текст для взлома» необходимый текст; * нажать кнопку «Взломать». | В поле «Результат взлома» отобразился понимаемый для человека текст. | Пройден  Рис. Б9-Б10 |

ЗАКЛЮЧЕНИЕ

Разработанная в ходе выполнения курсового проекта программа удовлетворяет всем требованиям технического задания, что подтверждается протоколом испытаний.

Разработанная программа может быть использована как в образовательных целях, так и в исследовательских для обучения и понимания основ криптографии.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд ., Язык программирования C#, 4-е издание, СПб:-ПИТЕР, 2020
2. Фленов М., Библия C#, 3-е издание, СПб:-БХВ, 2019
3. Культин Н., Microsoft Visual C# в задачах и примерах, 2-е издание, СПб:-БХВ, 2015
4. Описание языка C# [Электронный ресурс] — URL: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/ (дата обращения: 03.12.2024)
5. Стандарты Единой Системы Программной Документации:

ГОСТ 19.105-78 Общие требования к программным документам

ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом

ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению

ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению

ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 Описание программы. Требования к содержанию и оформлению

ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению

ГОСТ 19.701-90 Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

ПРИЛОЖЕНИЕ А

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования  
«Санкт-Петербургский политехнический университет Петра Великого»  
(ФГАОУ ВО «СПбПУ»)  
**Институт среднего профессионального образования**

УТВЕРЖДАЮ  
Председатель ПЦК   
\_\_\_\_\_\_\_\_\_\_\_ Андреев В.А.  
\_\_.\_\_. 2024

Программа для взлома шифра Цезаря  
**Техническое задание**  
Листов 6

ПРОВЕРИЛ  
Преподаватель   
\_\_\_\_\_\_\_\_\_\_\_ Ильин Ю.П.  
\_\_.\_\_. 2024

ВЫПОЛНИЛ  
Студент группы 329191/1   
\_\_\_\_\_\_\_\_\_\_\_ Шабанов Д.В  
\_\_.\_\_. 2024

2024

# ВВЕДЕНИЕ

* 1. Полное наименование программной разработки: Программа для взлома шифра Цезаря.
  2. Программа для взлома шифра Цезаря: человек вводит с клавиатуры либо загружает из файла исходный текст, далее он может его зашифровать, сдвинув все буквы текста на k позиций. Также зная ключ, может его расшифровать. На второй вкладке можно загрузить зашифрованный текст из файла и взломать его, программа сама подберет ключ и преобразует текст в нормальную форму.
  3. В соответствии с заданием программный продукт состоит из теоретической и практической частей. Теоретическая часть включает подробное описание работы с приложениями, показывающими схему работы системы и алгоритмы отдельных модулей. Практическая часть включает разработку и реализацию программных модулей программного продукта с использованием среды программирования Visual Studio 2017 и языка C#.
  4. Программа предназначена для взлома шифра Цезаря.

# ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1. Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем» МДК 01.01 «Разработка программных модулей» и утверждена Институтом среднего профессионального образования.

# НАЗНАЧЕНИЕ РАЗРАБОТКИ

* 1. Основное назначение программного продукта заключается во взломе шифра Цезаря.

4 ТРЕБОВАНИЯ К РАЗРАБОТКЕ

4.1 Требования к функциональным характеристикам:

**1 версия:**

* программа должна поддерживать загрузку текста из файла;
* программа должна поддерживать ввод текста с клавиатуры
* программа должна обеспечивать возможность корректировки текста и повторной записи его в файл.
* программа должна шифровать текст, заданным ключом
* программа должна дешифровать текст, введенный человеком, с заданным ключом.

**2 версия:**

* программа должна взламывать текст, зашифрованный Шифром Цезаря с неизвестным ключом.

4.2 Требования к надежности:

* использование лицензированного программного обеспечения;
* проверка программы на наличие вирусов;
* организация обработки исключительных ситуаций.

4.3 Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- процессор Intel\AMD или другой совместимый;

- объем свободной оперативной памяти ~ 130 Мб;

- объем необходимой памяти на жестком диске ~ 20Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- манипулятор «мышь».

4.4 Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы Microsoft Windows 10 или совместимой. Язык интерфейса – русский.

4.5 Требования к маркировке и упаковке

Программа должна поставляться на системном диске H: в виде исполняемого (еxе) файла, документации и проекта. Структура папок:

Корневой каталог: «Курсовой проект. Взлом Шифра Цезаря» с подкаталогами:

EXE (исполняемый файл);

Source (исходные файлы по каждой версии);

Vers 1;

Vers 2;

DOC (ТЗ и Пояснительная записка).

4.6 Требования к транспортировке и хранению

(Отсутствуют).

5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1 Предварительный состав программной документации:

- «Техническое задание»;

- разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии;

- разрабатываемое программное обеспечение должно включать справочную систему.

5.2 Перечень материалов пояснительной записки

Введение

1. Теоретические основы разработки
   1. Описание предметной области
   2. Анализ методов решения
   3. Обзор средств программирования
   4. Описание выбранного языка программирования
2. Практическая часть
   1. Постановка задачи
   2. Описание схем
   3. Текст программы
   4. Описание программы
   5. Руководство оператора
   6. Программа и методика испытаний
   7. Протокол испытаний

Заключение

Список использованных источников

Приложения

6 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели не рассчитываются.

7 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание стадии** | **Содержание этапа** | **Срок 2024 г.** | | **Форма  отчетности** |
| **начало** | **конец** |
| Техническое задание | Составление технического задания | 16.01 | 23.01 | Техническое задание |
| Эскизный проект | Разработка спецификаций | 24.01 | 08.02 | Спецификации программного обеспечения |
| Рабочий проект | Проектирование программы | 09.02 | 24.02 | Схема работы системы и спецификации компонентов |
| Составление программы | 25.02 | 20.03 | Программная документация |
| Приёмо-сдаточные испытания | 21.03 | 16.04 | Протокол испытаний (п. 2.7 пояснительной записки) |
| Приёмка | Защита курсового проекта | 17.04 | 24.04 | Оценка за курсовой проект |

8 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1 Порядок контроля

Контроль выполнения должен осуществляться руководителем курсового проекта (преподавателем) в соответствие с п.7.

8.2 Порядок приемки

Приемка должна осуществляться с участием руководителя после проведения приемо-сдаточных испытаний. В результате защиты курсового проекта должна быть выставлена оценка за курсовой проект.

ПРИЛОЖЕНИЕ Б



Рисунок Б1 Загрузка Исходного текста

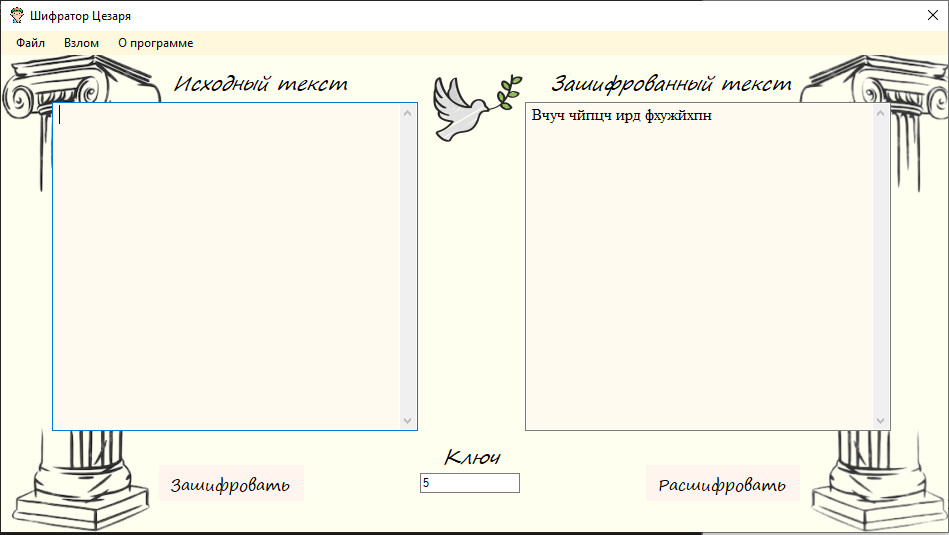


Рисунок Б2 Загрузка Зашифрованного текста

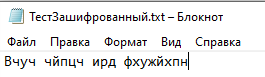


Рисунок Б3 Сохранение Зашифрованного текста

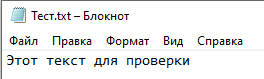


Рисунок Б4 Сохранение Исходного текста

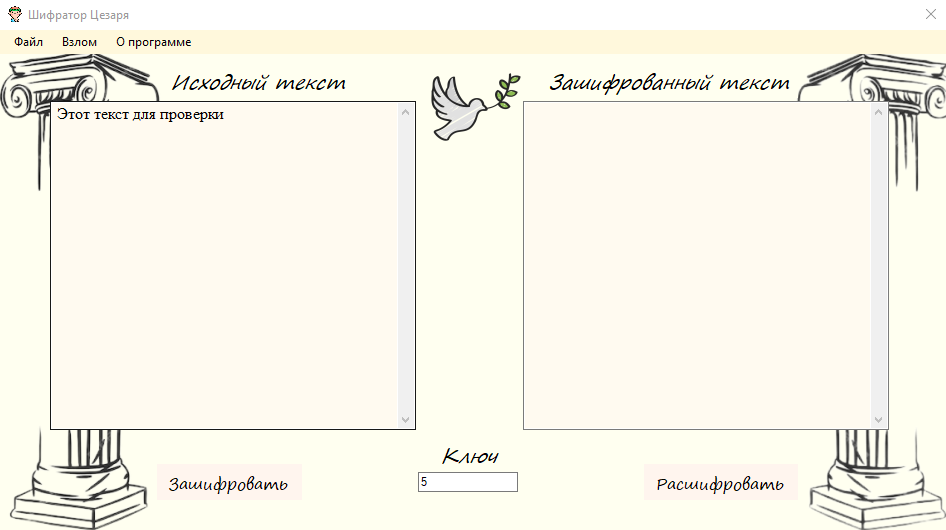


Рисунок Б5 Зашифровка текста

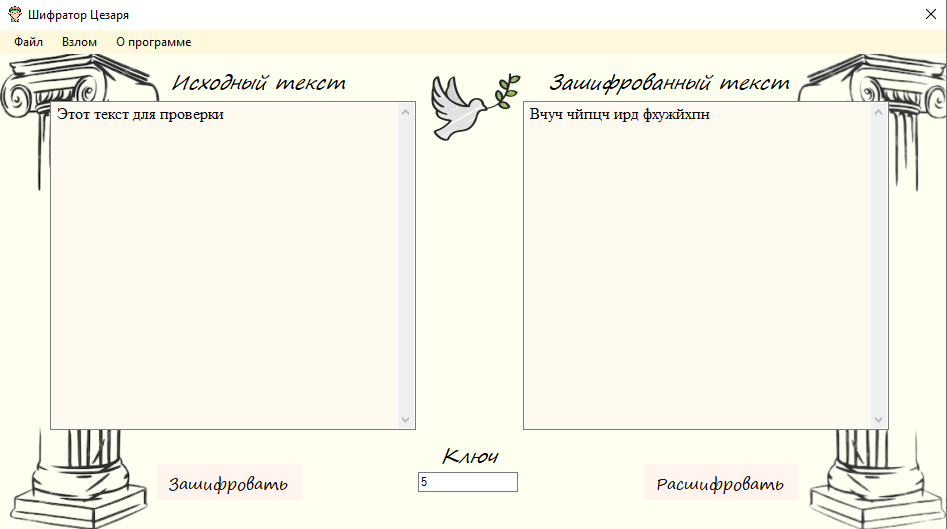


Рисунок Б6 Зашифровка текста

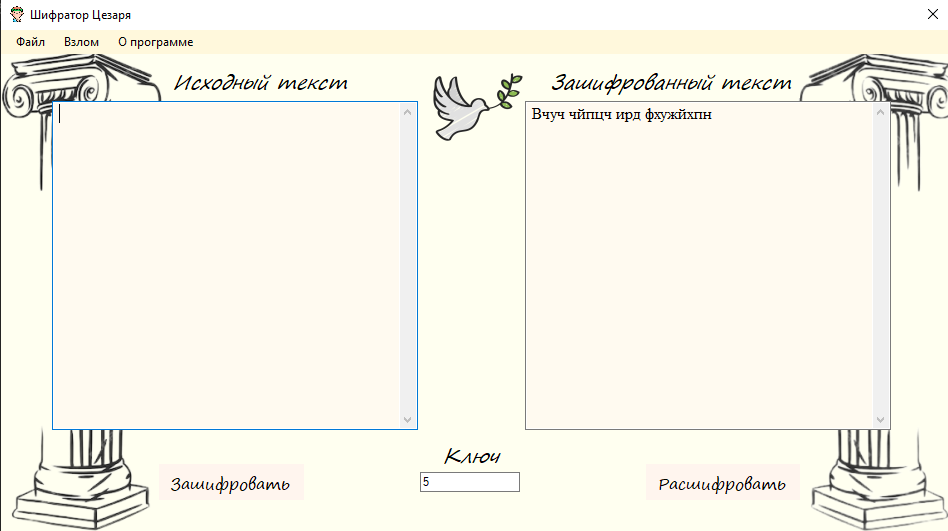
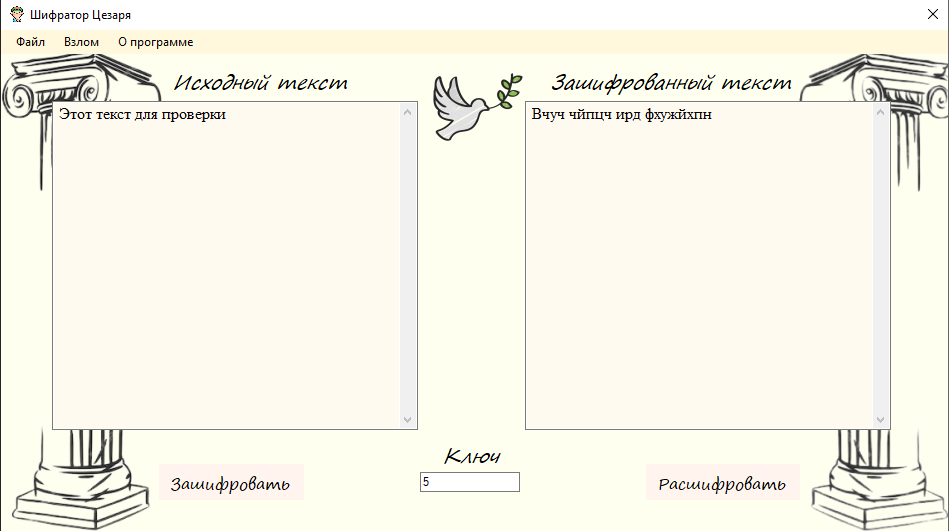
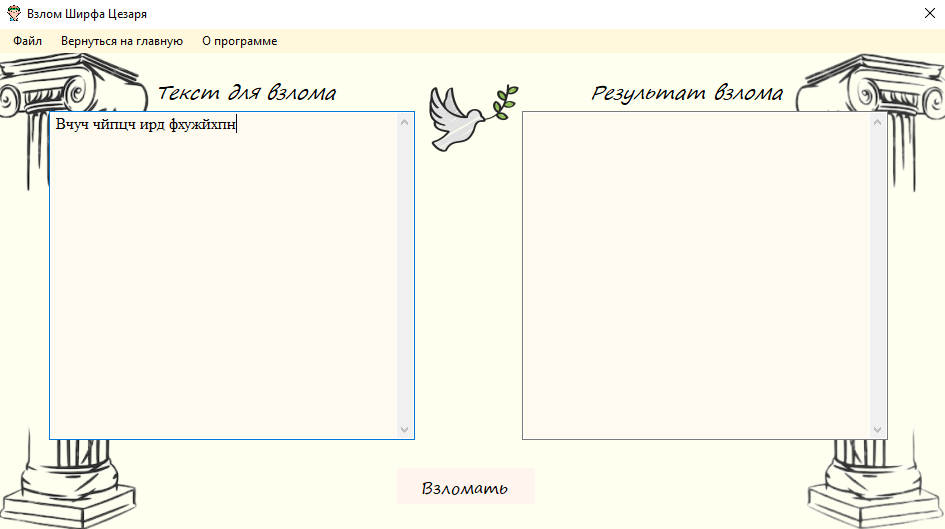


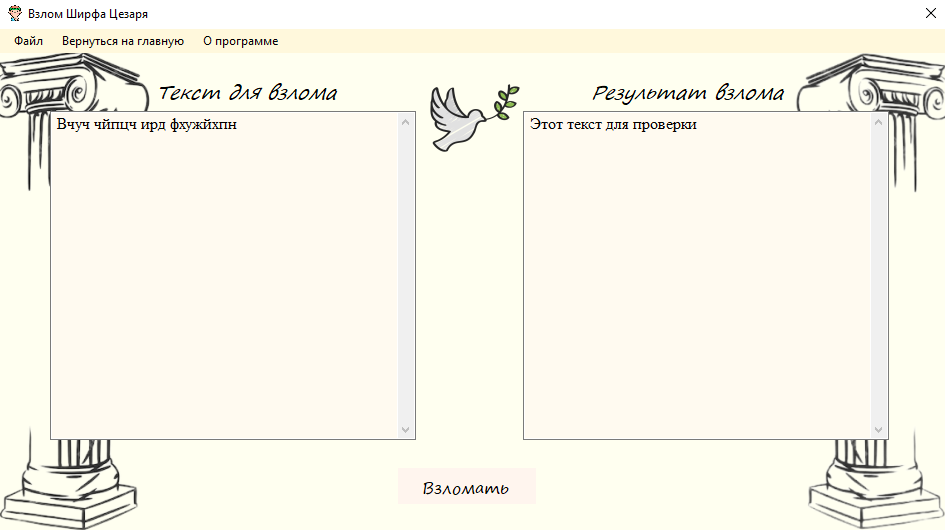
Рисунок Б7 Расшифровка текста



Б8 Расшифровка текста



Б9 Взлом зашифрованного текста



Б10 Взлом зашифрованного текста

ПРИЛОЖЕНИЕ В

**Текст программы**

## Модуль Form1.cs

using Lib\_Vzlom\_Cesarya;

using System;

using System.Windows.Forms;

namespace Forms\_Vzlom\_Shifra\_Cezarya

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

Cezar.LoadDictionary();

}

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

{

Form2 form2 = new Form2(this);

this.Hide();

form2.Show();

}

private void Kluch\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

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

{

if (SHAG.TextLength == 0)

{

MessageBox.Show("Поле \"Ключ\" не может быть пустым", "Ошибка!");

}

else if (Convert.ToByte(SHAG.Text) > 33)

{

MessageBox.Show("Поле \"Ключ\" не должно превышать размер Русского алфавита (33)", "Ошибка!");

}

else if (Ishodnoe.TextLength == 0)

{

MessageBox.Show("Поле исходного текста пустое", "Ошибка!");

}

else

{

Zashifrovannoe.Text = Cezar.Coding(Ishodnoe.Text, Convert.ToByte(SHAG.Text));

}

}

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

{

if (SHAG.TextLength == 0)

{

MessageBox.Show("Поле \"Ключ\" не может быть пустым", "Ошибка!");

}

else if (Convert.ToByte(SHAG.Text) > 33)

{

MessageBox.Show("Поле \"Ключ\" не должно превышать размер Русского алфавита (33)", "Ошибка!");

}

else if (Zashifrovannoe.TextLength == 0)

{

MessageBox.Show("Поле зашифрованного текста пустое", "Ошибка!");

}

else

{

Ishodnoe.Text = Cezar.Decoding(Zashifrovannoe.Text, Convert.ToByte(SHAG.Text));

}

}

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

{

OpenFileDialog openFile = new OpenFileDialog();

openFile.Title = "Открытие";

openFile.ShowDialog();

if (openFile.FileName != "")

{

Ishodnoe.Text = System.IO.File.ReadAllText(openFile.FileName);

}

}

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

{

OpenFileDialog openFile = new OpenFileDialog();

openFile.Title = "Открытие";

openFile.ShowDialog();

if (openFile.FileName != "")

{

Zashifrovannoe.Text = System.IO.File.ReadAllText(openFile.FileName);

}

}

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

{

SaveFileDialog saveFile = new SaveFileDialog();

saveFile.Title = "Сохранение";

saveFile.DefaultExt = "txt";

saveFile.ShowDialog();

if (saveFile.FileName != "")

{

System.IO.File.WriteAllText(saveFile.FileName, Ishodnoe.Text);

}

}

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

{

SaveFileDialog saveFile = new SaveFileDialog();

saveFile.Title = "Сохранение";

saveFile.DefaultExt = "txt";

saveFile.ShowDialog();

if (saveFile.FileName != "")

{

System.IO.File.WriteAllText(saveFile.FileName, Zashifrovannoe.Text);

}

}

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

{

About about = new About();

about.ShowDialog();

}

}

}

## Модуль Form2.cs

using System;

using System.Windows.Forms;

using Lib\_Vzlom\_Cesarya;

namespace Forms\_Vzlom\_Shifra\_Cezarya

{

public partial class Form2 : Form

{

Form FORM1;

public Form2(Form form1)

{

InitializeComponent();

FORM1 = form1;

}

private void Form2\_FormClosing(object sender, FormClosingEventArgs e)

{

FORM1.Show();

}

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

{

if (Shifr.TextLength == 0)

{

MessageBox.Show("Поле для взлома пустое, заполните его!","Ошибка!");

}

else

{

try

{

if (Shifr.Text.Length >= 600)

Results.Text = Cezar.Vzlom\_Shifra\_Chastotnim(Shifr.Text);

else

{

Results.Text = Cezar.Alphabet\_Attack(Shifr.Text);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

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

{

SaveFileDialog saveFile = new SaveFileDialog();

saveFile.Title = "Сохранение";

saveFile.DefaultExt = "txt";

saveFile.ShowDialog();

if (saveFile.FileName != "")

{

System.IO.File.WriteAllText(saveFile.FileName, Results.Text);

}

}

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

{

SaveFileDialog saveFile = new SaveFileDialog();

saveFile.Title = "Сохранение";

saveFile.DefaultExt = "txt";

saveFile.ShowDialog();

if (saveFile.FileName != "")

{

System.IO.File.WriteAllText(saveFile.FileName, Shifr.Text);

}

}

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

{

this.Dispose();

FORM1.Show();

}

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

{

OpenFileDialog openFile = new OpenFileDialog();

openFile.Title = "Открытие";

openFile.ShowDialog();

if (openFile.FileName != "")

{

Shifr.Text = System.IO.File.ReadAllText(openFile.FileName);

}

}

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

{

About about = new About();

about.Show();

}

}

}

## Модуль Cezar.cs

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

using System.Linq;

namespace Lib\_Vzlom\_Cesarya

{

public class Cezar

{

// Словарь, используемый для атаки на шифр Цезаря

private static Dictionary<char, List<string>> dictionary;

// Алфавиты для разных типов символов

static char[] alphabet = {'а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я' };

static char[] HeadAlphabet = { 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь', 'Э', 'Ю', 'Я' };

static char[] pun = {'.', ',', '(', ')', ':', ';', '@', '!', '?', '\*', '/', '|', '\'', '\"', '\\', '<', '>', '=', '+', '-', '\_', ' ', '#', '$', '%', '^', '&', '[', ']', '\n', '\r','№'};

static char[] num = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

// Метод шифрования текста шифром Цезаря

// Принимает на вход исходный текст (Text) и шаг сдвига (step).

// Возвращает зашифрованный текст.

public static string Coding(string Text,byte step)

{

try

{

StringBuilder CODED\_TEXT = new StringBuilder();

// Обработка каждого символа исходного текста

for (int i = 0; i < Text.Length; i++)

{

// Шифрование букв нижнего регистра

for (int j = 0; j < alphabet.Length; j++)

{

if (Text[i] == alphabet[j])

{

CODED\_TEXT.Append(alphabet[(j + step) % alphabet.Length]);

}

}

// Шифрование букв верхнего регистра

for (int j = 0; j < HeadAlphabet.Length; j++)

{

if (Text[i] == HeadAlphabet[j])

{

CODED\_TEXT.Append(HeadAlphabet[(j + step) % HeadAlphabet.Length]);

}

}

// Оставление символов пунктуации без изменений

for (int j = 0; j < pun.Length; j++)

{

if (Text[i] == pun[j])

{

CODED\_TEXT.Append(pun[(j)]);

}

}

// Шифрование цифр

for (int j = 0; j < num.Length; j++)

{

if (Text[i] == num[j])

{

CODED\_TEXT.Append(num[(j + step) % num.Length]);

}

}

}

return CODED\_TEXT.ToString();

}

catch

{

throw new Exception("Поле Шаг имеет некорректное значение");

}

}

// Метод дешифрования текста, зашифрованного шифром Цезаря

// Принимает на вход зашифрованный текст (Coded\_text) и шаг сдвига (step).

// Возвращает исходный текст.

public static string Decoding(string Coded\_text,byte step)

{

try

{

StringBuilder DECODED\_TEXT = new StringBuilder();

// Обработка каждого символа зашифрованного текста

for (int i = 0; i < Coded\_text.Length; i++)

{

// Дешифрование букв нижнего регистра

for (int j = 0; j < alphabet.Length; j++)

{

if (Coded\_text[i] == alphabet[j])

{

DECODED\_TEXT.Append(alphabet[(j - step + alphabet.Length) % alphabet.Length]);

}

}

// Дешифрование букв верхнего регистра

for (int j = 0; j < HeadAlphabet.Length; j++)

{

if (Coded\_text[i] == HeadAlphabet[j])

{

DECODED\_TEXT.Append(HeadAlphabet[(j - step + HeadAlphabet.Length) % HeadAlphabet.Length]);

}

}

// Оставление символов пунктуации без изменений

for (int j = 0; j < pun.Length; j++)

{

if (Coded\_text[i] == pun[j])

{

DECODED\_TEXT.Append(pun[(j)]);

}

}

// Дешифрование цифр

for (int j = 0; j < num.Length; j++)

{

if (Coded\_text[i] == num[j])

{

{

DECODED\_TEXT.Append(num[(j - (step % num.Length) + num.Length) % num.Length]);

}

}

}

}

return DECODED\_TEXT.ToString();

}

catch

{

throw new Exception("Поле Шаг имеет некорректное значение");

}

}

// Метод атаки на шифр Цезаря с использованием частотного анализа

// Принимает на вход зашифрованный текст (Coded\_text).

// Возвращает расшифрованный текст.

public static string Vzlom\_Shifra\_Chastotnim(string Coded\_text)

{

int counter = 0, max = 0;

char symbol = ' ';

byte Step=0;

// Подсчет частоты встречаемости каждой буквы в тексте

for (int i = 0; i < alphabet.Length; i++)

{

for (int j = 0; j < Coded\_text.Length; j++)

{

if (alphabet[i] == Coded\_text.ToLower()[j])

{

counter += 1;

}

}

// Определение символа с наибольшей частотой

if (counter > max)

{

max = counter;

symbol = alphabet[i];

counter = 0;

}

else

{

counter = 0;

}

}

// Определение шага сдвига для дешифровки

for (int i=0;i<alphabet.Length;i++)

{

if (alphabet[i]==symbol)

{

if (i>=15)

{

Step = Convert.ToByte(i - 15);

}

else

{

Step = Convert.ToByte((i - 15 + alphabet.Length) % alphabet.Length);

}

}

}

// Возвращение расшифрованного текста

return Decoding(Coded\_text,Step);

}

// Метод загрузки словаря из текстового файла для использования в атаке по словарю

public static void LoadDictionary()

{

dictionary = new Dictionary<char, List<string>>();

string[] lines = File.ReadAllLines("dictionary.txt", Encoding.UTF8);

foreach (char l in alphabet)

{

dictionary[l] = new List <string>();

}

foreach (string s in lines)

{

char firstletter = s[0];

dictionary[firstletter].Add(s);

}

}

// Метод атаки на шифр Цезаря с использованием словаря

// Принимает на вход зашифрованный текст (Coded\_text).

// Возвращает расшифрованный текст.

public static string Alphabet\_Attack(string Coded\_text)

{

string Coded\_text\_Checker = Coded\_text.ToLower();

foreach (char ch in Coded\_text\_Checker)

{

if (!alphabet.Contains(ch) && !num.Contains(ch) && !pun.Contains(ch))

{

throw new Exception("В строке присутствуют неизвестные символы");

}

}

byte step = 0;

string[] words = Coded\_text.Split(pun, StringSplitOptions.RemoveEmptyEntries);

string largestWordorigin = "";

string largestWordCash = "";

bool key = false;

foreach (string word in words)

{

if (word.Length > largestWordorigin.Length)

{

largestWordorigin = word;

}

}

largestWordorigin = largestWordorigin.ToLower();

while (key != true)

{

largestWordCash = "";

if (step == 33)

{

throw new Exception("Не найдено ни одного совпадения в словаре");

}

for (int i = 0; i < largestWordorigin.Length; i++)

{

for (int j = 0; j < alphabet.Length; j++)

{

if (largestWordorigin[i] == alphabet[j])

{

largestWordCash += alphabet[(j - step + alphabet.Length) % alphabet.Length];

}

}

}

foreach (var kvp in dictionary)

{

if (kvp.Key == largestWordCash[0])

{

foreach (string word in kvp.Value)

{

if (word == largestWordCash)

{

key = true;

break;

}

}

if (key == true)

{

break;

}

}

}

step++;

}

return Decoding(Coded\_text, Convert.ToByte(step - 1));

}

}

}