**МИНИСТЕРСТВО ОБРАЗОВАНИЯ НОВОСИБИРСКОЙ**

**ОБЛАСТИ**

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«НОВОСИБИРСКИЙ ПОЛИТЕХНИЧЕСКИЙ**

**КОЛЛЕДЖ»**

СПЕЦИАЛЬНОСТЬ 09.02.07 (230115)

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И

ПРОГРАММИРОВАНИЕ

**КУРСОВАЯ РАБОТА ПО УЧЕБНОЙ ПРАКТИКЕ**

**ТЕМА: СОЗДАНИЕ ПРОГРАММЫ ДЛЯ**

**ШИФРОВКИ/РАСШИФРОВКИ**

Студент: Толстыко Владимир

Группа: 121ИСП

Руководитель Гритчин И. В.

г. Новосибирск

2023 г.

**Содержание**

[**ВВЕДЕНИЕ**](#_Toc88424819)

[**1.** **Постановка задач** 5](#_Toc88424820)

[**2.** **Разработка** 6](#_Toc88424820)

[**3.** **Тестирование** 7](#_Toc88424821)

[**4.** **Внешний вид и функционал программы** 10](#_Toc88424822)

[**Заключение**](#_Toc88424824)

[**Список литературы**](#_Toc88424825)

**ВВЕДЕНИЕ**

Шифрование информации сегодня – как воздух. Вроде его не ощущаешь, но убери, и сразу станет плохо. Это такой же неотъемлемый процесс цифрового контента. Общаетесь вы в мессенджере, публикуете статьи, фото на сайте или отправляете электронное письмо боссу – в каждом случае будет задействован тот или иной алгоритм шифрования. Без шифрования любой ваш контент может стать достоянием общественности, а это не всегда хорошо. Корпоративная, личная информация, пароли, данные платежных средств – всё это может быть взломано и попасть в руки злоумышленников. Важные аспекты в хранении данных, будь то на внешних серверах или домашнем компьютере, – это прежде всего кодирования и шифрование.

**Определения и различия**

Кодирование– процесс преобразования доступной нам информации в информацию понятную компьютерную.

Шифрование– процесс изменения информации таким образом, чтобы её смогли получить только нужные пользователи.

Таким образом можно понять, что кодирование нужно для понимания кода компьютером, а шифрование для защиты информации от сторонних лиц.

Защита информации методом шифрования обладает рядом преимуществ:

* обеспечение целостности и блокировка корпоративных данных, хранящихся в электронном виде;
* охрана баз данных, почты и других систем от несанкционированного доступа;
* защита информации от копирования и обнародования;
* повышение уровня корпоративной этики за счет обеспечения безопасности обмена личными сообщениями.

Утечка данных может происходить при пересылке информации через интернет, и при копировании файлов сотрудниками, и при несанкционированном внедрении, и из-за неумышленных ошибок персонала. В любом из этих случаев шифрование данных в сети гарантирует их неизменность и полную безопасность, так как дешифровка для злоумышленников чаще всего оказывается просто невозможной.

**Краткая история шифрования информации**

Шифрование информации стало развиваться практически одновременно с появлением письменности. Представители древнейших цивилизаций в Месопотамии и Египте уже использовали различные виды кодирования записей. Одним из первых появился метод "Атбаш"(Рис. 1). Правило его шифрования состоит в замене i-й буквы алфавита буквой с номером n-i+1, где n — число букв в алфавите.

Шифр получил своё название в честь первой, последней, второй и предпоследней буквы Еврейского алфавита - «алеф», «тав», «бет», «шин». Такой шифр имеет низкую криптографическую стойкость, потому как алгоритм шифрования довольно прост. (более подробно в [[1](#_Список литературы)])

Криптографическая стойкость - способность криптографического алгоритма противостоять криптоанализу, или же простыми словами данный параметр отображает насколько просто дешифровать сообщение.

Ещё один вариант шифрования записей был придуман несколько позже в Римской империи. Это был «шифр сдвига», который применял Юлий Цезарь. (более подробно в [[1](#_Список литературы)]). Его-то прозвали, как и изобретателя - Шифр Цезаря.

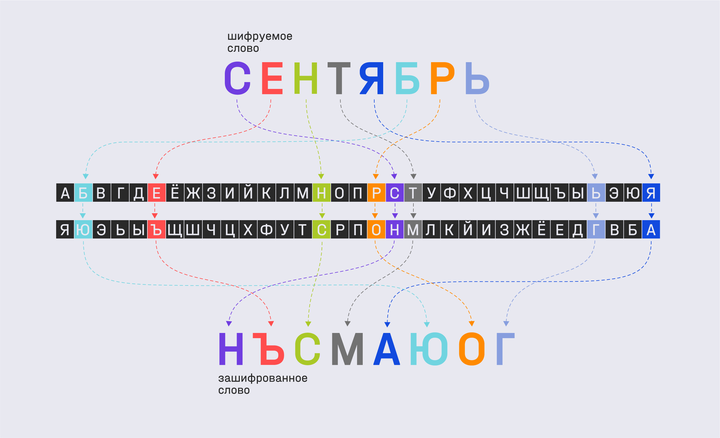


Рис. 1

1. **Постановка задач**

Создать программу, которая будет шифровать и расшифровывать сообщение по методу Цезаря. Программа будет выполнена на языке программирования C# с использованием интерфейса программирования приложений WinForms. Её функционал должен выполнять шифровку и дешифровку сообщения пользователя по его выбору. Программа будет очень простой и понятной для любого пользователя. На данный момент поставлена задача разработать интерфейс, с которым будет взаимодействовать пользователь, и код для шифровки/дешифровки сообщений.

1. **Разработка**

Был разработан код, выполняющий шифровку и расшифровку по методу Цезаря. Был изучен способ, по какому принципу можно написать код, для выполнения нужного функционала. В данном коде есть возможность задать нужное сообщение и после зашифровать его или расшифровать. Это изменённое программой сообщение показывается в второй текстовой строке с названием "Output". В коде также используются операции над переменными типа string.

Изначально я хотел сделать чтобы после шифровки буквы, она записывалась в отдельный массив типа string, но я столкнулся с проблемой. Тип данных string либо должен иметь фиксированный размер массива, либо иметь сразу заданный массив. Но каждое сообщение может быть уникальным по размеру. Сначала я попробовал задать размер массива при помощи переменной типа int, задав ей значение, равное длине сообщения в строке с названием "Input", но столкнулся с проблемой, что инициализация таким образом невозможна, либо я не понял, как это сделать. Следующее что мне пришло на ум — это сделать длину массива равной 99. Но это бы являлось костылём, так что продолжил искать решение. Потратив какое-то количество времени, я наткнулся на класс List.

**Костыль** — в программировании, быстрое «уродливое» решение проблемы, иначе требующей долгосрочного и ресурсоемкого исправления.

**Класс** — в программировании, представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализация поведения функций или методов.

После изучения его работы я понял, что он подходит для моих целей, так-как может взаимодействовать как тип данных string. После исправления проблемы, при которой я не знал, как записывать зашифрованное слово, я наткнулся на ещё одну проблему. Хоть классу List и не требовался размер, но при попытке создания новой ячейки с данными, компилятор выдавал ошибку. Вскоре я нашёл способ её исправления. Оказалось, что нельзя сразу создать ячейку с записанными данными. Нужно было сначала создать ячейку, и только потом присваивать ей значение. Конечно, есть возможность что я не до конца понял, как работает этот класс, но на решение данной ошибки у меня ушло достаточно много времени на поиск ответа в интернете.

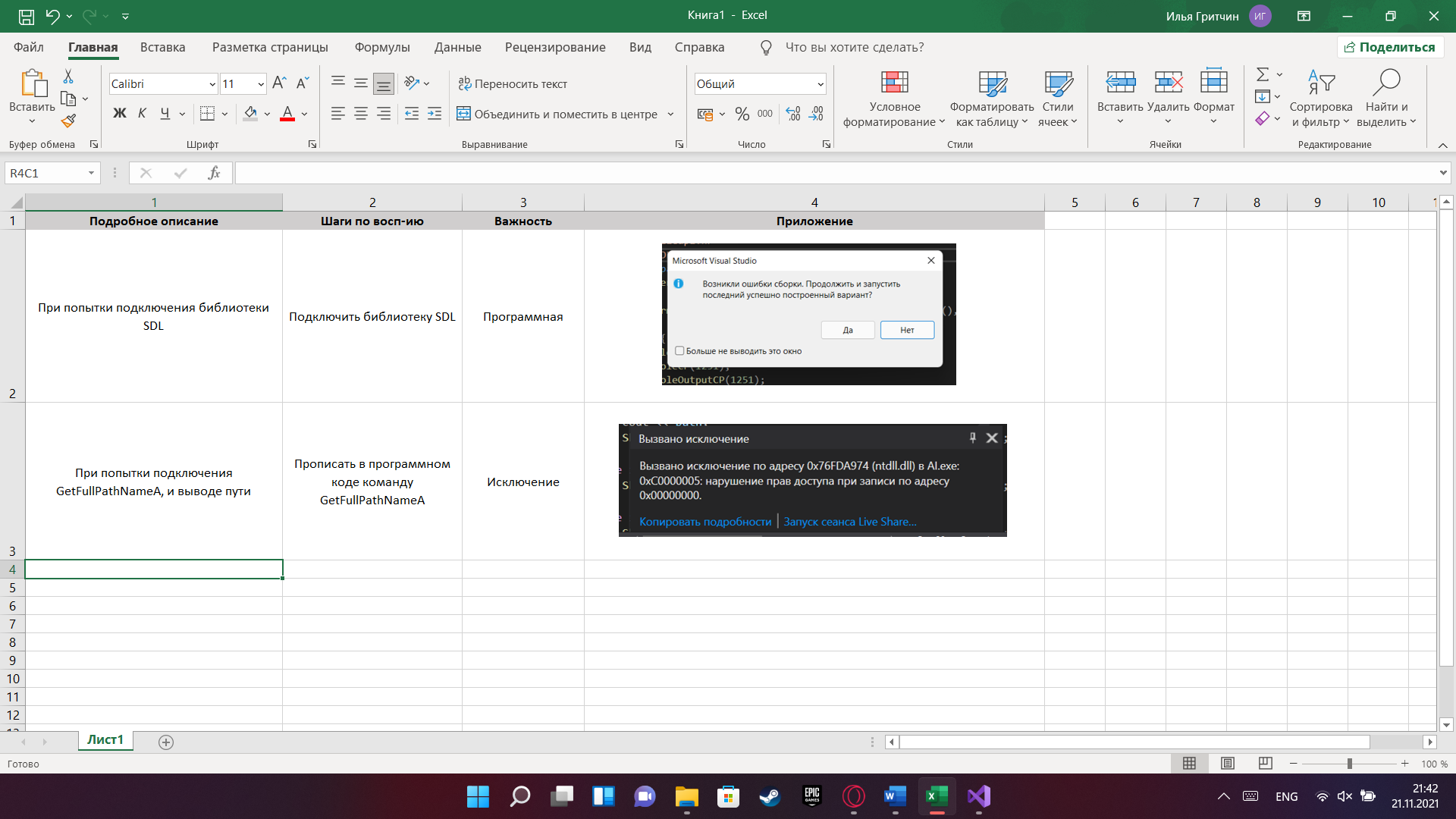
1. **Тестирование**

Тестирование (testing) программного обеспечения (ПО) – это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом. (более подробно в [[2](#_Список_литературы)])

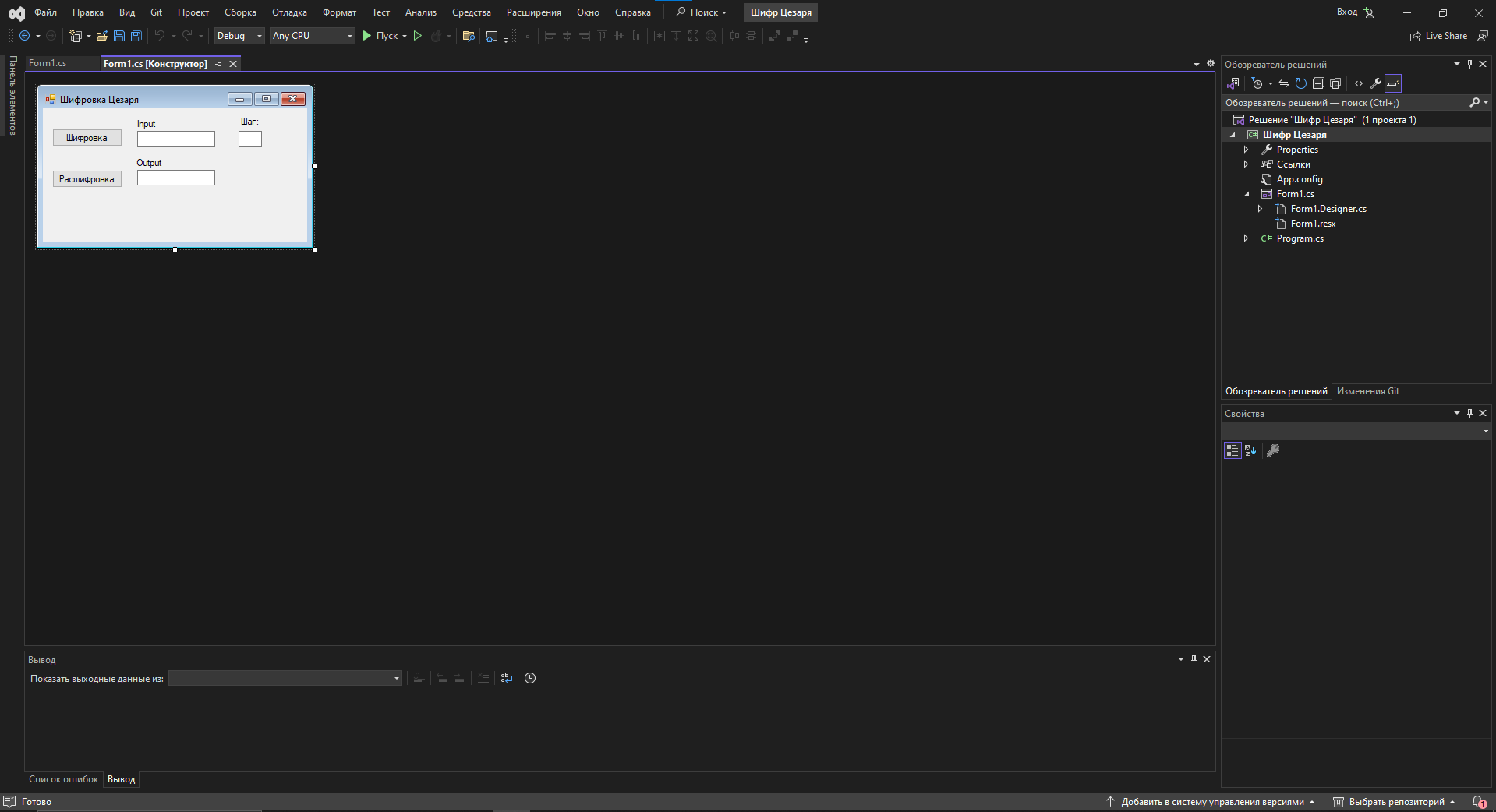
**Уровни тестирования:**

* **Модульное тестирование** – это процесс исследования ПО, при котором тестируется минимально возможный компонент, например, отдельный класс или функция. Часто модульное тестирование осуществляется разработчиками ПО.
* **Интеграционное тестирование** – это процесс исследования ПО, при котором тестируется интерфейсы между компонентами или подсистемами.
* **Системное тестирование** – это процесс исследования ПО, при котором тестируется интегрированная система на её соответствие требованиям заказчика. Альфа и Бета тестирование относятся к подкатегориям системного тестирования. (более подробно в [[2](#_Список_литературы)])

**Классификация видов тестирования:**

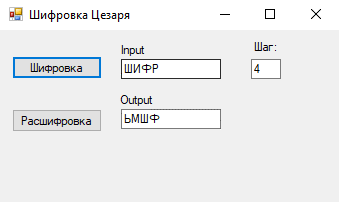
* **Функциональное тестирование (functional testing)** – тестирование ПО, направленное на проверку реализуемости функциональных требований. При функциональном тестировании проверяется способность ПО правильно решать задачи, необходимые пользователям.
* **Тестирование производительности (performance testing)** – тестирование ПО, позволяющее осуществлять оценку быстродействия программного продукта при определённой нагрузке.
* **Нагрузочное тестирование (load testing)** – тестирование ПО, позволяющее осуществлять оценку быстродействия программного продукта при плановых, повышенных и пиковых нагрузках.
* **Стресс-тестирование (stress testing)** – тестирование ПО, которое оценивает надёжность и устойчивость системы в условиях превышения пределов нормального функционирования.
* **Тестирование стабильности (stability/endurance/soak testing)** – тестирование ПО, при котором проверяется работоспособность ПО при длительном тестировании со среднем уровнем нагрузки.
* **Тестирование безопасности (security testing)** – тестирование ПО, которое проверяет фактическую реакцию защитных механизмов, встроенных в систему на проникновение злоумышленников.
* **Тестирование совместимости (compatibility testing)** - тестирование ПО, которое проверяет работоспособность ПО в определенном окружении.
* **Тестирование чёрного ящика (black box)** - тестирование ПО, при котором тестировщик имеет доступ к ПО только через интерфейсы заказчика, либо через внешние интерфейсы, позволяющие другому компьютеру или процессу подключиться к системе для тестирования.
* **Тестирование белого ящика (white box)** - тестирование ПО, при котором тестировщик имеет доступ к исходному коду программы и может писать код, связанный с библиотеками тестируемого ПО.
* **Альфа-тестирование** – это процесс имитации реальной работы разработчиков с программным продуктом, или реальная работа потенциальных пользователей с системой.
* **Бета-тестирование** – это распространение версий с ограничениями для некоторой группы лиц, с целью проверки содержания допустимо минимального количества ошибок в программном продукте.
* **Регрессионное тестирование (regression testing)** – тестирование ПО, при котором проводится проверка ранее найденных ошибок, а также проверка основной функциональности.
* **Дымовое тестирование (smoke testing)** - тестирование ПО, при котором выполняется набор тестов, после которого можно сказать, что программный продукт запускается.
* **Ручное тестирование (manual testing)** – тестирование, при котором не используются программные средства для выполнения тестов и проверки результатов выполнения.
* **Автоматизированное тестирование (automated testing)** – тестирование, при котором используются программные средства для выполнения тестов и проверки результатов выполнения.
* **Динамический анализ кода (runtime analysis)** – способ анализа программы непосредственно при ее выполнении.
* **Статический анализ кода (static analysis)** - анализ программы, производимый без реального выполнения исследуемых программ. (более подробно в [[2](#_Список_литературы)])

**3. Внешний вид и функционал программы**

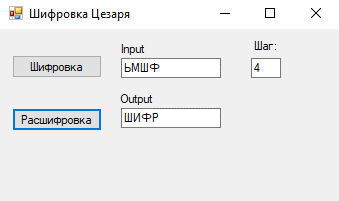


Ф1. Внешний вид программы.

  
Ф2. Запуск программы



Ф3. Работа кнопки "Шифровка"



Ф4. Работа кнопки "Расшифровка"

**Заключение**

Я написал программу на языке программирования C#. В его задачи вошли такие функции как шифровка и расшифровка сообщения по методу Цезаря. Также хочу сказать, что программа, написанная мной, работает нормально, но в будущем хотелось бы сделать выбор шифровки между несколькими шифрами.

В заключении хочу сказать, что в наше время шифровка сообщений практически неотъемлема. Без неё невозможно бы было передавать и хранить информацию, не опасаясь того, что она попадёт не в те руки. В будущем будут более сложные методы шифрования, которые безоговорочно усилят защиту данных от третьих лиц.

В наш век очень ценна информация. Вряд ли Вы хотите, чтобы Ваше сообщение прочитал кто-то другой, а не получатель. Многие передают в сообщениях очень много конфиденциальной и ценной информации. Как раз, чтобы эта информация не попала злоумышленнику, большинство сервисов обмена сообщений поддерживают шифрование. А самые безопасные — шифрование с закрытым ключом. Было издано много технической литературы на эту тему, а также много материалов в Интернете.

**Список литературы**

1. Кодирование и шифрование [Электронный ресурс] URL: <https://habr.com>
2. Тестирование ПО [Электронный ресурс] URL: [https://pvs-studio.com](https://pvs-studio.com/ru/blog/terms/0093/)