Содержание

Введение

Проблема защиты информации путем её преобразования волновала человечество с давних времен. С распространением письменности появилась потребность в обмене письмами и сообщениями, что вызвало необходимость сокрытия их содержимого от посторонних. Так начала формироваться наука под названием криптография и именно поэтому её можно считать ровесницей истории человеческого языка.

Первые криптосистемы встречаются уже в начале нашей эры. В основном развитию криптографии способствовали войны. Письменные приказы и донесения обязательно шифровались, чтобы пленение курьеров не позволило противнику получить важную информацию. Так, древнеримский политический деятель и полководец Гай Юлий Цезарь в своих переписках активно использовал уже более-менее систематический шифр, получивший его имя.

Бурное развитие криптографические системы получили в годы первой и второй мировых войн. Начиная с послевоенного времени и по нынешний день появление вычислительных средств ускорило разработку и совершенствование криптографических методов.

Почему проблема использования криптографических методов стала в настоящий момент особенно актуальна?

На сегодняшний день криптография является одним из наиболее мощных средств обеспечения конфиденциальности и контроля целостности информации. Во многих отношениях она занимает центральное место среди программно-технических регуляторов безопасности. Например, для портативных компьютеров, физически защитить которые крайне трудно, только криптография позволяет гарантировать конфиденциальность информации даже в случае кражи. Что уж говорить про важность криптографии в глобальной сети «Интернет», по которой ежедневно передаются колоссальные объемы информации государственного, военного,

коммерческого и частного характера, которая нуждается в защите от доступа к ней посторонних лиц.

Переоценить возможности криптографии для человечества сложно. С момента появления она прошла множество модификаций и сейчас представляет собой систему безопасности, которая практически не может быть взломана. Современные методы криптографии применяются практически во всех отраслях, в которых присутствует необходимость безопасной передачи или хранения данных.

Целью выпускной квалификационной работы является разработка программы шифрования файлов (и их дешифровки соответственно), которая позволила бы пользователю защитить желаемую информацию наиболее эффективными методами.

Для достижения поставленной цели необходимо выполнить ряд задач:

- детально изучить предметную область;
- провести сравнительный анализ и на его основе выбрать наиболее эффективные алгоритмы шифрования;
- разработать функциональные и нефункциональные требования к программе;
- составить необходимые диаграммы и схемы;
- выбрать подходящие для разработки инструментальные средства;
- реализовать функциональные и нефункциональные требования;
- провести тестирование и отладку программы;
- составить руководства по использованию программы.

Исходя из написанного выше можно сказать что объектом исследования будет являться – криптография, а предметом исследования – математические методы кодирования информации в современной криптографии.

Глава 1 Теоретическая часть

1.1 Предпроектное обследование

1.1.1 Описание предметной области

Криптография – это наука, изучающая способы сокрытия данных и обеспечения их конфиденциальности. Это одна из старейших наук и ее история насчитывает четыре тысячелетия. Сам термин «криптография» образовался от двух древнегреческих слов «крипто» – скрытый, «графо» – пишу.

До 1975 года криптография представляла собой шифровальный метод с секретным ключом, который предоставлял доступ к расшифровке данных. Позже начался период ее современного развития и были разработаны методы криптографии с открытым ключом, которые может передаваться по открытым каналам связи и использоваться для проверки данных.

Современная прикладная криптография представляет собой науку, образованную на стыке математики и информатики. Смежной наукой криптографии считается криптоанализ. Криптография и криптоанализ тесно взаимосвязаны между собой, только в последнем случае изучаются способы расшифровки сокрытой информации.

С модификацией до открытого ключа криптография получила более широкое распространение и стала применяться частными лицами и коммерческими организациями, а в 2009 году на ее основе была выпущена первая криптовалюта Биткоин. До этого времени она считалась прерогативой государственных органов правления.

В основе криптографических систем лежат различные виды криптографии. Всего различаю четыре основных криптографических примитива [40]:

- симметричное шифрование – метод предотвращает перехват данных третьими лицами и базируется на том, что отправитель и получатель данных имеет одинаковые ключи для разгадки шифра;

- асимметричное шифрование – в этом методе задействованы открытый и секретный ключ. Ключи взаимосвязаны – информация, зашифрованная открытым ключом, может быть раскрыта только связанным с ним секретным ключом. Применять для разгадки ключи из разных пар невозможно, поскольку они связаны между собой математической зависимостью;
- хэширование – метод основывается на преобразовании исходной информации в байты заданного образца. Преобразование информации называется хэш-функцией, а полученный результат хэш-кодом. Все хэш-коды имеют уникальную последовательность символов;
- электронная подпись – это преобразование информации с использованием закрытого ключа, позволяющее подтвердить подлинность документа и отсутствие искажений данных.

Изначально криптография использовалась правительством для безопасного хранения или передачи документов. Современные же асимметричные алгоритмы шифрования получили более широкое применение в сфере IT-безопасности, а симметричные методы сейчас применяются преимущественно для предотвращения несанкционированного доступа к информации во время хранения.

В частности, криптографические методы применяются для: безопасного хранения информации коммерческими и частными лицами, реализации систем цифровой электронной подписи, подтверждения подлинности сертификатов, защищенной передачи данных онлайн по открытым каналам связи [32].

1.1.2 Изучение аналогов

На мировом рынке существует множество программ по шифрованию файлов самыми различными способами. При поиске информации о них мне удалось выделить такие программы как:

- Pretty Good Privacy (PGP) Desktop;
- Folder Lock.

Начнём с PGP Desktop. Она представляет собой комплекс программ для шифрования, обеспечивающий гибкое многоуровневое шифрование. От Folder Lock, которая будет описана позднее, она отличается тем, что имеет тесную интеграцию в системную оболочку, а доступ к её функциям осуществляется через контекстное меню проводника. Программа позволяет выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, которая предоставляется в электронном виде, а также шифровать информацию на съёмных носителях.

Шифрование PGP осуществляется последовательно: хешированием, сжатием данных, шифрованием с симметричным ключом, и, наконец, шифрованием с открытым ключом, причём каждый этап может осуществляться одним из нескольких поддерживаемых алгоритмов. Такой подход позволяет достичь сильной защищённости информации.

К сожалению, PGP Desktop имеет низкую производительность и отсутствие поддержки русского языка [20].

Программа Folder Lock работает почти по тем же принципам. Заходя в программу сразу же хочется отметить простой и удобный интерфейс, который позволяет быстро начать работу. Используя эту программу можно:

- скрыть конфиденциальные данные в папках, на съёмных носителях и электронике;
- зашифровать конфиденциальные данные в папках, на съёмных носителях и электронике;
- устанавливать пароли;
- хранить важную информацию в «облаке» на сайте производителя;
- создавать виртуальные зашифрованные диски;
- шифрование сообщений электронной почты;
- полное удаление файлов без какой-либо возможности восстановления;
- другое.

Как видно, возможностей у программы предостаточно (особенно для персонального использования). Однако также, как и в PGP Desktop отсутствует поддержка русского языка, поэтому для пользователей не знакомых с английским, могут возникнуть некоторые сложности при работе [17].

Подводя итоги изучения аналогов можно сказать что в разрабатываемой программе должно в обязательном порядке присутствовать русский язык и разные алгоритмы шифрования файлов, при работе которых программа будет иметь высокую производительность.

1.1.3 Описание методов шифрования и принципов их работы

1.1.3.1 Транспозиция

Первым методом шифрования рассмотрим транспозицию или как его по-другому называют – шифр перестановки. Он относится к симметричным криптосистемам перестановочного типа. Принцип работы этого метода заключается в том, что элементы открытого текста меняются местами. Элементами текста могут быть как отдельные символы, так и их пары, тройки и так далее, а также комбинирование этих случаев [36].

Классическая криптография делит шифры перестановки на два класса:

- шифры простой перестановки – когда при шифровании символы открытого текста перемещаются с исходных позиций на новые один раз;
- шифры сложной перестановки – когда при шифровании символы открытого текста перемещаются с исходных позиций на новые несколько раз.

В разрабатываемой программе применяется первый класс подобных шифров – простой.

Как правило, при шифровании и дешифровании текста используется таблица перестановок, для примера, возьмём такую (рис. 1.1).

1	2	3	4	5
2	4	5	1	3

Рис. 1.1 – Таблица перестановок

Первая строка – номера символов в открытом тексте, вторая строка – номера позиций, которые должны занимать символы в шифрограмме (или проще говоря – это наш ключ, который задаёт порядок перестановки символов открытого текста).

Соответственно, для того, чтобы зашифровать исходное сообщение нужно разбить его на блоки, равные длине ключа. Как говорилось выше, кодирование осуществляется перестановкой букв (символов). Таким образом первый символ должен быть переставлен на второе место, второй на четвёртое, третий на пятое, четвёртый на первое, пятый на третье.

Если с помощью таблицы на рис. 1.1 зашифровать текст «пример маршрутной перестановки» получиться таблица, продемонстрированная на рис. 1.2.

Ключ →	2	4	5	1	3	
	п	р	и	м	е	← Блок 1
	р		м	а	р	← Блок 2
	ш	р	у	т	н	← Блок 3
	о	й		п	е	← Блок 4
	р	е	с	т	а	← Блок 5
	н	о	в	к	и	← Блок 6

Рис. 1.2 – Текст, зашифрованный методом транспозиции в табличном виде

В процессе шифрования символы текста переставляются в порядке, соответствующем заданному ключу по блокам (в данном примере по возрастанию) в результате чего получается следующая шифрограмма: «мпериарр мтшнрупоей траескниов».

Дешифрование производится в обратном порядке. На примере указанного ранее ключа: второй символ на первое, четвёртый на второе, третий на пятое и так далее.

Следует отметить, что при использовании любого блочного шифра могут возникать ситуации, когда текст не делиться на равные блоки. В таких случаях длину исходного текста увеличивают до тех пор, пока он не будет делиться на равные блоки длины заданного ключа.

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования так как символы всего лишь переставляются на другие позиции.

Минусами этого метода можно считать сохранение частотных характеристик текста и малое количество возможных ключей шифрования, что делает его уязвимым к криптоатакам. Главным недостатком этого и других симметричных алгоритмов шифрования можно считать передачу ключа. Ведь для того, чтобы он не попал в чужие руки для его передачи требуется обеспечить дополнительную безопасность, его также требуется регулярно обновлять, а после его смены опять же возникает нужда в его безопасной передаче.

1.1.3.2 Моноалфавитный шифр

Следующий метод шифрования – моноалфавитный шифр или по-другому – шифр простой замены. Он относится к симметричным криптосистемам подстановочного типа. К этому типу относятся, наверное, самый известный шифр – шифр Цезаря, в котором каждый символ алфавита сдвигается на три позиции правее. Принцип работы подстановочных шифров сводится к созданию таблицы шифрования (по определённому алгоритму), в которой каждой букве открытого текста соответствует единственная сопоставимая ей буква шифротекста. Само же шифрование заключается в замене букв согласно созданной таблице. Как можно заметить, всё очень просто [50].

Отмечу также, что в шифрах замены не всегда подразумевается замена буквы на какую-то другую букву. Допускается использовать замену на число. Соответственно в создаваемой таблице каждой букве используемого алфавита приравнивается любое число.

В разрабатываемой программе используется вариант замены буквы на букву, поэтому рассмотрим принцип работы на основе шифра Цезаря.

Для упрощения воспользуемся русским алфавитом, состоящим только из заглавных букв. Следуя принципу работы описанному выше создадим

таблицу шифрования, где каждая буква алфавита сдвигается на три позиции вправо. Из этого следует, что выбранный нами ключ равен числу три (рис. 1.3).

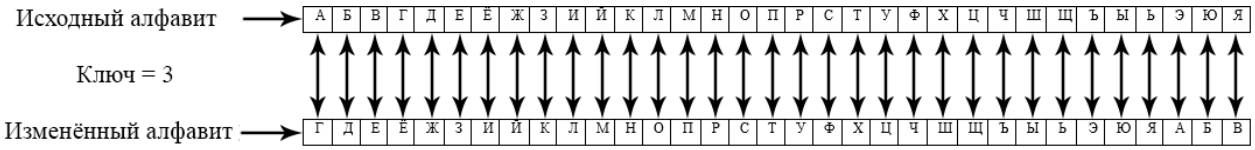

Рис. 1.3 – Исходный алфавит и алфавит, сдвинутый на три позиции

Теперь, глядя на рисунок мы чётко видим, как будут взаимозаменяться буквы при шифровании и дешифровании текста. При шифровании буква «А» будет заменена на «Г», «Б» на «Д» и так далее, а при дешифровании обратно. Таким образом, зашифровав сообщение «ШИФР ЦЕЗАРЯ» мы получим «ЫЛЧУ ЩЗКГУВ».

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования.

Минусами этого метода можно считать то, что в шифротексте не скрывается частота появления символов открытого текста, что делает его уязвимым к криптоатакам и то, что максимальный количество ключей равно количеству букв в используемом алфавите. Также здесь имеется проблема передачи больших объёмов текста (чем больше текст, тем легче его взломать). Так как алгоритм относится к симметричным опять же требуется дополнительная безопасность при передаче ключа и его регулярное обновление.

1.1.3.3 Полиалфавитный шифр

Рассмотрим метод шифрования – полиалфавитный шифр или по-другому – многоалфавитный шифр.

Как и моноалфавитный шифр он относится к симметричным криптосистемам подстановочного типа принцип работы которых был описан ранее.

Суть работы полиалфавитного шифра заключается в циклическом применении нескольких моноалфавитных шифров к некоторому количеству букв открытого текста [50].

Рассмотрим, как это работает на примере. Для упрощения воспользуемся русским алфавитом, состоящим только из заглавных букв. Пусть в качестве ключа будет использоваться слово «КЛЮЧ». Это слово мы можем поделить на четыре отдельных моноалфавита в каждом из которых будет произведён сдвиг на количество символов равное номеру отдельной буквы ключевого слова в алфавите. Сделав это, мы получим следующие моноалфавиты (рис. 1.4).

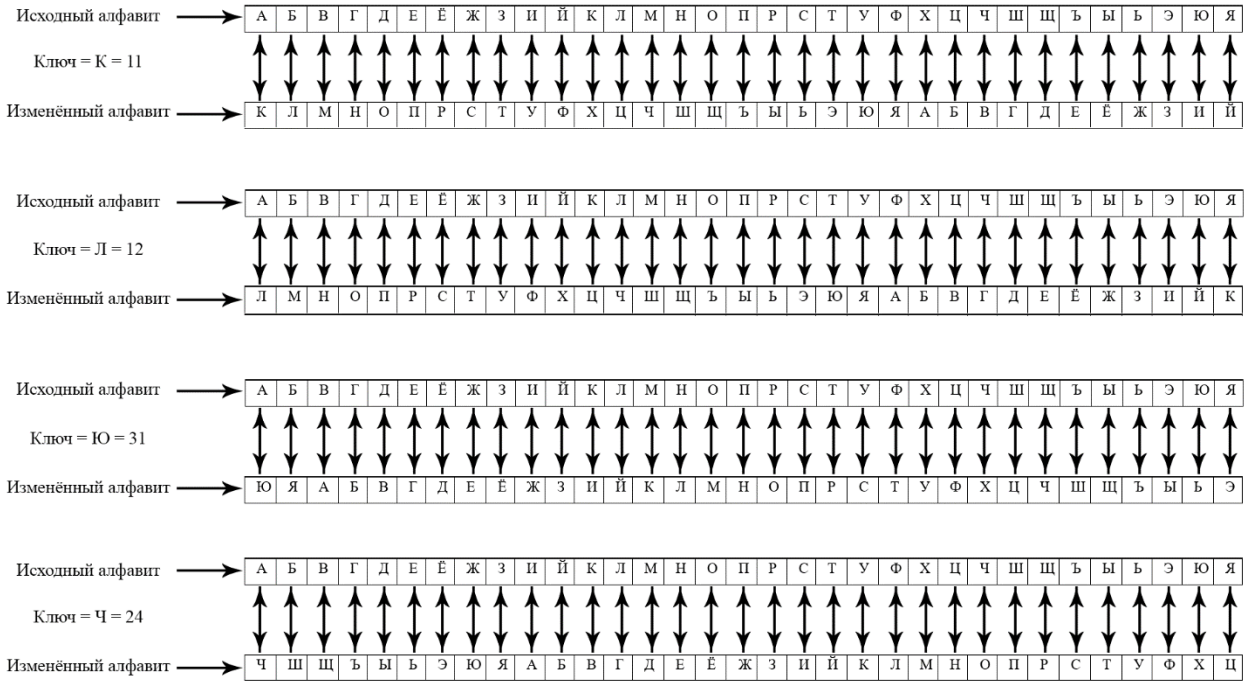

Рис. 1.4 – Моноалфавиты полученные при использовании ключевого слова «КЛЮЧ»

Теперь, когда у нас алфавиты, изменённые под выбранный ключ можно приступить к шифрованию. Зашифруем сообщение «ПОЛИАЛФАВИТНЫЙ ШИФР». Для этого первую буква открытого текста будем шифровать через моноалфавит буквы «К», вторую через моноалфавит буквы «Л», третью через моноалфавит буквы «Ю», четвёртую через моноалфавит буквы «Ч». Для последующих букв требуется повторять цикл до тех пор, пока весь открытый текст не будет зашифрован. В результате шифрования мы получим «ЪЪЙАКЧТЧМФРЕЁХ ПУАО» (рис. 1.5).

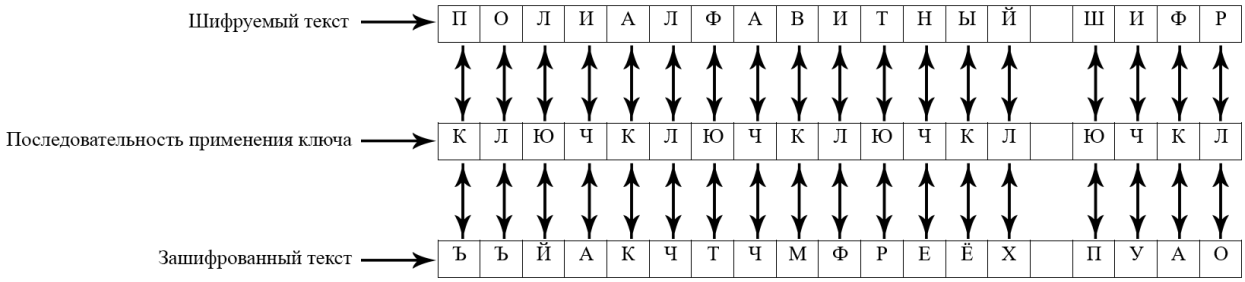

Рис. 1.5 – Развёртка, как шифруется каждая буква открытого текста

Дешифрование соответственно происходит в обратном порядке.

Обычно, если текст хотят зашифровать или расшифровать этим методом используют таблицу под названием «квадрат Виженера» (рис. 1.6).

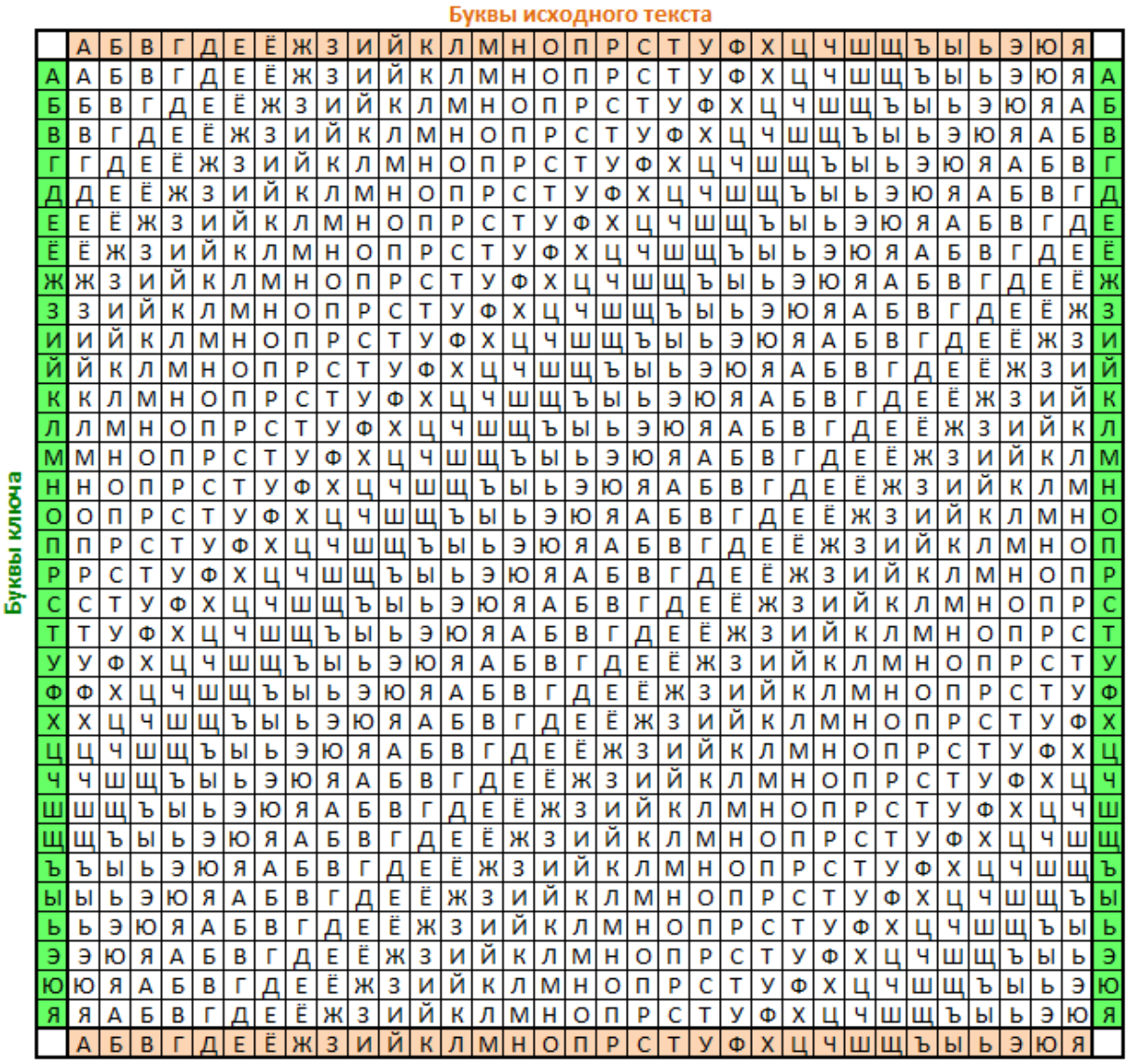

Рис. 1.6 – Квадрат Виженера для русского алфавита

Пользоваться ей довольно просто, по вертикали выбирается буква ключа, по горизонтали буква текста, на их пересечении и будет шифруемая буква.

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования, а также маскировку частот появления тех или иных букв в тексте.

Минусами этого метода можно считать передачу больших объёмов текста, а также распространение ключей и их обновление.

1.1.3.4 Исключающее ИЛИ (XOR)

Метод исключающего ИЛИ (XOR) относится к симметричным криптосистемам типа гаммирование. Принцип работы этого типа шифров заключается в «наложении» последовательности, состоящей из случайных чисел на открытый текст. То есть генератор случайных чисел выдаёт последовательность битов (гамму), которая накладывается на открытый текст с помощью побитовой операции исключающего ИЛИ, в результате чего получается шифротекст [43].

Метод является достаточно простым. Давайте представим русский алфавит (для упрощения без буквы «Ё») в двоичном виде (таблица 1.1).

Таблица 1.1 – Русский алфавит в двоичном представлении

Символ	Код	Символ	Код	Символ	Код	Символ	Код
А	00000	И	01000	Р	10000	Ш	11000
Б	00001	Й	01001	С	10001	Щ	11001
В	00010	К	01010	Т	10010	Ъ	11010
Г	00011	Л	01011	У	10011	Ы	11011
Д	00100	М	01100	Ф	10100	Ь	11100
Е	00101	Н	01101	Х	10101	Э	11101
Ж	00110	О	01110	Ц	10110	Ю	11110
З	00111	П	01111	Ч	10111	Я	11111

Как мы знаем из булевой алгебры, операция логического сложения имеет следующую семантику (рис. 1.7).

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 1.7 – Таблица истинности для операции исключающего ИЛИ (XOR)

Соответственно, если мы к примеру, захотим зашифровать букву «Х» (10101) используя в качестве ключа букву «У» (01011) результатом операции логического сложения будет буква «Ю» (11110). Для дешифровки нужно проделать те же самые действия, с тем же самым ключом, но уже над буквой «Ю». Отсюда вытекает свойство обратимости результата.

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования, а также его стойкость, определяющаяся гаммой (длительностью периода и равномерностью статических характеристик).

Минусами являются: распространение ключей и их обновление.

1.1.3.5 Одноразовый блокнот

Шифр Вернама или одноразовый блокнот – представляет собой систему симметричного шифрования типа гаммирование, так как использует булеву функцию «исключающее ИЛИ». Этот метод был изобретён в 1917 году Гилбертом Вернамом. При правильном использовании этого метода, текст, который был им зашифрован невозможно взломать. Этот метод является примером системы с абсолютной криптографической стойкостью при этом считаясь одной из простейших криптосистем [47].

Так как работает этот метод основываясь на XOR, алгоритм шифрования такой же, как был описан выше. Единственная разница в том, что длина ключа обязательно должна быть равна длине открытого текста.

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования, а также его абсолютную криптографическую стойкость при правильном использовании.

Минусами являются: существенный размер ключа, а также распространение ключей и их регулярное обновление (настолько регулярное, что ни один ключ не должен использоваться более одного раза).

1.1.3.6 Rivest, Shamir, Adleman (RSA)

Последний метод, который мы рассмотрим называется RSA (аббревиатура от фамилий его создателей Rivest, Shamir, Adleman). Он относится к асимметричным криптосистемам.

Эта криптосистема стала первой системой, пригодной как для шифрования, так и для цифровой подписи. Принцип её работы можно разделить на три шага: первый – создание открытого (публичного) и закрытого (секретного) ключей на основе взаимно простых чисел (тех, которые делятся только на единицу или сами на себя), второй шаг – шифрование сообщения и третий шаг – расшифровка.

Как же создать эти самые ключи? Для создания открытого ключа нужно: выбрать два простых числа, вычислить модуль их произведения, вычислить функцию Эйлера, выбрать открытую экспоненту, которая также будет являться простым числом, при этом будет меньше числа, полученного при вычислении функции Эйлера и наконец будет взаимно простым для этой функции. В результате этих манипуляций с формулами мы получим пару чисел, это и есть наш публичный ключ. Закрытый ключ создаётся вычислением обратной открытой экспоненты по модулю функции Эйлера, при этом модуль должен быть равен единице.

Теперь нужно отдать полученный открытый ключ человеку, который с помощью него планирует отправлять вам зашифрованные сообщения.

Допустим, что, выполнив все предыдущие операции мы получили числа 5 и 21 – это публичный ключ и числа 17 и 21 – это секретный ключ. Теперь допустим, что вы хотите зашифровать букву «С», в русском алфавите она располагается на 19 позиции. Для того чтобы зашифровать эту букву нужно возвести позицию нашей буквы в 5 степень и от полученного числа взять остаток от деления на 21. В результате получиться число 10 или буква «И» это и будут наши закодированные данные.

Шифрованные данные передаются владельцу секретного ключа. Тут следует обратить внимание на то, что открытый ключ не может расшифровать сообщение, а закрытый находится только у владельца (если он никому его не говорил), так что передача может осуществляться по открытому каналу. Итак, чтобы дешифровать сообщение нужно провести те же действия, что и при шифровании сообщения только используя закрытый ключ. Возьмём нашу зашифрованную букву «И» расположенную на позиции 10, возведём её в 17 степень и от полученного числа возьмём остаток от деления на 21. Результатом вычислений будет число 19 или буква «С» [27].

Подводя итоги, если необходимо передать зашифрованное сообщение владельцу ключей, то отправитель должен получить у него открытый ключ, зашифровать своё сообщение и передать его владельцу. При этом расшифровать сообщение не может никто, кроме владельца секретного ключа. Весь этот процесс представлен на рис. 1.8.

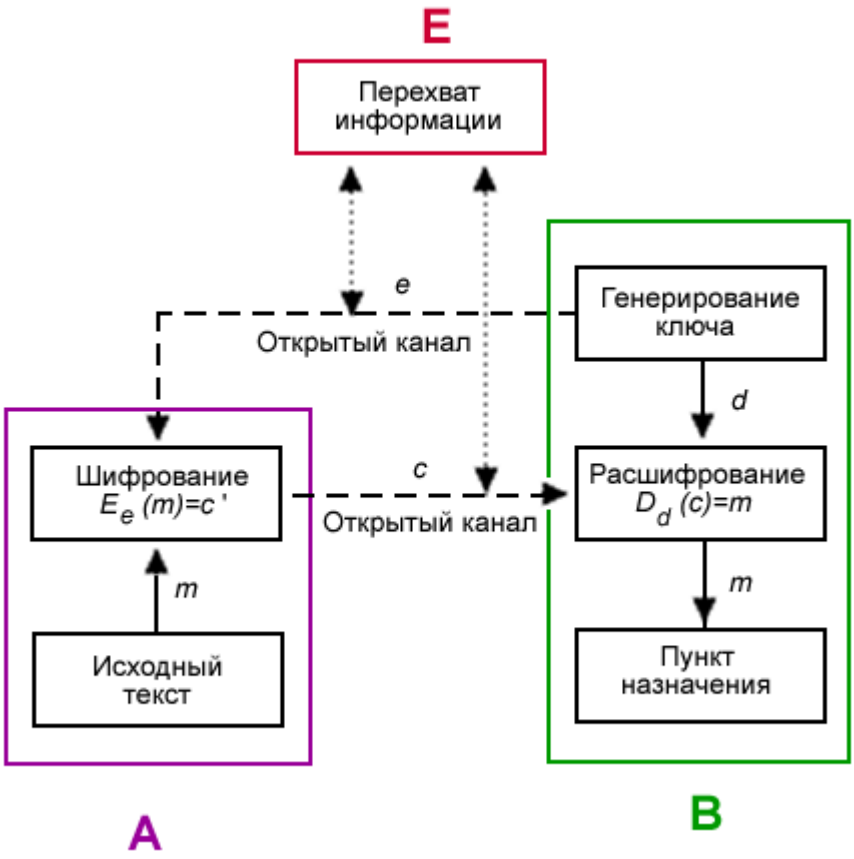

Рис. 1.8 – Процесс шифрования при помощи алгоритмы RSA

Надёжность такого шифрования обеспечивается тем, что третьему лицу очень трудно вычислить закрытый ключ по открытому. Хотя ключи вычисляются из одной пары чисел и по сути связаны между собой, установить их связь очень сложно, а чем большие числа были взяты, тем длиннее становится процесс вскрытия.

Плюсами этого метода можно считать удобство распространения ключей и высокую безопасность при передаче небольших сообщений.

Минусами являются: существенный размер ключа, низкая скорость шифрования, а также то, что шифрование происходит по буквам, то есть одна и та же буква будет шифроваться одним и тем же числом, если злоумышленник перехватит достаточно большое сообщение расшифровать его не составит никакого труда.

1.2 Характеристика инструментальных средств разработки

1.2.1 Характеристика среды программирования

Для написания программы использовалась среда программирования Microsoft Visual Studio 2017, представляющая собой полный набор средств разработки для создания веб-приложений Active Server Pages (ASP), eXtensible Markup Language (XML), настольных приложений и мобильных приложений. Visual Studio использует единую интегрированную среду разработки Integrated Development Environment (IDE), которая позволяет совместно использовать средства и упрощает создание решений на базе нескольких языков. Кроме того, в этих языках используются функциональные возможности платформы .NET Framework, которая позволяет получить доступ к ключевым технологиям, упрощающим разработку веб-приложений ASP и XML.

Microsoft Visual Studio объединяет в себе огромное количество функций, позволяющих осуществлять разработки для Windows всех версий, Интернета, SharePoint, различных мобильных устройств и облачных технологий. В Visual Studio реализуется новая среда разработчика, благодаря которой создавать приложения стало проще [19].

1.2.2 Описание системы Windows Presentation Foundation (WPF)

В качестве системы для построения клиентских приложений была выбрана WPF.

WPF – это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность.

В основе WPF лежит независимый от разрешения векторный модуль визуализации, использующий возможности современного графического оборудования. Возможности этого модуля расширяются с помощью комплексного набора функций разработки приложений, которые включают в

себя язык eXtensible Application Markup Language (XAML), элементы управления, привязку к данным, макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF входит в состав .NET Framework, поэтому вы можете создавать приложения, включающие другие элементы библиотеки классов .NET Framework.

При разработке поведения приложения главной задачей является обеспечение реакции на действия пользователя, включая обработку событий (таких как выбор пункта меню или нажатие на кнопку), и вызов в ответ бизнес-логики и логики доступа к данным. В WPF такое поведение реализуется в коде, связанном с разметкой. Этот код называется кодом программной части.

Возможности взаимодействия с пользователем, обеспечиваемые моделью приложения, реализуются с помощью сконструированных элементов управления. В WPF элемент управления – это общий термин, который относится к категории классов WPF, размещаемых в окне или на странице, имеющих пользовательский интерфейс и реализующих некоторое поведение [35].

1.2.3 Описание языка программирования

В качестве языка программирования был выбран язык C# версии 7.0. NET Framework 4.7.1.

Язык C# появился на свет в июне 2000 г., в результате кропотливой работы большой группы разработчиков компании Microsoft, возглавляемой Андерсом Хейлсбергом.

Создание инструментария для разработчиков с их полноценной поддержкой является одной из главных задач нового языка C#.

Авторы C# стремились создать язык, сочетающий простоту и выразительность современных объектно-ориентированных языков с богатством возможностей и мощью C++.

Особенности C#:

- полная поддержка классов и объектно-ориентированного программирования, включая наследование интерфейсов и реализаций, виртуальных функций и перегрузки операторов;
- полный и хорошо определенный набор основных типов;
- встроенная поддержка автоматической генерации XML-документации;
- возможность отметки классов и методов атрибутами, определяемыми пользователем. Это может быть полезно при документировании и способно воздействовать на процесс компиляции;
- автоматическое освобождение динамически распределенной памяти;
- полный доступ к библиотеке базовых классов .NET, а также легкий доступ к Windows Application Programming Interface;
- указатели и прямой доступ к памяти, если они необходимы. Однако язык разработан таким образом, что практически во всех случаях можно обойтись и без этого;
- поддержка свойств и событий в стиле Visual Basic;
- простое изменение ключей компиляции. Позволяет получать исполняемые файлы или библиотеки компонентов .NET, которые могут быть вызваны другим кодом так же, как элементы управления ActiveX;
- поддержка как Framework Class Library (FCL), так и Common Language Runtime (CLR);
- возможность использования C# для написания динамических web-страниц ASP.NET.

В настоящее время C# является одним из самых популярных языков программирования. Он применяется в разработке компьютерных, мобильных и веб-приложений. Также язык регулярно обновляется и имеет огромное количество документации, которая сильно упрощает и ускоряет процесс разработки программ на нём [15, 31].

1.2.4 Другие средства разработки

Для создания контекстной справки использовалась программа «Htm2chm». Она представляет собой проприетарный формат файлов контекстной справки, разработанный корпорацией Microsoft и выпущенный в 1997 году в качестве замены формата WinHelp. Содержит в себе набор HTML-страниц, может также включать в себя содержание со ссылками на страницы, предметный указатель, а также базу для полнотекстового поиска по содержимому страниц [18].

Преимуществами программы можно считать:

- размер файла меньше, чем у обычного HTML;
- используются все возможности форматирования, имеющиеся в HTML и CSS;
- возможность полнотекстового поиска;
- возможность просмотра множества .chm-файлов как один, с общим содержанием и предметным указателем.

Для построения схем и диаграмм при составлении технического задания и сопровождающей документации было решено использовать Draw.io. Draw.io – это сервис, предназначенный для формирования диаграмм и схем. Сервис разделён на три части – меню, панель объектов и сам документ.

С помощью веб-сервиса Draw.io можно создавать:

- диаграммы;
- UML-модели;
- вставка в диаграмму изображений;
- графики;
- блок-схемы;
- формы;
- другое.

Также доступен экспорт готовых схем в изображение и синхронизация полученных документов с Google Диском [16].

Глава 2 Практическая часть

2.1 Постановка задачи

2.1.1 Описание входных данных

Входные данные – данные, вводимые или выбранные пользователем в ходе работы программы. К ним относится: выбор операции, выбор способа шифрования или дешифрования, ввод исходного текста, ввод ключей, ввод значений для генерации ключей и файлы, загружаемые в программу.

Подробнее входные данные описаны в таблице 2.1.

Таблица 2.1 – Входные данные

Наименование	Идентификатор	Тип данных	Размер
Выбор операции	IsChecked	Bool	1 байт
Выбор способа шифрования или дешифрования	SelectedIndex	Int32	4 байта
Исходный текст введённый пользователем вручную или считанный из файла	soursetext openFileDlg	String String	255 байт 255 байт
Ключ введённый пользователем вручную или считанный из файла	shift key _key	String Int32 Int32	255 байт 4 байта 4 байта
Размер ключа	sizesourcetext	Int32	4 байта
Значение P	p	Int32	4 байта
Значение Q	q	Int32	4 байта

2.1.2 Описание выходных данных

Выходные данные – данные, полученные в ходе работы пользователя с программой. К ним относится: сгенерированные пользователем ключи, зашифрованный или дешифрованный текст.

Подробнее выходные данные описаны в таблице 2.2.

Таблица 2.2 – Выходные данные

Наименование	Идентификатор	Тип данных	Размер
Сгенерированные пользователем ключи	shift code e d n	Int32 String Long (Int64) Long (Int64) Long (Int64)	4 байта 255 байт 8 байт 8 байт 8 байт
Зашифрованный или дешифрованный текст	EncryptedData	String	255 байт
Сохраняемые изменения в тексте, результаты шифрования или дешифрования и ключи	saveFileDlg	String	255 байт

2.1.3 Математическая модель задачи

В программе реализовано шесть способов для шифрования текста и его дешифровки. Их работа основывается на формулах, о которых будет рассказано далее.

Начнём с формул для шифра транспозиции. Как говорилось ранее перед шифрованием этим способом исходный текст необходимо разбить на блоки равные длине ключа. Для этого используется следующая формула (2.1):

$$\sum_{i=0}^{n} \mathrm{input} = \mathrm{input}_i + \mathrm{input}_{i+1} + \mathrm{input}_{i+2} \dots \mathrm{input}_n \quad (2.1)\ [36]$$

где i – индекс суммирования, n – верхняя граница суммирования равная остатку от деления текста на размер ключа, input – переменная, обозначающая каждый член в серии.

После того, как текст будет разделён на блоки начинается процесс шифрования. Для шифрования используется следующая формула (2.2):

$$\mathrm{trans}_{m_j - 1} = \mathrm{input}_{i+j} \quad (2.2)\ [36]$$

где i – позиция символа в исходном тексте, j – позиция символа в блоке, m – ключ, trans – новая позиция символа, input – позиция символа.

Дешифрование этим способом проходит по похожему алгоритму за исключением того, что текст не нужно увеличивать до длины, которая будет делиться ровно на длину ключа, так как уже имеет подходящий размер. Следовательно, остаётся только переставить символы по следующей формуле (2.3):

$$\mathrm{trans_j} = \mathrm{input_{i+m_j-1}} \quad (2.3)\ [36]$$

где i – позиция символа в исходном тексте, j – позиция символа в блоке, m – ключ, trans – новая позиция символа, input – позиция символа.

Следующими алгоритмами рассмотрим моноалфавитный и полиалфавитный шифры. Они работают по одинаковым формулам. Если сопоставить каждому символу алфавита его порядковый номер, то шифрование и дешифрование можно выразить формулами 2.4:

$$y = (x + k) \bmod n$$
$$x = (y - k) \bmod n \quad (2.4)\ [37, 51]$$

где x – символ открытого текста, y – символ шифрованного текста, n – мощность алфавита (длина), k – ключ.

В шифрах исключающего ИЛИ и Вернама для шифрования и дешифрования текста используют следующую формулу (2.5):

$$x = (j \oplus k) \bmod l \quad (2.5)\ [23, 24, 43, 47]$$

где x – символ открытого или закрытого текста, j – номер символа в алфавите, k – номер символа ключа, l – мощность алфавита (длина).

Для шифрования алгоритмом RSA нужно создать открытый и закрытый ключи, для этого требуется найти модуль произведения простых чисел по формуле (2.6):

$$n = p * q \quad (2.6)\ [21, 27]$$

где n – модуль произведения простых чисел, p, q – простые, отличные друг от друга числа.

Далее вычисляется функция Эйлера (2.7):

$$\varphi(n) = (p - 1) * (q - 1) \quad (2.7)\ [21, 27]$$

где $\varphi(n)$ – значение функции Эйлера, n – модуль произведения простых чисел, p, q – простые, отличные друг от друга числа.

Затем ищется значение открытой экспоненты по следующим критериям (2.8):

$$1 < e < \varphi(n) \quad (2.8)\ [21, 27]$$

где $\varphi(n)$ – значение функции Эйлера, e – открытая экспонента.

Значения {e, n} – открытый ключ.

Чтобы вычислить значение для закрытого ключа, нужно подобрать такое число, которое было бы мультипликативно обратно к открытой экспоненте по модулю функции Эйлера, то есть число, удовлетворяющее сравнению. Оно находится по следующей формуле (2.9):

$$d * e \equiv 1 \quad (2.9)\ [21, 27]$$

где d – закрытая экспонента, e – открытая экспонента.

С ключами разобрались, теперь разберёмся как шифровать текст методом RSA. Для этого нужно взять открытый ключ и воспользоваться следующей формулой (2.10):

$$c = m^e \bmod n \quad (2.10)\ [21, 27]$$

где c – закрытый текст, e – открытая экспонента, m – открытый текст, n – модуль произведения простых чисел.

Дешифруется текст с использованием закрытого ключа по следующей формуле (2.11):

$$m = c^d \bmod n \quad (2.11)\ [21, 27]$$

где c – закрытый текст, d – закрытая экспонента, m – открытый текст, n – модуль произведения простых чисел.

2.1.4 Требования к программному обеспечению

2.1.4.1 Функциональные требования

Программа должна выполнять следующие функции:

- шифровать информацию с помощью выбранного алгоритма;
- дешифровать информацию с помощью выбранного алгоритма;

- загружать в программу уже существующие файлы;
- сохранять информацию, полученную в ходе работы программы;
- генерировать случайные ключи;
- увеличивать размер шрифта;
- уменьшать размер шрифта;
- редактировать текст прямо из программы;
- очищать поля от текста в одно нажатие;
- выполнять необходимые операции посредством нажатия определённых клавиш на клавиатуре;
- выводить подсказки в ходе работы;
- сообщать об ошибках, возникающих в ходе неправильной работы;
- вызывать справку с подробной информацией о программе;
- запрашивать у пользователя подтверждение для осуществления некоторых действий;
- работать в соответствии с заданным алгоритмом;
- производить бесперебойную работу по преобразованию информации.

Помимо этого, должно быть обеспечено правильное выполнение алгоритмов шифрования и дешифрования текста в соответствии со всеми формулами и уравнениями, используемыми в программе.

2.1.4.2 Нефункциональные требования

Разрабатываемый интерфейс должен:

- быть интуитивно понятным пользователю;
- быть выполнен в нейтральных цветах, а также в минималистическом стиле и при этом привлекательно смотреться;
- обеспечивать видимость всех элементов управления, которые необходимы для выполнения конкретной задачи;
- предоставлять пользователю помощь в освоении программы посредством подсказок;

- обладать ненавязчивостью в своём стремлении помочь, то есть не заставлять пользователя совершать лишних действий;
- минимизировать возможность ошибок пользователя.

В программе должна быть предусмотрена защита от таких действий пользователя как:

- ввод неверных ключей;
- ввод неверных параметров при генерации ключа;
- попытки выполнить операцию шифрования или дешифрования без выбранного алгоритма;
- попытка сохранить ключ при его генерации до того, как он будет сгенерирован;
- попытка перезаписать исходный текст в файл, для которого не указан путь;
- попытка открытия справки «О программе», если файл был повреждён или удалён.

Также программа должна быть реализована на языке C# версии 7.0. NET Framework 4.7.1. в системе для построения клиентских приложений WPF.

2.2 Анализ требований и определение спецификаций программного обеспечения

2.2.1 Диаграмма вариантов использования

Диаграмма вариантов использования представлена на рис. 2.1. Она описывает взаимоотношения и зависимости между группами вариантов использования и пользователем.

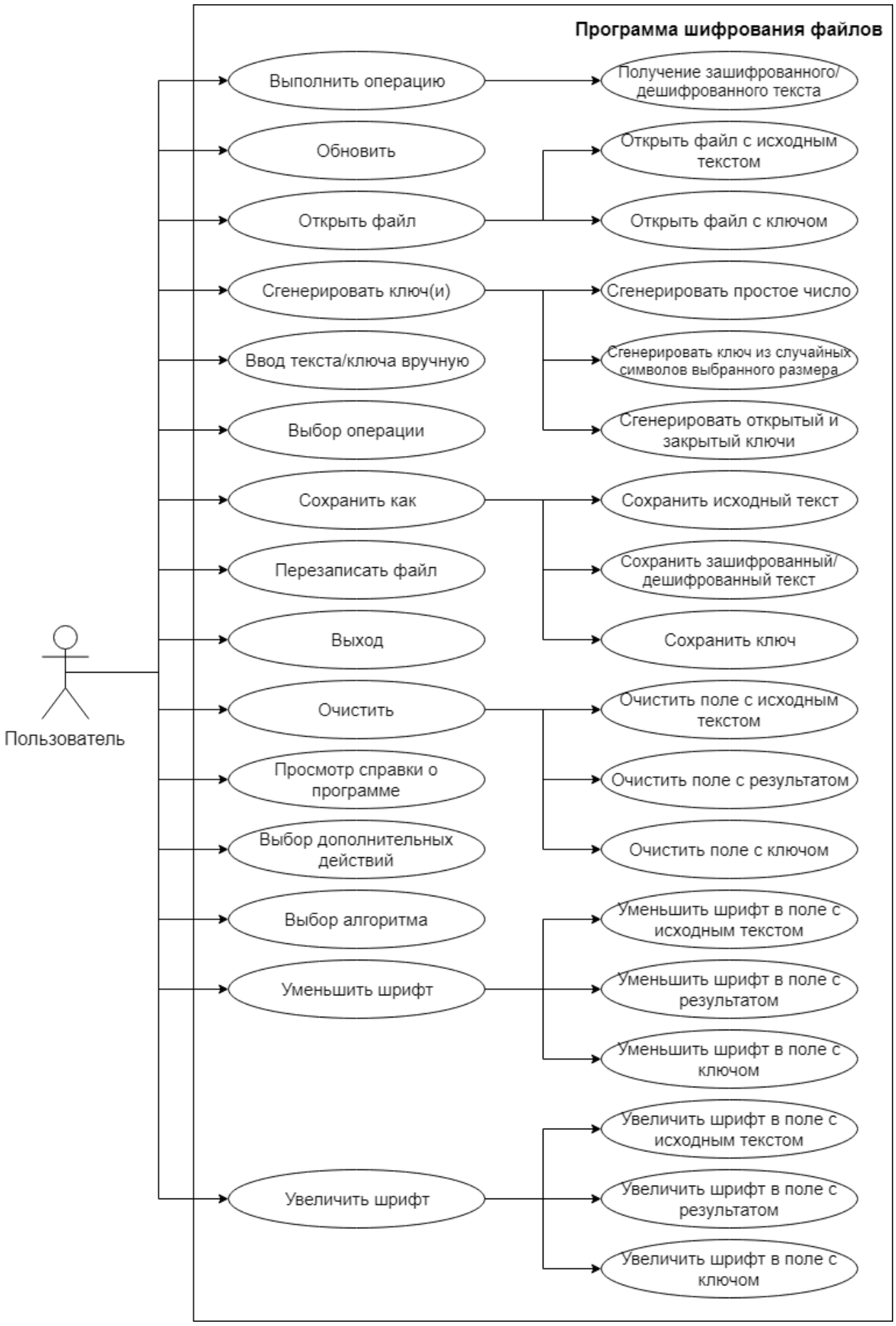

Рис. 2.1 – Диаграмма вариантов использования

2.2.2 Диаграмма потоков данных

Контекстная диаграмма потоков данных представлена на рис. 2.2. Она представляет собой взаимодействие пользователя и программы, связанных потоками данных.

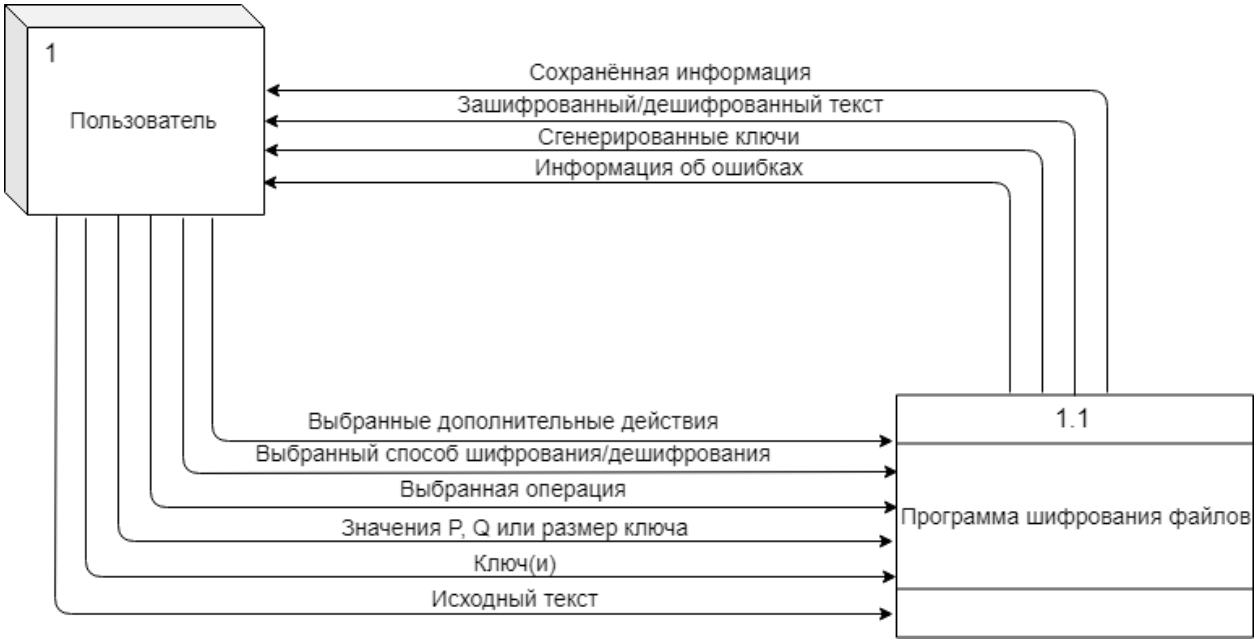

Рис. 2.2 – Контекстная диаграмма потоков данных

2.2.3 Функциональная диаграмма

Функциональная диаграмма – диаграмма, отражающая взаимосвязи функций разрабатываемого программного обеспечения (ПО). Контекстная функциональная диаграмма представлена на рис. 2.3.

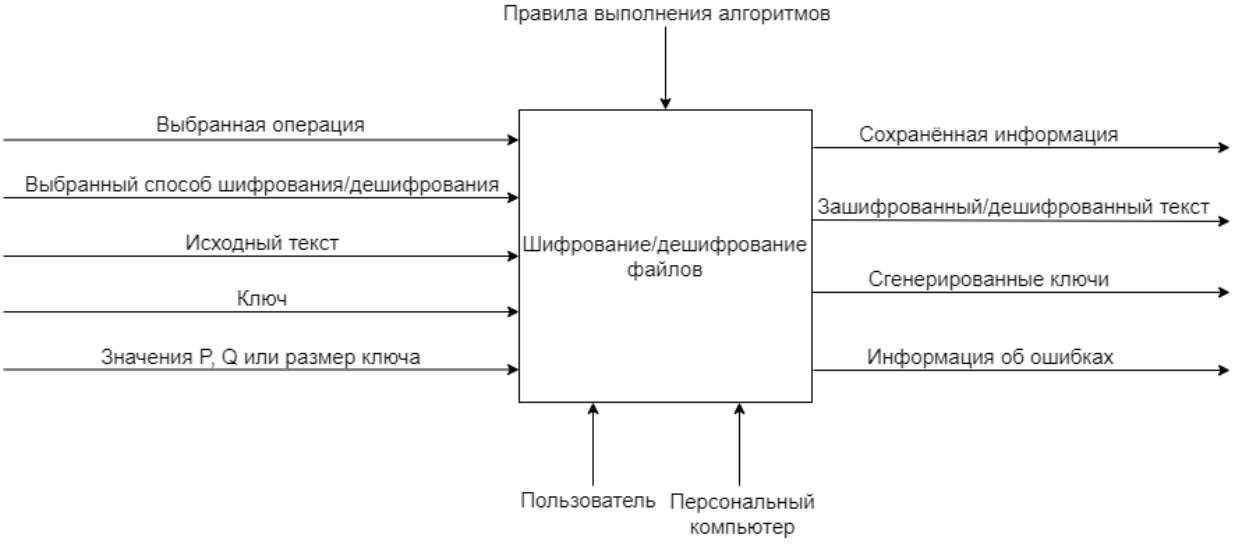

Рис. 2.3 – Контекстная функциональная диаграмма

2.3 Проектирование программного обеспечения

2.3.1 Структурная схема

Структурная схема представляет собой совокупность элементарных звеньев объекта и связей между ними. Под элементарным звеном следует понимать часть объекта, системы управления, которая реализует элементарную функцию. Структурная схема программы представлена на рис. 2.4.

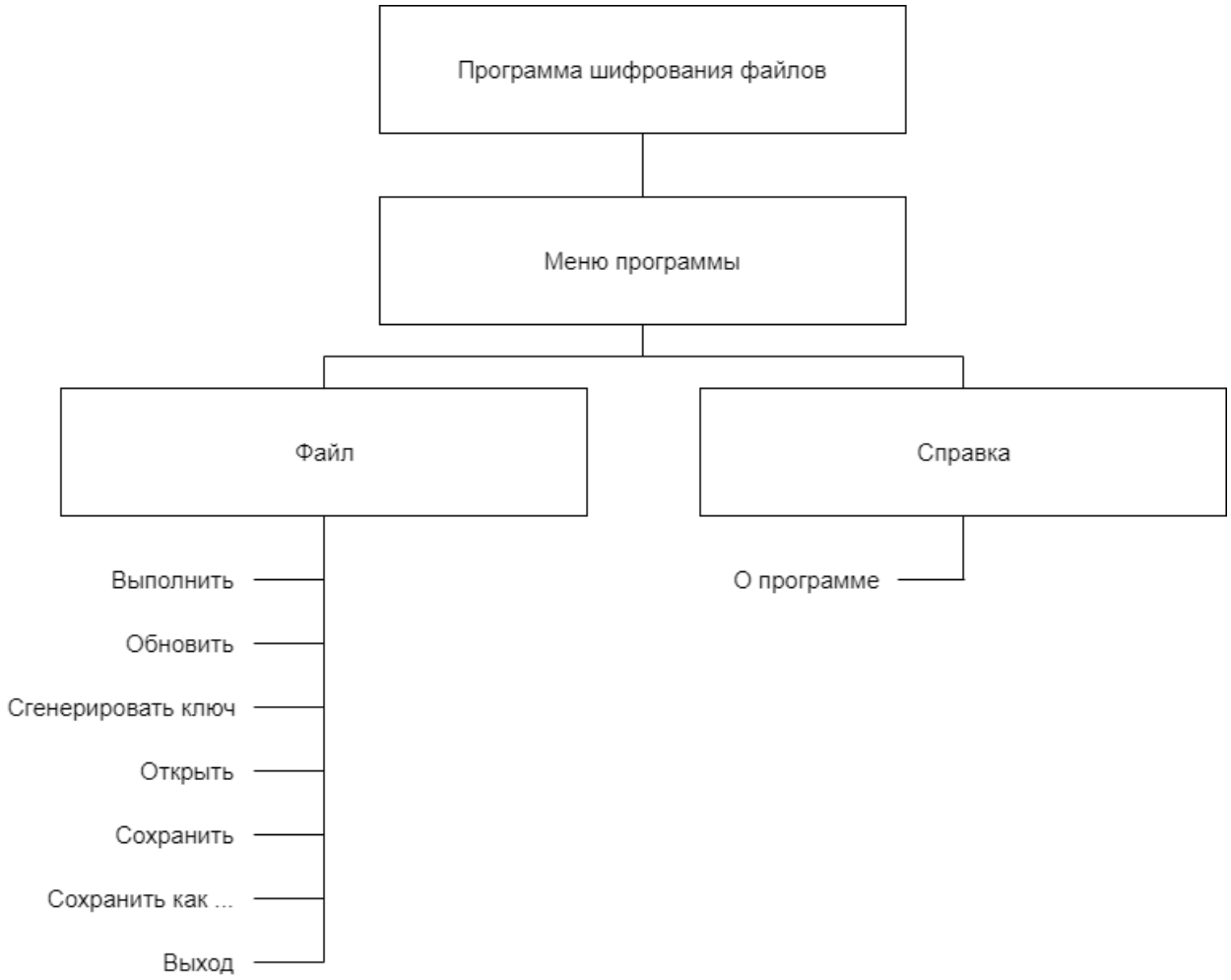

Рис. 2.4 – Структурная схема

2.3.2 Функциональная схема

Функциональная схема приведена на рис. 2.5. Она разъясняет процессы, протекающие в отдельных функциях программы и программы в целом.

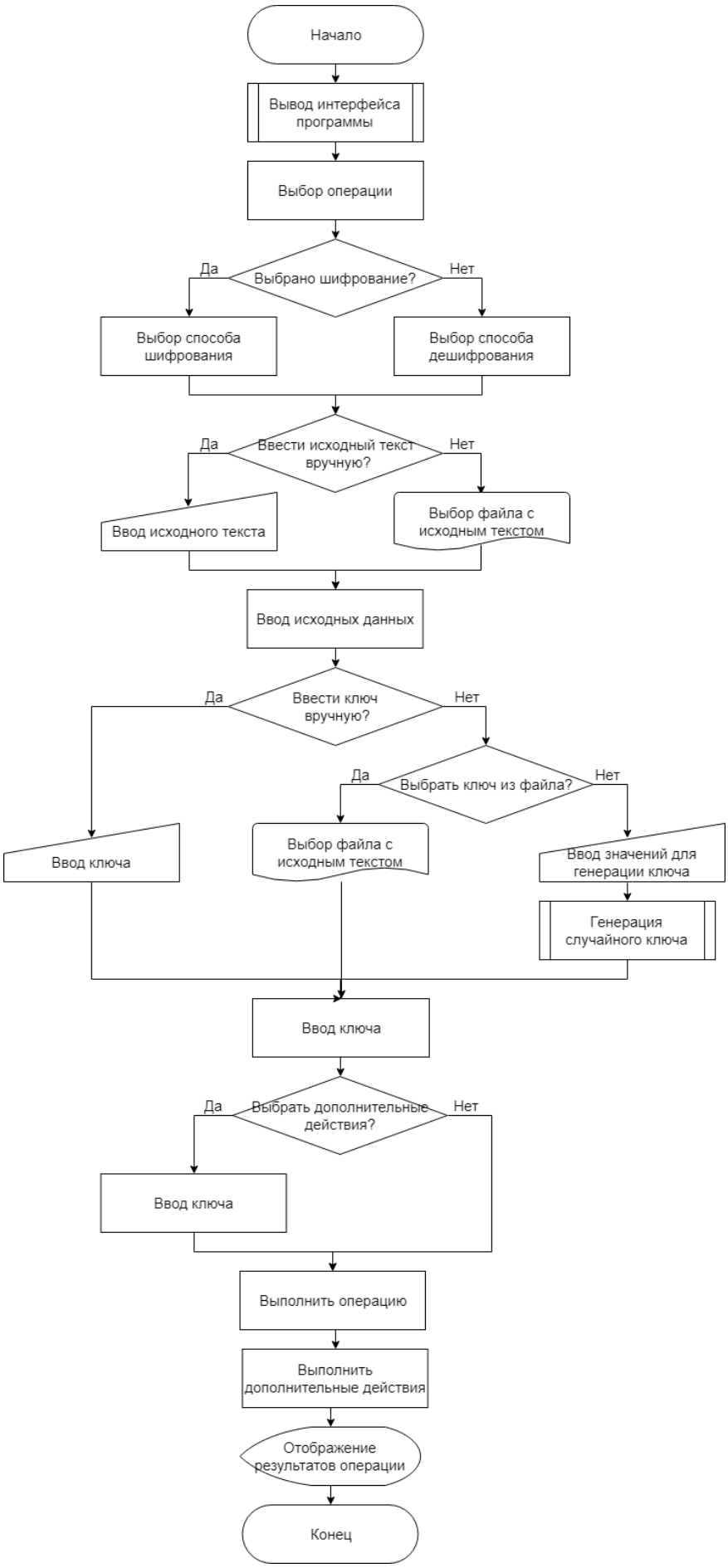

Рис. 2.5 – Функциональная схема

2.4 Разработка пользовательских интерфейсов

При запуске программы перед пользователем появляется следующее окно (рис. 2.6)

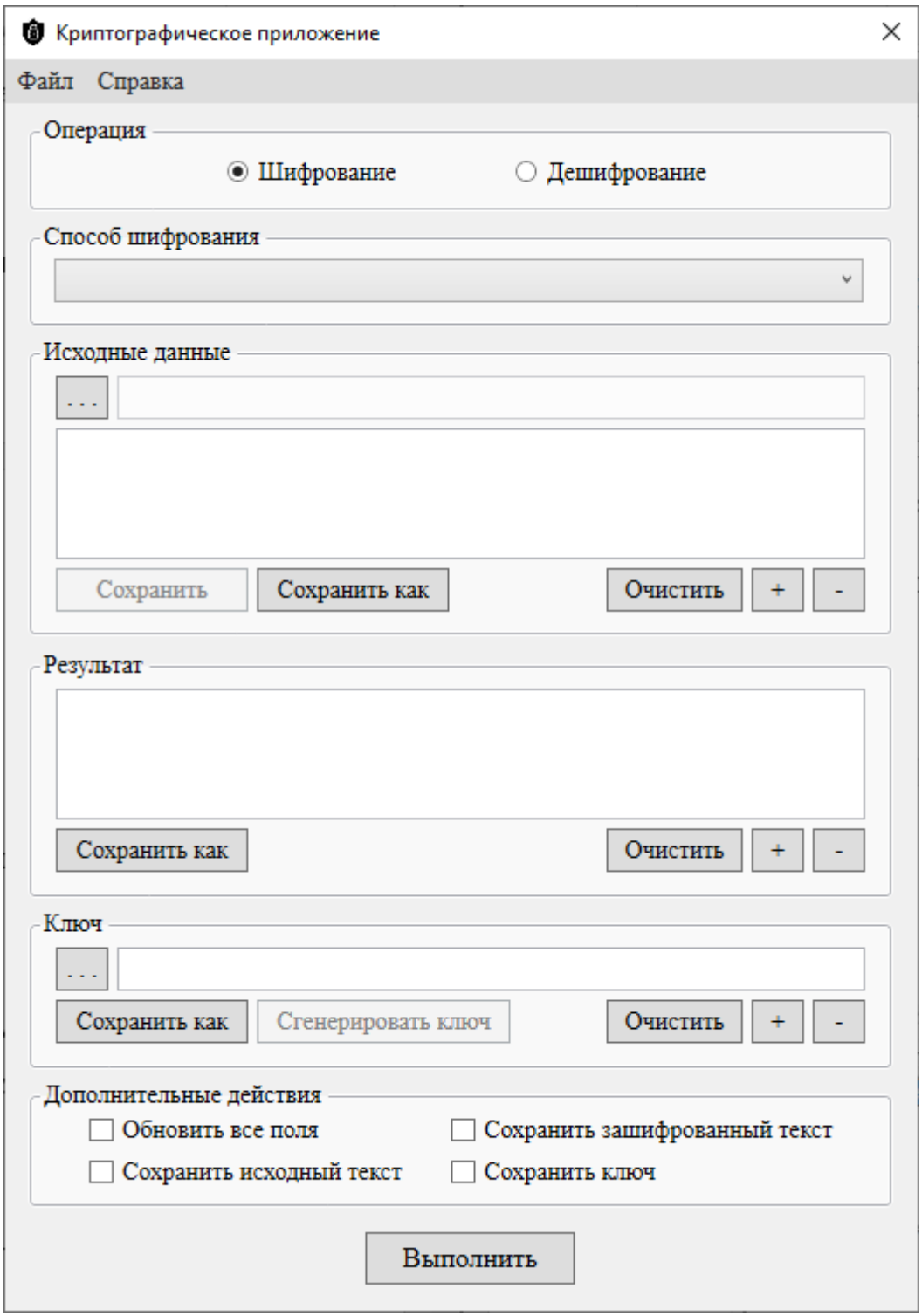

Рис. 2.6 – Главное окно программы

Так как окно является единственным, из него доступны все остальные функции программы (рис. 2.7).

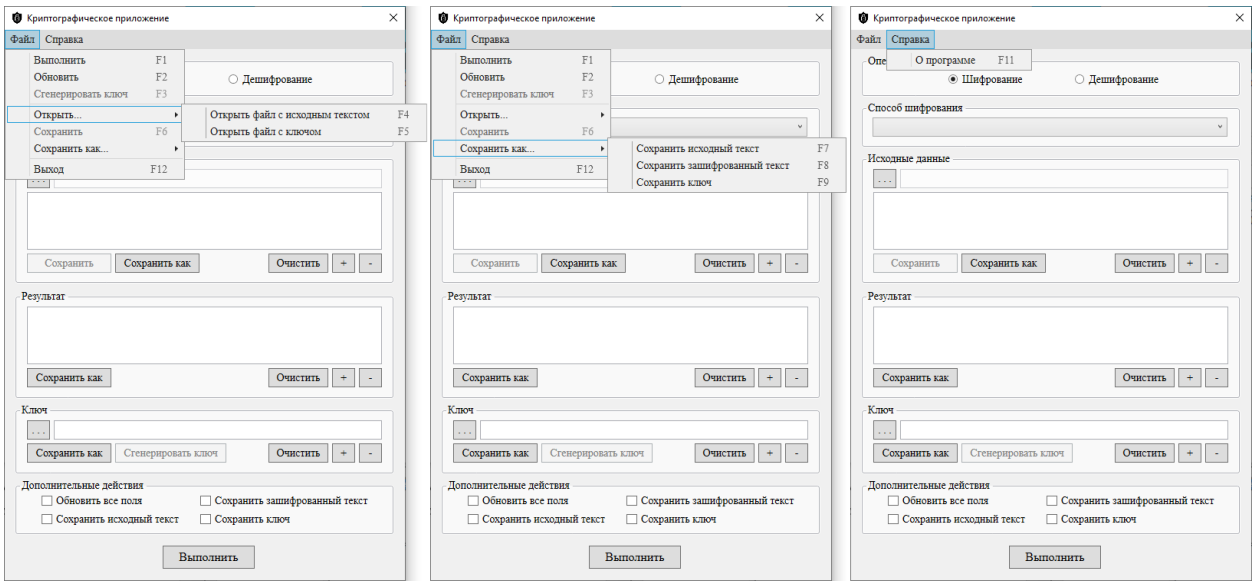

Рис. 2.7 – Большинство функций программы

Окно, открывающееся при сохранении файла с исходным текстом представлено на рис. 2.8.

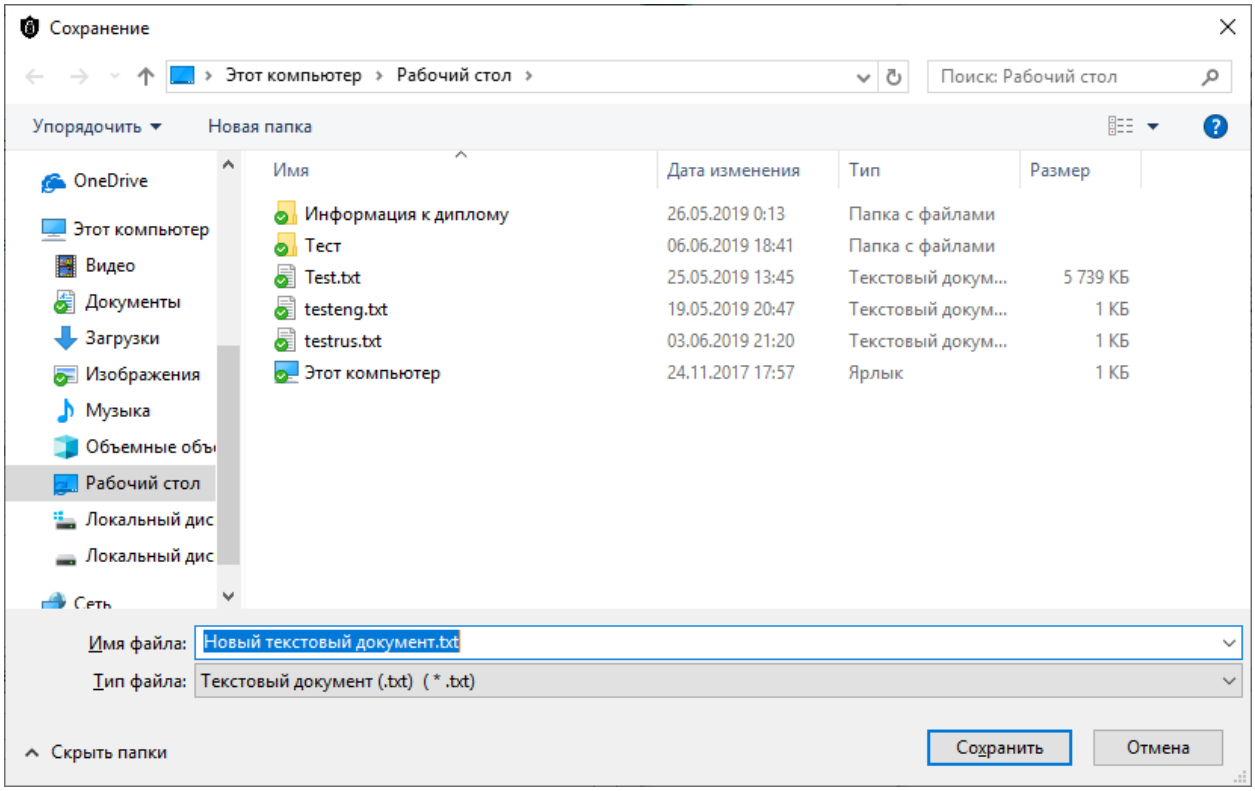

Рис. 2.8 – Окно, при сохранении файла с исходным текстом

В зависимости от того, какой файл вы хотите сохранить будут предложены разные изначальные названия для них (рис. 2.9).

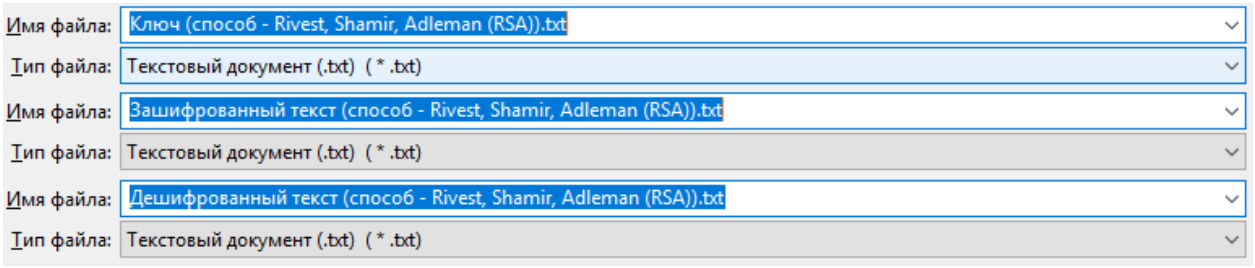

Рис. 2.9 – Некоторые варианты изначальных названий файлов

Окно, открывающееся при выборе, файла, который вы хотите открыть в программе представлено на рис. 2.10.

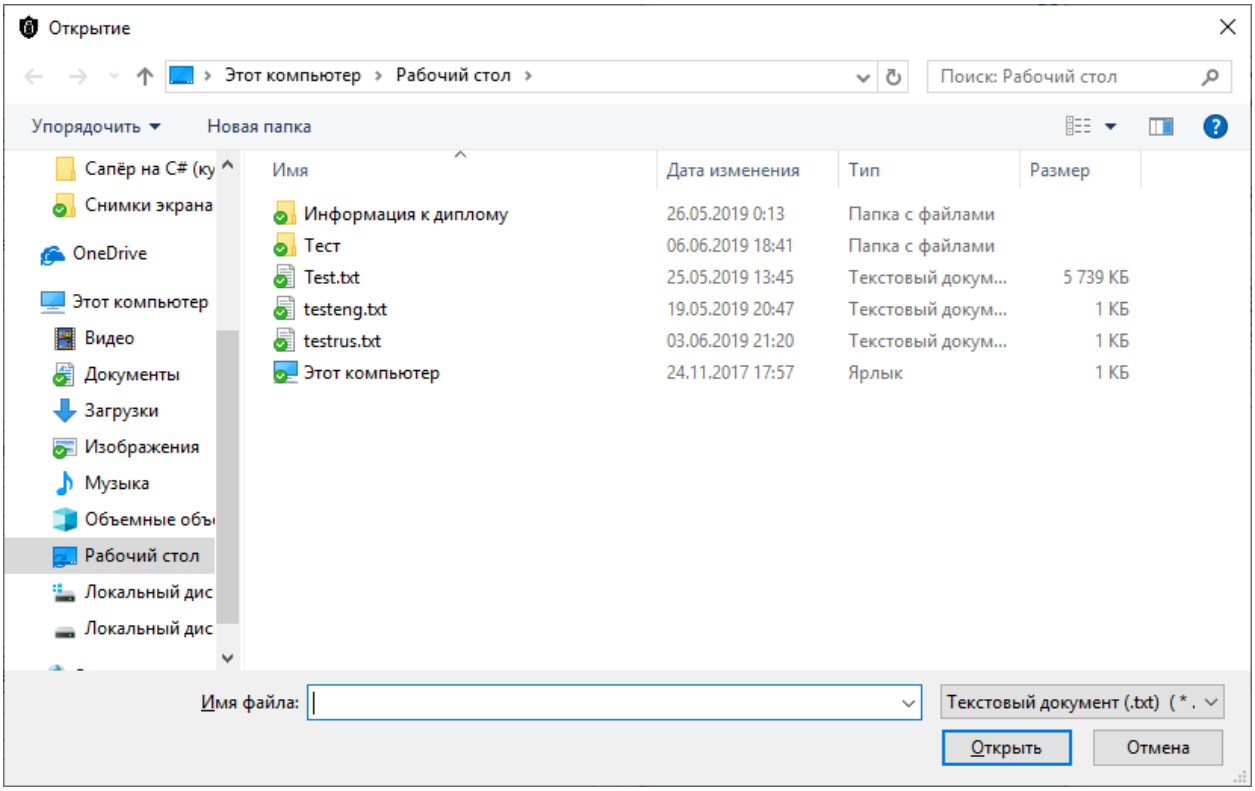

Рис. 2.10 – Окно, при открытии файла в программе

На рис. 2.11 представлены разные варианты интерфейса для генерации ключа, они меняются в зависимости от выбранного алгоритма.

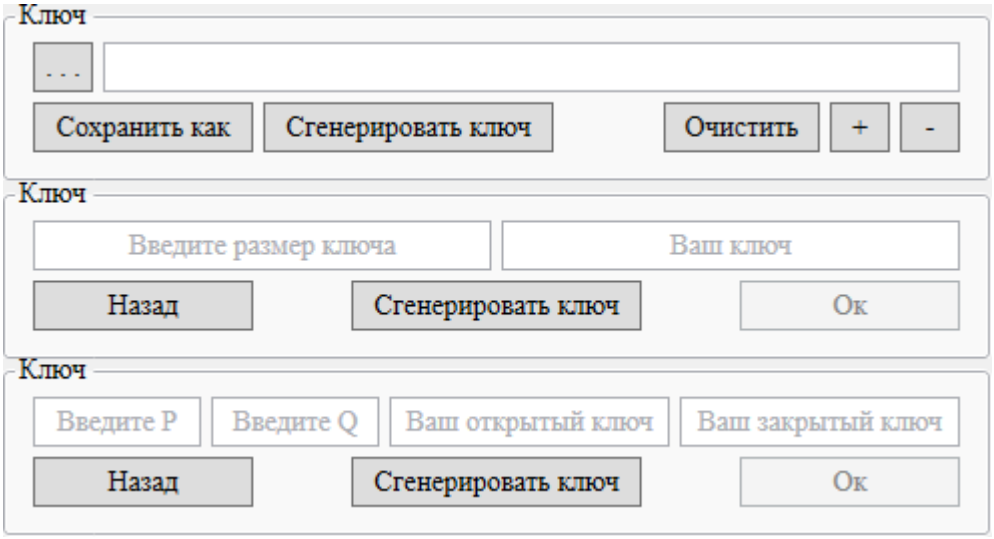

Рис. 2.11 – Варианты интерфейса при генерации ключа

Окно справки продемонстрированно на рис. 2.12.

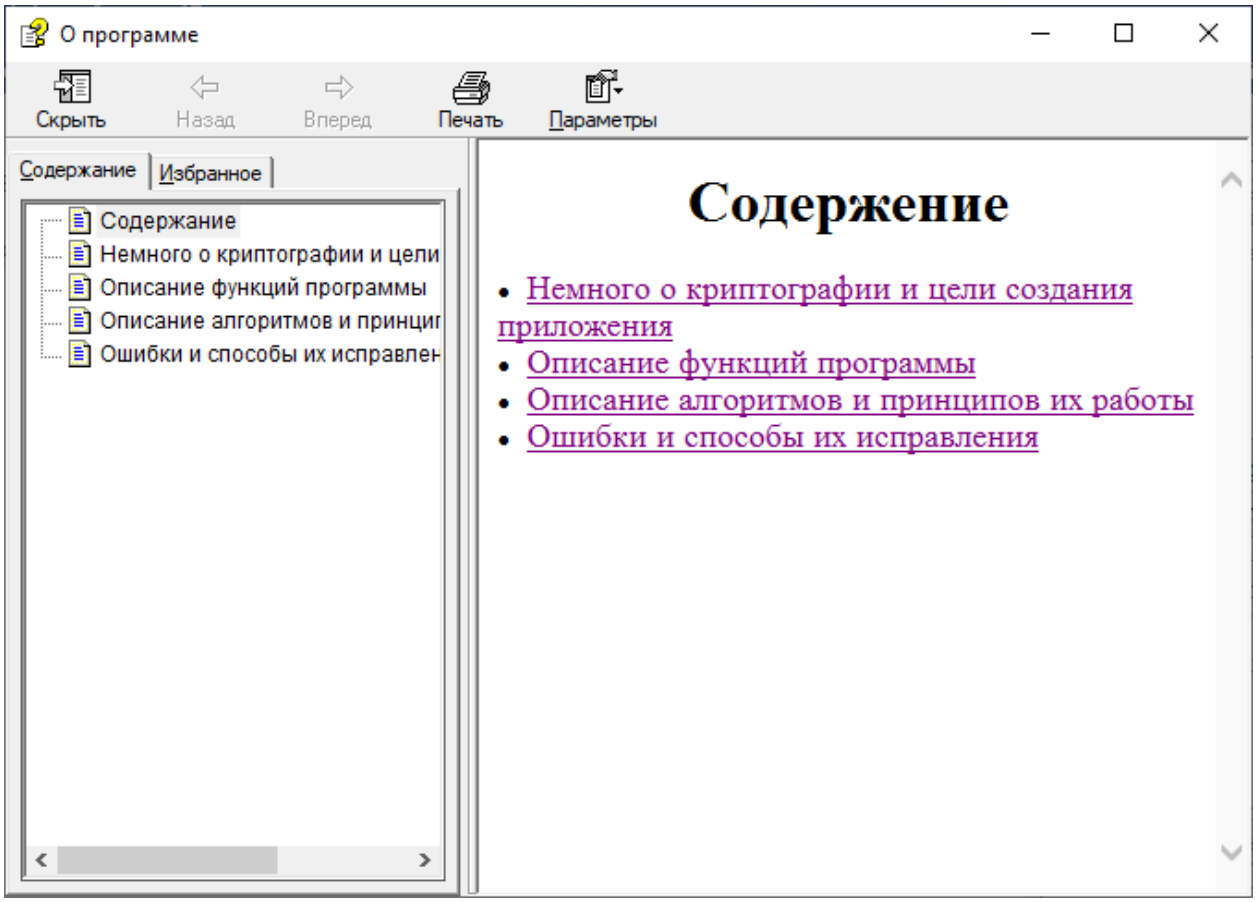

Рис. 2.12 – Окно справки

2.5 Тестирование и отладка программного обеспечения

В программе существует несколько моментов, где в ходе неправильной работы может произойти ошибка. Первый из них это отсутствие файла справки. Если при попытке открыть справку о программе, файл с ней будет не

найден или повреждён, перед пользователем появится окно, продемонстрированное на рис. 2.13.

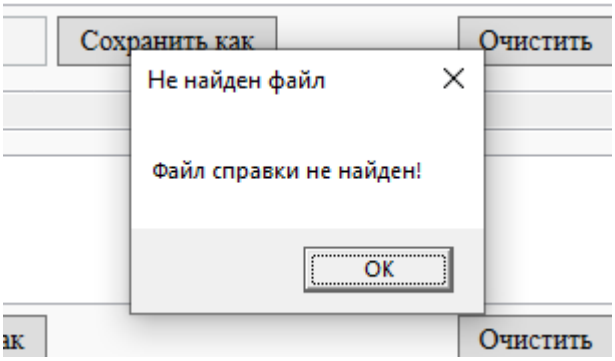

Рис. 2.13 – Реакция программы на возникновение ошибки, связанной с отсутствием или повреждением файла справки

Если с файлом справки всё впорядке перед пользователем откроется справка на вкладке «Содержание» (рис. 2.14).

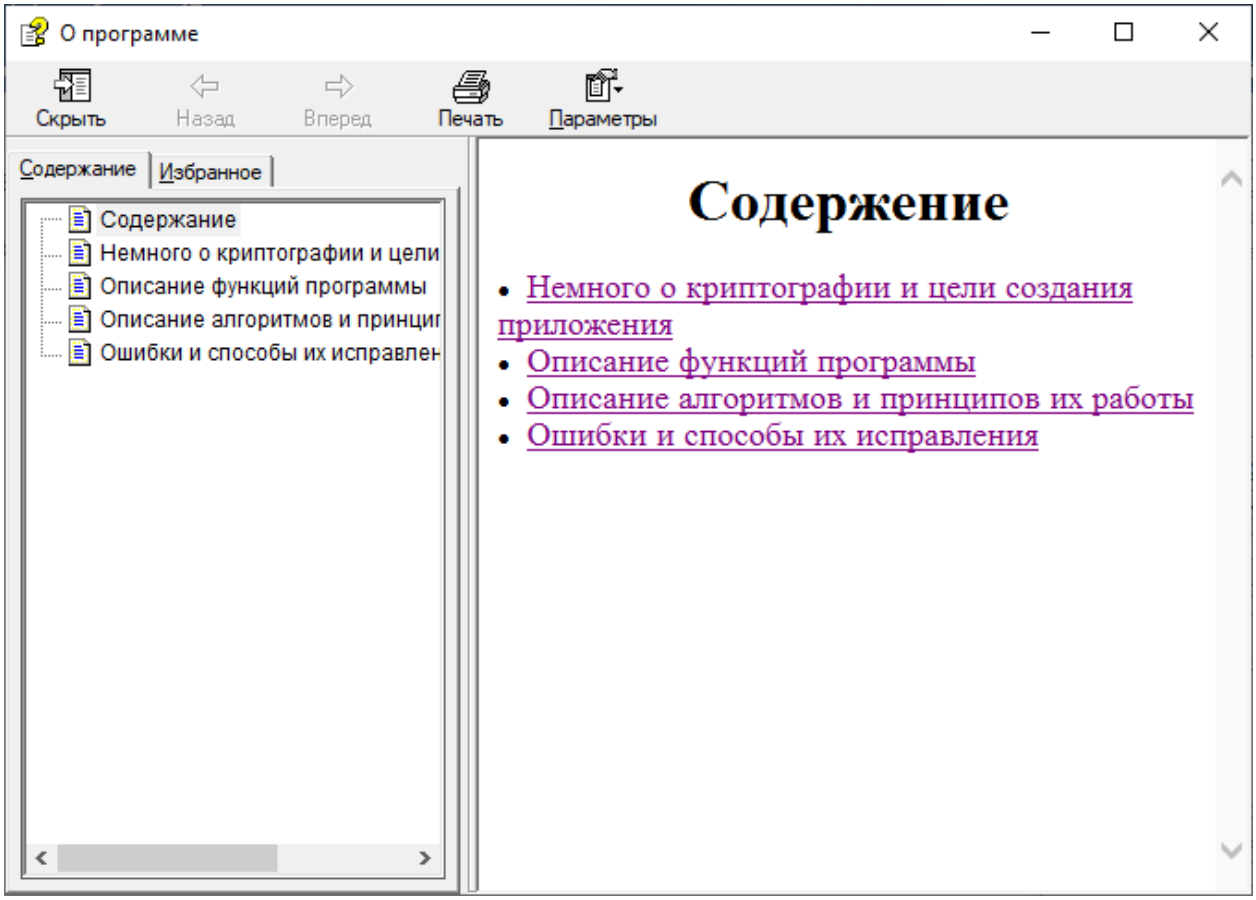

Рис. 2.14 – Реакция программы при отсутствии ошибки, связанной с отсутствием или повреждением файла справки

Следующий момент, когда может произойти ошибка, при попытке пользователя выполнить алгоритм с ключом, который не подходит для

выполнения выбранного метода. Тестовые данные и описание реакции программы на эти данные продемонстрирована в таблице 2.3.

Таблица 2.3 – Тестовые данные при вводе ключа для разных способов и реакция программы на них

№	Способ	Входные данные	Вводимое значение	Реакция программы
1	Транспозиция	«Текст»	Пустая строка	Ошибка (рис. 2.15)
2	Транспозиция	«Текст»	«буквы»	Ошибка (рис. 2.15)
3	Транспозиция	«Текст»	«3»	Ошибка (рис. 2.15)
4	Транспозиция	«Текст»	«1 3»	Ошибка (рис. 2.15)
5	Транспозиция	«Текст»	«1 п»	Ошибка (рис. 2.15)
6	Транспозиция	«Текст»	«1»	Выполнение (рис. 2.15)
7	Транспозиция	«Текст»	«2 1»	Выполнение (рис. 2.15)
8	Транспозиция	«Текст»	«3 2 1»	Выполнение (рис. 2.15)
9	Моноалфавит	«Текст»	Пустая строка	Ошибка (рис. 2.16)
10	Моноалфавит	«Текст»	«буквы»	Ошибка (рис. 2.16)
11	Моноалфавит	«Текст»	«1 3»	Ошибка (рис. 2.16)
12	Моноалфавит	«Текст»	«1 п»	Ошибка (рис. 2.16)
13	Моноалфавит	«Текст»	«4»	Выполнение (рис. 2.16)
14	Полиалфавит	«Текст»	Пустая строка	Ошибка (рис. 2.17)
15	Полиалфавит	«Текст»	«буквы»	Выполнение (рис. 2. 17)
16	Полиалфавит	«Текст»	«1 3»	Выполнение (рис. 2. 17)
17	Полиалфавит	«Текст»	«1 п»	Выполнение (рис. 2. 17)
18	Полиалфавит	«Текст»	«4»	Выполнение (рис. 2. 17)

Продолжение таблицы 2.3

№	Способ	Входные данные	Вводимое значение	Реакция программы
19	XOR	«Текст»	Пустая строка	Ошибка (рис. 2.18)
20	XOR	«Текст»	«буквы»	Выполнение (рис. 2.18)
21	XOR	«Текст»	«1 3»	Выполнение (рис. 2.18)
22	XOR	«Текст»	«1 п»	Выполнение (рис. 2.18)
23	XOR	«Текст»	«4»	Выполнение (рис. 2.18)
24	Блокнот	«Текст»	Пустая строка	Ошибка (рис. 2.19)
25	Блокнот	«Текст»	«1 3»	Ошибка (рис. 2.19)
26	Блокнот	«Текст»	«1 п»	Ошибка (рис. 2.19)
27	Блокнот	«Текст»	«4»	Ошибка (рис. 2.19)
28	Блокнот	«Текст»	«букв»	Ошибка (рис. 2.19)
29	Блокнот	«Текст»	«буквы»	Выполнение (рис. 2.19)
30	RSA	«Текст»	Пустая строка	Ошибка (рис. 2.20)
31	RSA	«Текст»	«буквы»	Ошибка (рис. 2.20)
32	RSA	«Текст»	«1 3»	Ошибка (рис. 2.20)
33	RSA	«Текст»	«1»	Ошибка (рис. 2.20)
34	RSA	«Текст»	«7 143»	Выполнение (рис. 2.20)
35	RSA	«Текст»	«103 143»	Выполнение (рис. 2.20)

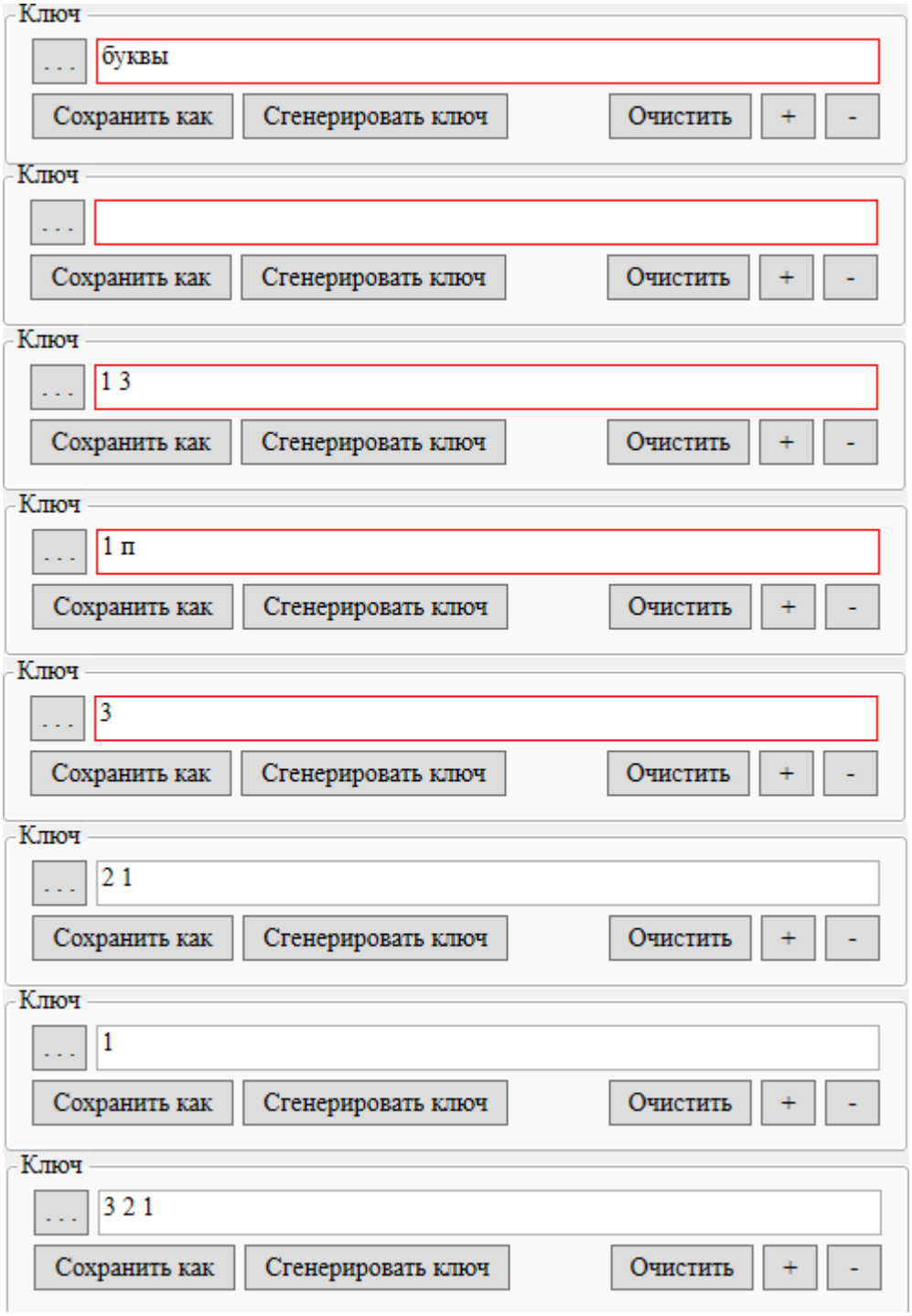

Рис. 2.15 – Реакция программы на тестовые данные при вводе ключа, способ «Транспозиция»

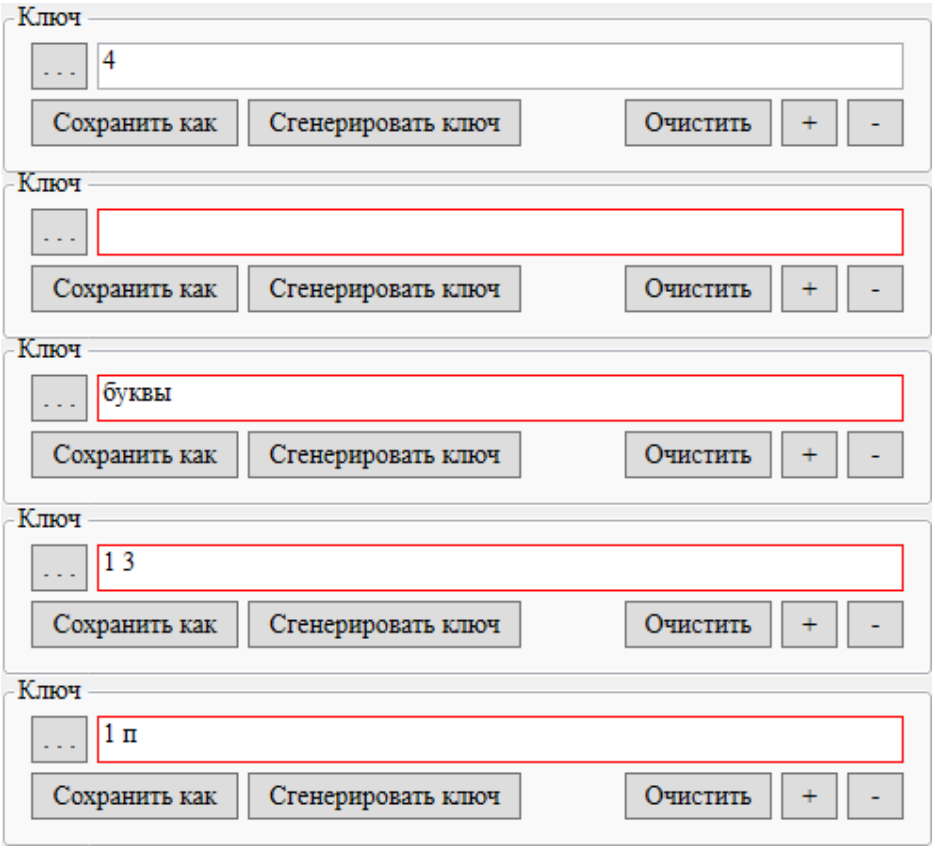

Рис. 2.16 – Реакция программы на тестовые данные при вводе ключа, способ «Моноалфавит»

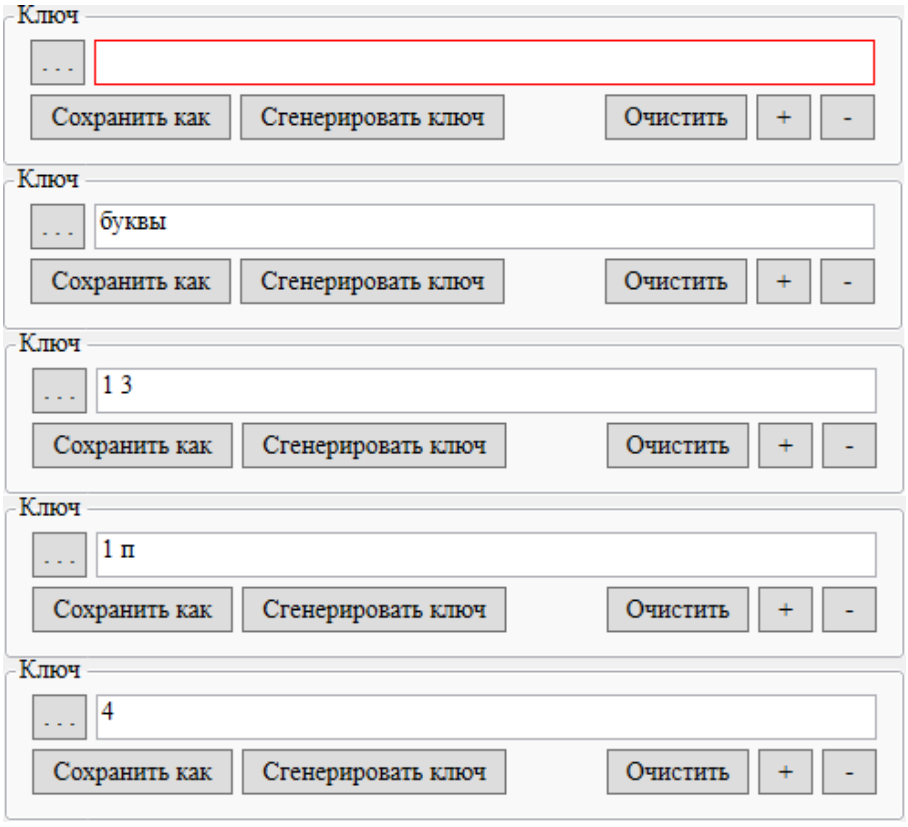

Рис. 2.17 – Реакция программы на тестовые данные при вводе ключа, способ «Полиалфавит»

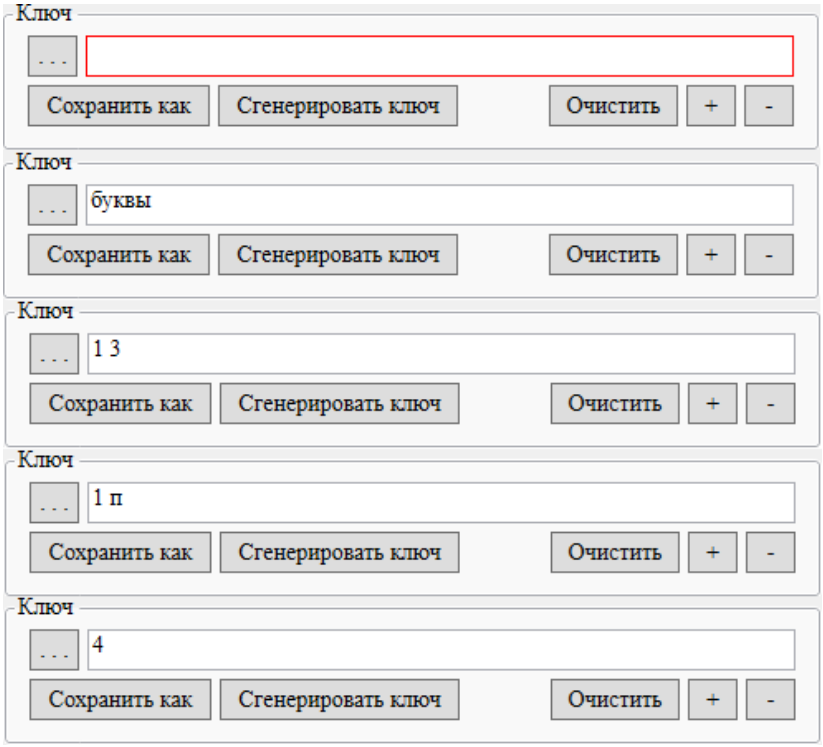

Рис. 2.18 – Реакция программы на тестовые данные при вводе ключа, способ «XOR»

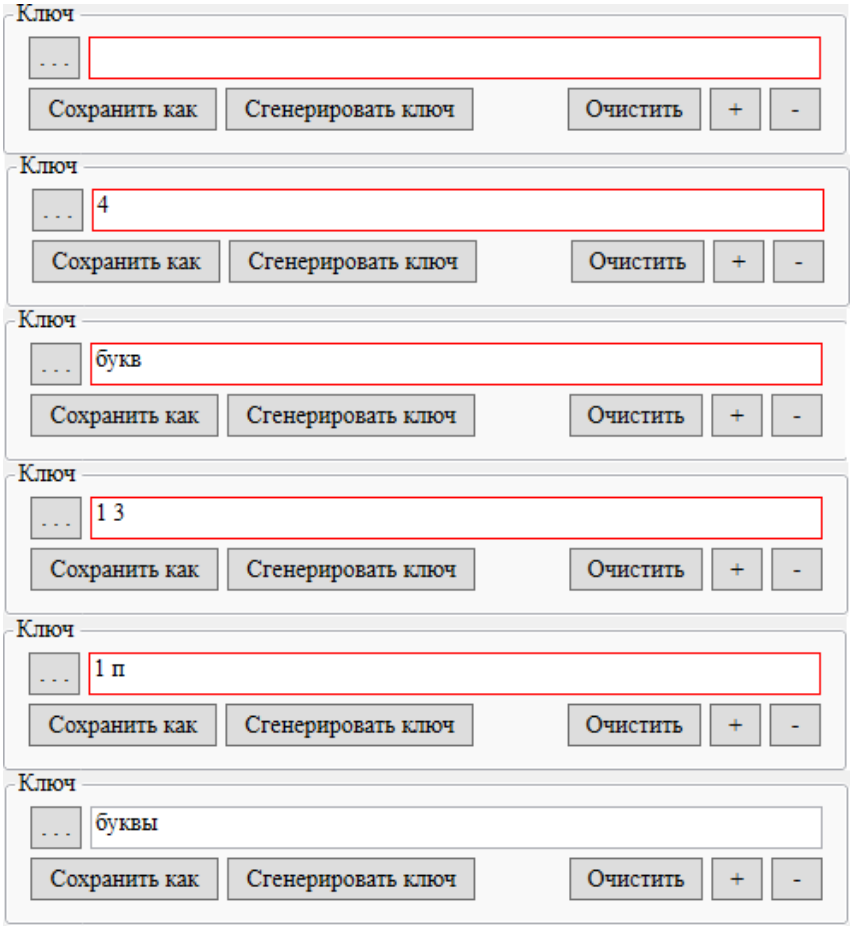

Рис. 2.19 – Реакция программы на тестовые данные при вводе ключа, способ «Одноразовый блокнот»

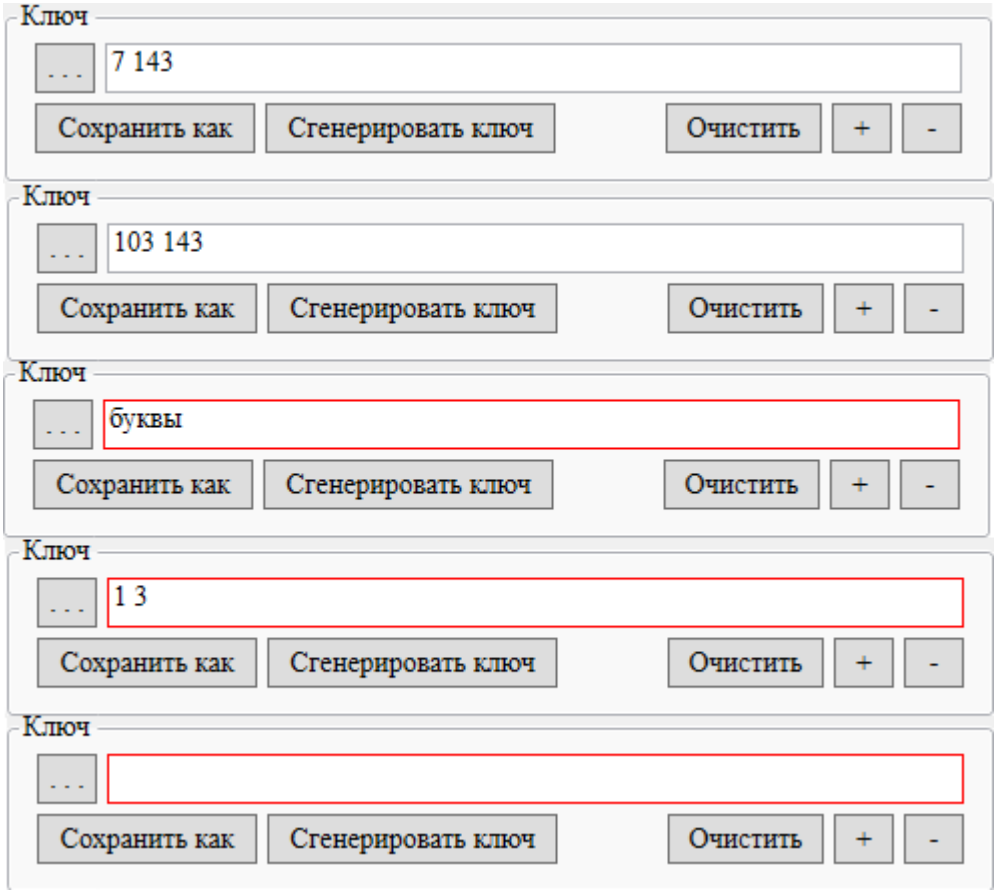

Рис. 2.20 – Реакция программы на тестовые данные при вводе ключа, способ «RSA»

Следующий момент, когда может произойти ошибка, при попытке пользователя сгенерировать ключ заданного размера. Тестовые данные и описание реакции программы на эти данные продемонстрирована в таблице 2.4.

Таблица 2.4 – Тестовые данные при вводе размера ключа для его генерации и реакция программы на них для разных способов

№	Способ	Вводимое значение	Реакция программы
1	Транспозиция	Пустая строка	Ошибка (рис. 2.21)
2	Транспозиция	«5»	Выполнение (рис. 2.21)
3	Полиалфавит	Пустая строка	Ошибка (рис. 2.22)
4	Полиалфавит	«5»	Выполнение (рис. 2.22)
5	XOR	Пустая строка	Ошибка (рис. 2.23)
6	XOR	«5»	Выполнение (рис. 2.23)

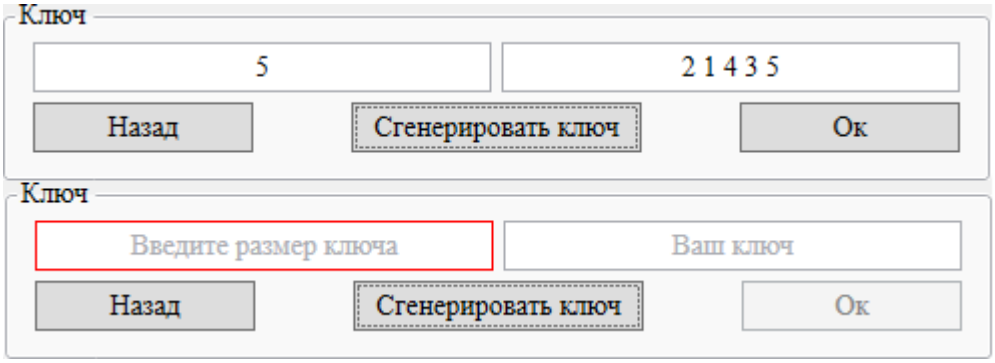

Рис. 2.21 – Реакция программы на тестовые данные при генерации размерных ключей, способ «Транспозиция»

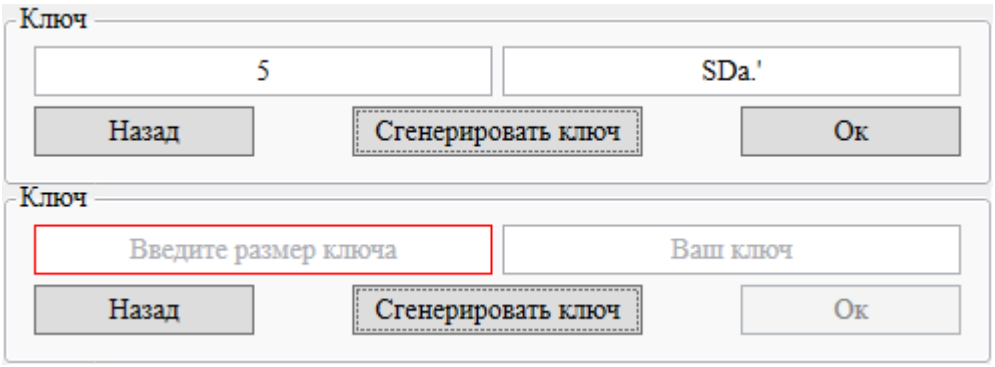

Рис. 2.22 – Реакция программы на тестовые данные при генерации размерных ключей, способ «Полиалфавит»

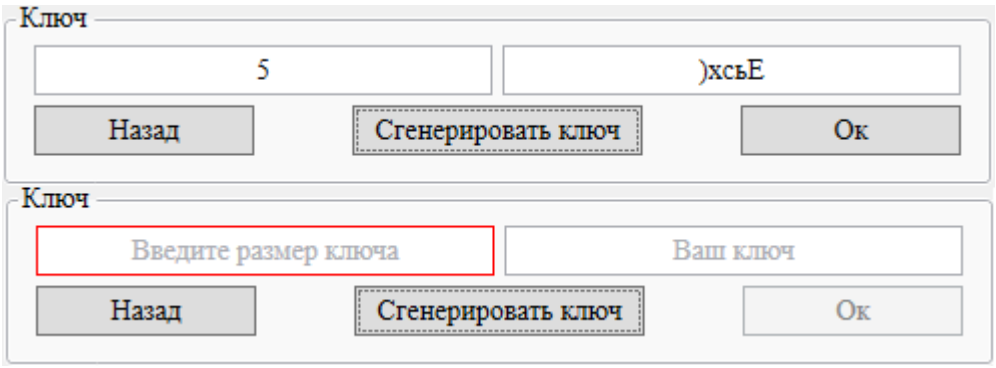

Рис. 2.23 – Реакция программы на тестовые данные при генерации размерных ключей, способ «XOR»

Следующий момент, когда может произойти ошибка, при попытке пользователя сгенерировать открытый и закрытый ключи. Тестовые данные и описание реакции программы на эти данные продемонстрирована в таблице 2.5.

Таблица 2.5 – Тестовые данные при вводе значений для генерации открытого и закрытого ключей и реакция программы на них

№	Способ	Вводимое значение	Реакция программы
1	RSA	Пустая строка P	Ошибка (рис. 2.24)
2	RSA	Пустая строка Q	Ошибка (рис. 2.24)
3	RSA	Пустые строки P и Q	Ошибка (рис. 2.24)
4	RSA	P = Q = 5	Ошибка (рис. 2.24)
5	RSA	P = Q = 11	Ошибка (рис. 2.24)
6	RSA	P = 11 Q = 13	Выполнение (рис. 2.24)
7	RSA	P = 13 Q = 11	Выполнение (рис. 2.24)

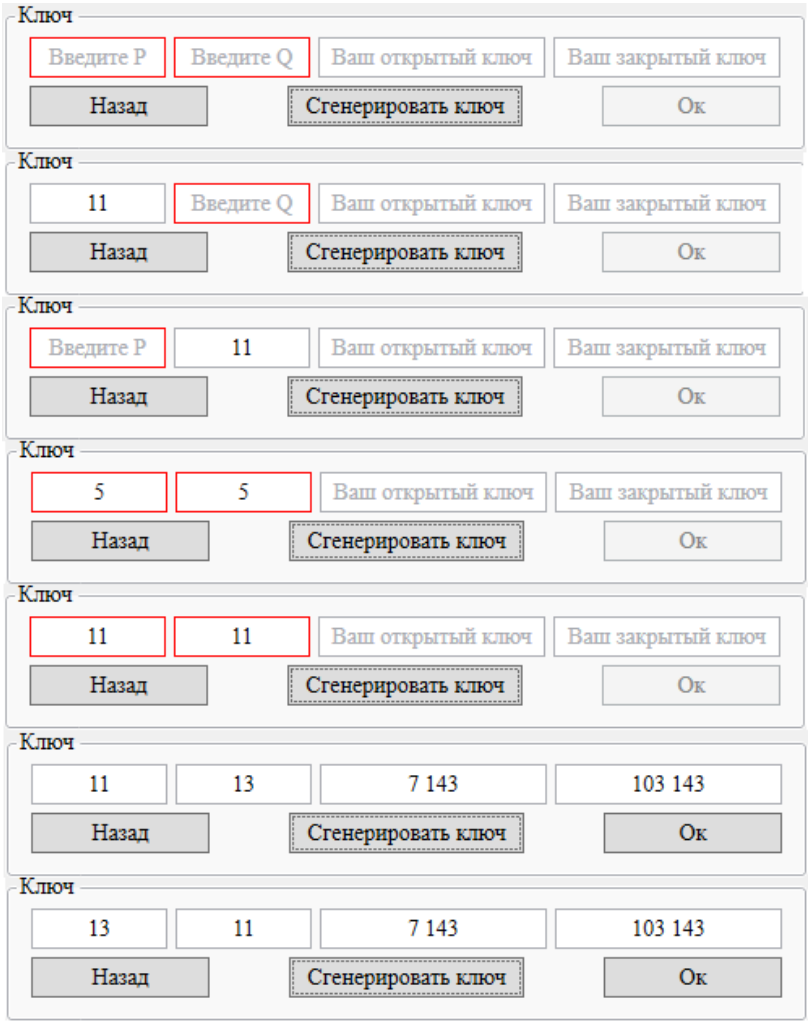

Рис. 2.24 – Реакция программы на тестовые данные при генерации открытых и закрытых ключей RSA

Последняя ситуация, когда может произойти ошибка, если пользователь пытается выполнить операцию, не выбрав способа для её выполнения. Реакция программы продемонстрирована на рис. 2.25 и рис. 2.26.

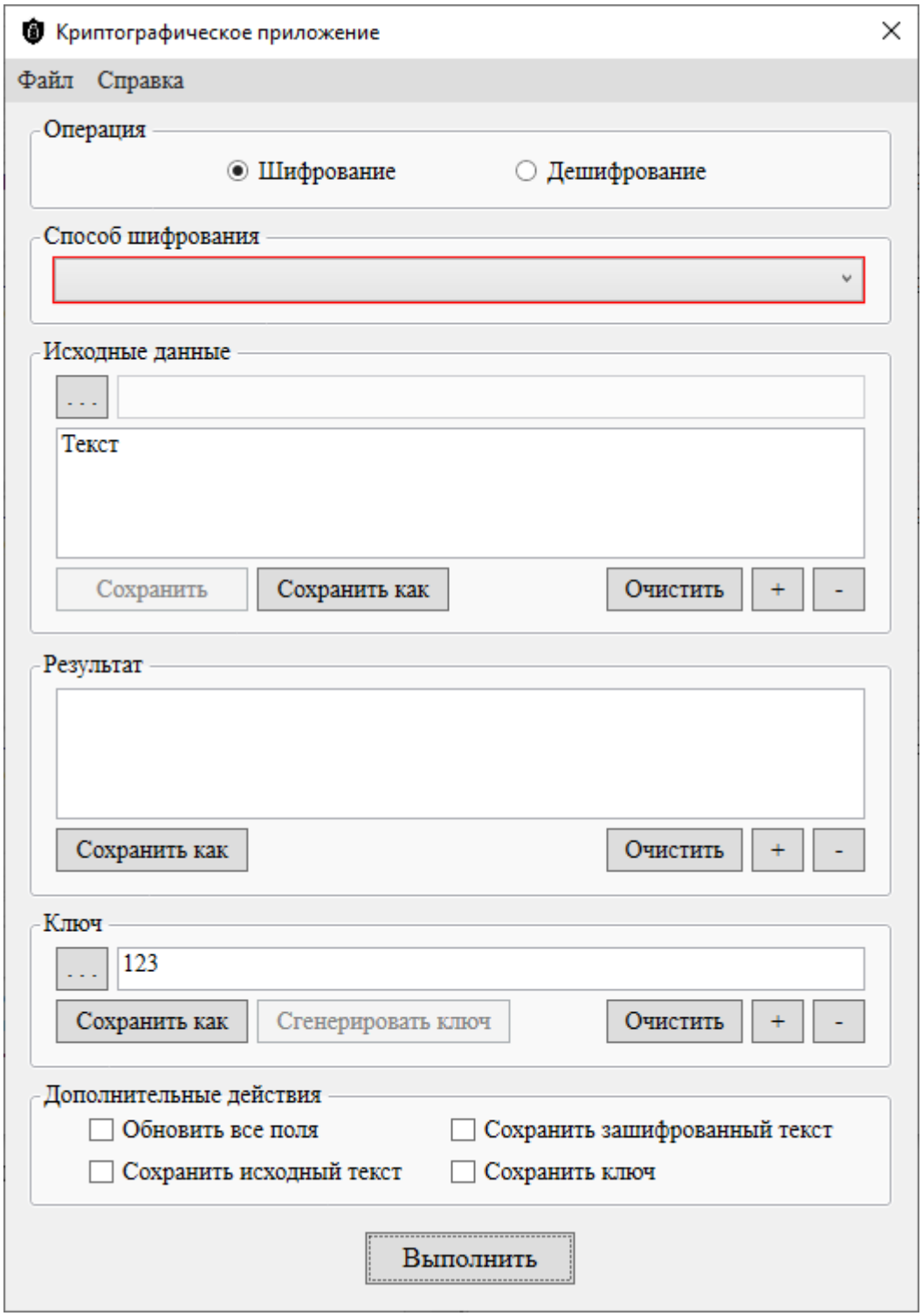

Рис. 2.25 – Реакция программы, если пользователь не выбрал способ перед тем как выполнить операцию

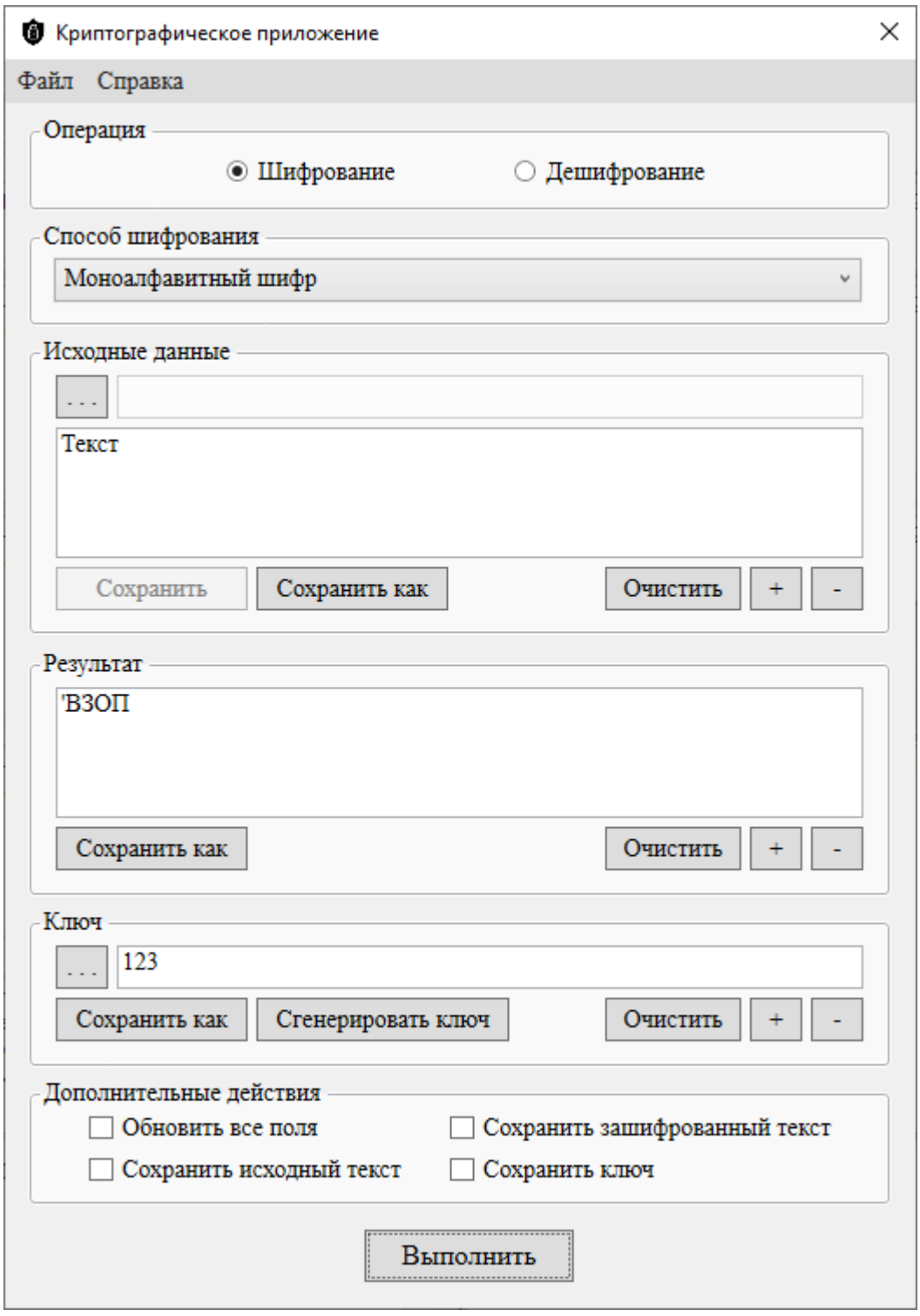

Рис. 2.26 – Реакция программы, если пользователь выбрал способ перед тем как выполнить операцию

Тестовые данные во всех таблицах были взяты с таким расчётом, чтобы при минимальном количестве тестов охватить все возможные ошибки, возникающие при работе с программой.

Подводя итоги тестирования и отладки можно сказать, что все функциональные и нефункциональные требования, поставленные в предпроектном обследовании, были выполнены.

2.6 Руководство по использованию программы

2.6.1 Руководство системного программиста

2.6.1.1 Общие сведения о программе

Программа шифрования файлов предназначена для шифрования и дешифрования файлов. Она призвана защитить информацию от несанкционированного доступа к ней посторонних лиц и для этого использует самые популярные и относительно надёжные алгоритмы.

Программа выполняет следующие функции:

- обновление всех полей;
- сохранение исходного текста;
- сохранение зашифрованного или дешифрованноно текста;
- сохранение ключа;
- выполнение выбранной операции;
- генерация случайных ключей;
- открытие файлов в программе;
- перезапись данных в выбранном файле;
- очистка полей;
- уменьшение или увеличение размеров шрифта.

Также программа способна вызывать файл справки формата chm.

2.6.1.2 Структура программы

Ознакомиться со структурой программы можно на рис. 2.27.

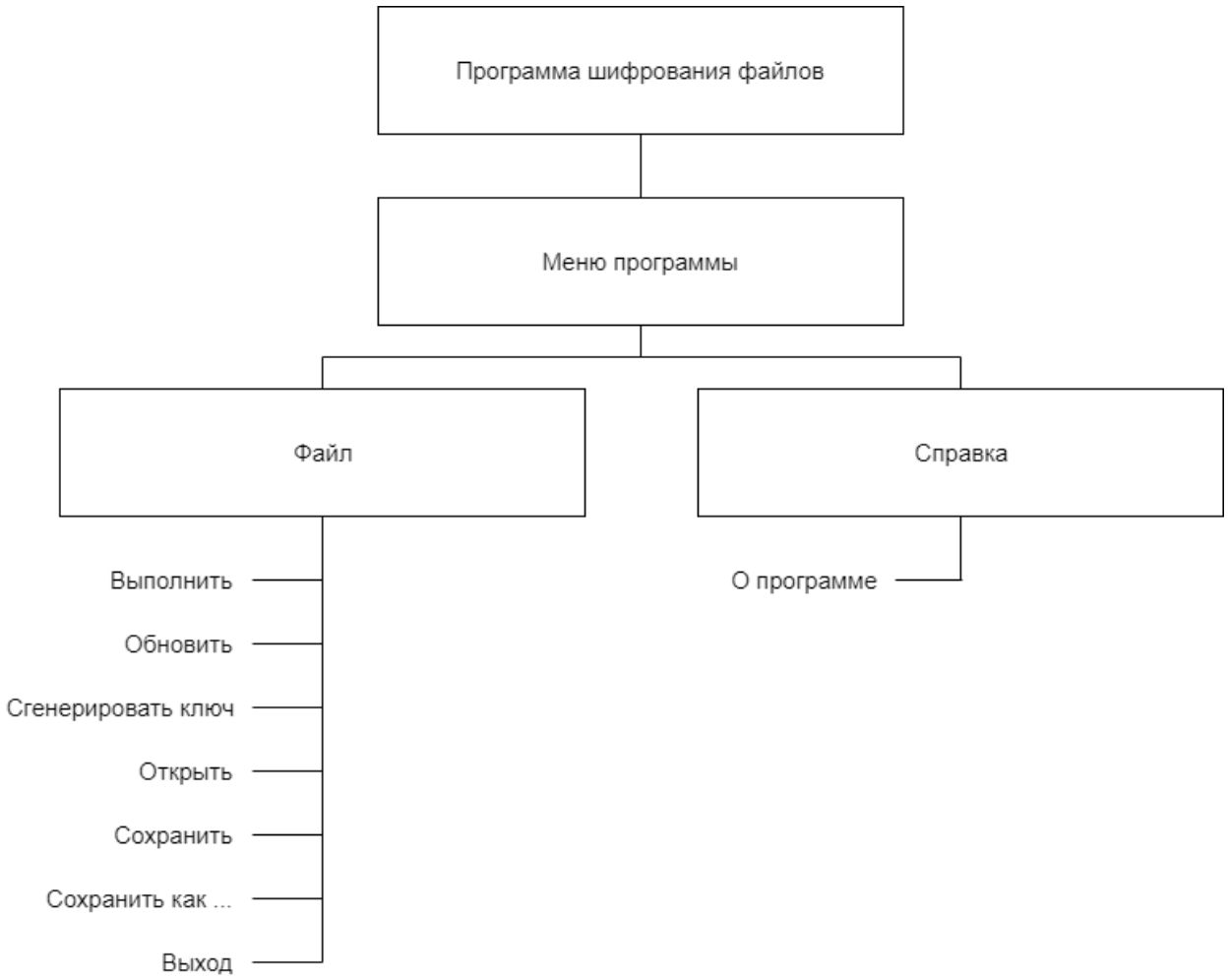

Рис. 2.27 – Структура программы

2.6.1.3 Настройка программы

Программа не требует дополнительных настроек и после запуска уже готова к использованию.

2.6.1.4 Проверка программы

Самый простой способ проверить работоспособность программы – это выполнить шифрование и дешифрование любого текста любым методом. Для этого выберите желаемый метод, в поле исходного текста введите сообщение или откройте файл с текстом, хранящийся на компьютере, введите ключ, верный для выбранного метода и нажмите кнопку «Выполнить». В случае обнаружения ошибок программа укажет на это. Если ошибок не было обнаружено, в поле результата будет выведен зашифрованный текст.

Теперь необходимо проверить операцию дешифрования. Для этого поменяйте операцию, скопируйте полученный шифротекст и вставьте его на

на место исходного. Не меняя ключа и метода ещё раз нажмите кнопку «Выполнить». Результатом правильного выполнения программы будет текст, который вы до этого зашифровали.

2.6.1.5 Дополнительные возможности

В программе имеются следующие дополнительные возможности:

- «Обновить все поля»;
- «Сохранить исходный текст»;
- «Сохранить зашифрованный текст»;
- «Сохранить ключ».

Каждая из этих функций выполняется только в случае успешного завершения операции и, если пользователь отметил их на выполнение. Для того чтобы отметить их необходимо навестись курсором на квадратик, расположенный рядом с дополнительным действием и отметить его щелчком левой кнопки мыши.

Дополнительные функции в программе продемонстрированы на рис. 2.28.

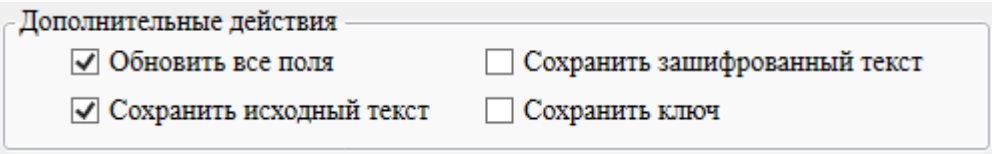

Рис. 2.28 – Дополнительные функции в окне программы

2.6.1.6 Сообщения системному программисту

Для системного программиста в программе содержиться всего одно сообщение. Оно появляется при возникновении ошибки связанной с файлом справки. Если файл справки был повреждён, случайно удалён или другое, тогда при попытке открыть его в ходе выполнения программы появится следующее сообщение (рис. 2.29).

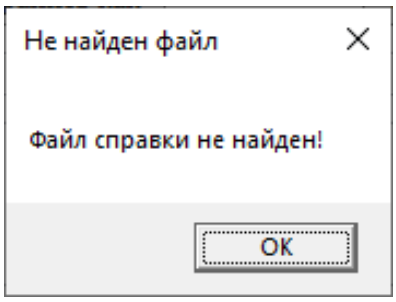

Рис. 2.29 – Реакция программы на возникновение ошибки, связанной с отсутствием или повреждением файла справки

Чтобы устранить эту ошибку нужно просто переустановить программу на рабочий компьютер, либо откатить систему к моменту, когда файл справки был в рабочем состоянии.

2.6.2 Руководство программиста

2.6.2.1 Назначение и условия применения программы

Программа призвана защитить информацию от несанкционированного доступа к ней посторонних лиц используя для этого самые популярные и относительно надёжные алгоритмы.

Программа выполняет следующие функции:

- обновление всех полей;
- сохранение исходного текста;
- сохранение зашифрованного или дешифрованноно текста;
- сохранение ключа;
- выполнение выбранной операции;
- генерация случайных ключей;
- открытие файлов в программе;
- перезапись данных в выбранном файле;
- очистка полей;
- уменьшение или увеличение размеров шрифта.

Также программа способна вызывать файл справки формата chm.

Системные требования для выполнения программы:

- ОС: 32-битная Windows 7/8/10;
- процессор: Intel Core i3-4030U 1.90 GHz;

- ОЗУ: 4 ГБ;
- DirectX: видеокарта, совместимая с версией 11.0 или аналогичная ей;
- свободное место на жестком диске: 512 МБ.

При соблюдении этих требований программа будет успешно выполняться.

2.6.2.2 Характеристики программы

Приложение разработано на языке программирования C# версии 7.0. NET Framework 4.7.1., в среде разработки Visual Studio 2017, в системе для построения клиентских приложений WPF.

2.6.2.3 Обращение к программе

Для запуска программы необходимо дважды кликнуть левой кнопкой мыши по исполняемому файлу «CryptographicApplication.exe».

2.6.2.4 Входные и выходные данные

Входные данные – данные, вводимые или выбранные пользователем в ходе работы программы. К ним относится: выбор операции, выбор способа шифрования или дешифрования, ввод исходного текста, ввод ключей, ввод значений для генерации ключей и файлы, загружаемые в программу.

Выходные данные – данные, полученные в ходе работы пользователя с программой. К ним относится: сгенерированные пользователем ключи, зашифрованный или дешифрованный текст.

2.6.2.5 Сообщения

В программе не содержится сообщений, выдаваемых программисту в ходе её выполнения.

2.6.3 Руководство пользователя

2.6.3.1 Назначение программы

Программа шифрования файлов предназначена для шифрования и дешифрования файлов. Она призвана защитить информацию от несанкционированного доступа к ней посторонних лиц и для этого использует самые популярные и относительно надёжные алгоритмы.

2.6.3.2 Условия выполнения программы

Системные требования:

- ОС: 32-битная Windows 7/8/10;
- процессор: Intel Core i3-4030U 1.90 GHz;
- ОЗУ: 4 ГБ;
- DirectX: видеокарта, совместимая с версией 11.0 или аналогичная ей;
- свободное место на жестком диске: 512 МБ.

При соблюдении этих требований программа будет успешно выполняться.

2.6.3.3 Выполнение программы

Перед тем, как начать использование программы её нужно установить. Сделать это можно запустив файл «Установщик программы шифрования файлов.exe» (рис. 2.30).

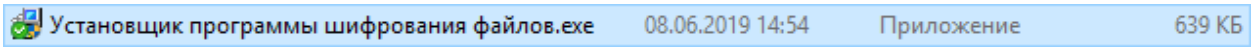

Рис. 2.30 – Файл для установки программы шифрования файлов

После того, как файл будет запущен, появиться окно выбора языка установки (рис. 2.31).

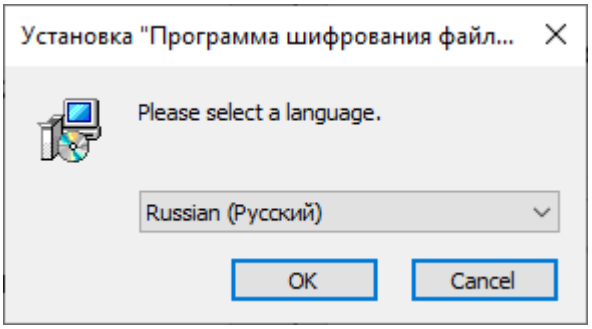

Рис. 2.31 – Окно выбора языка установки

Выберите язык, на котором будет происходить дальнейшая установка программы и нажмите кнопку «ОК» для перехода к следующему этапу.

В следующем этапе появиться окно приветствия (рис. 2.32).

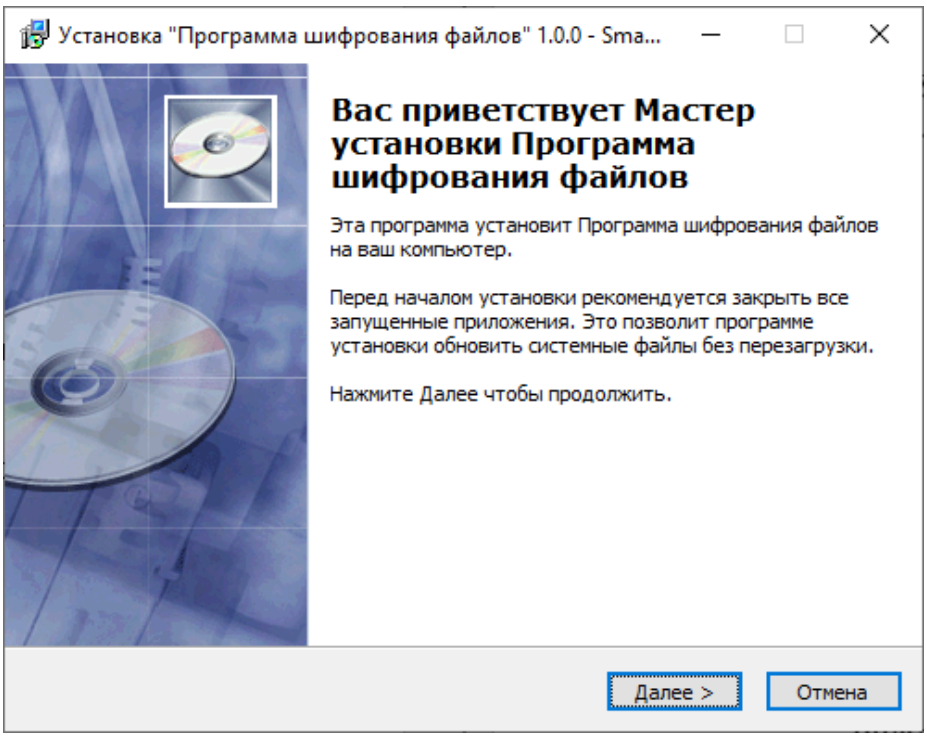

Рис. 2.32 – Окно приветствия

Здесь написаны некоторые действия, которые рекомендуется сделать перед началом установки. Нажмите кнопку «Далее» для перехода к следующему этапу.

В следующем этапе появиться окно выбора места установки программы (рис. 2.33).

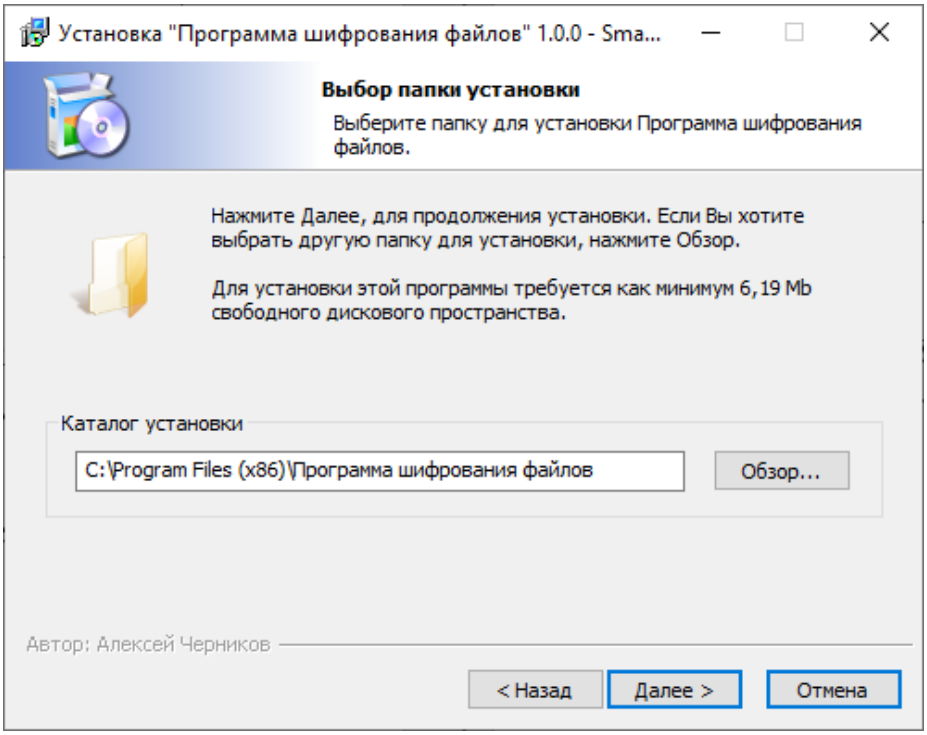

Рис. 2.33 – Окно выбора места установки программы

По умолчанию программа устанавливается по пути «C:\Program Files (x86)\Программа шифрования файлов». Если вы хотите изменить место установки нажмите кнопку «Обзор» и выберите подходящее место. После того как место установки программы будет выбрано нажмите кнопку «Далее».

В следующем этапе появиться окно выбора места установки ярлыков программы в меню «Пуск» (рис. 2.34).

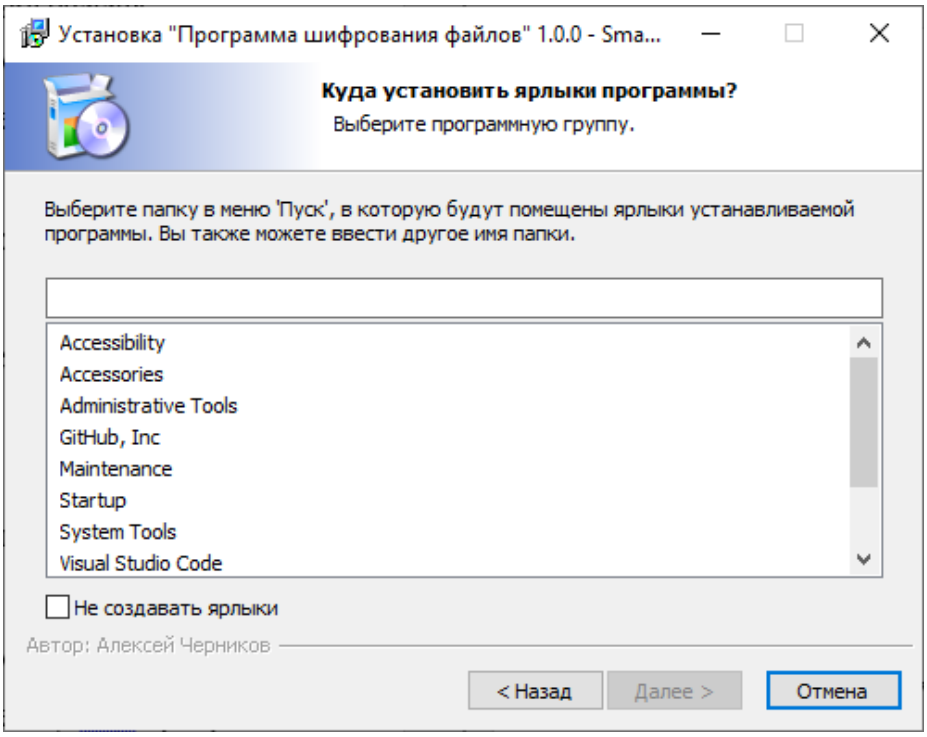

Рис. 2.34 – Окно выбора места установки ярлыков программы в меню «Пуск»

Выберите место, куда будут помещены ярлыки устанавливаемой программы. Если вам не требуются ярлыки программы в меню «Пуск» отметьте пункт «Не создавать ярлыки». Для перехода к следующему этапу нажмите кнопку «Далее».

В следующем этапе появиться окно выбора дополнительных ярлыков (рис. 2.35).

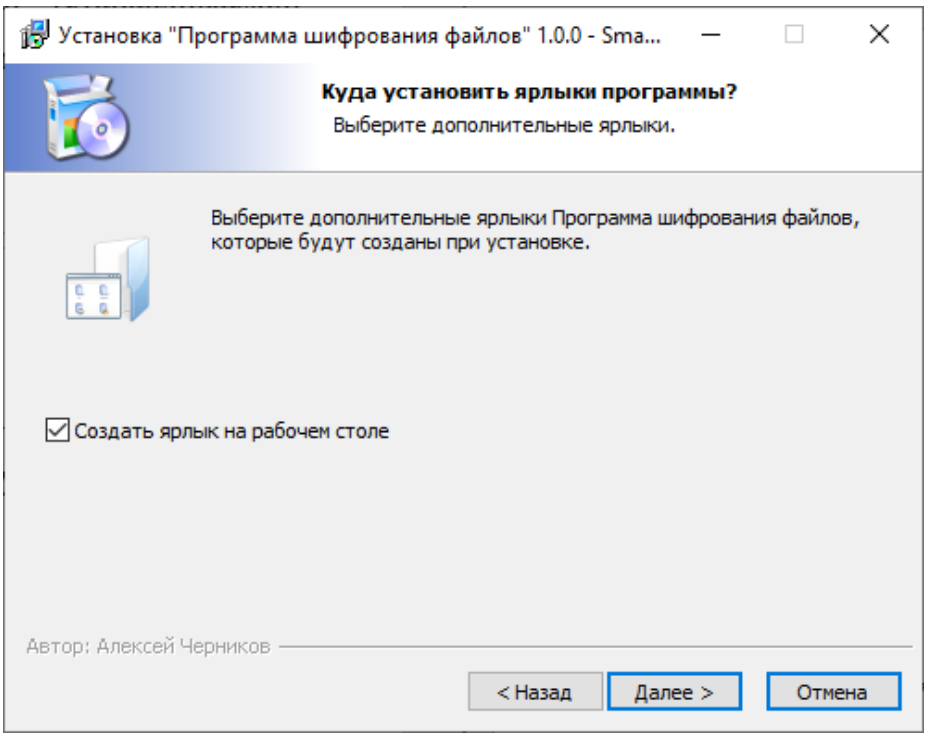

Рис. 2.35 – Окно выбора дополнительных ярлыков

Выберите дополнительные ярлыки, которые будут созданы при установке и нажмите кнопку «Далее».

В следующем этапе появиться окно проверки информации для установки (рис. 2.36).

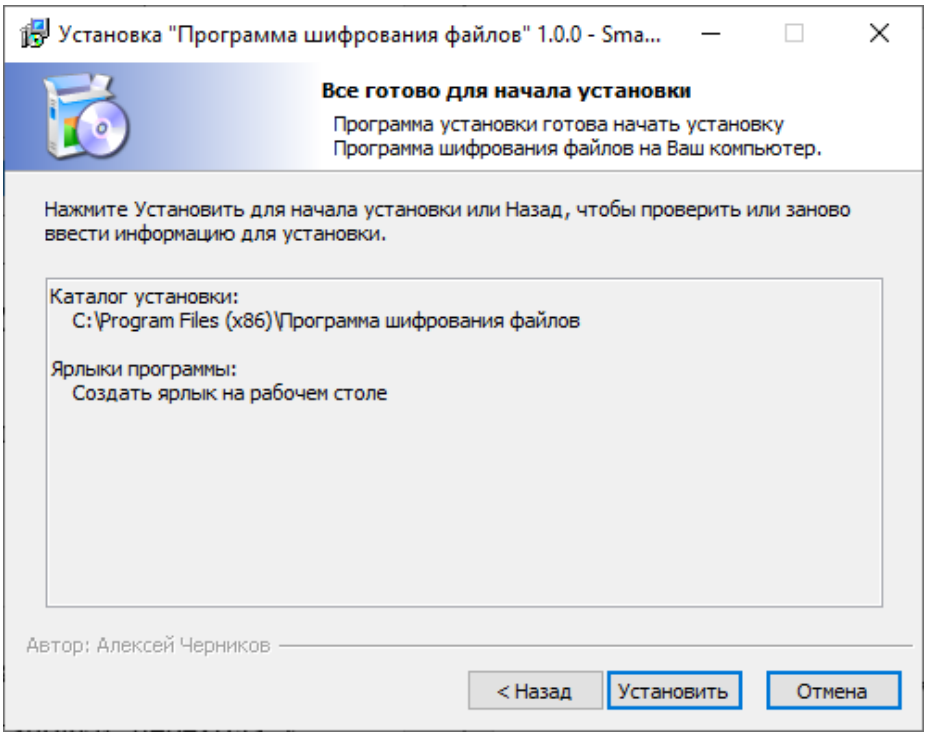

Рис. 2.36 – Окно проверки информации для установки

Здесь будут описаны функции, которые вы выбирали на предыдущих этапах. Нажмите кнопку «Установить» для начала установки или «Назад», чтобы проверить или заново ввести информацию для установки.

Окно установки программы представлено на рис. 2.37.

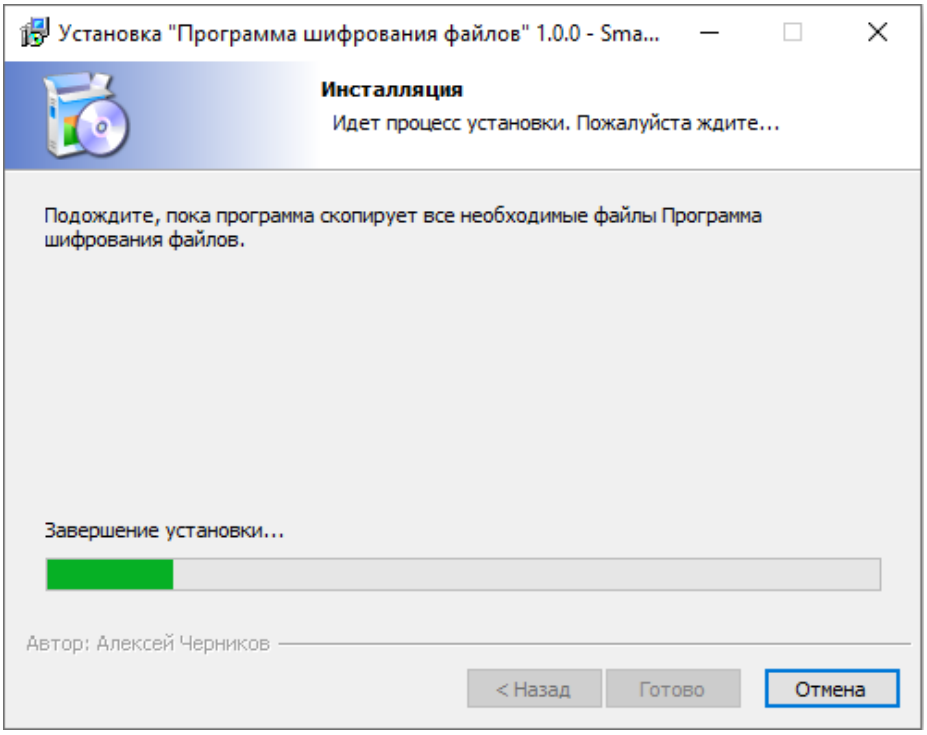

Рис. 2.37 – Окно установки программы

После завершения установки появиться следующее окно (рис. 2.38).

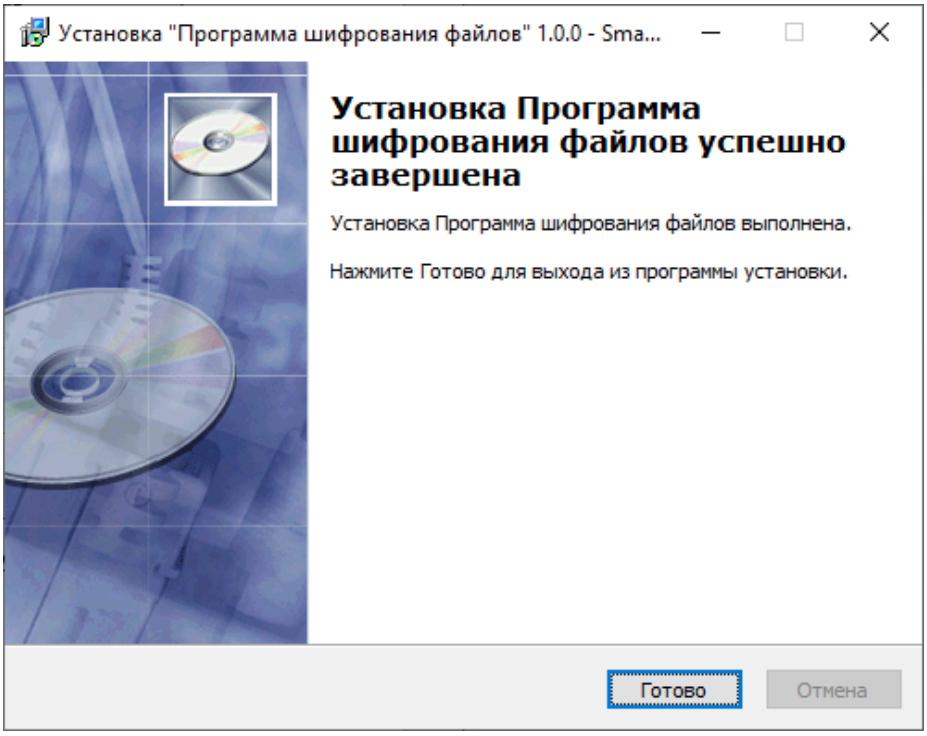

Рис. 2.38 – Окно завершения установки

Нажмите кнопку «Готово» для завершения установки.

Теперь запустите установленную программу двойный нажатием левой кнопки мыши по файлу «Программа шифрования файлов» (рис. 2.39).

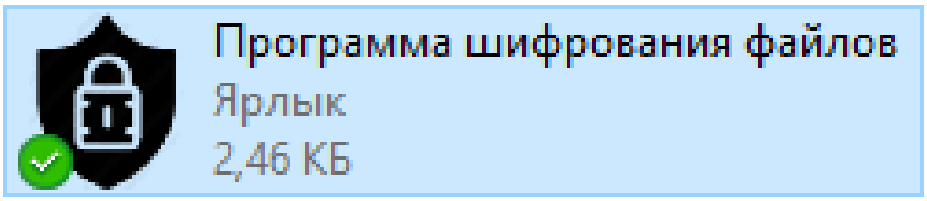

Рис. 2.39 – Файл «Программа шифрования файлов»

После запуска программы появиться следующее окно (рис. 2.40).

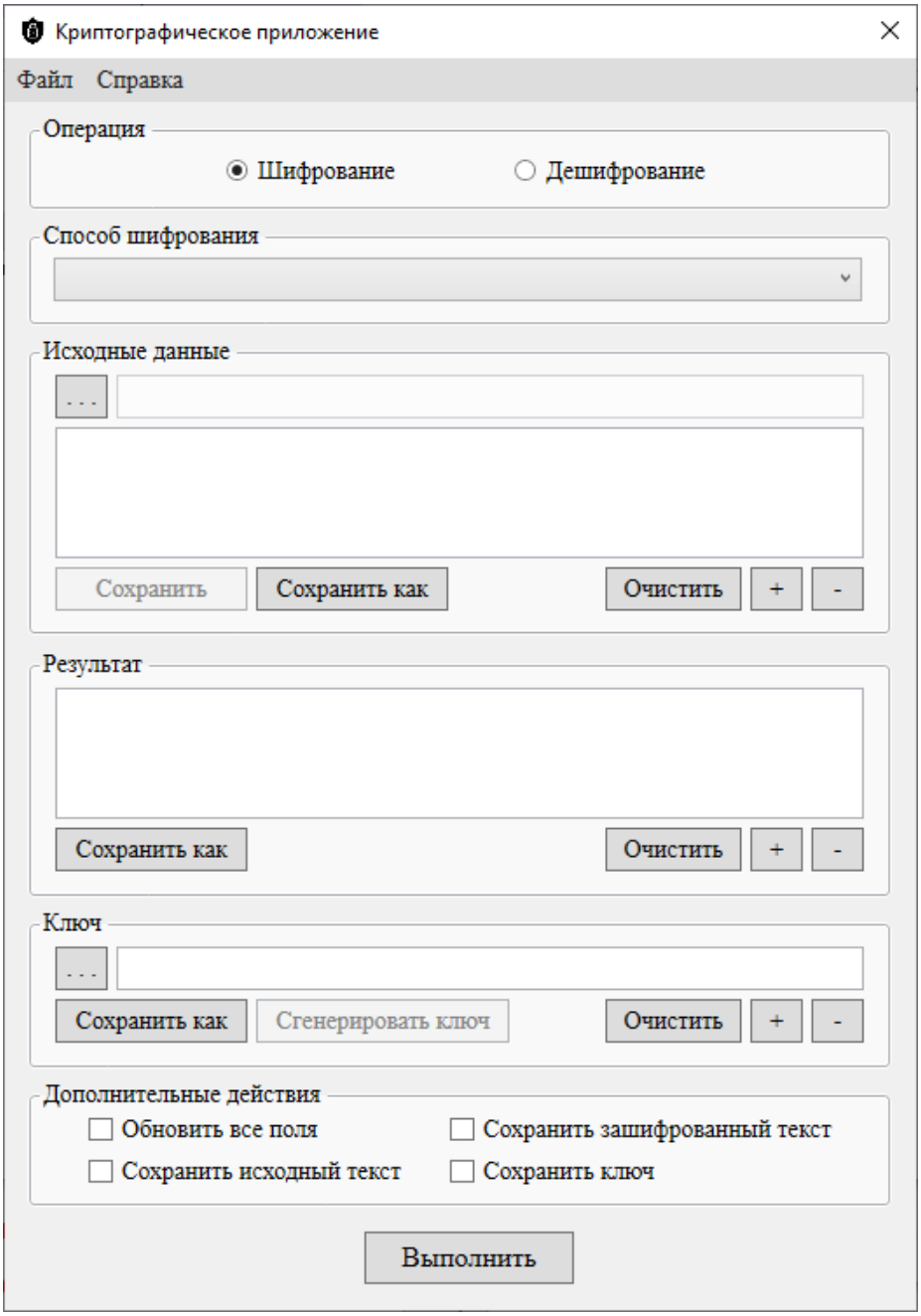

Рис. 2.40 – Главное окно программы

Это главное окно программы. Отсюда доступны все остальные функции программы. Необходимые функциями можно выбирать при помощи компьютерной мыши или нажатием определённых клавиш на клавиатуре, список которых можно посмотреть в приложении 1, таблице 1.1 «горячие» клавиши программы.

Ознакомимся с меню программы. Оно имеет две вкладки: «Файл» и «Справка». В первой вкладке находятся основные функции программы, во второй – дополнительные (рис. 2.41).

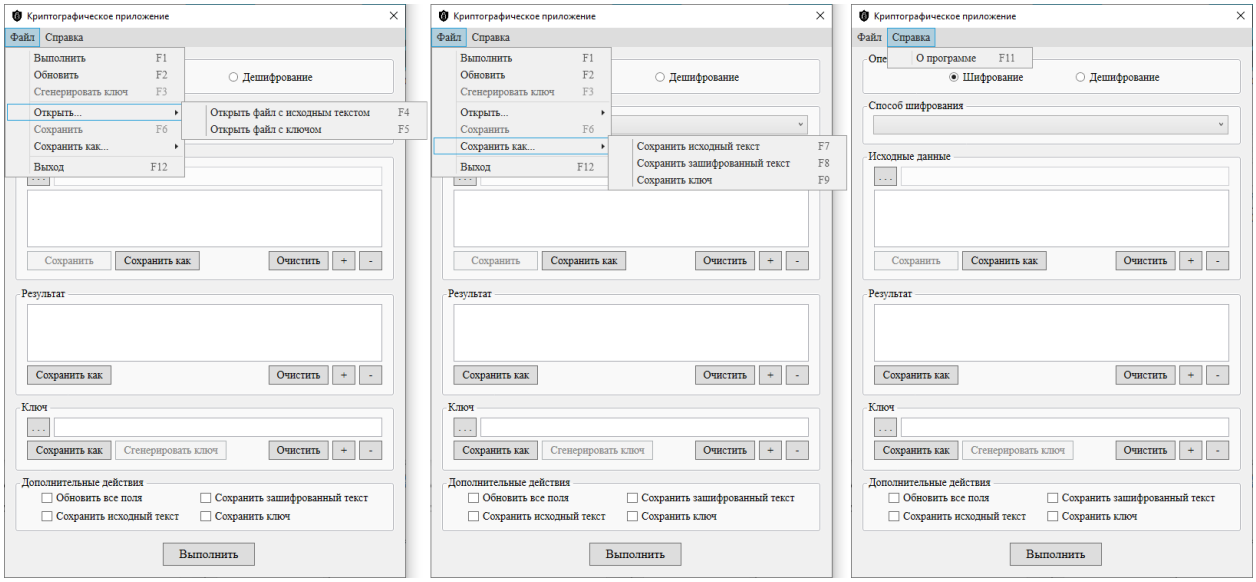

Рис. 2.41 – Функции программы, доступные из меню

Как можно заметить некоторые функции изначально заблокированы, они станут доступны после выполнения определённых действий, о которых будет рассказано позже. Остальные функции будут продемонстрированны в ходе демонстрации основных задач программы, о которых речь пойдёт далее.

Если вы работаете с программой в первые, рекомендуется ознакомиться со основной информацией о ней. Основную информацию о программе можно прочитать в её справке. Чтобы вызвать справку нужно перейти в пункт меню «Справка» и выбрать подпункт «О программе». Нажав на него левой кнопкой мыши или вызвав его при помощи клавиши «F11» перед вами откроется следующее окно (рис. 2.42).

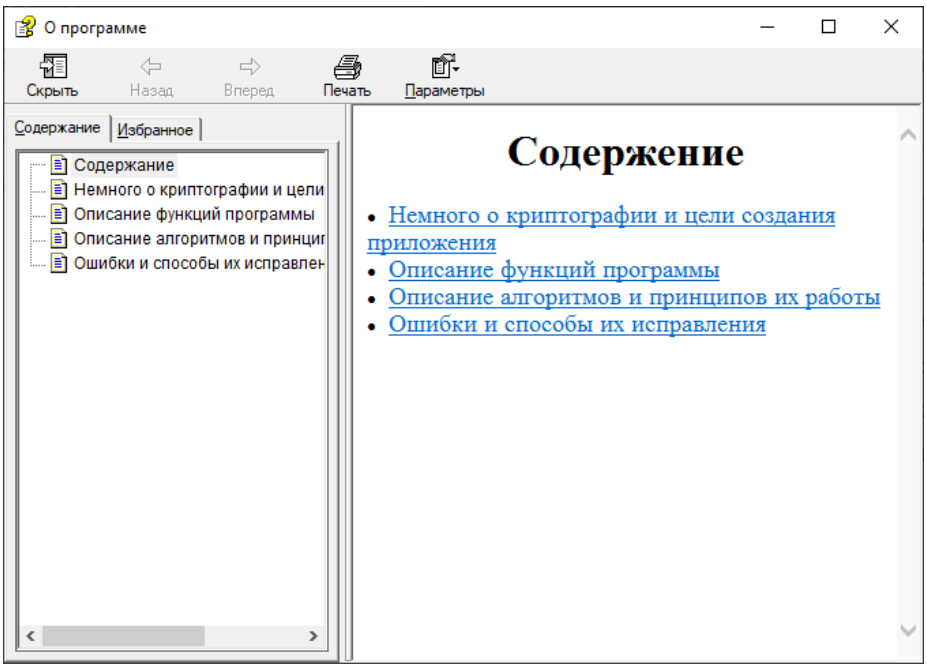

Рис. 2.42 – Окно справки, вкладка «Содержание»

Всего в справке содержится пять вкладок:

- «Содержание»;
- «Немного о криптографии и цели создания приложения»;
- «Описание функций программы»;
- «Описание алгоритмов и принципов их работы»;
- «Ошибки и способы их исправления».

Перемещаться между вкладками можно как при помощи содержания в левой части окна справки, так и при помощи открытого изначально в правой части, как на рис. 2.42.

Перейдём на вкладку «Немного о криптографии и цели создания приложения» (рис. 2.43).

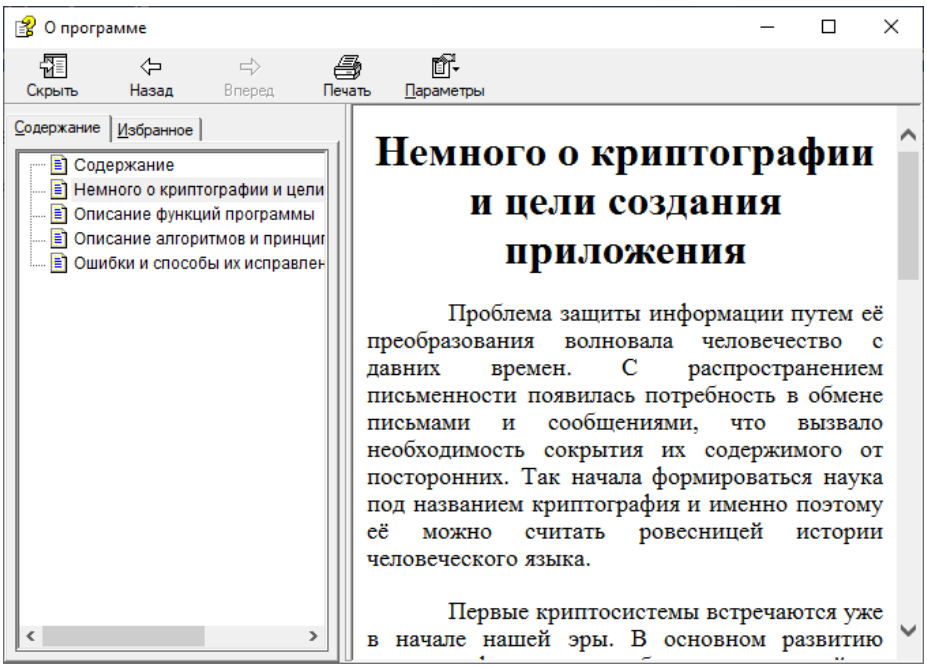

Рис. 2.43 – Окно справки, вкладка «Немного о криптографии и цели создания приложения»

Эта вкладка содержит информацию о той части науки криптологии, которая отвечает за защиту информации от посторонних лиц, то есть о криптографии. Здесь кратко рассказывается история развития этой науки и её роль в современном мире, а также с какой целью была созданна используемая программа.

Следующая вкладка «Описание функций программы» (рис. 2.44).

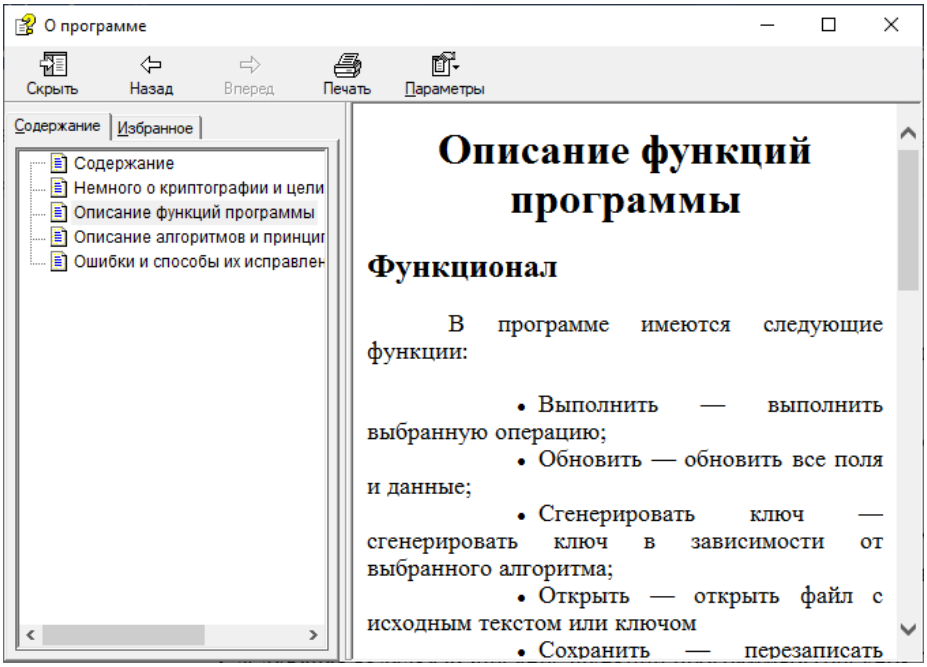

Рис. 2.44 – Окно справки, вкладка «Описание функций программы»

На этой вкладке описаны все функции программы и что они делают. Также здесь можно посмотреть «горячие» клавиши.

Вкладка «Описание алгоритмов и принципов их работы» представлена на рис. 2.45.

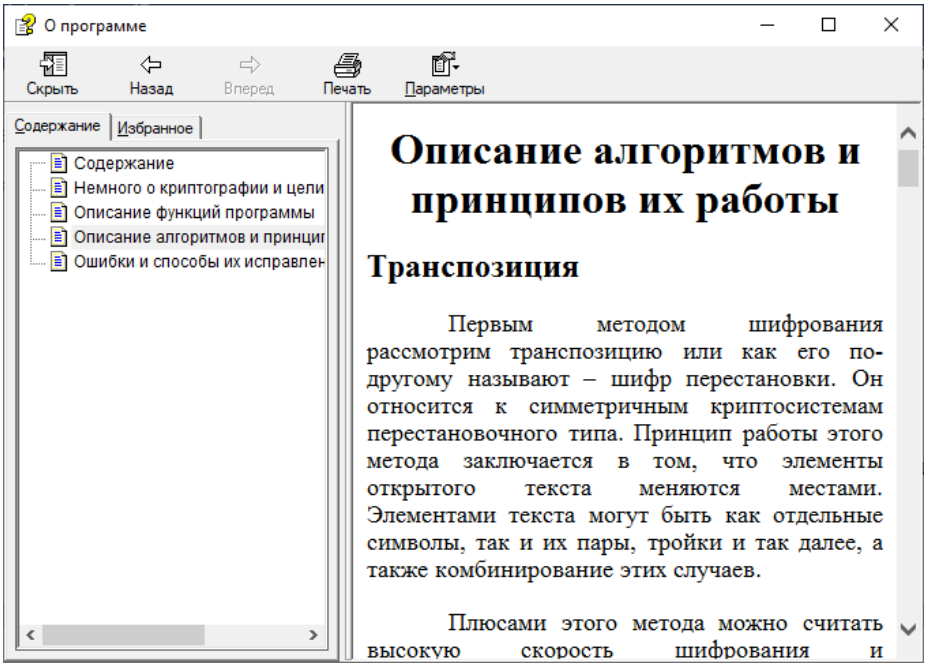

Рис. 2.45 – Окно справки, вкладка «Описание алгоритмов и принципов их работы»

Здесь описан каждый способ шифрования и дешифрования, который есть в программе. Описано к какому типу они относятся, как они работают, их плюсы и минусы.

Последняя вкладка «Ошибки и способы их исправления» (рис. 2.46).

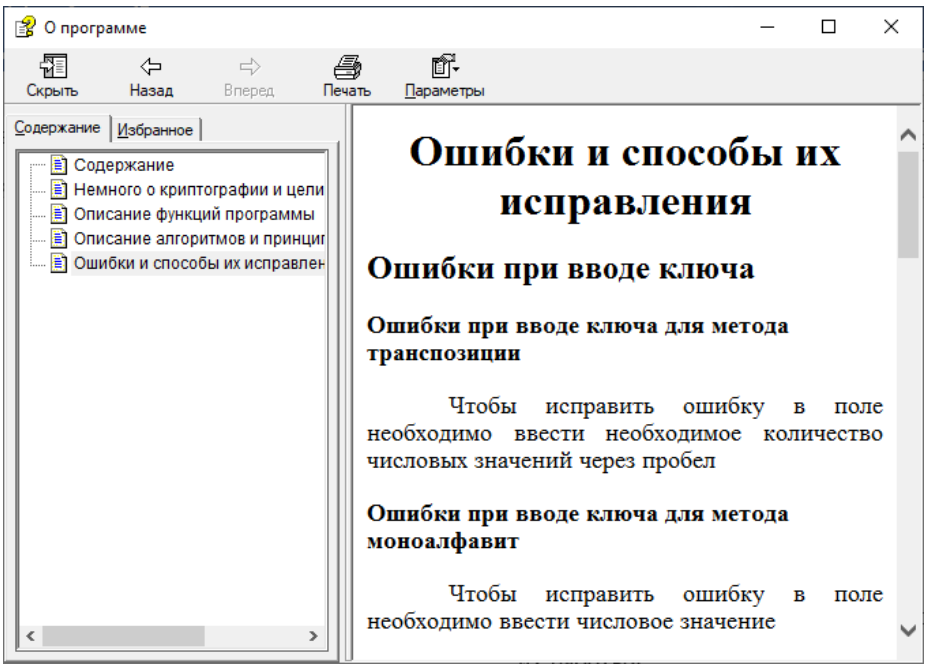

Рис. 2.46 – Окно справки, вкладка «Ошибки и способы их исправления»

Вкладка содержит информацию о всех типах ошибок, которые могут встретиться в программе и описывает последовательность действий для их исправления.

Теперь рассмотрим основные функции, для которых и была создана программа, а именно шифрование и дешифрование текстовой информации. Рассматривать будем по способам, демонстрируя для каждого способа операцию шифрования и дешифрования. Чтобы выбрать способ шифрования или дешифрования необходимо нажать на следующее поле (рис. 2.47)

Рис. 2.47 – Поле выбора способа

Нажав на него вылетит список со всеми доступными способами (рис. 2.48).

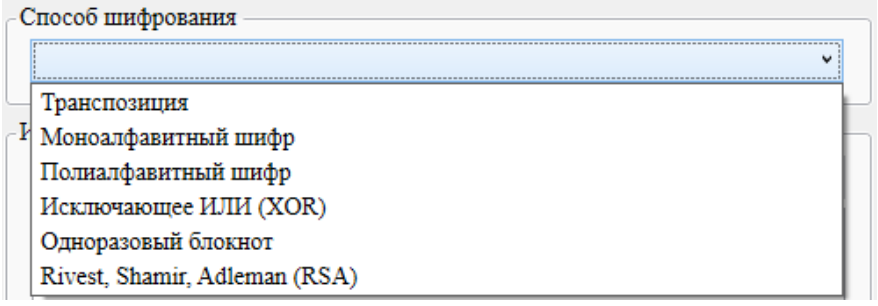

Рис. 2.48 – Список способов сокрытия информации

Первый способ шифрования текста, который будет рассмотрен – «Транспозиция». Выберем его в списке. После этого он отобразится в поле (рис. 2.49).

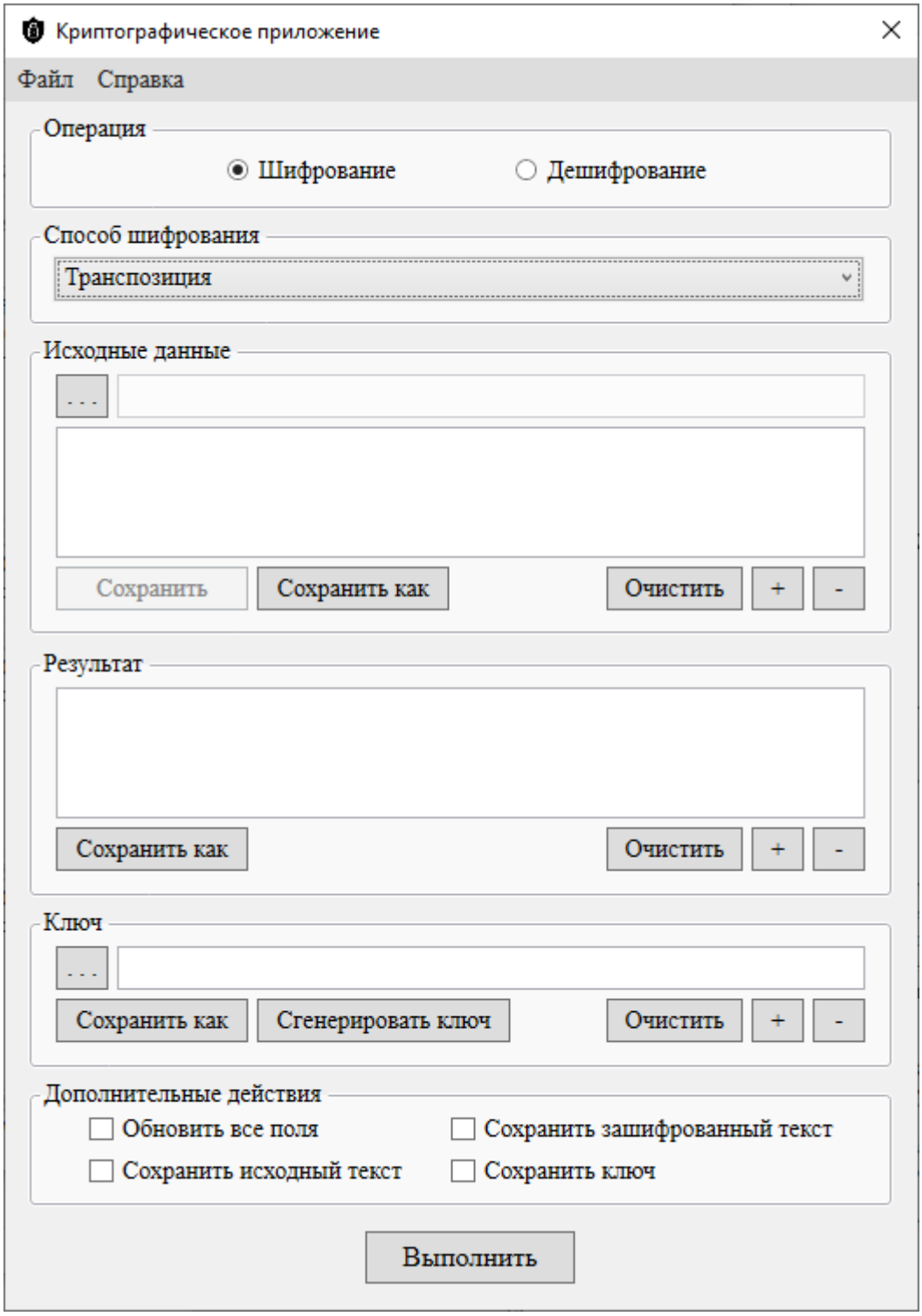

Рис. 2.49 – Способ, отображённый в поле выбора способа

После выбора способа можно заметить, что теперь стала доступна функция «Сгенерировать ключ», которая ранее была недоступна. Подробнее о ней будет рассказано позднее.

У нас выбран способ, теперь необходимо ввести текст, который нужно зашифровать. Это можно сделать либо вручную, либо выбрав уже существующий файл на компьютере. В целях демонстрации всех возможностей программы, мы выберем уже существующий файл. Сделать это можно выбрав пункт меню «Файл», подпункт «Открыть…», подподпункт «Открыть файл с исходным текстом», либо нажать клавишу «F4», либо нажать на кнопку «…» в разделе «Исходные данные». При выполнении любого из этих способов появиться следующее окно (рис. 2.50).

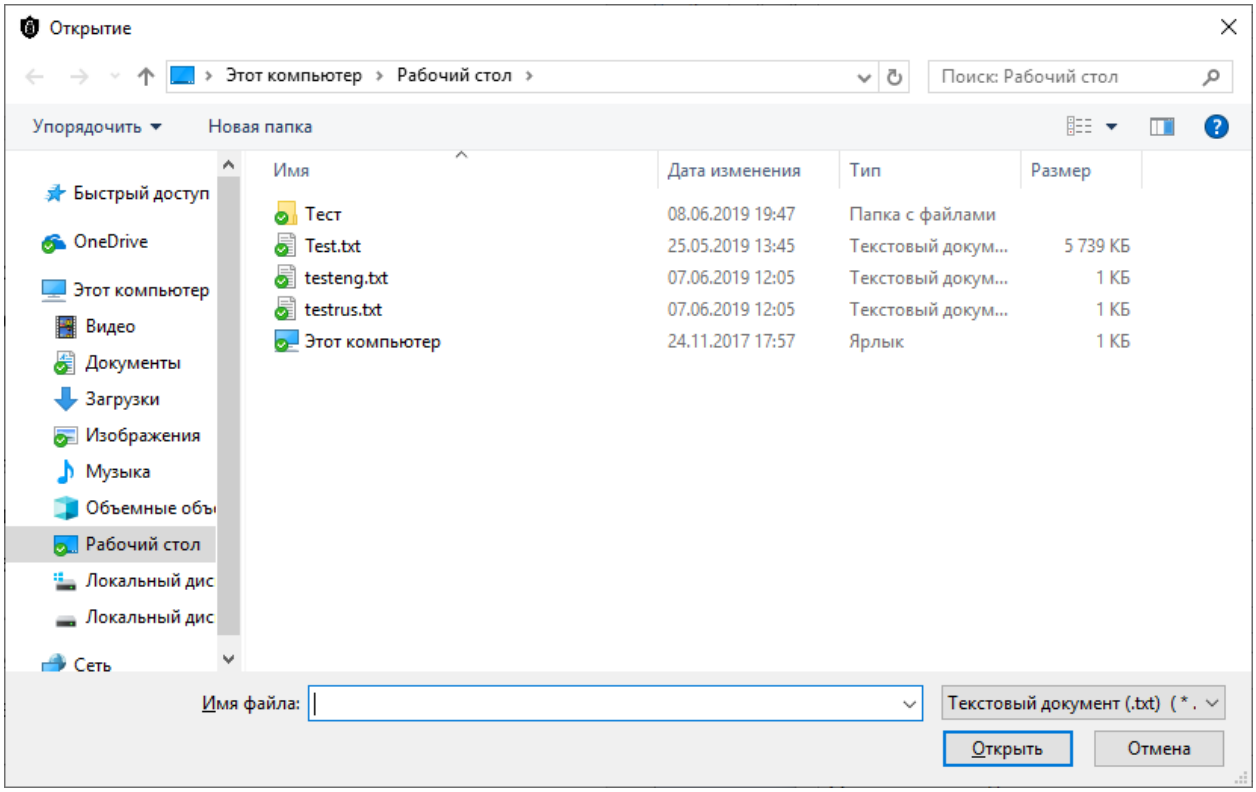

Рис. 2.50 – Окно выбора файла

Выберите нужный файл и нажмите кнопку «Открыть». Файл и путь к нему отобразиться в соответствующих полях программы (рис. 2.51).

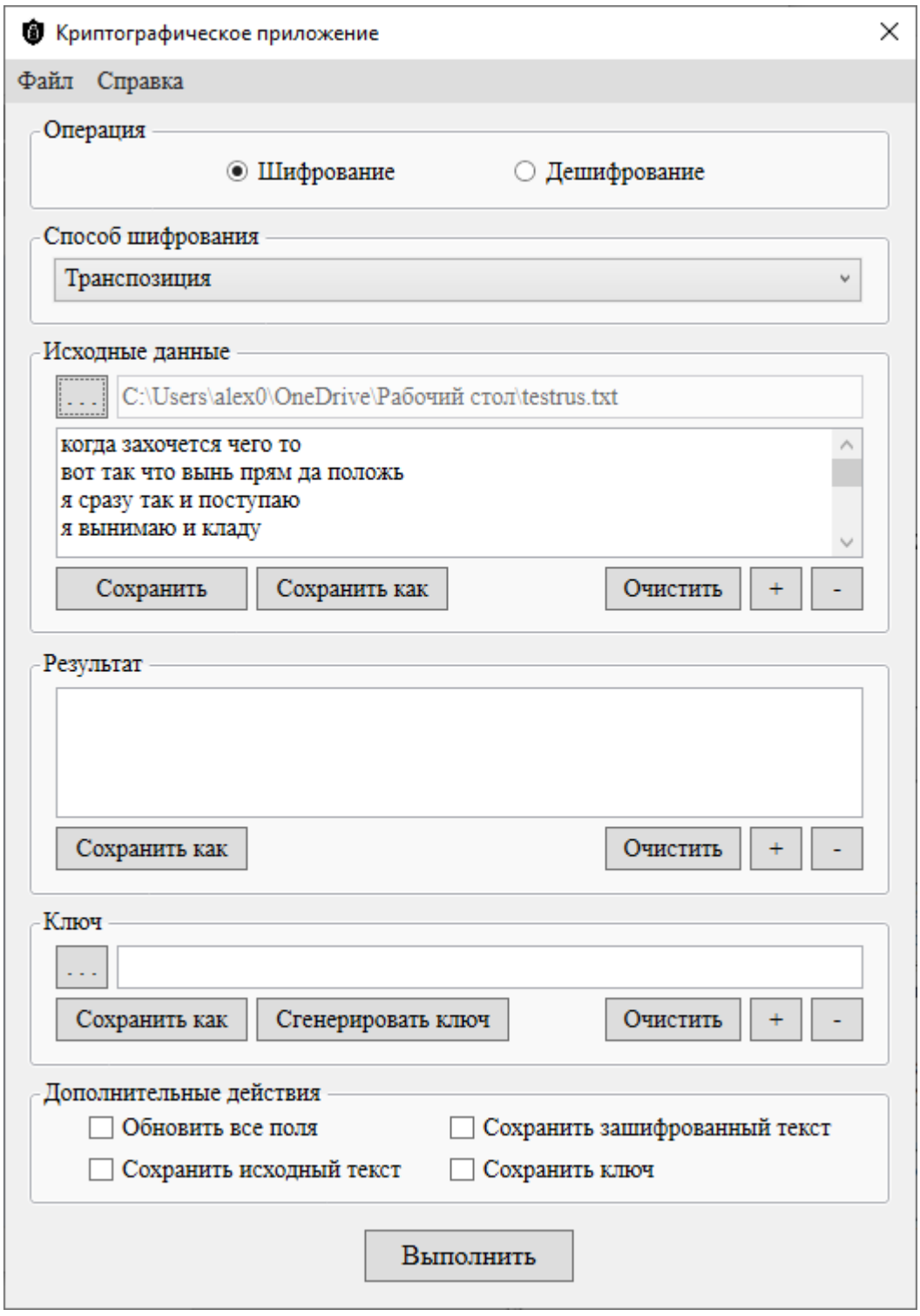

Рис. 2.51 – Открытый в программе файл с исходными данными

Следует отметить, что программа открывает только файлы формата txt.

После выбора файла нам стала доступна вторая ранее недоступная функция «Сохранить». Она отвечает за перезапись исходного файла, то есть если вы захотите изменить текст, вы можете сделать это прямо из программы и нажав на эту кнопку, либо клавишу «F6», либо выбрав соответсвующий

подпункт в пункте меню «Файл», программа перезапишет изменённый исходный текст в выбранный ранее файл.

Ещё есть возможность сохранить изменённый текст в новый файл. Для этого нажмите на кнопку «Созранить как», либо клавишу «F7», либо выбрав подподпункт «Сохранить исходный текст», в подпункте «Сохранить как…», в пункте меню «Файл». При выполнении любого из этих способов откроется следующее окно (рис. 2.52).

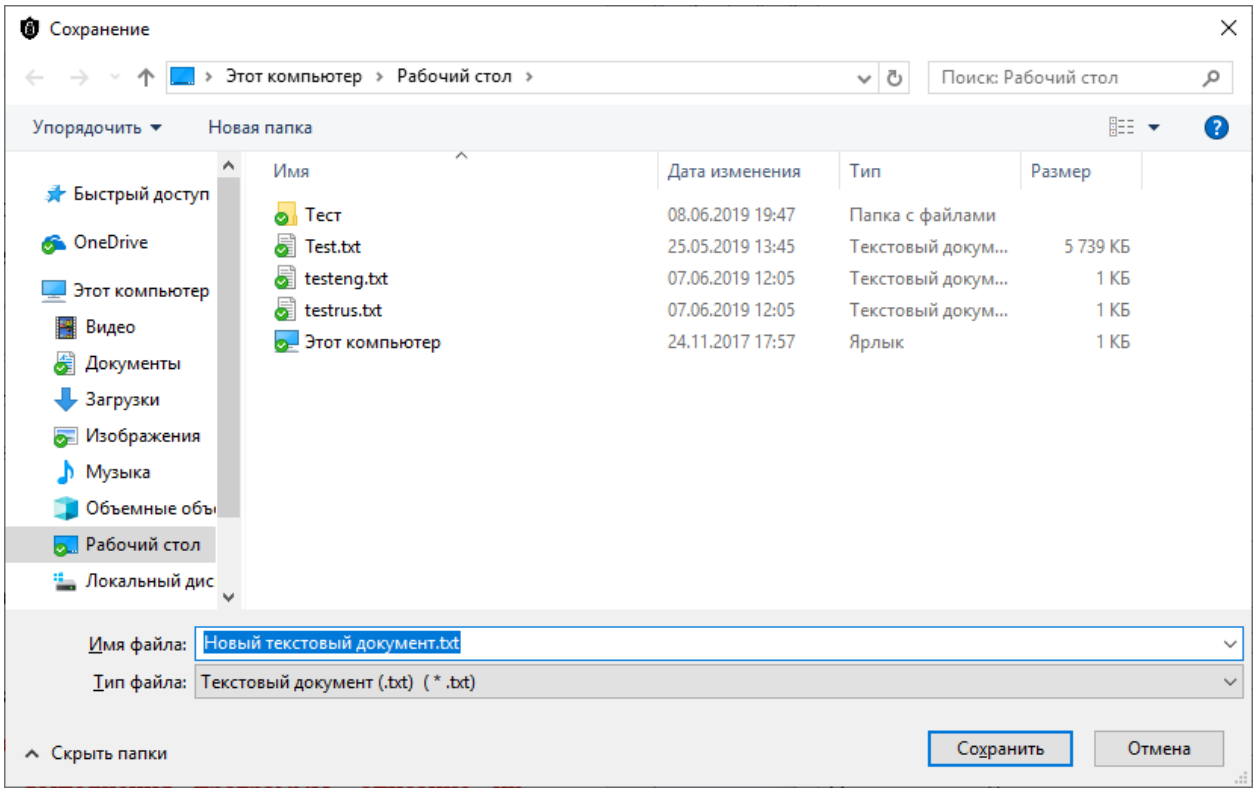

Рис. 2.52 – Окно сохранения исходного текста

Название файла по умолчанию «Новый текстовый документ», при необходимости его можно изменить. Как только определитесь с названием файла и местом его сохранения нажмите кнопку «Сохранить» для сохранения нового файла.

Вернёмся к шифрованию текста. Чтобы его зашифровать нужно ввести ключ. Его также можно выбрать из файла или сгенерировать. Для демонстрации возможностей будем генерировать. Для этого нажмите на кнопку «Сгенерировать ключ», либо клавишу «F3», либо выбрав соответствующий подпункт в пункте меню «Файл». При выполнении любого

из этих способов в главном окне программы изменится раздел «Ключ», перед вами появиться форма для создания ключа выбранного размера (рис. 2.53).

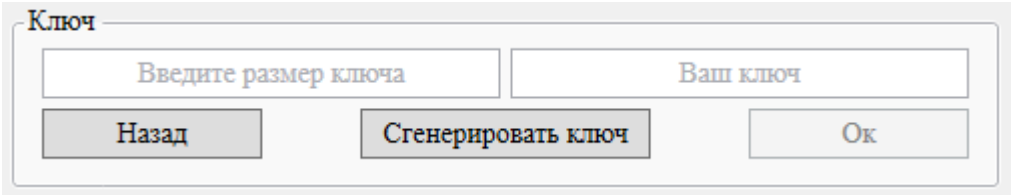

Рис. 2.53 – Раздел «Ключ» в главном окне программы после нажатия на кнопку «Сгенерировать ключ»

На этой форме генерируется случайный ключ указанного размера. Кнопка «Ок» будет заблокирована до тех пор, пока вы не сгенерируете ключ. Если вы передумали и хотите воспользоваться своим ключом, либо по другим причинам вернуться к предыдущей форме нажмите кнопку «Назад». А для того, чтобы сгенерировать ключ нужно ввести его размер в соответствующее поле. Сделайте это и нажмите на кнопку «Сгенерировать ключ» (рис. 2.54).

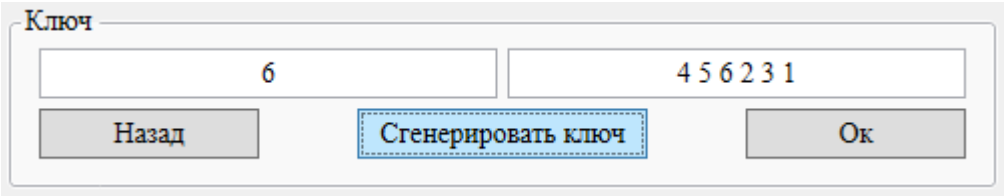

Рис. 2.54 – Раздел «Ключ» в главном окне программы после генерации ключа, способ «Транспозиция»

Вот ключ для данного способа и сгенерирован. Если вам не нравится сгенерированный ключ нужно снова нажать на кнопку «Сгенерировать ключ». Можете делать так сколько угодно пока не найдёте понравившийся ключ. Если вас устраивает сгенерированный ключ нажмите кнопку «Ок» (кстати она стала доступна) и программа предложит вам сохранить сгенерированный ключ (рис. 2.55).

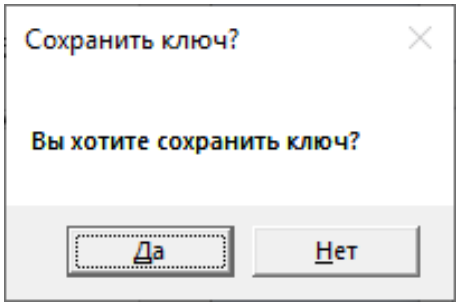

Рис. 2.55 – Предложение сохранить сгенерированный ключ

Если нажмёте нет «Нет» вернётесь на первичную форму раздела «Ключ». Если нажмёте «Да» перед вами всплывёт окно сохранения файла, где нужно также указать место сохранения файла и его название, если вас не устраивает название по умолчанию, нажмите кнопку «Сохранить» и также вернётесь на первичную форму раздела «Ключ».

Ваш сгенерированный ключ в любом случае отобразится в поле на первичной форме (рис. 2.56).

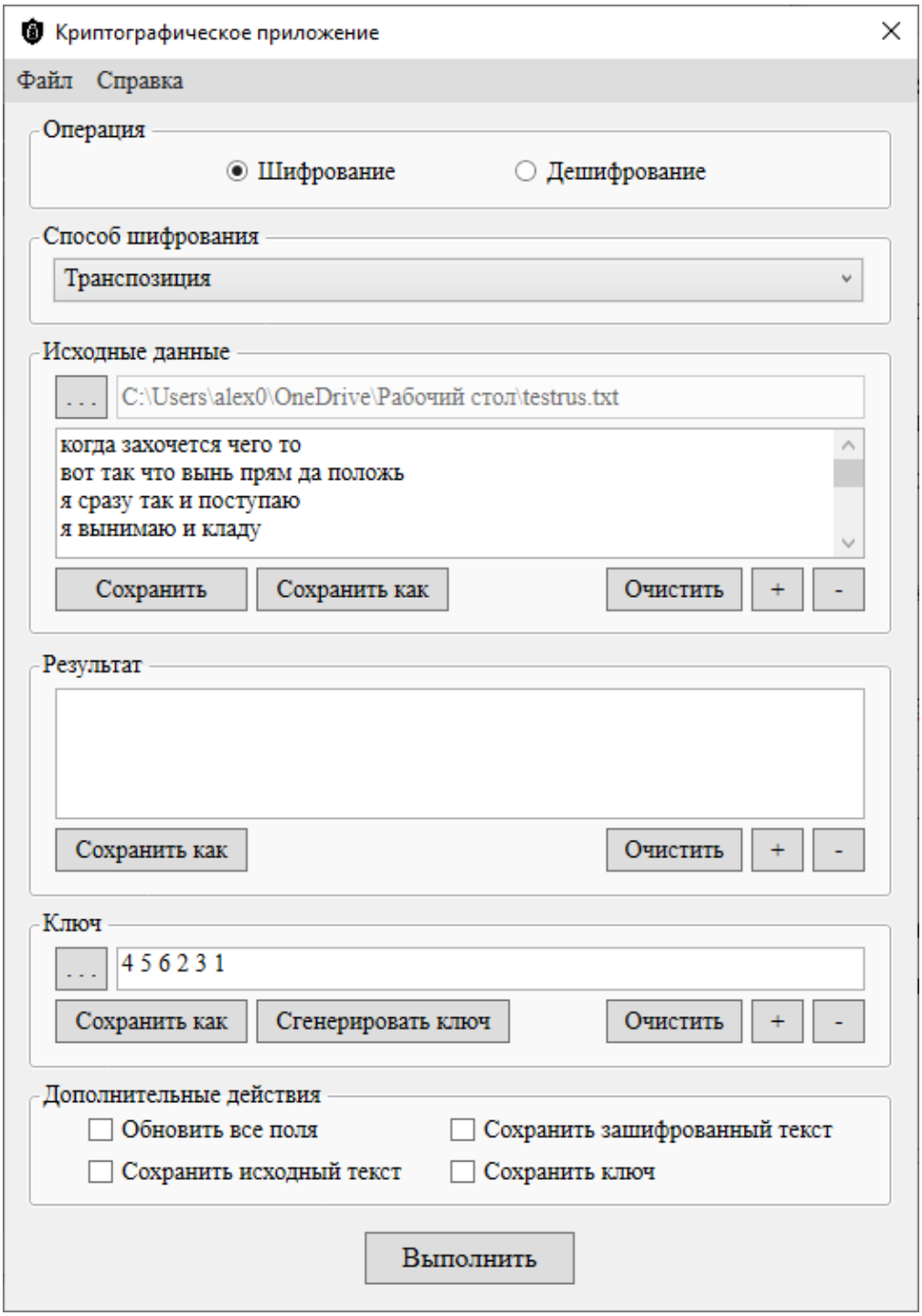

Рис. 2.56 – Изменения в главном окне программы после генерации ключа

Следующим этапом, который является не обязательным можно выбрать дополнительные действия (рис. 2.57).

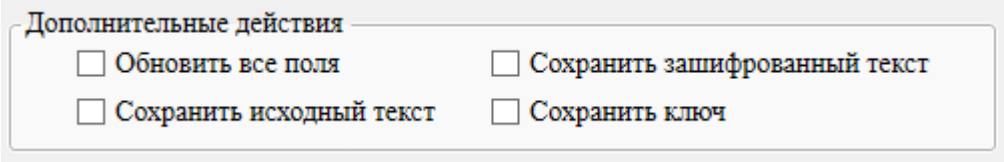

Рис. 2.57 – Раздел «Дополнительные действия»

Как видно на рис. 2.57, доступно всего четыре дополнительных действия:

- «Обновить все поля»;
- «Сохранить исходный текст»;
- «Сохранить зашифрованный текст»;
- «Сохранить ключ».

Из их названия понятно за что какое дополнительное действие отвечает. Каждая из этих действий выполняется только в случае успешного завершения операции и, если пользователь отметил их на выполнение. Для того чтобы отметить их необходимо навестись курсором на квадратик, расположенный рядом с дополнительным действием и отметить его щелчком левой кнопки мыши.

Теперь наконец зашифруем исходный текст нажав на кнопку «Выполнить» и в разделе «Результат» появится зашифрованный выбранным ключом текст (рис. 2.58).

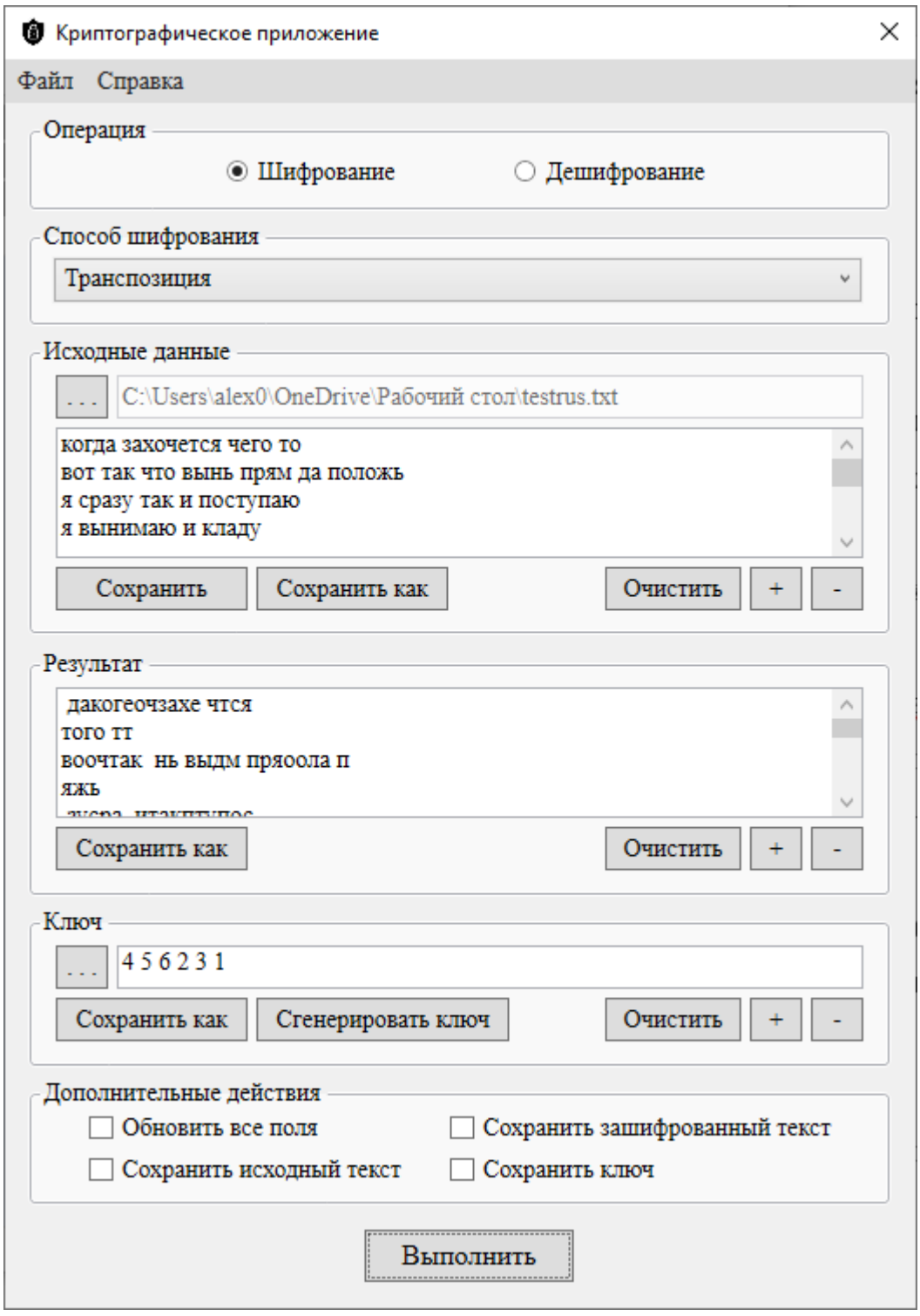

Рис. 2.58 – Главное окно программы после выполнения операции шифрования способом «Транспозиция»

Если вы выбирали какие-либо дополнительные действия перед вами будут показаны соответствующие окна.

Далее дешифруем только что зашифрованный текст. Для этого поменям операцию на «Дешифрование» в разделе «Операция» (рис. 2.59).

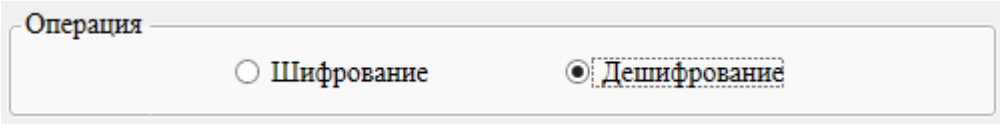

Рис. 2.59 – Смена операции

Перенеся зашифрованный текст в исходные данные и не меняя ключа нажмите кнопку «Выполнить» (рис. 2.60).

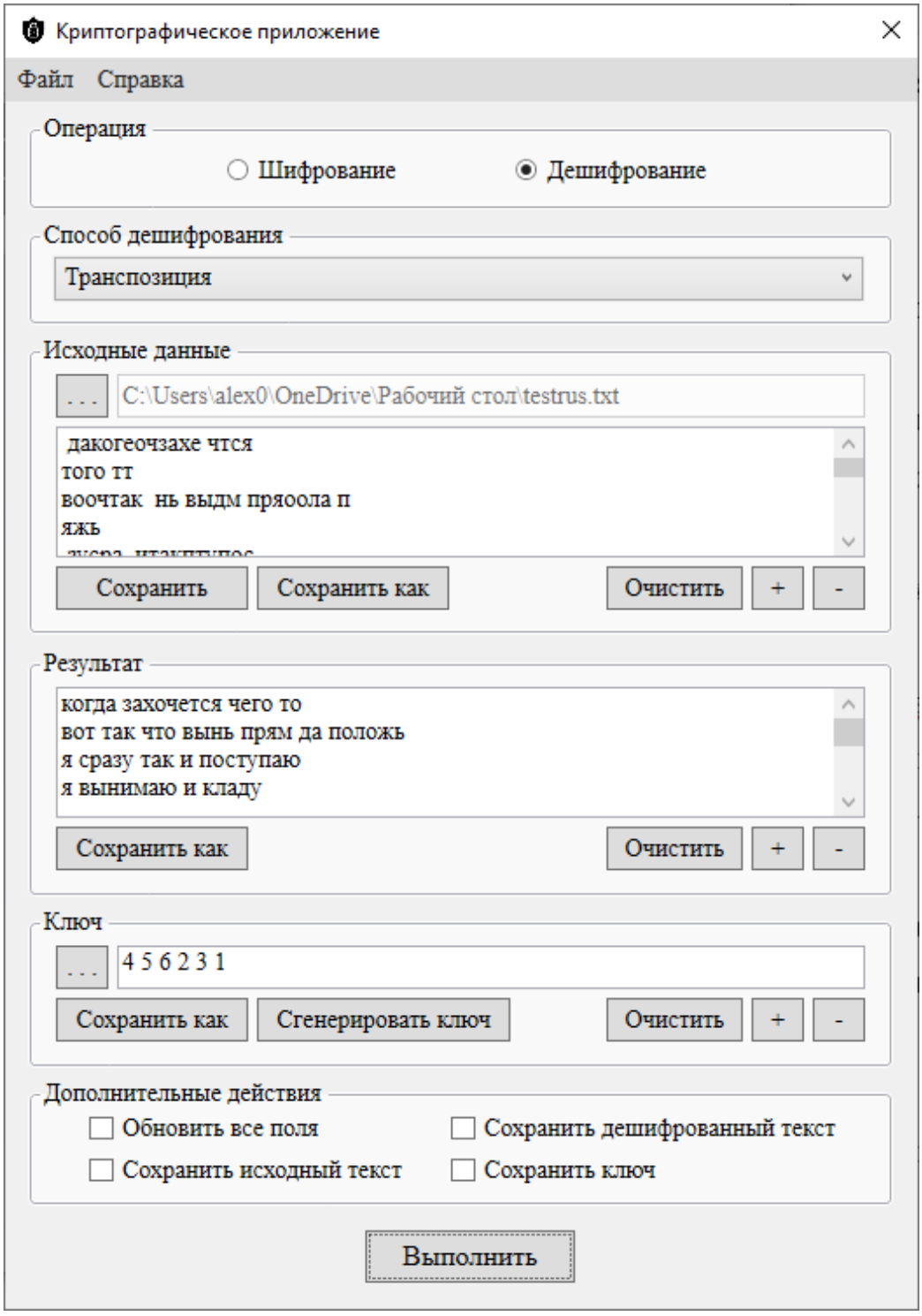

Рис. 2.60 – Главное окно программы после выполнения операции дешифрования способом «Транспозиция»

Как видно мы получили назад наш ранее зашифрованный текст.

Теперь рассмотрим способ защиты информации «Моноалфавитный шифр». Для этого выберем его в соответсвующем разделе. При демонстрации этого и последующих методов будем использовать тот же текст что и при предыдущем способе.

Сгенерируем ключ и нажмём кнопку «Выполнить», главное окно программы преобразится следующим образом (рис. 2.61).

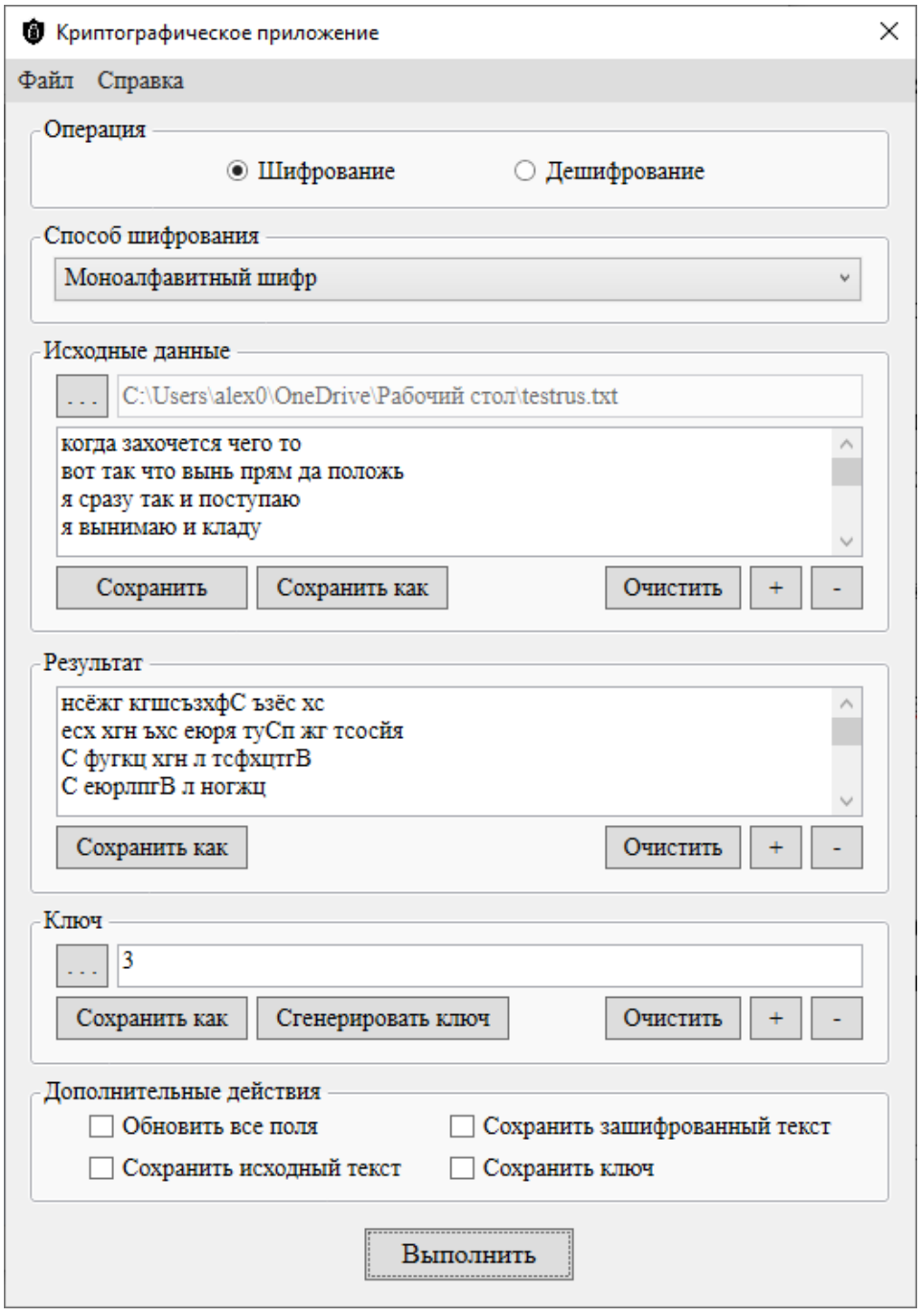

Рис. 2.61 – Главное окно программы после выполнения операции шифрования способом «Моноалфавитный шифр»

Теперь дешифруем полученный шифротекст (рис. 2.62).

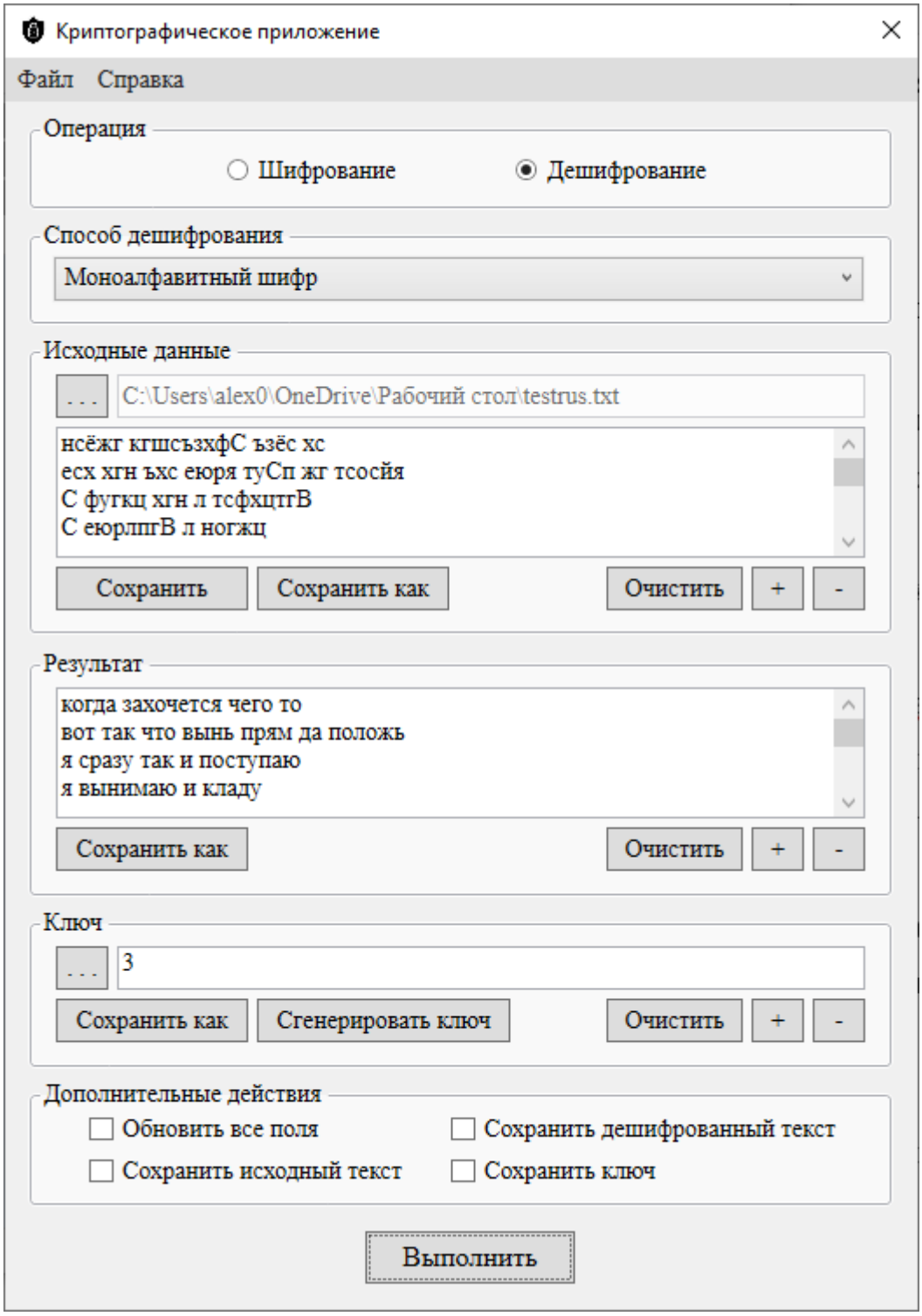

Рис. 2.62 – Главное окно программы после выполнения операции дешифрования способом «Моноалфавитный шифр»

Теперь рассмотрим способ защиты информации «Полиалфавитный шифр». Для этого выберем его в соответсвующем разделе.

Сгенерируем ключ. При генерации ключа вы попадёте на уже знакомую вам форму, но теперь ключ будет не случайной последовательностью числе, а случайным набором символов (рис. 2.63).

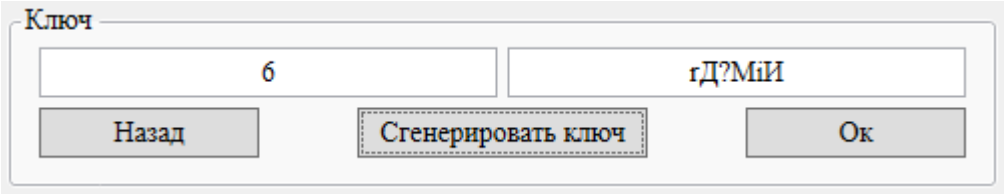

Рис. 2.63 – Раздел «Ключ» в главном окне программы после генерации ключа, способ «Полиалфавитный шифр»

Нажмём кнопку «Ок» и вернёмся к предыдущей форме.

Выполним шифрование и дешифрование этим методом (рис. 2.64).

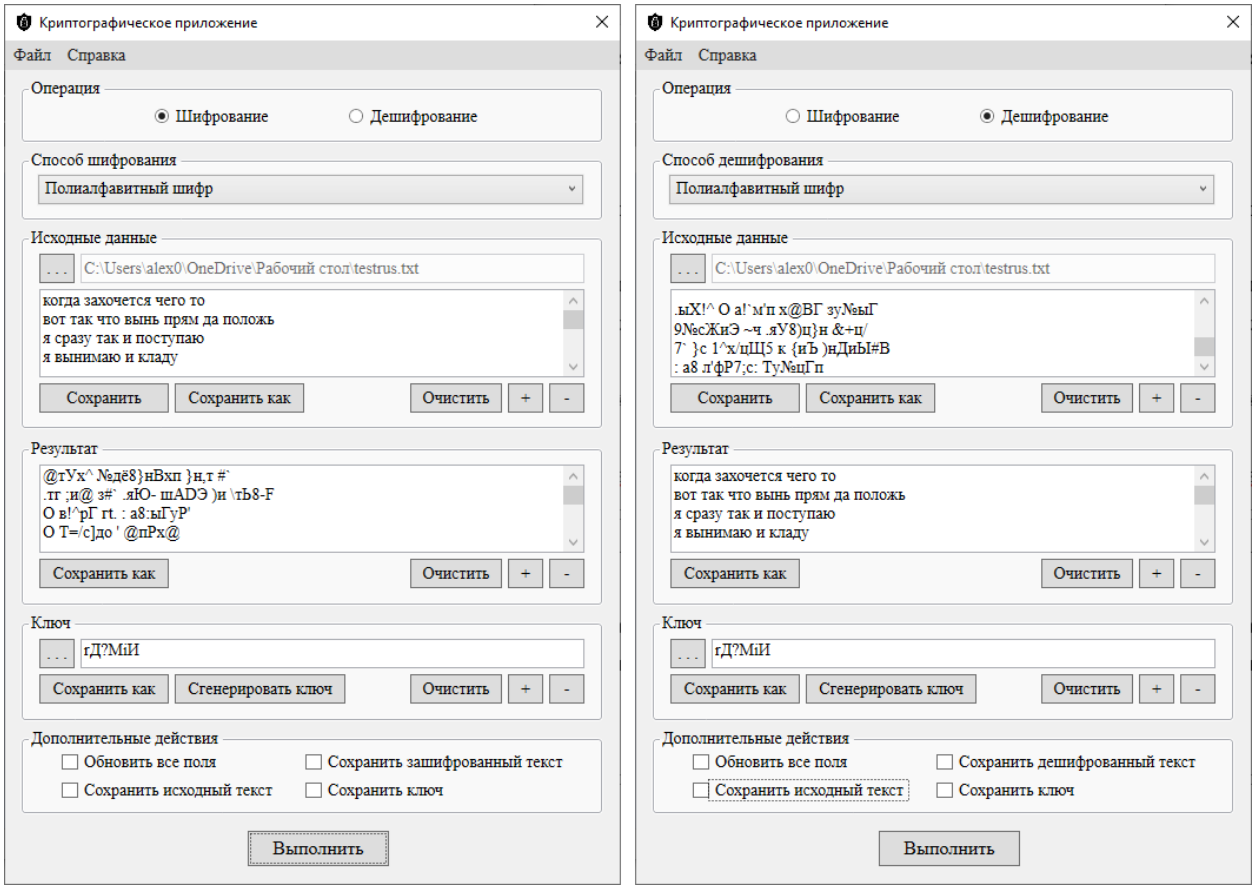

Рис. 2.64 – Главное окно программы после выполнения операций шифрования и дешифрования способом «Полиалфавитный шифр»

Теперь рассмотрим способ «Исключающее ИЛИ (XOR)». Выполним шифрование и дешифрование этим методом (рис. 2.65).

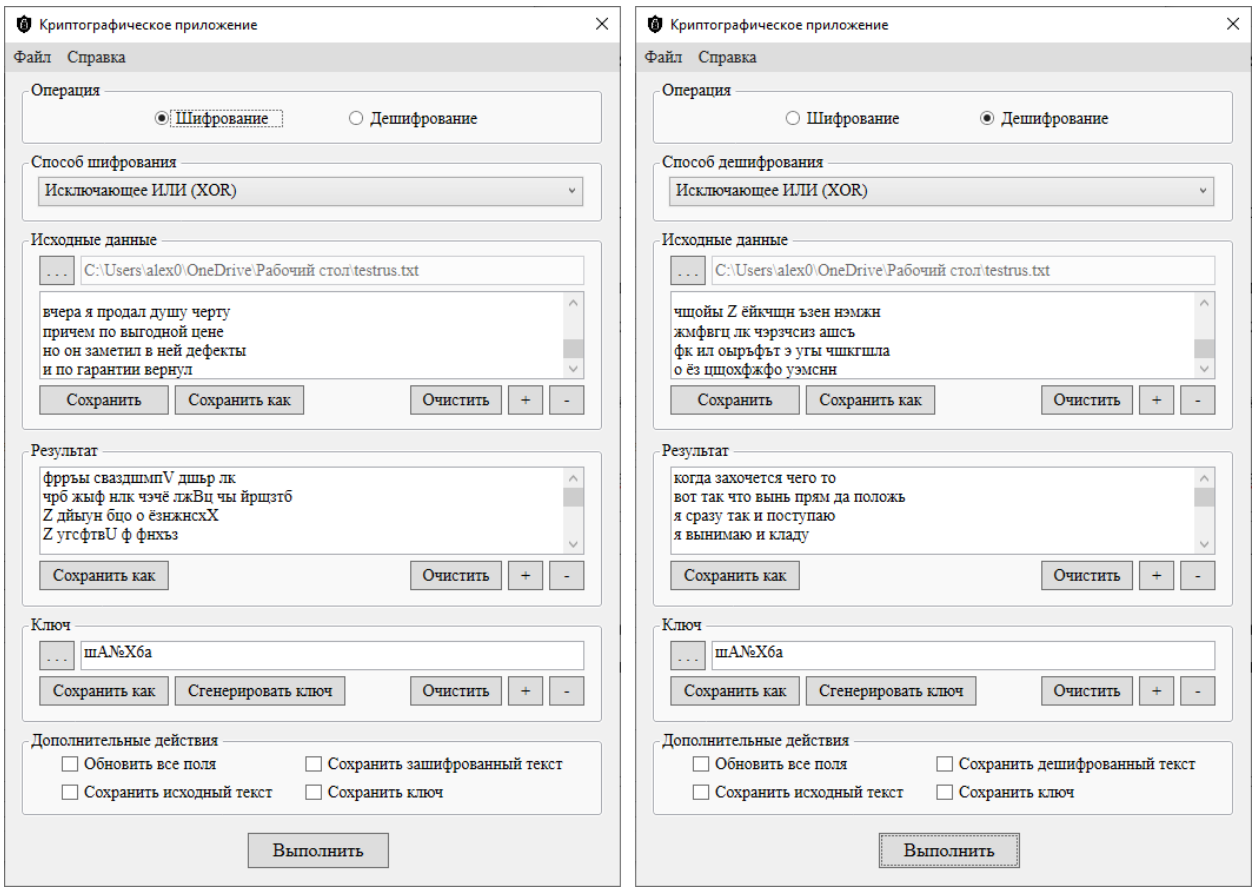

Рис. 2.65 – Главное окно программы после выполнения операций шифрования и дешифрования способом «Исключающее ИЛИ (XOR)»

Теперь рассмотрим способ «Одноразовый блокнот». Выполним шифрование и дешифрование этим методом (рис. 2.66).

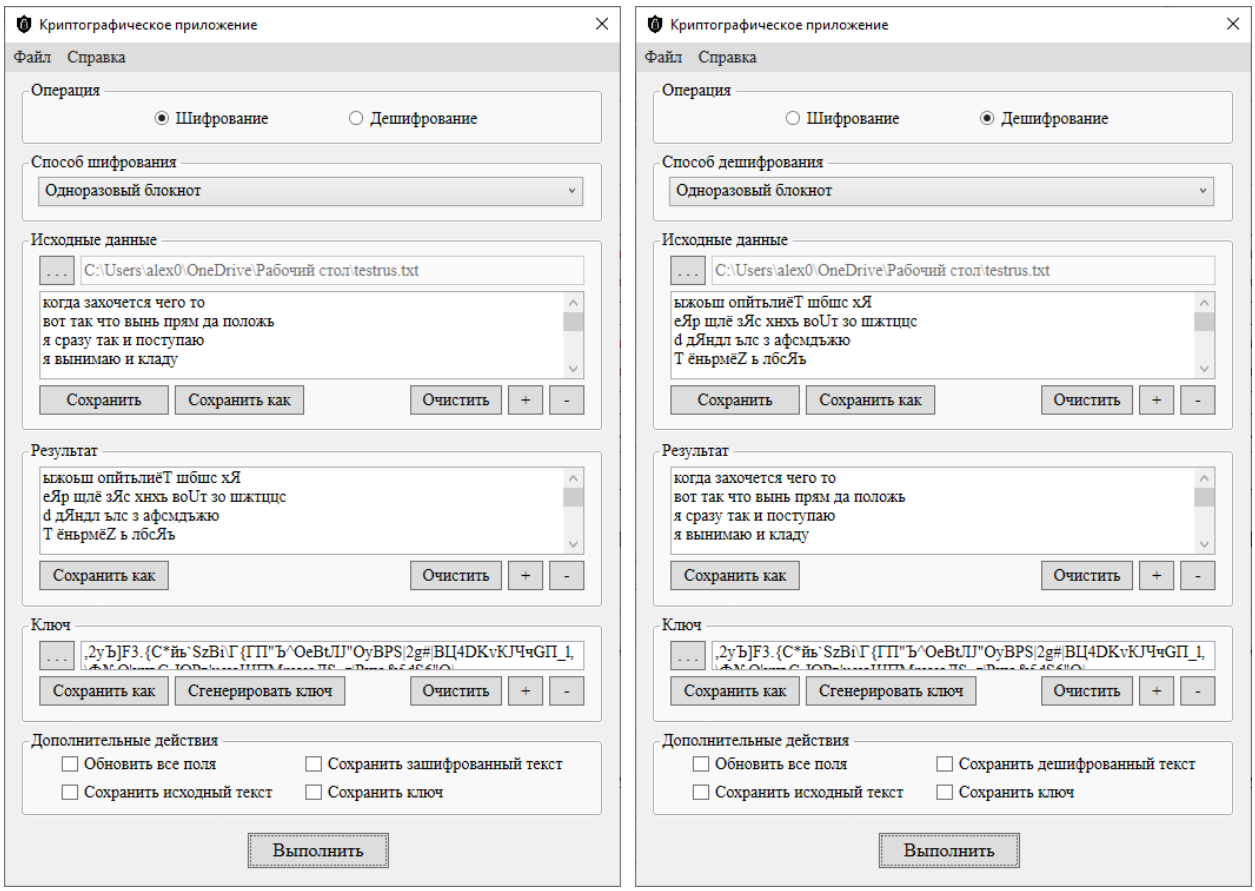

Рис. 2.66 – Главное окно программы после выполнения операций шифрования и дешифрования способом «Одноразовый блокнот»

Рассмотрим последний способ «Rivest, Shamir, Adleman (RSA)». Генерируя ключ для этого метода в разделе «Ключ» форма изменится следующим образом (рис. 2.67).

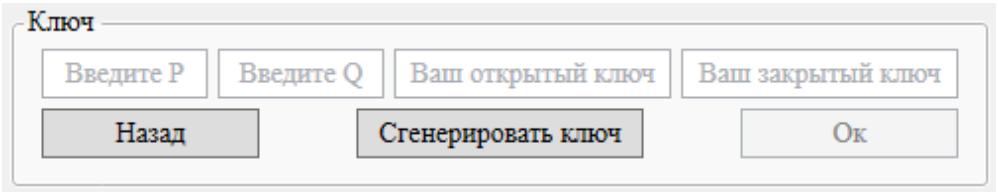

Рис. 2.67 – Раздел «Ключ» в главном окне программы после нажатия на кнопку «Сгенерировать ключ»

Введём необходимые числа и получим два ключа: открытый и закрытый (рис. 2.68).

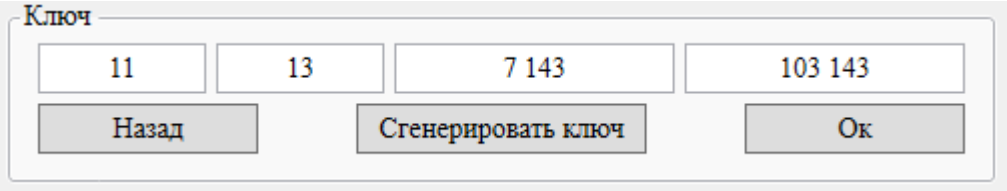

Рис. 2.68 – Раздел «Ключ» в главном окне программы после генерации ключа, способ «Rivest, Shamir, Adleman (RSA)»

Нажмём «Ок» чтобы вернуться к изначальной форме (рис. 2.69).

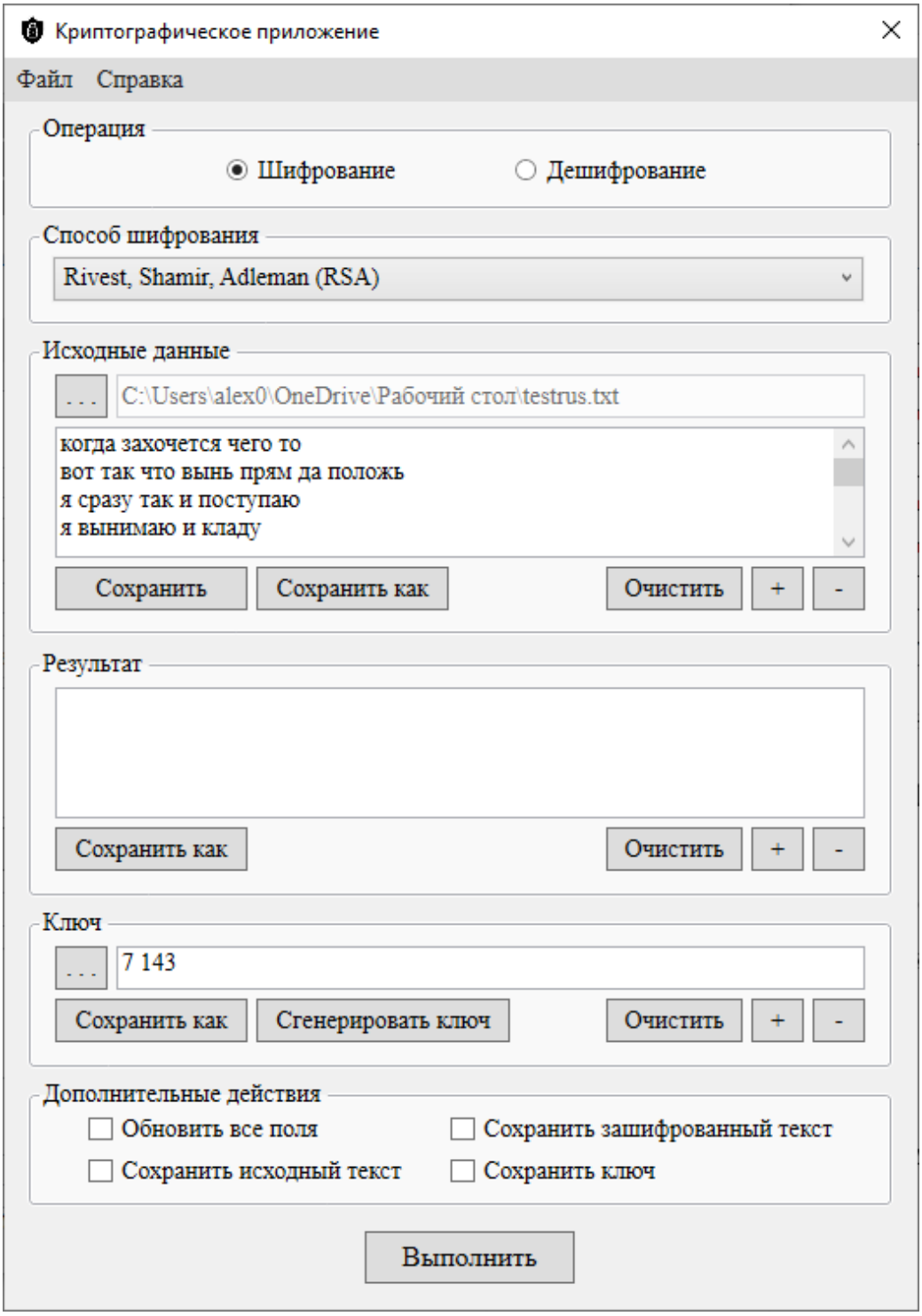

Рис. 2.69 – Изменения в главном окне программы после генерации ключа

Зашифруем текст при помощи открытого ключа (рис. 2.70).

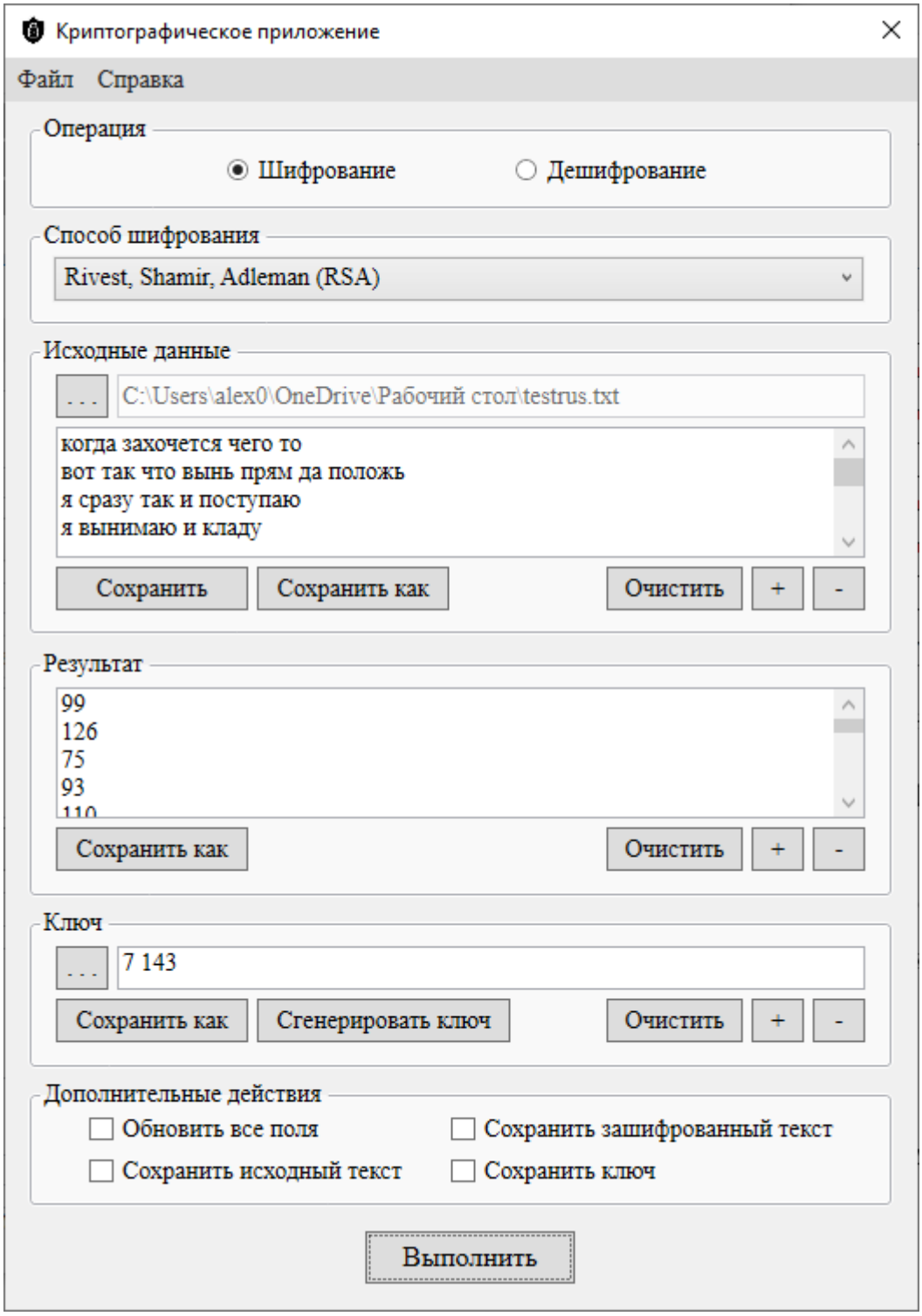

Рис. 2.70 – Главное окно программы после выполнения операции шифрования способом «Rivest, Shamir, Adleman (RSA)»

Теперь дешифруем текст при помощи закрытого ключа (рис. 2.71).

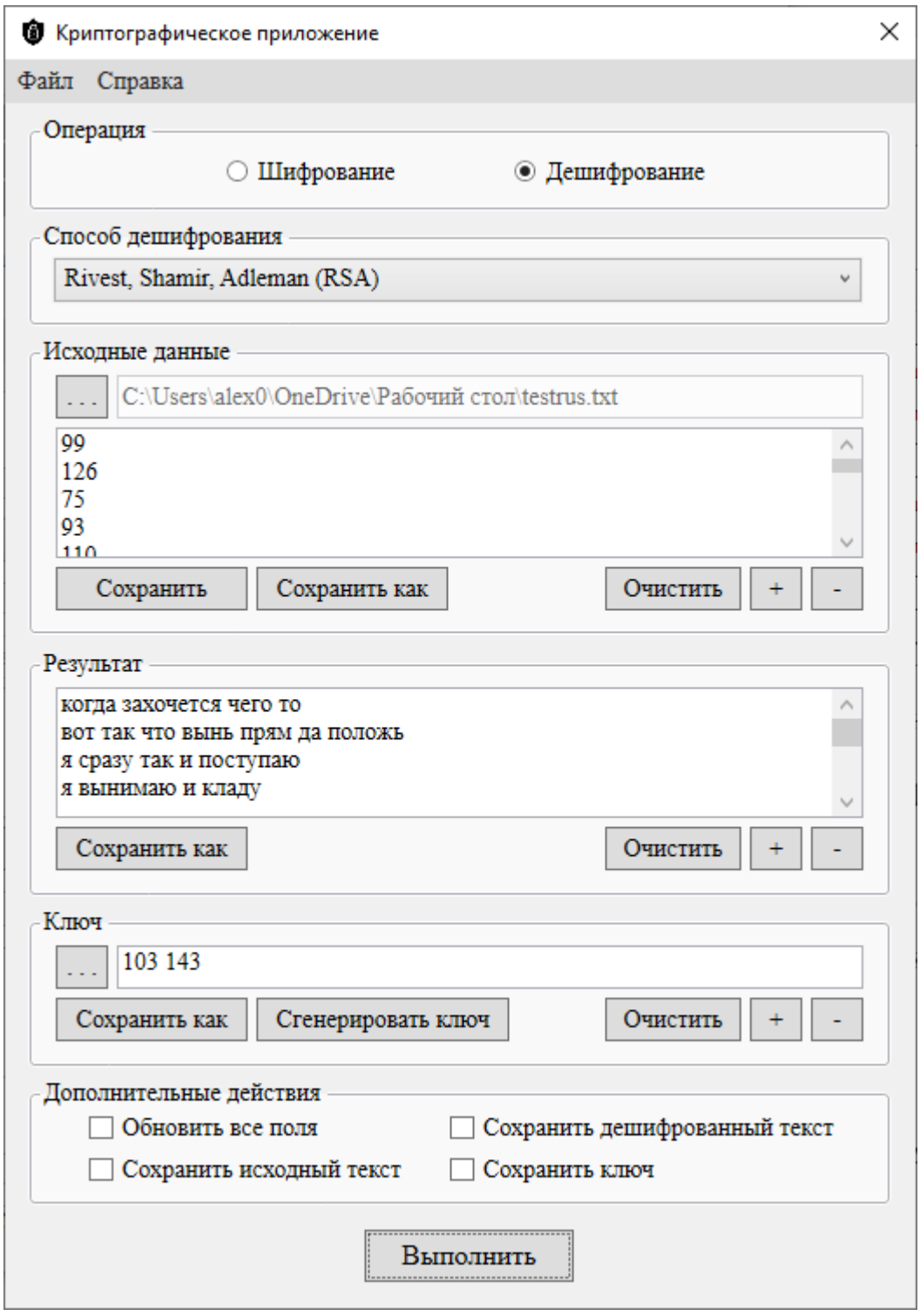

Рис. 2.71 – Главное окно программы после выполнения операции дешифрования способом «Rivest, Shamir, Adleman (RSA)»

Вот мы и рассмотрели все способы шифрования и дешифрования текста. Напоследок рассмотим второстепенные функции. Например, увеличение размера шрифта, для этого нажмём на кнопку «+» (рис. 2.72).

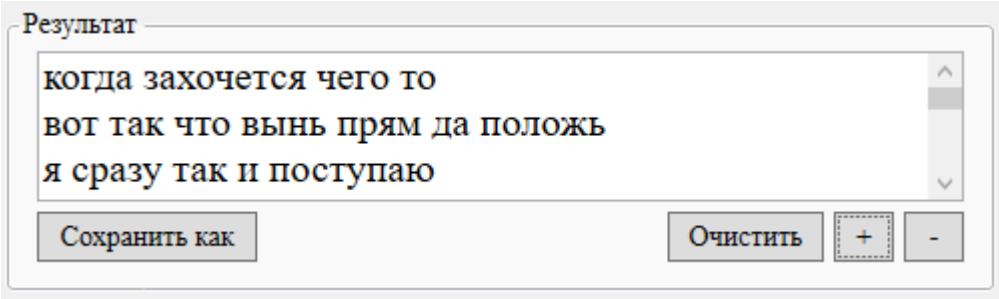

Рис. 2.72 – Демонстрация увеличения размера шрифта в разделе «Результат»

Для уменьшения размера шрифта нажмём на кнопку «-» (рис. 2.73).

Рис. 2.73 – Демонстрация уменьшения размера шрифта в разделе «Результат»

Чтобы очистить поле от текста нажмём на кнопку «Очистить» (рис. 2.74).

Рис. 2.74 – Демонстрация очищения поля от теста в разделе «Результат»

Чтобы полностью обновить все поля нажмите на клавишу клавишу «F2», либо выберите соответствующий подпункт в пункте меню «Файл». Программа после обновления всех полей представлена на рис. 2.75.

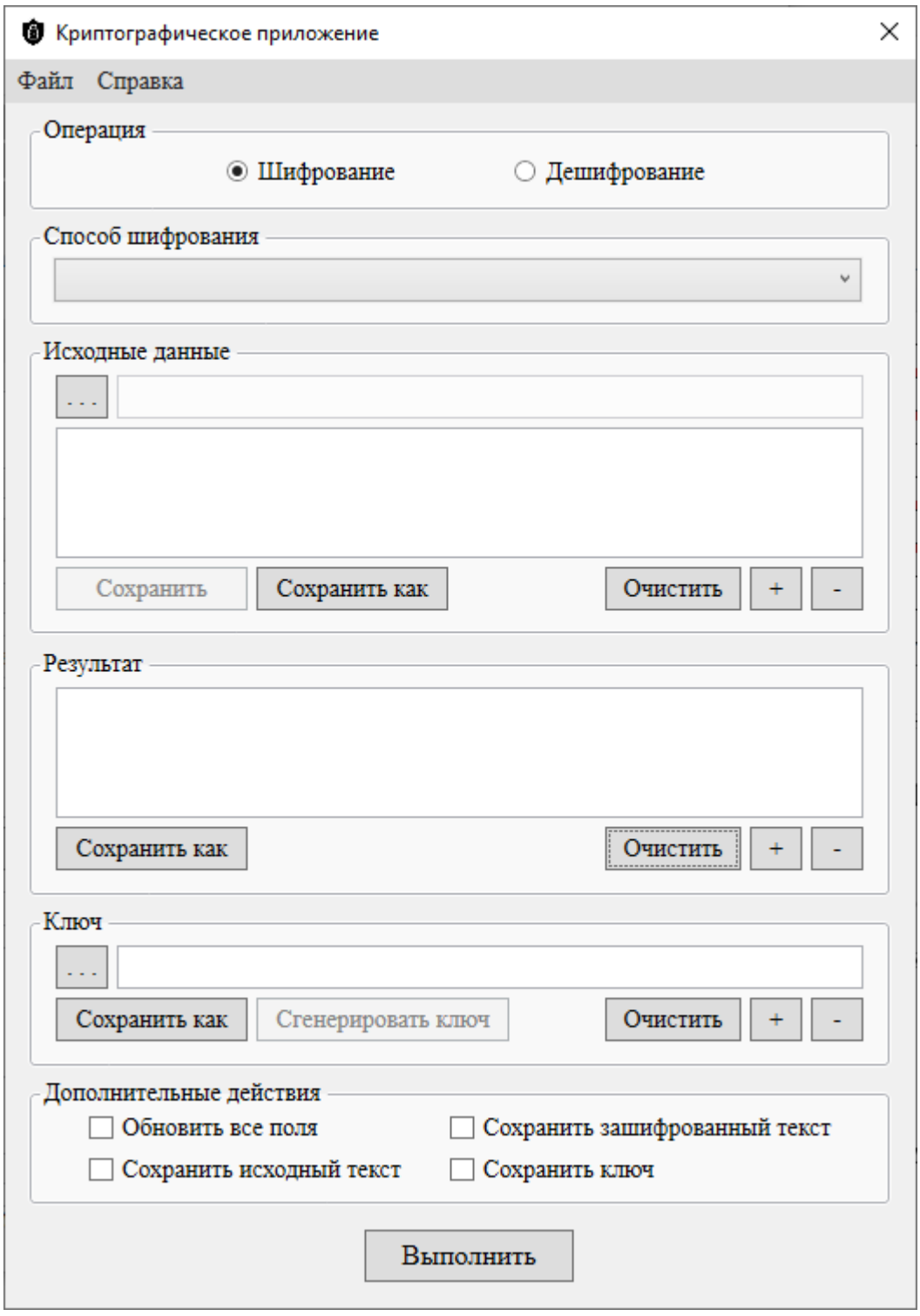

Рис. 2.75 – Главное окно программы после обновления всех полей

Последняя функция, которую хотелось бы рассмотреть это выход из программы. Помимо простого нажатия на крестик в правом верхнем углу окна для закрытия программы, можно также воспользоваться функцией, которая расположена в пункте меню «Файл», подпункте «Выход». Нажав на неё левой

кнопкой мыши или нажав клавишу «F12», программа попросит подтвердить выход из неё как на рис. 2.76.

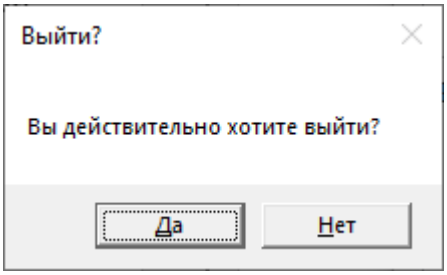

Рис. 2.76 – Подтверждение выхода из программы

После подтверждения программа завершит свою работу.

2.6.3.4 Сообщения пользователю

В программе сообщения пользователю можно разделить на три типа:

- подсказки при работе;
- подтверждение действий;
- сообщения об ошибках.

Подсказки, возникающие при работе отображены в таблице 2.6.

Таблица 2.6 – Подсказки при работе

Действие для появления	Текст сообщения	Демонстрация в программе
Наведение курсора на кнопку «…»	Выбрать файл	Рис. 2.77
Наведение курсора на кнопку «Сохранить»	Перезаписать текст в выбранный файл	Рис. 2.77
Наведение курсора на кнопку «Сохранить как» в разделе «Исходные данные»	Сохранить текст в новый файл	Рис. 2.77
Наведение курсора на кнопку «Сохранить как» в разделе «Результат»	Сохранить результат в новый файл	Рис. 2.77

Продолжение таблицы 2.6

Действие для появления	Текст сообщения	Демонстрация в программе
Наведение курсора на кнопку «Сохранить как» в разделе «Ключ»	Сохранить ключ в новый файл	Рис. 2.77
Наведение курсора на кнопку «Сгенерировать ключ»	Сгенерировать случайный ключ	Рис. 2.77
Наведение курсора на кнопку «Назад»	Вернуться назад	Рис. 2.77
Наведение курсора на кнопку «Ок»	Воспользоваться сгенерированным ключом	Рис. 2.77
Наведение курсора на поле «Введите P»	P должно: - быть простым числом; - быть больше 10; - быть отличным от Q.	Рис. 2.77
Наведение курсора на поле «Введите Q»	Q должно: - быть простым числом; - быть больше 10; быть отличным от P.	Рис. 2.77
Наведение курсора на кнопку «Очистить»	Очистить поле с исходным текстом	Рис. 2.77
Наведение курсора на кнопку «+»	Увеличить размер шрифта	Рис. 2.77
Наведение курсора на кнопку «-»	Уменьшить размер шрифта	Рис. 2.77

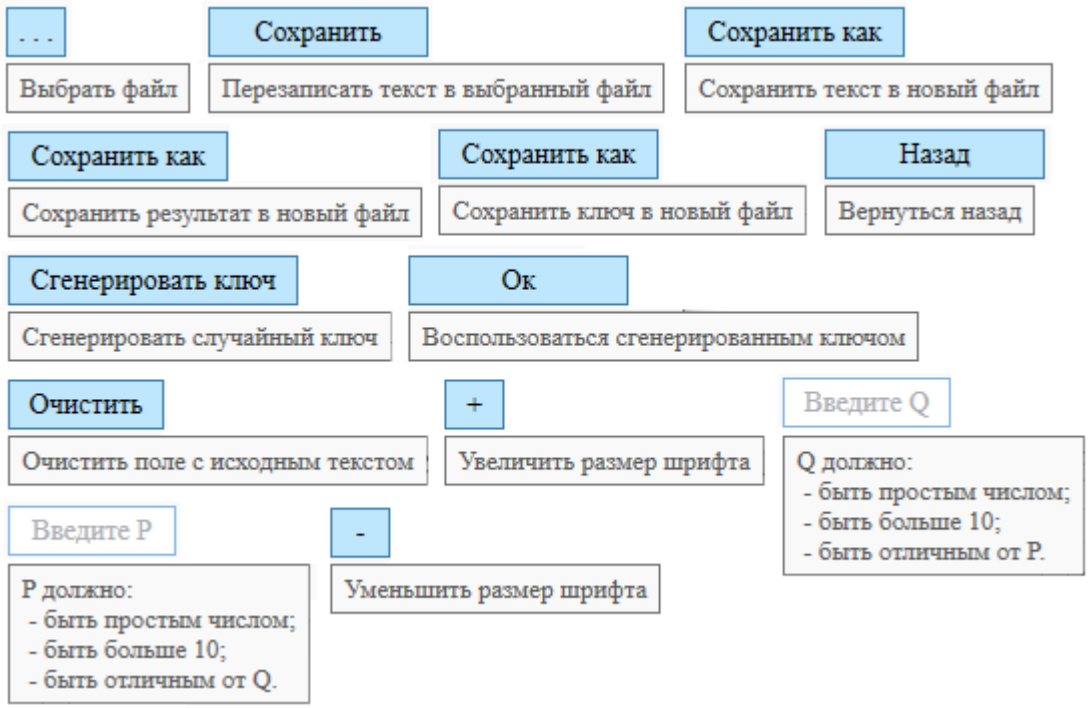

Рис. 2.77 – Подсказка возникающие при наведении на элементы

Подсказки подтверждения действий приведены на рис. 2.78 и рис. 2.79.

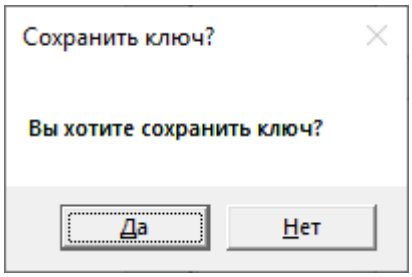

Рис. 2.78 – Предложение сохранить ключ

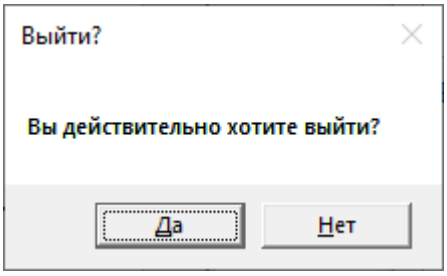

Рис. 2.79 – Подтверждение выхода из программы

Сообщение, возникающее при ошибке, связанной с повреждением или отсутствием файла справки (рис. 2.80).

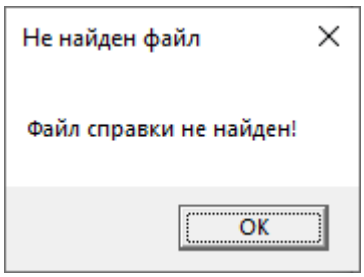

Рис. 2.80 – Реакция программы на возникновение ошибки, связанной с отсутствием или повреждением файла справки

Заключение

В этой выпускной квалификационной работе была разработана программа шифрования файлов. При её написании было изучено множество аспектов: специальная литература, алгоритмы и принципы работы для шифрования и дешифрования текста разными методами, новые библиотеки, программы и способы написания кода на языке C#.

Для достижения поставленной цели были решены следующие задачи:

- детальное изучение предметной области;
- проведение сравнительного анализ и на его основе выбор наиболее эффективных алгоритмов шифрования и дешифрования;
- реализация функциональных и нефункциональных требований к программе;
- составление необходимых диаграмм и схем;
- выбор подходящих для разработки инструментальных средств;
- тестирование и отладка правильности работы программы.

Самым сложным и одновременно интересным при реализации программы было написание алгоритмов шифрования и дешифрования текста. Для этого пришлось вспоминать булеву алгебру, логику и другие предметы, на что потребовалось достаточно большое количество времени. Это было необходимо так как при написании кода программы небольшие ошибки в нём могли привести к полной неправильности выполнения операций.

Отдельно хочу отметить момент изучения предметной области. Никогда бы не подумал, что криптология может быть такой увлекательной наукой, котораю я ещё точно буду изучать в будущем, уже не в рамках проекта.

Изложенное мною ранее подразумевает, что все цели проекта были выполнены, функции реализованы и протестированы.

Достоинствами этого проекта можно считать:

- простой, понятный, удобный и приятный взгляду интерфейс с множеством подсказок;

- наличие нескольких методов для защиты информации;
- возможность генерации ключей для каждого метода;
- открытие и сохранение файлов;
- наличие «горячих» клавиш для основных функций программы;
- наличие справки с инструкцией, которую можно вызвать прямо из программы.

К сожалению, в проект не была добавлена возможность работы с файла формата отличного от txt.

Проведенные испытания показали корректность работы программы. Следовательно, все вышеизложенное позволяет рекомендовать её для защиты конфеденциальной или личной информации от посторонних лиц.

Подводя итоги всему написанному ранее, можно сказать что работа над данным проектом многому меня научила, показала мне, что перед написанием программы нужно продумать каждую деталь до мелочей, ведь если придумывать всё на ходу, придётся менять огромное количество кода, что по времязатратам может выйти даже больше, чем если бы всё было продумано с самого начала. К тому же, если учесть, что программирование это прежде всего командная деятельность огромную роль играет умение писать «чистый» код. Я также понял то, что для написания подобных проектов умение только программировать даёт очень немного, помимо этого необходимо разбираться во множестве областей и не только технических.

Считаю выпускную квалификационную работу успешно завершённой.

Список литературы

Законодательные и нормативные акты

1. ГОСТ 2.316-2008 Правила нанесения надписей, технических требований и таблиц на графических документах
2. ГОСТ 7.1. – 2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
3. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание.
4. ГОСТ 7.32 – 2001. Система стандартов по информацию, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21 с.
5. ГОСТ 8 417 2002 Единицы величин
6. ГОСТ Р 21.1101-2013 Основные требования к проектной и рабочей документации
7. ГОСТ СН 528-80 Перечень единиц физических величин
8. ГОСТ_2.105-95 Общие требования к текстовым документам
9. Единая система программной документации. – М.: Стандартинформ, 2005. – 128 с.

Учебная и научная литература

10. Гуриков С.Р. Введение в программирование на языке Visual C#: учебное пособие. – М.: ФОРУМ: ИНФРА-М, 2013.
11. Душкин Р.В. Математика и криптография. Тайны шифров и логическое мышление. – М.: АСТ, 2017.
12. Саймон Сингх. Книга шифров. – М.: Астрель, 2007.
13. Рудаков А.В. Технология разработки программных продуктов: учебник для студ. учреждений сред. проф. образования. – М.: Академия, 2013.

14. Рудаков А.В., Федорова Г.Н. Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования. – М.: Академия, 2013.

Интернет-документы

15. C Sharp – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/C_Sharp
16. Draw.io – [Электронный ресурс]. – https://startpack.ru/application/draw-io
17. Folder Lock – приложение для защиты данных – [Электронный ресурс]. – http://bezwindowsa.ru/programmy/folder-lock-prilozhenie-dlya-zashhityi-dannyih.html
18. HTMLHelp – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/HTMLHelp
19. Microsoft Visual Studio – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio
20. PGP – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/PGP
21. RSA – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/RSA#%D0%9A%D0%BE%D1%80%D1%80%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D1%81%D1%85%D0%B5%D0%BC%D1%8B_RSA
22. Xaml – Как динамически изменять содержимое окна (пример: главное окно -> настройки)? – [Электронный ресурс]. – https://toster.ru/q/280542
23. Булева алгебра – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0
24. Булева логика – [Электронный ресурс]. – http://wikiredia.ru/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0
25. Виды алгоритмов симметричного шифрования – [Электронный ресурс]. – https://lektsii.org/15-5320.html

26. Гаммирование – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5
27. Иллюстрация работы RSA на примере – [Электронный ресурс]. – http://www.michurin.net/computer-science/rsa.html
28. Как написать введение дипломной работы? – [Электронный ресурс]. – https://studlance.ru/blog/kak-napisat-vvedenie-diplomnoj-raboty
29. Как написать заключение к дипломной работе? – [Электронный ресурс]. – https://studlance.ru/blog/kak-napisat-zaklyuchenie-k-diplomnoj-rabote
30. Как написать заключение к дипломной работе? – [Электронный ресурс]. – https://studlance.ru/blog/kak-napisat-zaklyuchenie-k-diplomnoj-rabote
31. Краткий обзор языка C# – [Электронный ресурс]. – https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/
32. Криптографическая защита информации – [Электронный ресурс]. – https://studfiles.net/preview/6210714/
33. Криптосервисы .NET Framework – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81%D1%8B_.NET_Framework
34. Объект и предмет исследования – в чём разница? – [Электронный ресурс]. – https://nauchniestati.ru/blog/obekt-i-predmet-issledovaniya/
35. Особенности платформы WPF – [Электронный ресурс]. – https://metanit.com/sharp/wpf/1.php
36. Перестановочный шифр – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80
37. Полиалфавитный шифр – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%B0%

D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80

38. Пользовательский интерфейс – [Электронный ресурс]. – https://studwood.ru/1615608/informatika/polzovatelskiy_interfeys

39. Поточные шифры – [Электронный ресурс]. – https://studfiles.net/preview/4574848/page:2/

40. Предмет криптографии – [Электронный ресурс]. – https://studfiles.net/preview/4483753/page:2/

41. Преимущества и недостатки алгоритма шифрования RSA – [Электронный ресурс]. – https://studwood.ru/1685074/informatika/preimuschestva_nedostatki_algoritma_shifrovaniya

42. Пример простейшей программы для шифрования текста – [Электронный ресурс]. – https://habr.com/ru/post/140373/

43. Простое XOR-шифрование. Алгоритм. – [Электронный ресурс]. – http://words-are-dust.blogspot.com/2012/01/xor.html

44. Руководство по программированию на C# – [Электронный ресурс]. – https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/

45. Список простых чисел – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB

46. Сравнение настольных программ для шифрования – [Электронный ресурс]. – https://habr.com/ru/company/cybersafe/blog/252561/

47. Шифр Вернама – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B5%D1%80%D0%BD%D0%B0%D0%BC%D0%B0

48. Шифр Виженера на C# – [Электронный ресурс]. – http://code-enjoy.ru/shifr_vizhnera_na_c_sharp/

49. Шифр перестановки – [Электронный ресурс]. – https://vscode.ru/prog-lessons/shifr-perestanovki.html
50. Шифр подстановки – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%BF%D0%BE%D0%B4%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B8
51. Шифр Цезаря – [Электронный ресурс]. – https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F

Приложение 1

«Горячие» клавиши программы

Таблица 1.1 – «Горячие» клавиши программы

Клавиша	Функция
F1	Выполнить
F2	Обновить
F3	Сгенерировать ключ
F4	Открыть файл с исходным текстом
F5	Открыть файл с ключом
F6	Сохранить (перезаписать файл)
F7	Сохранить исходный текст
F8	Сохранить зашифрованный/дешифрованный текст
F9	Сохранить ключ
F11	Открыть справку «О программе»
F12	Выход из программы

Исходный код программы

MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Diagnostics;

namespace CryptographicApplication
{
   /// <summary>
   /// Логика взаимодействия для MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window
   {

      #region Переменные

      Functional func_obj;
      RandomKeyGeneration rndkey_obj;
      Transposition trans_obj;
      Monoalphabetic mono_obj;
      Polyalphabetic poly_obj;
      XOR xor_obj;
      Vernam vernam_obj;
      RSA rsa_obj;

      public bool transition = false;

      #endregion

      public MainWindow()
      {
         InitializeComponent();

         rb_Encryption.IsChecked = true;
         func_obj = new Functional();
         trans_obj = new Transposition();
         mono_obj = new Monoalphabetic();
         poly_obj = new Polyalphabetic();
         xor_obj = new XOR();
         vernam_obj = new Vernam();
         rsa_obj = new RSA();
         rndkey_obj = new RandomKeyGeneration();
      }
```

```
        #region Алгоритмы

        public void Transposition_Cipher()
        {
            trans_obj.SetKey(tb_Key.Text);

            if (rb_Encryption.IsChecked == true)
            {
                tb_EncryptedData.Text = trans_obj.Encrypt(tb_SourceData.Text);
            }
            else
            {
                tb_EncryptedData.Text = trans_obj.Decrypt(tb_SourceData.Text);
            }
        }

        public void Monoalphabetic_Cipher()
        {
            if (rb_Encryption.IsChecked == true)
            {
                tb_EncryptedData.Text = mono_obj.Encrypt(tb_SourceData.Text,
Convert.ToInt32(tb_Key.Text));
            }
            else
            {
                tb_EncryptedData.Text = mono_obj.Decrypt(tb_SourceData.Text,
Convert.ToInt32(tb_Key.Text));
            }
        }

        public void Polyalphabetic_Cipher()
        {
            if (rb_Encryption.IsChecked == true)
            {
                tb_EncryptedData.Text = poly_obj.Encrypt(tb_SourceData.Text, tb_Key.Text);
            }
            else
            {
                tb_EncryptedData.Text = poly_obj.Decrypt(tb_SourceData.Text, tb_Key.Text);
            }
        }

        public void Vernam_Cipher()
        {
            tb_EncryptedData.Text = vernam_obj.Encrypt_and_Decrypt(tb_SourceData.Text,
tb_Key.Text);
        }

        public void XOR_Cipher()
        {
```

```
            tb_EncryptedData.Text = xor_obj.Encrypt_and_Decrypt(tb_SourceData.Text,
tb_Key.Text);
        }

        public void RSA_Cipher()
        {
            rsa_obj.SetKey(tb_Key.Text);

            if (rb_Encryption.IsChecked == true)
            {
                tb_EncryptedData.Text = rsa_obj.Encrypt(tb_SourceData.Text);
            }
            else
            {
                String[] s = tb_SourceData.Text.Split(new String[] { "\n" },
StringSplitOptions.RemoveEmptyEntries);
                List<string> sourcetext = new List<string>(s);

                tb_EncryptedData.Text = rsa_obj.Decrypt(sourcetext);
            }
        }

        #endregion

        #region Элементы формы

        #region Меню

        private void Menu_btn_ExecuteOperation_Click(object sender, RoutedEventArgs e)
        {
            List_of_Operations_to_Perform();
        }

        private void Menu_btn_Refresh_Click(object sender, RoutedEventArgs e)
        {
            Refresh();
        }

        private void Menu_btn_Key_Generation_Click(object sender, RoutedEventArgs e)
        {
            List_of_Key_Generation_Methods();
        }

        private void Menu_btn_FileSelection_Source_Click(object sender, RoutedEventArgs
e)
        {
            func_obj.File_Selection(tb_FileName_Source, tb_SourceData,
tb_EncryptedData);
        }

        private void Menu_btn_FileSelection_Key_Click(object sender, RoutedEventArgs e)
        {
```

```
            func_obj.File_Selection(tb_Key);
        }

        private void Menu_btn_SaveFile_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Save_File(tb_FileName_Source, tb_SourceData);
        }

        private void Menu_btn_SaveFileAs_Source_Click(object sender, RoutedEventArgs
e)
        {
            func_obj.Save_File_As(tb_FileName_Source, tb_SourceData);
        }

        private void Menu_btn_SaveFileAs_Encrypted_Click(object sender,
RoutedEventArgs e)
        {
            func_obj.Save_File_As(cb_Algorithms, tb_EncryptedData, rb_Encryption, true);
        }

        private void Menu_btn_SaveFileAs_Key_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Save_File_As(cb_Algorithms, tb_Key, rb_Encryption, false);
        }

        private void Menu_btn_Exit_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private void Menu_btn_About_the_Program_Click(object sender, RoutedEventArgs
e)
        {
            Calling_Help();
        }

        #endregion

        #region Исходные данные

        private void Btn_FileSelection_Source_Click(object sender, RoutedEventArgs e)
        {
            func_obj.File_Selection(tb_FileName_Source,                tb_SourceData,
tb_EncryptedData);
        }

        private void Btn_SaveFile_Source_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Save_File(tb_FileName_Source, tb_SourceData);
        }

        private void Btn_SaveFileAs_Source_Click(object sender, RoutedEventArgs e)
```

```
        {
            func_obj.Save_File_As(tb_FileName_Source, tb_SourceData);
        }

        private void Tb_FileName_Source_TextChanged(object sender,
TextChangedEventArgs e)
        {
            func_obj.Changed_File_Name_TB(tb_FileName_Source, btn_SaveFile_Source,
menu_btn_SaveFile);
        }

        private void Btn_Clear_Source_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Clear_tb(tb_SourceData);
        }

        private void Btn_Increase_Source_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Font_Size(tb_SourceData, true);
        }

        private void Btn_Reduce_Source_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Font_Size(tb_SourceData, false);
        }

        private void Tb_SourceData_TextChanged(object sender, TextChangedEventArgs e)
        {
            tb_EncryptedData.Text = "";
        }

        #endregion

        #region Результат

        private void Btn_SaveFileAs_Encrypted_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Save_File_As(cb_Algorithms, tb_EncryptedData, rb_Encryption, true);
        }

        private void Btn_Clear_Encrypted_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Clear_tb(tb_EncryptedData);
        }

        private void Btn_Increase_Encrypted_Click(object sender, RoutedEventArgs e)
        {
            func_obj.Font_Size(tb_EncryptedData, true);
        }

        private void Btn_Reduce_Encrypted_Click(object sender, RoutedEventArgs e)
        {
```

```
    func_obj.Font_Size(tb_EncryptedData, false);
}

#endregion

#region Ключ

#region Основное меню

private void Btn_FileSelection_Key_Click(object sender, RoutedEventArgs e)
{
    func_obj.File_Selection(tb_Key);
}

private void Btn_SaveFileAs_Key_Click(object sender, RoutedEventArgs e)
{
    func_obj.Save_File_As(cb_Algorithms, tb_Key, rb_Encryption, false);
}

private void Btn_Clear_Key_Click(object sender, RoutedEventArgs e)
{
    func_obj.Clear_tb(tb_Key);
}

private void Btn_Increase_Key_Click(object sender, RoutedEventArgs e)
{
    func_obj.Font_Size(tb_Key, true);
}

private void Btn_Reduce_Key_Click(object sender, RoutedEventArgs e)
{
    func_obj.Font_Size(tb_Key, false);
}

private void Tb_Key_TextChanged(object sender, TextChangedEventArgs e)
{
    tb_Key.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));
}

#endregion

#region Меню для генерации ключей с выбором размера

private void Btn_Key_Generation_1_Click(object sender, RoutedEventArgs e)
{
    Key_Generation_Selection();
}

private void Btn_Backward_Click(object sender, RoutedEventArgs e) //назад
{
    transition = false;
```

```
            Grid_Main_Key_Menu.Visibility = Visibility.Visible;
            menu_btn_SaveFileAs_Key.IsEnabled = true;

            tb_Key_Size.Text = "";
            tb_Key_1.Text = "";
            func_obj.Check_the_Cursor(tb_Key_Size, 0);
            func_obj.Check_the_Cursor(tb_Key_1, 1);

            Grid_Generation_Key_Menu_1.Visibility = Visibility.Collapsed;
        }

        private void Btn_OK_Click(object sender, RoutedEventArgs e) //ок
        {

            if (func_obj.Confirm_Action(0))
            {
                func_obj.Save_File_As(cb_Algorithms, tb_Key_1, rb_Encryption, false);
            }

            transition = false;

            Grid_Main_Key_Menu.Visibility = Visibility.Visible;
            menu_btn_SaveFileAs_Key.IsEnabled = true;

            tb_Key.Text = tb_Key_1.Text;
            tb_Key_Size.Text = "";
            tb_Key_1.Text = "";
            func_obj.Check_the_Cursor(tb_Key_Size, 0);
            func_obj.Check_the_Cursor(tb_Key_1, 1);

            Grid_Generation_Key_Menu_1.Visibility = Visibility.Collapsed;
        }

        private void Tb_Key_Size_PreviewMouseDown(object sender,
MouseButtonEventArgs e)
        {
            func_obj.Check_for_Text(tb_Key_Size);
        }

        private void Tb_Key_Size_LostFocus(object sender, RoutedEventArgs e)
        {
            func_obj.Check_the_Cursor(tb_Key_Size, 0);
        }

        private void Tb_Key_Size_PreviewKeyDown(object sender, KeyEventArgs e)
        {
            func_obj.Without_a_Space(tb_Key_Size, e);
        }

        private void Tb_Key_Size_PreviewTextInput(object sender,
TextCompositionEventArgs e)
        {
```

```
            func_obj.Only_Number(e);
        }

        private void Tb_Key_Size_TextChanged(object sender, TextChangedEventArgs e)
        {
            tb_Key_Size.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173,
179));
        }

        private void Tb_Key_1_PreviewMouseDown(object sender, MouseButtonEventArgs
e)
        {
            func_obj.Check_for_Text(tb_Key_1);
        }

        private void Tb_Key_1_LostFocus(object sender, RoutedEventArgs e)
        {
            func_obj.Check_the_Cursor(tb_Key_1, 1);
        }

        private void Tb_Key_1_PreviewKeyDown(object sender, KeyEventArgs e)
        {
            func_obj.Without_a_Space(tb_Key_1, e);
        }

        private void Tb_Key_1_TextChanged(object sender, TextChangedEventArgs e)
        {
            try
            {
                if (tb_Key_1.Text == "Ваш ключ" || tb_Key_1.Text == "")
                {
                    btn_OK.IsEnabled = false;
                }
                else
                {
                    btn_OK.IsEnabled = true;
                }
            }
            catch (Exception)
            {

            }
        }

        #endregion

        #region Меню для генерации ключей RSA

        private void Btn_Key_Generation_2_Click(object sender, RoutedEventArgs e)
        {
            Key_Generation_Selection();
        }
```

```
private void Btn_Backward_1_Click(object sender, RoutedEventArgs e)
{
   transition = false;

   Grid_Main_Key_Menu.Visibility = Visibility.Visible;
   menu_btn_SaveFileAs_Key.IsEnabled = true;

   tb_Key_P.Text = "";
   tb_Key_Q.Text = "";
   tb_Public_Key.Text = "";
   tb_Privat_Key.Text = "";
   func_obj.Check_the_Cursor(tb_Key_P, 2);
   func_obj.Check_the_Cursor(tb_Key_Q, 3);
   func_obj.Check_the_Cursor(tb_Public_Key, 4);
   func_obj.Check_the_Cursor(tb_Privat_Key, 5);

   Grid_Generation_Key_Menu_2.Visibility = Visibility.Collapsed;
}

private void Btn_OK_1_Click(object sender, RoutedEventArgs e)
{
   if (func_obj.Confirm_Action(1))
   {
      func_obj.Save_File_As(cb_Algorithms, tb_Public_Key, true);
      func_obj.Save_File_As(cb_Algorithms, tb_Privat_Key, false);
   }

   transition = false;

   Grid_Main_Key_Menu.Visibility = Visibility.Visible;
   menu_btn_SaveFileAs_Key.IsEnabled = true;

   if (rb_Encryption.IsChecked == true)
   {
      tb_Key.Text = tb_Public_Key.Text;
   }
   else
   {
      tb_Key.Text = tb_Privat_Key.Text;
   }

   tb_Key_P.Text = "";
   tb_Key_Q.Text = "";
   tb_Public_Key.Text = "";
   tb_Privat_Key.Text = "";
   func_obj.Check_the_Cursor(tb_Key_P, 2);
   func_obj.Check_the_Cursor(tb_Key_Q, 3);
   func_obj.Check_the_Cursor(tb_Public_Key, 4);
   func_obj.Check_the_Cursor(tb_Privat_Key, 5);

   Grid_Generation_Key_Menu_2.Visibility = Visibility.Collapsed;
```

```
        }

        private void Tb_Key_P_PreviewTextInput(object sender,
TextCompositionEventArgs e)
        {
            func_obj.Only_Number(e);
        }

        private void Tb_Key_P_PreviewMouseDown(object sender, MouseButtonEventArgs
e)
        {
            func_obj.Check_for_Text(tb_Key_P);
        }

        private void Tb_Key_P_PreviewKeyDown(object sender, KeyEventArgs e)
        {
            func_obj.Without_a_Space(tb_Key_P, e);
        }

        private void Tb_Key_P_LostFocus(object sender, RoutedEventArgs e)
        {
            func_obj.Check_the_Cursor(tb_Key_P, 2);
        }

        private void Tb_Key_P_TextChanged(object sender, TextChangedEventArgs e)
        {
            tb_Key_P.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));
        }

        private void Tb_Key_Q_TextChanged(object sender, TextChangedEventArgs e)
        {
            tb_Key_Q.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));
        }

        private void Tb_Key_Q_LostFocus(object sender, RoutedEventArgs e)
        {
            func_obj.Check_the_Cursor(tb_Key_Q, 3);
        }

        private void Tb_Key_Q_PreviewKeyDown(object sender, KeyEventArgs e)
        {
            func_obj.Without_a_Space(tb_Key_Q, e);
        }

        private void Tb_Key_Q_PreviewMouseDown(object sender,
MouseButtonEventArgs e)
        {
            func_obj.Check_for_Text(tb_Key_Q);
        }

        private void Tb_Key_Q_PreviewTextInput(object sender,
TextCompositionEventArgs e)
```

```
        {
            func_obj.Only_Number(e);
        }

        //открытый

        private void Tb_Public_Key_LostFocus(object sender, RoutedEventArgs e)
        {
            func_obj.Check_the_Cursor(tb_Public_Key, 4);
        }

        private void Tb_Public_Key_PreviewKeyDown(object sender, KeyEventArgs e)
        {
            func_obj.Without_a_Space(tb_Public_Key, e);
        }

        private void Tb_Public_Key_PreviewMouseDown(object sender,
MouseButtonEventArgs e)
        {
            func_obj.Check_for_Text(tb_Public_Key);
        }

        private void Tb_Public_Key_TextChanged(object sender, TextChangedEventArgs e)
        {
            Unlock_Btn();
        }

        //закрытый

        private void Tb_Privat_Key_LostFocus(object sender, RoutedEventArgs e)
        {
            func_obj.Check_the_Cursor(tb_Privat_Key, 5);
        }

        private void Tb_Privat_Key_PreviewKeyDown(object sender, KeyEventArgs e)
        {
            func_obj.Without_a_Space(tb_Privat_Key, e);
        }

        private void Tb_Privat_Key_PreviewMouseDown(object sender,
MouseButtonEventArgs e)
        {
            func_obj.Check_for_Text(tb_Privat_Key);
        }

        private void Tb_Privat_Key_TextChanged(object sender, TextChangedEventArgs e)
        {
            Unlock_Btn();
        }

        #endregion
```

```
        #endregion

        #region Прочее

        public void List_of_Operations_to_Perform() //меню, определяющее какой метод выполнять
        {
            try
            {
                tb_Key.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

                switch (cb_Algorithms.SelectedIndex)
                {
                    case -1: cb_Algorithms_Border.Background = Brushes.Red; break;
                    case 0: Transposition_Cipher(); break;
                    case 1: Monoalphabetic_Cipher(); break;
                    case 2: Polyalphabetic_Cipher(); break;
                    case 3: XOR_Cipher(); break;
                    case 4: Vernam_Cipher(); break;
                    case 5: RSA_Cipher(); break;
                }

                Additional_Func();
                //tb_Key.ToolTip = "Ваш ключ";

            }
            catch (Exception)
            {
                tb_Key.ToolTip = "Неверный ключ";
                tb_Key.BorderBrush = Brushes.Red;
            }
        }

        public void Displaying_the_Key_Generation_Menu(bool a) //переходы
        {
            if (a == true)
            {
                transition = true;
                menu_btn_SaveFileAs_Key.IsEnabled = false;
                Grid_Main_Key_Menu.Visibility = Visibility.Collapsed;
                Grid_Generation_Key_Menu_1.Visibility = Visibility.Visible;
                Grid_Generation_Key_Menu_2.Visibility = Visibility.Collapsed;
            }
            else
            {
                transition = true;
                menu_btn_SaveFileAs_Key.IsEnabled = false;
                Grid_Main_Key_Menu.Visibility = Visibility.Collapsed;
                Grid_Generation_Key_Menu_1.Visibility = Visibility.Collapsed;
                Grid_Generation_Key_Menu_2.Visibility = Visibility.Visible;
            }
        }
```

```
        public void List_of_Key_Generation_Methods() //меню, определяющее для какого способа генерировать ключ
        {
            switch (cb_Algorithms.SelectedIndex)
            {
                case 0:
                    if (transition == false)
                    {
                        Displaying_the_Key_Generation_Menu(true);
                    }
                    else
                    {
                        Key_Generation_Selection();
                    }
                    break;

                case 1: Key_Generation_Selection(); break;

                case 2:
                    if (transition == false)
                    {
                        Displaying_the_Key_Generation_Menu(true);
                    }
                    else
                    {
                        Key_Generation_Selection();
                    }
                    break;

                case 3:
                    if (transition == false)
                    {
                        Displaying_the_Key_Generation_Menu(true);
                    }
                    else
                    {
                        Key_Generation_Selection();
                    }
                    break;

                case 4: Key_Generation_Selection(); break;

                case 5:
                    if (transition == false)
                    {
                        Displaying_the_Key_Generation_Menu(false);
                    }
                    else
                    {
                        Key_Generation_Selection();
                    }
```

```
                break;
            }
        }

        public void Key_Generation_Selection() //меню, для выбора генерации ключей
        {
            try
            {
                switch (cb_Algorithms.SelectedIndex)
                {
                    case 0:
                        tb_Key_1.Text = rndkey_obj.Rand_Key_Generation_Transposition(Convert.ToInt32(tb_Key_Size.Text));
                        tb_Key_1.Foreground = Brushes.Black;
                        break;

                    case 1: tb_Key.Text = rndkey_obj.Rand_Key_Generation(); break;
                    case 2:
                        tb_Key_1.Text = rndkey_obj.Rand_Key_Generation(Convert.ToInt32(tb_Key_Size.Text));
                        tb_Key_1.Foreground = Brushes.Black;
                        break;

                    case 3:
                        tb_Key_1.Text = rndkey_obj.Rand_Key_Generation(Convert.ToInt32(tb_Key_Size.Text));
                        tb_Key_1.Foreground = Brushes.Black;
                        break;

                    case 4: tb_Key.Text = rndkey_obj.Rand_Key_Generation(tb_SourceData.Text.Length); break;
                    case 5:
                        int p = 0, q = 0, i = 0;

                        p = func_obj.Check(p, tb_Key_P);
                        q = func_obj.Check(q, tb_Key_Q);

                        if (p > 10 && p != q && rsa_obj.IsTheNumberSimple(p))
                        {
                            i++;
                        }
                        else
                        {
                            tb_Key_P.BorderBrush = Brushes.Red;
                        }

                        if (q > 10 && q != p && rsa_obj.IsTheNumberSimple(q))
                        {
                            i++;
                        }
                        else
                        {
```

```
                    tb_Key_Q.BorderBrush = Brushes.Red;
                }

                if (i == 2)
                {
                    tb_Key_P.BorderBrush = new SolidColorBrush(Color.FromRgb(171,
173, 179));
                    tb_Key_Q.BorderBrush = new SolidColorBrush(Color.FromRgb(171,
173, 179));

                    long n = p * q;
                    long fi = (p - 1) * (q - 1);
                    long e = rsa_obj.Calculate_e(fi);
                    long d = rsa_obj.Calculate_d(e, fi);

                    tb_Public_Key.Foreground = Brushes.Black;
                    tb_Privat_Key.Foreground = Brushes.Black;

                    tb_Public_Key.Text = Convert.ToString(e) + " " + Convert.ToString(n);
                    tb_Privat_Key.Text = Convert.ToString(d) + " " + Convert.ToString(n);
                }

                    break;
                }
            }
            catch (Exception)
            {
                if (cb_Algorithms.SelectedIndex == 0 || cb_Algorithms.SelectedIndex == 2 ||
cb_Algorithms.SelectedIndex == 3)
                {
                    tb_Key_Size.BorderBrush = Brushes.Red;
                }
            }
        }

        public void Calling_Help()
        {
            try
            {
                string commandText = "Help.chm";
                var proc = new Process();
                proc.StartInfo.FileName = commandText;
                proc.StartInfo.UseShellExecute = true;
                proc.Start();
            }
            catch
            {
                MessageBox.Show("Файл справки не найден!", "Не найден файл");
            }
        }

        public void Refresh() //обновить всё
```

```
{
    transition = false;

    //операция
    rb_Encryption.IsChecked = true;

    //способ шифрования
    cb_Algorithms.SelectedIndex = -1;
    cb_Algorithms_Border.Background = this.Background;

    //исходные
    tb_FileName_Source.Text = "";
    tb_SourceData.Text = "";
    tb_SourceData.FontSize = 14;

    //результат
    tb_EncryptedData.Text = "";
    tb_EncryptedData.FontSize = 14;

    //ключ
    menu_btn_SaveFileAs_Key.IsEnabled = true;
    tb_Key.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));
    tb_Key.Text = "";
    tb_Key.FontSize = 14;

    //Дополнительные действия
    chb_SaveKey.IsChecked = false;
    chb_SaveEncryptText.IsChecked = false;
    chb_SaveSourceText.IsChecked = false;
    chb_Refresh.IsChecked = false;

    //генарация ключа с вводом размера
    tb_Key_Size.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173,
179));
    tb_Key_Size.Text = "";
    tb_Key_1.Text = "";
    func_obj.Check_the_Cursor(tb_Key_Size, 0);
    func_obj.Check_the_Cursor(tb_Key_1, 1);

    //генарация ключа RSA
    tb_Key_P.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));
    tb_Key_Q.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));
    tb_Key_P.Text = "";
    tb_Key_Q.Text = "";
    tb_Public_Key.Text = "";
    tb_Privat_Key.Text = "";
    func_obj.Check_the_Cursor(tb_Key_P, 2);
    func_obj.Check_the_Cursor(tb_Key_Q, 3);
    func_obj.Check_the_Cursor(tb_Public_Key, 4);
    func_obj.Check_the_Cursor(tb_Privat_Key, 5);

    Grid_Main_Key_Menu.Visibility = Visibility.Visible;
```

```
            Grid_Generation_Key_Menu_1.Visibility = Visibility.Collapsed;
            Grid_Generation_Key_Menu_2.Visibility = Visibility.Collapsed;
        }

        public void Additional_Func()
        {
            if (chb_SaveSourceText.IsChecked == true)
            {
                func_obj.Save_File_As(tb_FileName_Source, tb_SourceData);
            }

            if (chb_SaveEncryptText.IsChecked == true)
            {
                func_obj.Save_File_As(cb_Algorithms, tb_EncryptedData, rb_Encryption,
true);
            }

            if (chb_SaveKey.IsChecked == true)
            {
                func_obj.Save_File_As(cb_Algorithms, tb_Key, rb_Encryption, false);
            }

            if (chb_Refresh.IsChecked == true)
            {
                Refresh();
            }
        }

        public void Unlock_Btn()
        {
            try
            {
                if (tb_Public_Key.Text == "Ваш открытый ключ" || tb_Public_Key.Text == ""
                    || tb_Privat_Key.Text == "Ваш закрытый ключ" || tb_Privat_Key.Text == "")
                {
                    btn_OK_1.IsEnabled = false;
                }
                else
                {
                    btn_OK_1.IsEnabled = true;
                }
            }
            catch (Exception)
            {

            }
        }

        private void Window_KeyDown(object sender, KeyEventArgs e) //горячие клавиши
        {
            switch (e.Key)
            {
```

```
                case Key.F1: List_of_Operations_to_Perform(); break; //выполнить
                case Key.F2: Refresh(); break; //обновить
                case Key.F3: if (cb_Algorithms.SelectedIndex != -1) List_of_Key_Generation_Methods(); break; //сгенерировать ключ
                case Key.F4: func_obj.File_Selection(tb_FileName_Source, tb_SourceData, tb_EncryptedData); break; //открыть файл с исходным текстом
                case Key.F5: func_obj.File_Selection(tb_Key); break; //открыть файл с ключом
                case Key.F6: if (tb_FileName_Source.Text != "") func_obj.Save_File(tb_FileName_Source, tb_SourceData); break; //сохранить
                case Key.F7: func_obj.Save_File_As(tb_FileName_Source, tb_SourceData); break; //сохранить исходный текст
                case Key.F8: func_obj.Save_File_As(cb_Algorithms, tb_EncryptedData, rb_Encryption, true); break; //сохранить зашифрованный/дешифрованный текст
                case Key.F9: if (Grid_Main_Key_Menu.Visibility == Visibility.Visible) func_obj.Save_File_As(cb_Algorithms, tb_Key, rb_Encryption, false); break; //сохранить ключ
                case Key.F11: Calling_Help(); break; //о программе
                case Key.F12: this.Close(); break; //выход
            }
        }

        private void Cb_Algorithms_SelectionChanged(object sender, SelectionChangedEventArgs e) //изменения комбобокса
        {
            transition = false;
            Grid_Main_Key_Menu.Visibility = Visibility.Visible;

            tb_Key_Size.Text = "";
            tb_Key_1.Text = "";
            func_obj.Check_the_Cursor(tb_Key_Size, 0);
            func_obj.Check_the_Cursor(tb_Key_1, 1);
            Grid_Generation_Key_Menu_1.Visibility = Visibility.Collapsed;

            tb_Key_P.Text = "";
            tb_Key_Q.Text = "";
            tb_Public_Key.Text = "";
            tb_Privat_Key.Text = "";
            func_obj.Check_the_Cursor(tb_Key_P, 2);
            func_obj.Check_the_Cursor(tb_Key_Q, 3);
            func_obj.Check_the_Cursor(tb_Public_Key, 4);
            func_obj.Check_the_Cursor(tb_Privat_Key, 5);
            Grid_Generation_Key_Menu_2.Visibility = Visibility.Collapsed;

            cb_Algorithms_Border.Background = this.Background;
            menu_btn_SaveFileAs_Key.IsEnabled = true; //разблокировать кнопку сохранение ключа в меню
            func_obj.Changed_CB(cb_Algorithms, btn_Key_Generation, menu_btn_Key_Generation); //разблокирует или блокирует кнопку "Сгенерировать ключ"
        }

        private void Rb_Encryption_Checked(object sender, RoutedEventArgs e)
        {
```

```
            gb_algs.Header = "Способ шифрования";
            chb_SaveEncryptText.Content = "Сохранить зашифрованный текст";
            menu_btn_SaveFileAs_Encrypted.Header = "Сохранить зашифрованный текст";
        }

        private void Rb_Decryption_Checked(object sender, RoutedEventArgs e)
        {
            gb_algs.Header = "Способ дешифрования";
            chb_SaveEncryptText.Content = "Сохранить дешифрованный текст";
            menu_btn_SaveFileAs_Encrypted.Header = "Сохранить дешифрованный текст";
        }

        private void Btn_ExecuteOperation_Click(object sender, RoutedEventArgs e)
        {
            List_of_Operations_to_Perform();
        }

        private void Btn_Key_Generation_Click(object sender, RoutedEventArgs e)
        {
            List_of_Key_Generation_Methods();
        }

        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (MessageBox.Show("Вы действительно хотите выйти?", "Выйти?", MessageBoxButton.YesNo) == MessageBoxResult.Yes)
            {
                e.Cancel = false;
            }
            else
            {
                e.Cancel = true;
            }
        }

        #endregion

        #endregion
    }
}
```

Alphabet.cs

```
using System;

namespace CryptographicApplication
{
    class Alphabet
    {
```

```
        public char[] lang = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюяABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!\"#$%^&*()+=-_'?.,|/`~№:;@[]{}\\".ToCharArray();
    }
}
```

Functional.cs

```
using System;
using System.Text;
using System.Windows.Controls;
using System.IO;
using Microsoft.Win32;
using System.Windows.Media;
using System.Windows.Input;
using System.Windows;

namespace CryptographicApplication
{
    class Functional
    {
        public void Clear_tb(TextBox a)
        {
            a.Text = "";
        }

        public void Font_Size(TextBox a, bool b)
        {
            if (b == true)
            {
                if (a.FontSize < 20)
                    a.FontSize++;
            }
            else
            {
                if (a.FontSize > 1)
                    a.FontSize--;
            }
        }

        public void File_Selection(TextBox localFileText) // для ключа
        {
            OpenFileDialog openFileDlg = new OpenFileDialog();

            openFileDlg.DefaultExt = ".txt";
            openFileDlg.Filter = "Текстовый документ (.txt) | * .txt";
            openFileDlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

            Nullable<bool> result = openFileDlg.ShowDialog();

            if (result == true)
```

```
            {
                localFileText.Text = File.ReadAllText(openFileDlg.FileName,
Encoding.Default);
            }
        }

        public void File_Selection(TextBox localFileName, TextBox localFileText, TextBox
clearFileText) // для исходного
        {
            OpenFileDialog openFileDlg = new OpenFileDialog();

            openFileDlg.DefaultExt = ".txt";
            openFileDlg.Filter = "Текстовый документ (.txt) | * .txt";
            openFileDlg.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

            Nullable<bool> result = openFileDlg.ShowDialog();

            if (result == true)
            {
                localFileName.Text = openFileDlg.FileName;
                localFileText.Text = File.ReadAllText(openFileDlg.FileName,
Encoding.Default);
                clearFileText.Text = "";
            }
        }

        public void Save_File(TextBox localFileName, TextBox TextToSave)
        {
            File.WriteAllText(localFileName.Text, TextToSave.Text, Encoding.Default);
        }

        public void Save_File_As(TextBox NewFileName, TextBox TextToSave) // для
исходного
        {
            SaveFileDialog saveFileDlg = new SaveFileDialog();

            saveFileDlg.DefaultExt = ".txt";
            saveFileDlg.Filter = "Текстовый документ (.txt) | * .txt";
            saveFileDlg.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            saveFileDlg.FileName = "Новый текстовый документ";

            Nullable<bool> result = saveFileDlg.ShowDialog();

            if (result == true)
            {
                NewFileName.Text = saveFileDlg.FileName;
                File.WriteAllText(saveFileDlg.FileName, TextToSave.Text,
Encoding.Default);
            }
        }
```

```
        public void Save_File_As(ComboBox cb, TextBox TextToSave, RadioButton operation, bool a) // для ключа и зашифрованного
        {
            SaveFileDialog saveFileDlg = new SaveFileDialog();

            saveFileDlg.DefaultExt = ".txt";
            saveFileDlg.Filter = "Текстовый документ (.txt) | * .txt";
            saveFileDlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

            if (a == true)
            {
                if (operation.IsChecked == true)
                {
                    saveFileDlg.FileName = "Зашифрованный текст (способ - " + cb.Text + ")";
                }
                else
                {
                    saveFileDlg.FileName = "Дешифрованный текст (способ - " + cb.Text + ")";
                }
            }
            else
            {
                saveFileDlg.FileName = "Ключ (способ - " + cb.Text + ")";
            }

            Nullable<bool> result = saveFileDlg.ShowDialog();

            if (result == true)
            {
                File.WriteAllText(saveFileDlg.FileName, TextToSave.Text, Encoding.Default);
            }
        }

        public void Save_File_As(ComboBox cb, TextBox TextToSave, bool a) // для открытого и закрытого ключей
        {
            SaveFileDialog saveFileDlg = new SaveFileDialog();

            saveFileDlg.DefaultExt = ".txt";
            saveFileDlg.Filter = "Текстовый документ (.txt) | * .txt";
            saveFileDlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

            if (a == true)
            {
                saveFileDlg.FileName = "Открытый ключ (способ - " + cb.Text + ")";
            }
            else
            {
```

```
            saveFileDlg.FileName = "Закрытый ключ (способ - " + cb.Text + ")";
        }


        Nullable<bool> result = saveFileDlg.ShowDialog();

        if (result == true)
        {
            File.WriteAllText(saveFileDlg.FileName, TextToSave.Text,
Encoding.Default);
        }
    }

    public void Changed_File_Name_TB(TextBox ChangedTB, Button BtnUsed,
MenuItem MIUsed)
    {
        if (ChangedTB.Text == "")
        {
            BtnUsed.IsEnabled = false;
            MIUsed.IsEnabled = false;
        }
        else
        {
            BtnUsed.IsEnabled = true;
            MIUsed.IsEnabled = true;
        }
    }

    public void Changed_CB(ComboBox ChangedCB, Button BtnUsed, MenuItem
MIUsed)
    {
        if (ChangedCB.SelectedIndex == -1)
        {
            BtnUsed.IsEnabled = false;
            MIUsed.IsEnabled = false;
        }
        else
        {
            BtnUsed.IsEnabled = true;
            MIUsed.IsEnabled = true;
        }
    }

    public void Check_for_Text(TextBox a)
    {
        if (a.Foreground != Brushes.Black)
        {
            a.Text = "";
            a.Foreground = Brushes.Black;
        }
    }
```

```
public void Check_the_Cursor(TextBox a, int b)
{
   if (a.Text == "")
   {
      a.Foreground = new SolidColorBrush(Color.FromRgb(171, 173, 179));
      switch (b)
      {
         case 0: a.Text = "Введите размер ключа"; break;
         case 1: a.Text = "Ваш ключ"; break;
         case 2: a.Text = "Введите P"; break;
         case 3: a.Text = "Введите Q"; break;
         case 4: a.Text = "Ваш открытый ключ"; break;
         case 5: a.Text = "Ваш закрытый ключ"; break;
      }
   }
}

public void Only_Number(TextCompositionEventArgs e)
{
   e.Handled = !(Char.IsDigit(e.Text, 0));
}

public void Without_a_Space(TextBox a, KeyEventArgs e)
{
   if (e.Key == Key.Space)
   {
      e.Handled = true;
   }
}

public int Check(int value, TextBox a)
{
   try
   {
      value = Convert.ToInt32(a.Text);
      return value;
   }
   catch (Exception)
   {
      a.BorderBrush = Brushes.Red;
      return 0;
   }
}

public bool Confirm_Action(int a) //подтверждение выхода или сохранения файла
{
   bool choice = false;

   switch (a)
   {
      case 0:
```

```
                    if (MessageBox.Show("Вы хотите сохранить ключ?", "Сохранить ключ?",
MessageBoxButton.YesNo) == MessageBoxResult.Yes)
                    {
                       choice = true;
                    }
                    else
                    {
                       choice = false;
                    }
                    break;

                 case 1:
                    if (MessageBox.Show("Вы хотите сохранить ключи?", "Сохранить
ключи?", MessageBoxButton.YesNo) == MessageBoxResult.Yes)
                    {
                       choice = true;
                    }
                    else
                    {
                       choice = false;
                    }
                    break;
              }

              return choice;
           }
        }
     }
```

RandomKeyGeneration.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Controls;

namespace CryptographicApplication
{
   class RandomKeyGeneration
   {
      Alphabet alph = new Alphabet();
      RSA RSA_obj = new RSA();

      Random rnd = new Random();

      public string Rand_Key_Generation()
      {
         int value = rnd.Next(1, alph.lang.Length);
         return Convert.ToString(value);
      }

      public string Rand_Key_Generation(int sizesourcetext)
      {
```

```
            StringBuilder code = new StringBuilder();

            int value;

            for (int i = 0; i < sizesourcetext; i++)
            {
                value = rnd.Next(0, alph.lang.Length);
                code.Append(alph.lang[value]);
            }

            return code.ToString();
        }

        public string Rand_Key_Generation_Transposition(int sizesourcetext)
        {
            StringBuilder code = new StringBuilder();
            int[] arr = new int[sizesourcetext];
            int value1, value2, temp;

            for (int i = 0; i < sizesourcetext; i++)
            {
                arr[i] = i + 1;
            }

            for (int i = 0; i < sizesourcetext; i++) //перемешиваем элементы
            {
                value1 = rnd.Next(0, sizesourcetext);
                value2 = rnd.Next(0, sizesourcetext);

                temp = arr[value1];
                arr[value1] = arr[value2];
                arr[value2] = temp;
            }

            for (int i = 0; i < sizesourcetext; i++)
            {
                if (i == sizesourcetext - 1)
                {
                    code.Append(arr[i]);
                }
                else
                {
                    code.Append(arr[i] + " ");
                }
            }

            return code.ToString();
        }
    }
}
```

Transposition.cs

```
using System;

namespace CryptographicApplication
{
   class Transposition
   {
      private int[] key = null;

      public void SetKey(int[] _key)
      {
         key = new int[_key.Length];

         for (int i = 0; i < _key.Length; i++)
            key[i] = _key[i];
      }

      public void SetKey(string[] _key)
      {
         key = new int[_key.Length];

         for (int i = 0; i < _key.Length; i++)
            key[i] = Convert.ToInt32(_key[i]);
      }

      public void SetKey(string _key)
      {
         SetKey(_key.Split(' '));
      }

      public string Encrypt(string input)
      {
         for (int i = 0; i < input.Length % key.Length; i++)
            input += input[i];

         string result = "";

         for (int i = 0; i < input.Length; i += key.Length)
         {
            char[] transposition = new char[key.Length];

            for (int j = 0; j < key.Length; j++)
               transposition[key[j] - 1] = input[i + j];

            for (int j = 0; j < key.Length; j++)
               result += transposition[j];
         }

         return result;
      }

      public string Decrypt(string input)
      {
```

```
            string result = "";

            for (int i = 0; i < input.Length; i += key.Length)
            {
                char[] transposition = new char[key.Length];

                for (int j = 0; j < key.Length; j++)
                    transposition[j] = input[i + key[j] - 1];

                for (int j = 0; j < key.Length; j++)
                    result += transposition[j];
            }

            return result;
        }
    }
}
```

Monoalphabetic.cs

```
using System;
using System.Text;

namespace CryptographicApplication
{
    class Monoalphabetic
    {
        Alphabet alph = new Alphabet();

        public string Encrypt(string sourcetext, int shift)
        {
            StringBuilder code = new StringBuilder();

            for (int i = 0; i < sourcetext.Length; i++)
            {
                //поиск символа в алфавите
                for (int j = 0; j < alph.lang.Length; j++)
                {
                    //если символ найден
                    if (sourcetext[i] == alph.lang[j])
                    {
                        code.Append(alph.lang[(j + shift) % alph.lang.Length]);
                        break;
                    }
                    //если символ не найден
                    else if (j == alph.lang.Length - 1)
                    {
                        code.Append(sourcetext[i]);
                    }
                }
            }

            return code.ToString();
```

```
        }

        public string Decrypt(string sourcetext, int shift)
        {
            StringBuilder code = new StringBuilder();

            for (int i = 0; i < sourcetext.Length; i++)
            {
                //поиск символа в алфавите
                for (int j = 0; j < alph.lang.Length; j++)
                {
                    //если символ найден
                    if (sourcetext[i] == alph.lang[j])
                    {
                        code.Append(alph.lang[(j - shift + alph.lang.Length) % alph.lang.Length]);
                        break;
                    }
                    //если символ не найден
                    else if (j == alph.lang.Length - 1)
                    {
                        code.Append(sourcetext[i]);
                    }
                }
            }

            return code.ToString();
        }
    }
}
```

Polyalphabetic.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CryptographicApplication
{
    class Polyalphabetic
    {
        Alphabet alph = new Alphabet();

        public string Encrypt(string sourcetext, string key)
        {
            StringBuilder code = new StringBuilder();
            int[] key_id = new int[key.Length];
            int t = 0;

            //поиск индексов букв ключа
            for (int i = 0; i < key.Length; i++)
            {
```

```
        for (int j = 0; j < alph.lang.Length; j++)
        {
            if (key[i] == alph.lang[j])
            {
                key_id[i] = j;
                break;
            }
        }
    }

    for (int i = 0; i < sourcetext.Length; i++)
    {
        //поиск символа в алфавите
        for (int j = 0; j < alph.lang.Length; j++)
        {
            //если символ найден
            if (sourcetext[i] == alph.lang[j])
            {
                    if (t > key.Length - 1)
                    {
                        t = 0;
                    }
                    code.Append(alph.lang[(j + key_id[t]) % alph.lang.Length]);
                    t++;

                break;
            }
            //если символ не найден
            else if (j == alph.lang.Length - 1)
            {
                code.Append(sourcetext[i]);
                t++;
            }
        }
    }

    return code.ToString();
}

public string Decrypt(string sourcetext, string key)
{
    StringBuilder code = new StringBuilder();
    int[] key_id = new int[key.Length];
    int t = 0;

    //поиск индексов букв ключа
    for (int i = 0; i < key.Length; i++)
    {
        for (int j = 0; j < alph.lang.Length; j++)
        {
            if (key[i] == alph.lang[j])
            {
```

```
                    key_id[i] = j;
                    break;
                }
            }
        }

        for (int i = 0; i < sourcetext.Length; i++)
        {
            //поиск символа в алфавите
            for (int j = 0; j < alph.lang.Length; j++)
            {
                //если символ найден
                if (sourcetext[i] == alph.lang[j])
                {
                    if (t > key.Length - 1)
                    {
                        t = 0;
                    }
                    code.Append(alph.lang[(j + alph.lang.Length - key_id[t]) %
alph.lang.Length]);
                    t++;
                    break;
                }
                //если символ не найден
                else if (j == alph.lang.Length - 1)
                {
                    code.Append(sourcetext[i]);
                    t++;
                }
            }
        }

        return code.ToString();
    }
  }
}
```

XOR.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CryptographicApplication
{
  class XOR
  {
    Alphabet alph = new Alphabet();

    public string Encrypt_and_Decrypt(string sourcetext, string key)
    {
```

```
            StringBuilder code = new StringBuilder();
            int[] key_id = new int[key.Length];
            int t = 0;

            //поиск индексов букв ключа
            for (int i = 0; i < key.Length; i++)
            {
               for (int j = 0; j < alph.lang.Length; j++)
               {
                  if (key[i] == alph.lang[j])
                  {
                     key_id[i] = j;
                     break;
                  }
               }
            }

            for (int i = 0; i < sourcetext.Length; i++)
            {
               //поиск символа в алфавите
               for (int j = 0; j < alph.lang.Length; j++)
               {
                  //если символ найден
                  if (sourcetext[i] == alph.lang[j])
                  {
                     if (t > key.Length - 1)
                     {
                        t = 0;
                     }
                     code.Append(alph.lang[(j ^ key_id[t] % 32) % alph.lang.Length]);
                     t++;
                     break;
                  }
                  //если символ не найден
                  else if (j == alph.lang.Length - 1)
                  {
                     code.Append(sourcetext[i]);
                     t++;
                  }
               }
            }

            return code.ToString();
         }
      }
   }
```

Vernam.cs

```
using System;
using System.Linq;
using System.Text;
using System.Windows.Controls;
```

```
namespace CryptographicApplication
{
   class Vernam
   {
      Alphabet alph = new Alphabet();

      public string Encrypt_and_Decrypt(string sourcetext, string key)
      {
         StringBuilder code = new StringBuilder();
         int[] key_id = new int[key.Length];

         //поиск индексов букв ключа
         for (int i = 0; i < key.Length; i++)
         {
            for (int j = 0; j < alph.lang.Length; j++)
            {
               if (key[i] == alph.lang[j])
               {
                  key_id[i] = j;
                  break;
               }
            }
         }

         for (int i = 0; i < sourcetext.Length; i++)
         {
            //поиск символа в алфавите
            for (int j = 0; j < alph.lang.Length; j++)
            {
               //если символ найден
               if (sourcetext[i] == alph.lang[j])
               {
                  code.Append(alph.lang[(j ^ key_id[i] % 32) % alph.lang.Length]);
                  break;
               }
               //если символ не найден
               else if (j == alph.lang.Length - 1)
               {
                  code.Append(sourcetext[i]);
               }
            }
         }

         return code.ToString();
      }
   }
}
```

RSA.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Text;
using System.Numerics;

namespace CryptographicApplication
{
   class RSA
   {
      Alphabet alph = new Alphabet();

      private int[] key = null;

      public void SetKey(int[] _key)
      {
         key = new int[_key.Length];

         for (int i = 0; i < _key.Length; i++)
            key[i] = _key[i];
      }

      public void SetKey(string[] _key)
      {
         key = new int[_key.Length];

         for (int i = 0; i < _key.Length; i++)
            key[i] = Convert.ToInt32(_key[i]);
      }

      public void SetKey(string _key)
      {
         SetKey(_key.Split(' '));
      }

      public string Encrypt(string sourcetext)
      {
         StringBuilder code = new StringBuilder();

         long e = Convert.ToInt64(key[0]);
         long n = Convert.ToInt64(key[1]);

         List<string> result = RSA_Endoce(sourcetext, e, n);

         foreach (string item in result)
            code.Append(item + "\n");

         return code.ToString();
      }

      public string Decrypt(List<string> sourcetext)
      {
         StringBuilder code = new StringBuilder();

         long d = Convert.ToInt64(key[0]);
```

```
    long n = Convert.ToInt64(key[1]);

    code.Append(RSA_Dedoce(sourcetext, d, n));

    return code.ToString();
}

private List<string> RSA_Endoce(string sourcetext, long e, long n) //шифрование
{
    List<string> result = new List<string>();

    BigInteger bi;

    for (int i = 0; i < sourcetext.Length; i++)
    {
        int index = Array.IndexOf(alph.lang, sourcetext[i]);

        bi = new BigInteger(index);
        bi = BigInteger.Pow(bi, (int)e);

        BigInteger bn = new BigInteger((int)n);

        bi = bi % bn;

        result.Add(bi.ToString());
    }

    return result;
}

private string RSA_Dedoce(List<string> sourcetext, long d, long n) //дешифрование
{
    string result = "";
    int i = 0;

    BigInteger bi;

    foreach (string item in sourcetext)
    {
        bi = new BigInteger(Convert.ToDouble(item));
        bi = BigInteger.Pow(bi, (int)d);

        BigInteger bn = new BigInteger((int)n);

        bi = bi % bn;

        int index = Convert.ToInt32(bi.ToString());

        if (index == -1)
        {
            if (i < 1)
            {
```

```
                result += " ";
                i++;
            }
            else
            {
                result += "\n";
                i = 0;
            }
        }
        else
        {
            result += alph.lang[index].ToString();
            i = 0;
        }
    }

    return result;
}

public bool IsTheNumberSimple(int n) //является ли число простым
{
    if (n < 2)
        return false;

    if (n == 2)
        return true;

    for (int i = 2; i < n; i++)
        if (n % i == 0)
            return false;

    return true;
}

public long Calculate_e(long fi) //вычисление открытой экспоненты
{
    long e = 2;

    for (int i = 2; i <= fi; i++)
        if ((fi % i == 0) && (e % i == 0) && e < fi) //если имеют общие делители
        {
            e++;
            i = 1;
        }

    return e;
}

public long Calculate_d(long e, long fi) //вычисление закрытой экспоненты
{
    long d = 2;
```

```
            while (true)
            {
                if ((d * e % fi == 1) && d != e)
                    break;
                else
                    d++;
            }

            return d;
        }
    }
}
```