МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт Факультет приборостроения, информационных технологий и электроники	Кафедра «Информационная безопасность систем и технологий»
Специальность 10.05.03 –	Информационная безопасность автоматизированных систем
Специализация -	Защищенные автоматизированные системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ИЗУЧЕНИЯ МЕТОДОВ ПОИСКА НЕДЕКЛАРИРОВАННЫХ ВОЗМОЖНОСТЕЙ ПРИ ФУНКЦИОНАЛЬНОМ ТЕСТИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Студент _____ Есьман Юлия Александровна

Руководитель _____ Егорова Н.А.

Нормоконтролёр _____ Фатеев А.Г.

Рецензент ведущий инженер АО «ПНИЭИ» _____ Ананьев Е.А.

Работа допущена к защите (протокол заседания кафедры от 15.06.20 № 13)

Заведующий кафедрой _____ С.Л. Зефиров

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 26.06.20 № 11)

Секретарь ГЭК _____ А.П. Иванов

Пенза 2020

Оглавление

Термины и определения…….....4
Перечень сокращений и определений.....7
Введение.....8
1 Анализ основных разновидностей недекларированных возможностей…….12
2 Анализ угроз ИБ, возникающих от недекларированных возможностей.....14
3 Анализ основных методов поиска недекларированных возможностей: при функциональном тестировании программных средств; при анализе исходного кода программных средств.....16
4 Разработка алгоритма создаваемой программы.....22
5 Описание разработки программы шифрования с введёнными в неё недекларированными возможностями.....30
6 Руководство пользователя программы.....48
6.1 Назначение программы……...48
6.2 Условия выполнения программы……...48
6.3 Выполнение программы……49
6.4 Сообщения оператору……49
7 Руководство по поиску недекларированных возможностей в разработанной программе шифрования.....51
7.1 НДВ № 1: программа добавляет ключ к файлу после процедуры шифрования, если изображение черно-белое…….....51
7.2 НДВ № 2: программа производит инверсию бит файла вместо его шифрования, если размер изображения не превышает 20 килобайт……53
7.3 НДВ № 3: программа вырабатывает периодическую гамму шифра для файлов формата BMP……56
8 Вредные психофизиологические факторы, влияющие на

психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....61
Заключение.....67
Список используемых источников.....70
Приложение А Алгоритм шифрования файлов……..…74
Приложение Б Алгоритм выработки гаммы шифра…….……...…82
Приложение В Код программы шифрования изображений…..……...91

Термины и определения

В настоящем отчете используются следующие термины с соответствующими определениями:

– защита информации - деятельность, направленная на предотвращение утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую информацию [1];

– источник угрозы безопасности информации - субъект (физическое лицо, материальный объект или физическое явление), являющийся непосредственной причиной возникновения угрозы безопасности информации [1];

– патч - часть программы, или небольшая отдельная программа, используемая для устранения проблем в программном обеспечении [2];

– программная закладка - преднамеренно внесенный в программное обеспечение функциональный объект, который при определенных условиях инициирует реализацию недекларированных возможностей программного обеспечения [3];

– угроза информационной безопасности - совокупность условий и факторов, создающих потенциальную или реально существующую опасность нарушения безопасности информации [1];

– злоумышленник - любое лицо, намеренно использующее уязвимости технических и нетехнических средств безопасности в целях захвата или компрометации информационных систем и сетей или затруднения доступа авторизованных пользователей к ресурсам информационной системы и сетевым ресурсам [4];

– автоматизированная система в защищённом исполнении - автоматизированная система, реализующая информационную технологию выполнения установленных функций в соответствии с требованиями стандартов и/или других нормативных документов по защите информации [5];

– программное средство - объект, состоящий из программ, процедур, правил, а также, если предусмотрено, сопутствующих им документации и данных, относящихся к функционированию системы обработки информации [6];

– несанкционированный доступ - доступ к информации или к ресурсам автоматизированной информационной системы, осуществляемый с нарушением установленных прав и (или) правил доступа [7];

– бэкдор - вредоносная программа, предназначенная для скрытого удалённого управления злоумышленником пораженным компьютером [8];

– руткит - программа или набор программ, использующих технологии сокрытия системных объектов (файлов, процессов, драйверов, сервисов, ключей реестра, открытых портов, соединений и пр.) посредством обхода механизмов системы [9];

– кейлоггер - любой компонент ПО или оборудования, который умеет перехватывать и записывать все манипуляции с клавиатурой компьютера [10];

– недекларированные возможности - функциональные возможности ПО, не описанные или не соответствующие описанным в документации, при использовании которых возможно нарушение конфиденциальности, доступности или целостности обрабатываемой информации [11];

– отказ в обслуживании - препятствие санкционированному доступу к ресурсам системы или задержка операций и функций системы [4];

– вредоносная программа - программа, предназначенная для осуществления несанкционированного доступа к информации и (или) воздействия на информацию или ресурсы информационной системы [2];

– инцидент информационной безопасности - единичное событие или серия нежелательных или неожиданных событий, связанных с информационной безопасностью, которые со значительной вероятностью

способны нанести ущерб бизнес-операциям (деловым операциям) и поставить под угрозу информационную безопасность [4];

– безопасность информации - состояние защищенности информации [данных], при котором обеспечены ее [их] конфиденциальность, доступность и целостность [1];

– событие информационной безопасности - идентифицированное появление определенного состояния системы, сервиса или сети, указывающего на возможное нарушение политики ИБ или отказ защитных мер, или возникновение неизвестной ранее ситуации, которая может иметь отношение к безопасности [27].

Перечень сокращений и определений

В настоящем отчете применяют следующие обозначения и сокращения:

– АСУ ТП - автоматизированная система управления технологическим процессом;

– НДВ - недекларированные возможности;

– ПО - программное обеспечение.

Введение

На современном этапе столь динамичного развития и применения информационных технологий неизбежно возникает вопрос защиты информации. Основная особенность информационных технологий состоит в том, что их определяющим компонентом является ПО. В большинстве случаев именно ПО и выступает источником угроз информационной безопасности. Одним из видов опасности для ПО является наличие НДВ.

НДВ – достаточно распространенное среди разработчиков явление.

НДВ программы наделяются не только из злых побуждений, чаще наоборот. Возможность удаленного управления системой, получения отладочных данных, установки патчей позволяет разработчикам сэкономить немало времени, сил и средств на этапе опытного внедрения. Даже известные и уважаемые разработчики нередко грешат преднамеренным внесением соответствующих программных закладок.

Собственно, и большинство заказчиков осведомлено об этой «серой» части технологического процесса. Другое дело, что по завершении опытной эксплуатации и сдаче продукта в эксплуатацию промышленную все эти лазейки должны быть перекрыты. Всегда ли это происходит на практике – вопрос, скорее, риторический.

Итак, можно прийти к выводу, что НДВ являются практически неизбежным компаньоном программных продуктов. Несмотря на то, что в принципе закладки могут и не быть направлены на реализацию угроз информационной безопасности, проблема заключается в том, что имеющиеся лазейки могут использовать злоумышленники. При этом следует учитывать, что в отдельных случаях программные закладки вносятся действительно со злым умыслом или «на всякий случай», для того, чтобы иметь средство воздействия на заказчика или пользователя программы. Известно немало случаев шантажа и диверсий, которые устраивали недовольные своим работодателем

программисты. В отсутствие контроля над результатами работы сотрудников установить «логическую бомбу» не составляет особого труда.

Именно поэтому необходимо осознавать проблему доверия к используемому ПО и исключать возможности появления угроз безопасности информации.

В современных условиях развития информационных технологий для организаций актуально знать и правильно применять существующие способы выявления НДВ. Ведь в целях снижения риска потери производительности бизнес-операций необходимо качественно и оперативно выявлять потенциально опасные и уязвимые участки кода.

Целью выпускной квалификационной работы является разработка программы шифрования файлов, содержащих изображения формата BMP и JPG, с внесёнными в неё недекларированными возможностями.

Данная программа предназначена для выполнения обучаемыми курсового проекта по дисциплине «Разработка и эксплуатация защищённых автоматизированных систем».

Для достижения данной цели необходимо выполнить следующие задачи:

— обзор методов поиска недекларированных возможностей в программных средствах;

— разработка программного средства для изучения методов поиска недекларированных возможностей при функциональном тестировании программного обеспечения;

— разработка руководства по поиску НДВ в разработанном программном средстве.

В первом разделе выпускной квалификационной работы описаны основные разновидности недекларированных возможностей. В зависимости от целей НДВ были разделены на несколько категорий.

Во втором разделе выпускной квалификационной работы, согласно Постановлению Правительства РФ №1119 [12] для информационных систем,

описаны типы угроз ИБ, возникающие в случае реализации НДВ. Также описаны виды угроз информационной безопасности, которые могут быть реализованы с помощью НДВ.

В третьем разделе выпускной квалификационной работы сформулированы основные методы поиска НДВ при функциональном тестировании программных средств и при анализе исходного кода программных средств. Также описаны цели, сущность и задачи методов поиска НДВ.

В четвертом разделе выпускной квалификационной работы описан алгоритм разработанной программы шифрования изображений, функции, разработанные в целях реализации данного алгоритма, блок-схема, по которой производится генерация гаммы-шифра, а также заголовки форматов файлов, с которыми работает программа.

В пятом разделе выпускной квалификационной работы описана программа шифрования изображений, разработанная по выше указанному алгоритму на языке программирования Си++. Описана реализация функций и переменные, используемые в программе. Также в разделе описаны НДВ, реализованные в программе с целью последующего описания методов поиска НДВ в ней.

В шестом разделе выпускной квалификационной работы представлено руководство пользователя программы. Описаны назначение программы, её условия выполнения, а также процесс выполнения программы, включая перечень сообщений, выводимых оператору в процессе выполнения программы.

В седьмом разделе выпускной квалификационной работы представлено руководство по поиску НДВ в разработанной программе шифрования. Описаны методы динамического анализа, используемые в процессе поиска реализованных НДВ в программном средстве. Промоделированы действия обучаемых при выполнении курсового проекта по дисциплине «Разработка и эксплуатация защищённых автоматизированных систем».

В восьмом разделе выпускной квалификационной работы описаны вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Определены методы и средства физической культуры, снижающие их воздействие, как рекомендации к применению в организации мер по регулированию психофизиологического состояния специалиста.

1 Анализ основных разновидностей недекларированных возможностей

НДВ могут быть как случайными ошибками программистов, так и следствием преднамеренно видоизменённой части ПО, благодаря которой можно получить скрытый несанкционированный доступ в защищаемую компьютерную систему, исходя из [13].

Таким образом, если добавить в программу НДВ, она сможет как напрямую выполнять необходимую злоумышленнику функциональность, так и умышленно создавать «дыры» в безопасности, являться своего рода бэкдором [28].

В зависимости от целей НДВ можно разделить на несколько категорий [18]:

— получение полного доступа к удаленному компьютеру или системе. По сути такие программы аналогичны хакерским руткитам и бэкдорам с той лишь разницей, что их функциональность заложена в одну из легальных программ пользователя. Могут применяться для шпионажа либо для нарушения работы компьютеров, телекоммуникационного оборудования и смартфонов коммерческих и общественных организаций, широких групп граждан;

— кража паролей доступа, то есть функции кейлоггера. Наличие пароля удаленного доступа к компьютеру обеспечивает столь же обширные возможности, как самый лучший бэкдор, а доступ к аккаунтам электронной почты и Skype позволит отслеживать переговоры и переписку даже в тех случаях, когда человек использует для общения другие компьютеры, на которых программ-закладок нет. Особый интерес кража паролей представляет в случаях, когда нужен доступ ко всей внутренней сети, где работает компьютер с закладкой;

— несанкционированное изменение данных и вывод из строя компьютерных систем. Наибольшую угрозу такие программы представляют в

системах АСУ ТП, в особенности критически важных объектов, управления техникой военного или двойного назначения. Установленные программные закладки позволяют, в случае необходимости, вывести из строя инфраструктурные и военные объекты потенциального противника.

2 Анализ угроз ИБ, возникающих от недекларированных возможностей

Под угрозами информационной безопасности защищаемых данных понимается совокупность условий и факторов, создающих опасность несанкционированного, в том числе случайного, доступа к защищаемым данным при их обработке в информационной системе, результатом которого могут стать уничтожение, изменение, блокирование, копирование, предоставление, распространение защищаемых данных, а также иные неправомерные действия.

Согласно Постановлению Правительства РФ №1119 [12] для информационных систем различают следующие типы угроз:

— угрозы 1-го типа актуальны для информационной системы, если для неё в том числе актуальны угрозы, связанные с наличием недокументированных (недекларированных) возможностей в системном программном обеспечении, используемом в информационной системе;

— угрозы 2-го типа актуальны для информационной системы, если для неё в том числе актуальны угрозы, связанные с наличием недокументированных (недекларированных) возможностей в прикладном программном обеспечении, используемом в информационной системе;

— угрозы 3-го типа актуальны для информационной системы, если для нее актуальны угрозы, не связанные с наличием недокументированных (недекларированных) возможностей в системном и прикладном программном обеспечении, используемом в информационной системе.

Следовательно, при наличии НДВ в информационной системе, для неё могут быть актуальны угрозы 1-го и 2-го типа, что приводит к необходимости обеспечения, в зависимости от деятельности организации, 1, 2 или 3 уровня защищённости данных при их обработке в информационной системе.

Угрозы информационной безопасности, которые могут быть реализованы с помощью НДВ, чрезвычайно обширны.

Это вызвано спектром возможностей, предоставляемых злоумышленнику с помощью данной уязвимости ПО.

Виды угроз могут составлять:

— угроза неавторизованного доступа;

— угроза отказа в обслуживании;

— угроза внедрения вредоносного кода;

— угроза сбора информации;

— угроза несоответствующего использования.

Чтобы вовремя и наиболее эффективно среагировать на выявленные инциденты информационной безопасности и предотвратить угрозы информационной безопасности, используются различные методы поиска НДВ программных средств.

3 Анализ основных методов поиска недекларированных возможностей: при функциональном тестировании программных средств; при анализе исходного кода программных средств

Защищённость программных средств напрямую зависит от надежности и порядочности разработчиков программы, а также мер по анализу исходного кода и поиску НДВ. Защититься от программ-закладок труднее, чем от иного вредоносного ПО, антивирус их видит, а аномалий в работе может не быть. Поэтому проверка на посторонний код и недокументированные функции обязательна.

На сегодняшний день применяются два основных метода поиска уязвимостей кода: структурный статический и динамический анализ исходного кода, регламентируемые руководящим документом.

Надежность методов статического анализа определяется вероятностью ошибок первого и второго рода. Под ошибкой первого рода понимается принятие за НДВ безопасной программы, а под ошибкой второго рода — объявление программы безопасной, когда на самом деле она содержит НДВ.

Сущность статического анализа исходного текста ПО заключается в проверке семантического и синтаксического соответствия текста ПО определенным правилам [14-17].

Основной задачей статического анализа ПО, в общем случае, является восстановление алгоритма по исходному тексту программ. В дальнейшем восстановленный алгоритм используется для верификации программы и заявленным в программной документации ее спецификациям (декларируемым возможностям).

Частными задачами статического анализа ПО являются:

— проверка модульной структуры ПО, а также логической структуры отдельных модулей и сравнения этих структур с приведенными в программной документации;

— подготовка исходных данных для проведения динамического анализа ПО и разработки плана тестирования ПО;

— оценка конструктивных характеристик программы, степени простоты модификации и сопровождения программы;

— определение наличия несовершенств в программе, неиспользуемых участков программы, лишних переменных;

— оценка текстовой сложности программы, затрат на ее разработку и освоение;

— экспертиза идентичности программ для установления авторства и решения правовых споров;

— определение количественных характеристик при оценке уровня качества программы.

Простейшие формы статического анализа осуществляются компиляторами. Обычно набор обнаруживаемых ими ошибок ограничен нарушениями синтаксиса и правил совместимости типов. На другом конце спектра систем статического анализа находятся средства автоматического доказательства правильности программ. Эти средства требуют детальной формальной спецификации свойств программы, которые надо доказать, поэтому являются требовательными к ресурсам.

Проведение статического анализа «вручную» подготовленным экспертом заключается в детальном просмотре исходного текста программ.

Учитывая большие объемы и сложность построения алгоритмических конструкций современных программных средств, применение данного метода выполнения статического анализа приводит к большим затратам времени и человеческих ресурсов и требует привлечения эксперта высокой квалификации и как программиста, и как эксперта качества. При этом точность полученных результатов, как правило, не получается удовлетворительной. В ряде случаев человеческие возможности просто не позволяют провести статический анализ «вручную».

Автоматизированные инструментальные средства способны вычислить или хотя бы помочь в определении самых трудоемких характеристик, позволяя, таким образом, эксперту сосредоточить внимание на интерпретации полученных характеристик в уровень качества программы.

В процессе проведения статического анализа используются следующие методы:

— выполнение контрольной сборки ПО. Выявление файлов, не участвующих в процессе сборки ПО;

— анализ соответствия результатов контрольной сборки ПО исходным текстам ПО и данными документации;

— выявление процедур (функций), которые в ПО только объявлены, но к которым нет обращений из функциональных объектов;

— выявление отношений по передачам управления между функциональными объектами;

— выявление типов информационных объектов, которые только объявлены, но нигде не используются;

— выявление переменных, которые только объявлены, но нигде не используются;

— выявление отношений по использованию информационных объектов различными функциональными объектами;

— формирование перечня информационных объектов, подлежащих контролю. Анализ всех участков кода, в которых осуществляется обращение к указанным в перечне информационным объектам;

— построение по исходным кодам цепочки вызовов функциональных объектов (маршрутов).

Дополнительно к частным методам статического анализа также относятся анализ указателей, поиск и устранение мертвого кода, минимизация количества переменных.

Дополнительное применение метрик сложности позволяет проводить статический анализ программ при отсутствии семантического значения используемых функций и процедур.

Статический анализ, в свою очередь, предоставляет больше возможностей по рассмотрению всех путей выполнения программы, нежели динамический, а работа на уровне исполняемого кода повышает точность за счёт того, что он является конечным результатом процесса сборки и не подлежит изменениям перед выполнением на ЭВМ конечного пользователя.

Целью динамического анализа является контроль соответствия заявленных функциональных возможностей программного средства реально реализованным в программе.

Анализ осуществляется на особым образом подготовленной версии ПО, в которую встроены специальные программные средства контроля и идентификации (датчики, счетчики и обращения к ним), позволяющие определять маршруты (пути) выполнения заявленных функций ПО с точностью до линейных участков, а также контролировать используемое адресное пространство.

В процессе динамического анализа основное внимание уделяется выявленным при статическом анализе подозрительным фрагментам программ.

Для этого на этапе их выявления (статического анализа) планируются и формируются специальные тесты.

Вместе с тем возможно и принудительное выполнение маршрутов с подозрительными фрагментами путем выделения текста исходной программы, соответствующего данному маршруту.

На этапе динамического анализа с использованием моделей управляющих структур функциональных объектов решаются следующие задачи:

— разбиение множества линейных участков на два подмножества, соответственно задействованных и не задействованных при выполнении заявленных в программной документации команд (процессов);

— уточнение функционального назначения программных компонентов;

— выявление потенциальных НДВ.

Реализация алгоритма служебной подпрограммы-датчика сводится к фиксации факта обращения к ней и записи входных параметров на внешний носитель информации для последующей статистической обработки.

Подготовленная версия исходных текстов комплекса с вставленными подпрограммами-датчиками компилируется обычным порядком, и полученная версия комплекса в дальнейшем используется при проведении динамического тестирования.

В процессе проведения динамического анализа используются следующие методы:

— контроль процесса выполнения ПО;

— выявление расхождений в маршрутах выполнения функциональных объектов, полученных на этапах статического и динамического анализа.

В процессе выполнения программы фиксируется время выполнения отдельных функций и сравнивается с заданным значением в спецификации требований (техническом задании). Для оценки времени восстановления моделируются соответствующие сбойные ситуации. При подготовке тестов следует учитывать, что показатель полноты тестирования проекта коррелируется с показателем работоспособности программы только при условии достаточного уровня качества тестов, адекватных заданным функциональным спецификациям программы. Действительно, обработка программы без ошибок, даже при полноте тестирования, равной 100 %, на тестовых данных, не соответствующих особенностям реальных данных, не доказывает, что на реальных данных программа также будет работоспособна.

С практической точки зрения при обеспечении безопасности информационных систем методы динамического анализа обладают рядом преимуществ, связанных с их привязкой к конкретной системе и программе, а

также с их надежностью в отношении ошибок объявления программы безопасной, когда на самом деле она содержит НДВ. Однако затраты, необходимые для организации процесса тестирования, являются определенной преградой для их применения.

В результате проведения статического и динамического анализа в исследуемом ПО с помощью автоматизированных средств формируется перечень потенциальных НДВ, т.е. фрагментов программного кода (программных конструкций), для которых получены неудовлетворительные результаты соответствия правилам программирования, алгоритмизации и заявленной функциональности программного средства.

По окончании проведения статического и динамического анализа ПО квалифицированным экспертом выполняется сравнение результатов статического и динамического анализа, анализ и обработка полученных данных с целью принятия решения по каждой выявленной потенциально опасной конструкции.

С методической точки зрения методы статического и динамического анализа ПО выглядят более предпочтительными, так как основываются на формальном подходе и приближают перспективное решение проблемы, связанное с доказательством разрешимости множества НДВ. Кроме того, они позволяют создать легко применяемые средства анализа, независящие от анализируемых программ. Однако на данное время любой из этих методов имеет существенный недостаток — исследование безопасности проводится лишь относительно некоторого подмножества НДВ.

4 Разработка алгоритма создаваемой программы

Алгоритм шифрования файла был разработан с помощью функций `main()`, `Pr_K()`, `inv()`, `Pr_ChB()`, алгоритм работы которых представлен в приложении А на рисунках А.1, А.2, А.3, А.4 соответственно.

В функции `main()` реализована выгрузка кода файла, введённого пользователем, в массив и получение размера обрабатываемого файла.

Посредством анализа байтов кода заголовка происходит инициализация необходимых параметров. Описание кода заголовка формата BMP представлено на рисунке 1 [19], формата JPG — в таблице 1.

Имя	Длина	Смещение	Описание
Заголовок файла (BitMapFileHeader)			
Type	2	0	Сигнатура "BM"
Size	4	2	Размер файла
Reserved 1	2	6	Зарезервировано
Reserved 2	2	8	Зарезервировано
OffsetBits	4	10	Смещение изображения от начала файла
Информационный заголовок (BitMapInfoHeader)			
Size	4	14	Длина заголовка
Width	4	18	Ширина изображения, точки
Height	4	22	Высота изображения, точки
Planes	2	26	Число плоскостей
BitCount	2	28	Глубина цвета, бит на точку
Compression	4	30	Тип компрессии (0 - несжатое изображение)
SizeImage	4	34	Размер изображения, байт
XpelsPerMeter	4	38	Горизонтальное разрешение, точки на метр
YpelsPerMeter	4	42	Вертикальное разрешение, точки на метр
ColorsUsed	4	46	Число используемых цветов (0 - максимально возможное для данной глубины цвета)
ColorsImportant	4	50	Число основных цветов
Таблица цветов (палитра) (ColorTable)			
ColorTable	1024	54	256 элементов по 4 байта
Данные изображения (BitMap Array)			
Image	Size	1078	Изображение, записанное по строкам слева направо и снизу вверх

Рисунок 1 - Описание заголовка файла формата BMP

Таблица 1 — Основные маркеры заголовка файла формата JPG

Маркер	Байты	Длина	Назначение	Комментарии
SOI	0xFFD8	Нет	Начало изображения	-
SOF0	0xFFC0	Переменный размер	Начало фрейма (базовый, ДКП)	Показывает, что изображение кодировалось в базовом режиме с использованием ДКП и кода Хаффмана. Маркер содержит число строк и длину строки изображения (двухбайтовые поля со смещением соответственно 5 и 7 относительно начала маркера), количество компонентов (байтовое поле со смещением 9 относительно начала маркера), число бит на компонент — строго 8 (байтовое поле со смещением 4 относительно начала маркера), а также соотношение компонентов (например, 4:2:0).
SOF1	0xFFC1	Переменный размер	Начало фрейма (расширенный, ДКП, код Хаффмана)	Показывает, что изображение кодировалось в расширенном (extended) режиме с использованием ДКП и кода Хаффмана. Маркер содержит число строк и длину строки изображения, количество компонентов, число бит на компонент (8 или 12), а также соотношение компонентов (например, 4:2:0).
SOF2	0xFFC2	Переменный размер	Начало фрейма (прогрессивный, ДКП, код Хаффмана)	Показывает, что изображение кодировалось в прогрессивном режиме с использованием ДКП и кода Хаффмана. Маркер содержит число строк и длину строки изображения, количество компонентов, число бит на компонент (8 или 12), а также соотношение компонентов (например, 4:2:0).
DHT	0xFFC4	Переменный размер	Содержит таблицы Хаффмана	Задает одну или более таблиц Хаффмана.
DQT	0xFFDB	Переменный размер	Содержит таблицы квантования	Задает одну или более таблиц квантования.
DRI	0xFFDD	4 байта	Указывает длину рестарт-интервала	Задает интервал между маркерами RST n в макроблоках. При отсутствии DRI появление в потоке кодированных данных маркеров RSTn недопустимо и считается ошибкой. Если при кодировании маркеры RST n не применяются, маркер DRI либо не используется вовсе, либо интервал повторений в нём указывается равным 0.

Продолжение таблицы 1

Маркер	Байты	Длина	Назначение	Комментарии
SOS	0xFFDA	Переменный размер	Начало сканирования	Начало первого или очередного скана изображения с направлением обхода слева направо сверху вниз. Если использовался базовый режим кодирования, используется один скан. При использовании прогрессивных режимов используется несколько сканов. Маркер SOS является разделяющим между информативной (заголовком) и закодированной (собственно сжатыми данными) частями изображения.
RSTn	0xFFDn	Нет	Перезапуск	Маркеры перезапуска используются для сегментирования кодированных энтропийным кодером данных. В каждом сегменте данные декодируются независимо, что позволяет распараллелить процедуру декодирования. При повреждении кодированных данных в процессе передачи или хранения JPEG-файла использование маркеров перезапуска позволяет ограничить потери (макроблоки из неповреждённых сегментов будут восстановлены правильно). Вставляется в каждом r-м макроблоке, где r — интервал перезапуска DRI маркера. Не используется при отсутствии DRI маркера. n, младшие 3 бита маркера кода, циклы от 0 до 7.
APPn	0xFFEn	Переменный размер	Задаётся приложением	Например, в EXIF JPEG-файла используется маркер APP1 для хранения метаданных, расположенных в структуре, основанной на TIFF.
COM	0xFFFE	Переменный размер	Комментарий	Содержит текст комментария.
EOI	0xFFD9	Нет	Конец закодированной части изображения	-

Для начала определяется переменная `Pr1` значением первых двух байт кода файла. Если `Pr1` соответствует коду «`424D`», делается вывод о том, что введённый файл является форматом BMP [20]. Иначе, если переменная принимает значение «`FFD8`», — форматом JPG [21].

Далее определяется байт начала кода пиксельных данных. Для формата BMP определяется значение 4 байт, записанных в заголовке файла со смещением в 10 байт. Этот участок кода файла называется `OffsetBits` (смещение изображения от начала файла).

Для файла формата JPG определяется положение маркера `SOS`, который является разделяющим между информативной (заголовком) и закодированной (собственно сжатыми данными) частями изображения. При нахождении данного маркера среди кода файла определяется его размер из следующих двух байт кода заголовка и следующий байт после конца кода маркера `SOS` определяется началом кода пиксельных данных. Результат записывается в переменную `st`.

В участке кода, реализующем шифрование изображений формата BMP, происходит проверка условия на соответствие третьему требованию к реализациям НДВ в программе шифрования изображений. Для этого вызывается функция `Pr_ChB()`, возвращающая тип обрабатываемого изображения.

Затем проверяется условие второго требования к реализациям НДВ в программе шифрования изображений. Если размер файла меньше 20 Кбайт, вызывается функция `inv()`. Иначе происходит генерация гаммы и шифрование кода файла в функции `sh()`.

В конце кода `main()`, согласно всем инициализированным параметрам, происходит запись модифицированного кода в файл, введённого пользователем для сохранения результата.

В функции `Pr_K()` происходит проверка вводимого ключа шифрования. Если введённые значения файла не соответствуют необходимому размеру ключа и содержат символы, не равные «0» и «1», программа выдаёт сообщение о несоответствии ключа шифрования необходимым условиям и заканчивает выполнение кода программы.

В функции `inv()` описан код, выполняющийся, если изображение имеет объём, меньший 20 Кб. В этом случае, исходя из требований к техническому заданию, происходит инверсия бит.

Функциия `Pr_ChB()` определяет, является ли изображение чёрно-белым. Для этого в заголовке BMP файла находятся два байта, называемых в заголовке `BitCount`, со смещением, равным 28 байтам. В них хранится количество бит, использованных при кодировании пикселей изображений.

Если в двух байтах записано значение, соответствующее «1», изображение признаётся монохромным и попадает под условие требования к НДВ. Если при кодировании битов изображения использовалось большее количество разрядов, происходил анализ байт на соответствие значений кода пикселей «FF» (черный цвет) и «00» (белый цвет). Если все значения кодов пикселей подходили под это условие, изображение обрабатывалось как черно-белое.

Алгоритм выработки гаммы шифра описан функцией `sh()`, представленной на рисунке Б.1 в приложении Б. Функция генерирует гамму шифра по описанным в следующем разделе условиям и производит шифрование кода файла методом гаммирования.

Реализация шифрования методом гаммирования происходила согласно схеме выработки гаммы шифра, представленной на рисунке 2.

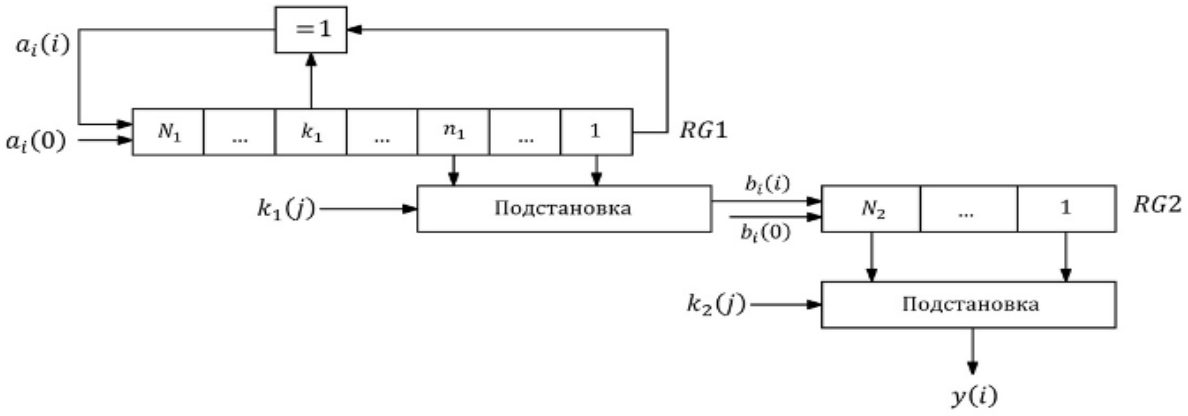

Рисунок 2 — Схема выработки гаммы шифра

На данной схеме изображены два регистра сдвига, один из которых (RG1) является регистром сдвига с обратной связью со следующими параметрами:

— $a_i(i)$ - результат сложения по модулю 2 элемента k_1 с последним элементом для осуществления процедуры сдвига регистра сдвига RG1;

— $a_i(0)$ - начальное заполнение регистра сдвига RG1;

— $k_1(j)$ - ключ первой подстановки;

— $k_2(j)$ - ключ второй подстановки;

— $b_i(i)$ - результат первой подстановки, $y_i(i)$ - результат первой подстановки.

При генерации гаммы шифра использовалось следующее уравнение:

$$y = x^{151} + x^{67} + 1, n_1 = 32, N_2 = 48.$$

Здесь:

— N_1 равно 151 - размер регистра сдвига RG1;

— N_2 равно 48 - размер регистра сдвига RG2;

— n_1 равно 32 - число элементов первой подстановки;

— k_1 равно 67 - номер элемента, складывающегося по модулю два с последним элементом для осуществления процедуры сдвига регистра сдвига RG1.

Операция подстановки была реализована с помощью трёх таблиц, хранящихся во внешних файлах.

Ключ подстановки и таблицы представлены соответственно в key.txt (см. рисунок 3), tab.txt (см. рисунок 4), tab16.txt (см. рисунок 5), tab32.txt (см. рисунок 6).

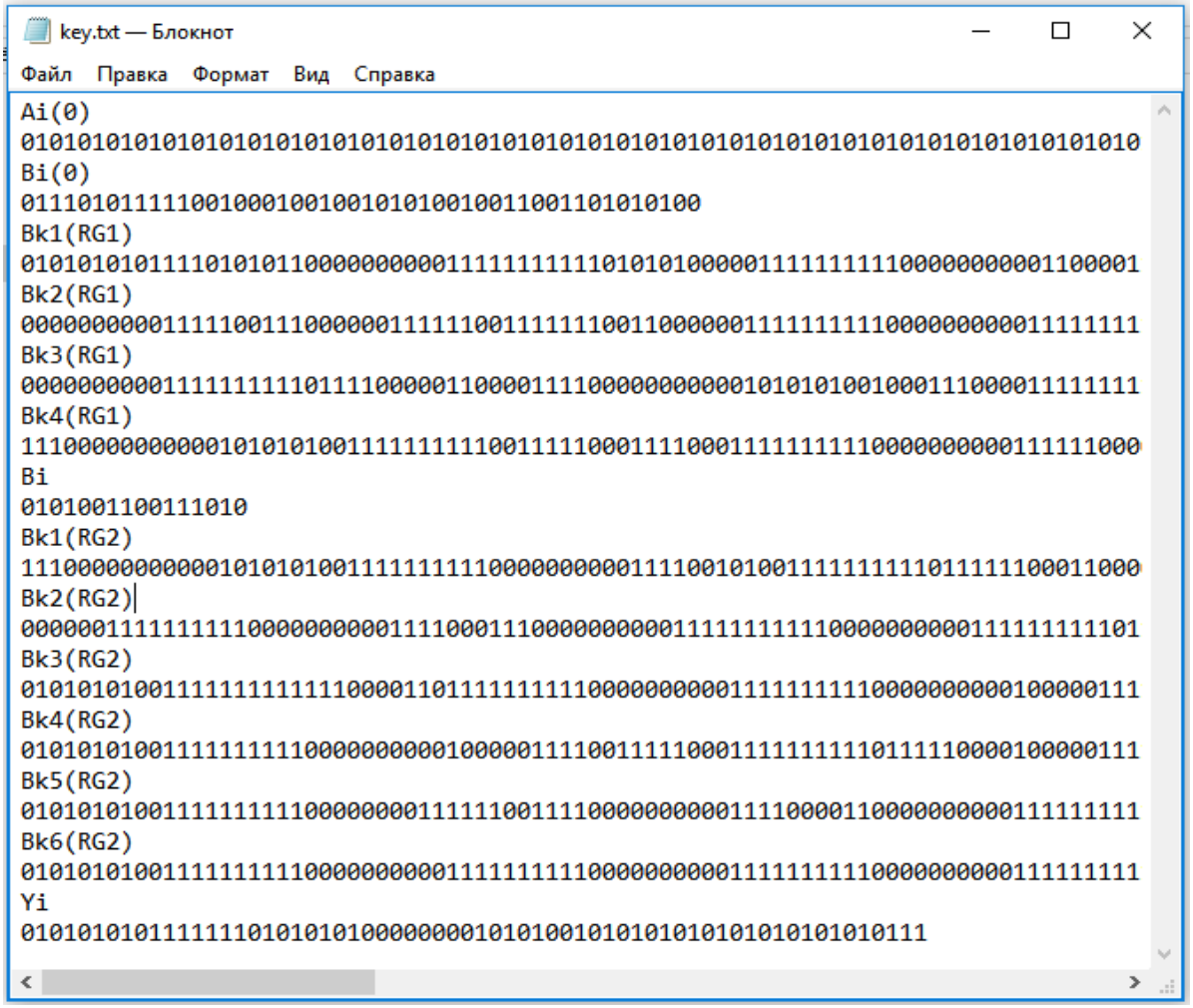

Рисунок 3 — Демонстрация содержимого внешнего файла `key.txt`

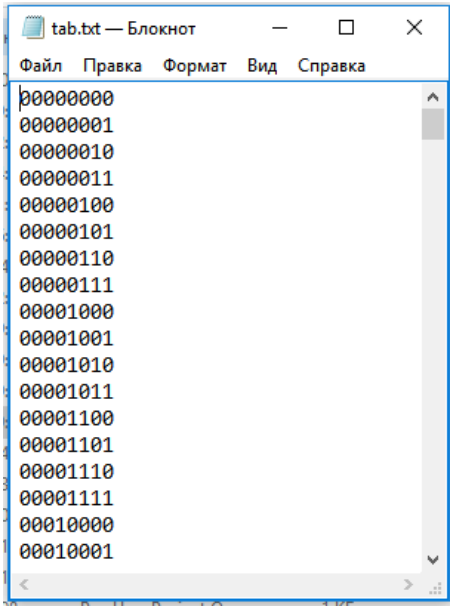

Рисунок 4 — Демонстрация содержимого внешнего файла `tab.txt`

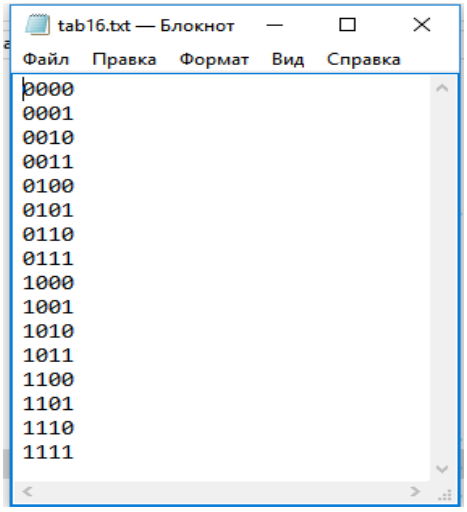

Рисунок 5 — Демонстрация содержимого внешнего файла `tab16.txt`

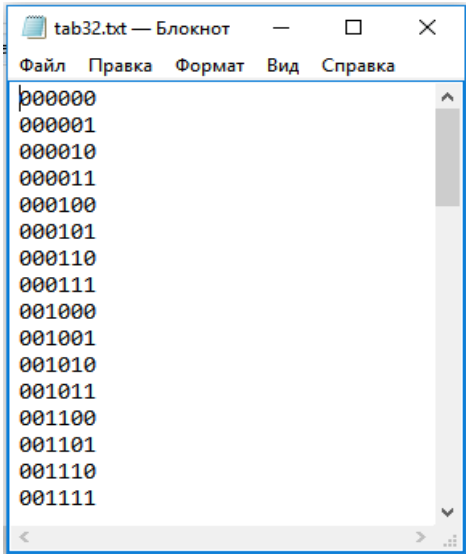

Рисунок 6 — Демонстрация содержимого внешнего файла `tab32.txt`

В результате выполнения этапа были разработаны алгоритмы работы программы шифрования изображений.

5 Описание разработки программы шифрования с введёнными в неё недекларированными возможностями

В целях реализации алгоритма шифрования изображений на языке Си++ был написан код программы шифрования изображений, представленный в приложении В.

Разработанная программа шифрования изображений реализована в виде консольного приложения для ОС Windows [22].

Согласно разработанному алгоритму программа шифрования состоит из пяти функций, реализующих одноимённые части алгоритма программы шифрования изображений.

Функция `main()` выполняет выгрузку кода файла, введённого пользователем, в массив и получение размера обрабатываемого файла. Посредством анализа байтов кода заголовка происходит: определение, какого формата введён файл, границы конца кода заголовка, является ли изображение черно-белым, меньше ли изображение 20 Кбайт. В конце происходит запись модифицированного кода в код файла, введённого пользователем для сохранения результата. Выходным значением функции является код «0». Если программа была выполнена без ошибок, выводится сообщение об удачном шифровании файла, в ином случае - сообщение об ошибке, согласно разработанному алгоритму.

В функции `main()` используются следующие переменные:

— `FILE* fp` для открытия файла, подлежащего шифрованию;

— `FILE* fp1` для открытия файла с целью сохранения результата;

— `int i` для организации циклов;

— `int j` для организации цикла;

— `int st` для хранения числа смещения пиксельных данных относительно кода файла;

— `long lSize` для хранения размера файла в байтах;

— `char p[20]` для хранения названия файла, подлежащего шифрованию;

— `char d[20]` для хранения названия файла с целью сохранения результата;

— `int q` для подсчёта числа смещения маркера SOS относительно кода файла;

— `int m` для хранения размера массива, передаваемого в файл с целью сохранения результата;

— `char* bal[]` для хранения кода файла, подлежащего шифрованию;

— `string Pr1` для хранения первых двух байт файла с целью определения формата файла, подлежащего шифрованию;

— `int ChB` для хранения результата определения типа изображения (чёрно-белое (монохромное) или цветное);

— `int bal2[4]` для подсчёта числа смещения пиксельных данных относительно кода файла;

— `int* bal1[]` для хранения пиксельного кода файла, подлежащего шифрованию.

Функция `Pr_K()` осуществляет проверку вводимого ключа шифрования, согласно разработанному алгоритму. Выходными значениями функции являются: «0», если вводимый ключ шифрования соответствует предъявляемым требованиям, «4», если в ключе шифрования были найдены ошибки.

В функции `Pr_K()` используются следующие переменные:

— `string p` для хранения пути файла, содержащего ключ шифрования;

— `string s` для ввода строк из файла, содержащего ключ шифрования;

— `string u` для ввода строк из файла, содержащего ключ шифрования;

— `int r` для вычисления длины строк, вводимых из файла, содержащего ключ шифрования;

— `int i` для организации циклов;

— `ifstream fin(p)` для открытия файла, содержащего ключ шифрования.

Функция `inv()` инвертирует биты пиксельного кода файла, согласно разработанному алгоритму. Входными переменными являются:

— `char* bal` — массив для записи зашифрованного кода файла;

— `int* bal1`, в которой передаётся массив пиксельного кода;

— `long lSize`, в которой передаётся размер файла в байтах;

— `int st`, в которой передаётся смещение пиксельного кода относительно начала;

— `int f`, в которой определяется формат обрабатываемого файла.

Выходными значениями функции являются: «0», если вводимый ключ шифрования соответствует предъявляемым требованиям, «5», если в ключе шифрования были найдены ошибки.

В функции `inv()` используются следующие переменные:

— `int k` для организации цикла;

— `char Per5` для осуществления операции инверсии;

— `char Per6` для осуществления операции инверсии.

Функция `Pr_ChB()` определяет, является ли изображение чёрно-белым, согласно разработанному алгоритму. Входными переменными являются переменные, описанные у функции `inv()`. Выходными значениями функции являются: «9», если вводимое пользователем изображение является черно-белым, «5», если изображение цветное.

В функции `Pr_ChB()` используются следующие переменные:

— `string p` для хранения пути файла, содержащего ключ шифрования;

— `int i` для организации циклов;

— `int bal4[2]` для хранения двух байт, определяющих количество бит на пиксель, использованных при кодировании файла;

— `int ch` для хранения двух байт, определяющих количество бит на пиксель, использованных при кодировании файла;

— `int chh` для сравнения байта файла с возможными значениями черно-белого изображения.

Функция `sh()`, генерирует гамму шифра согласно разработанному алгоритму. Входными переменными являются переменные, описанные у функции `inv()`, а также переменная `int ChB`, указывающая, цветным или черно-белым является изображение. Выходными значениями функции являются: «0», если вводимый ключ шифрования соответствует предъявляемым требованиям, «5», если в ключе шифрования были найдены ошибки.

В функции `sh()` используются следующие переменные:

— `int i` для организации циклов;

— `int j` для организации цикла периодического шифрования пиксельного кода файла;

— `int k` для организации цикла генерации гаммы шифра;

— `string p` для хранения пути файла, содержащего ключ шифрования;

— `string t` для хранения пути файла, содержащего таблицу первой подстановки;

— `string sh` для хранения пути файла, содержащего таблицу второй подстановки (генерация Bi);

— `string yh` для хранения пути файла, содержащего таблицу второй подстановки (генерация Yi);

— `char RG1[151]` для хранения содержимого регистра сдвига RG1;

— `char RG2[48]` для хранения содержимого регистра сдвига RG2;

— `char Per1RG1` для реализации процедуры сдвига регистра RG1;

— `char Per2RG1` для реализации процедуры сдвига регистра RG1;

— `char Pod1[8]` для реализации подстановки для RG1;

— `char Pod2[8]` для реализации подстановки для RG1;

— `char Pod3[8]` для реализации подстановки для RG1;

— `char Pod4[8]` для реализации подстановки для RG1;

— `char Bk_RG1[256]` для хранения ключа подстановки для реализации подстановки для RG1;

— `char b[4]` для хранения промежуточного результата перестановки для RG1;

— `char y[6]` для хранения промежуточного результата перестановки для RG2;

— `char Yi[8]` для хранения промежуточного результата перестановки для RG2;

— `char bi` для хранения результата перестановки для RG1;

— `char Pod11[8]` для реализации подстановки для RG2;

— `char Pod22[8]` для реализации подстановки для RG2;

— `char Pod33[8]` для реализации подстановки для RG2;

— `char Pod44[8]` для реализации подстановки для RG2;

— `char Pod55[8]` для реализации подстановки для RG2;

— `char Pod66[8]` для реализации подстановки для RG2;

— `ifstream fin(p)` открытие файла, содержащего ключ шифрования, для начального заполнения регистров сдвига;

— `string s` для хранения строк, выводимых из файла, содержащего ключ шифрования;

— `string POD1` для осуществления поиска в файле с таблицей

первой подстановки для реализации подстановки для RG1;

— `string POD2` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG1;

— `string POD3` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG1;

— `string POD4` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG1;

— `ifstream ffin1(t)` для открытия файла с таблицей первой подстановки для определения 1 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o1` счётчик для определения 1 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream ffin2(t)` для открытия файла с таблицей первой подстановки для определения 2 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o2` счётчик для определения 2 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream ffin3(t)` для открытия файла с таблицей первой подстановки для определения 3 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o3` счётчик для определения 3 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream ffin4(t)` для открытия файла с таблицей первой подстановки для определения 4 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o4` счётчик для определения 4 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream fin1(p)` для открытия файла, содержащего ключ

шифрования, для вывода в массив 1 значения ключа промежуточной подстановки для RG1;

— `ifstream fin2(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 2 значения ключа промежуточной подстановки для RG1;

— `ifstream fin3(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 3 значения ключа промежуточной подстановки для RG1;

— `ifstream fin4(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 4 значения ключа промежуточной подстановки для RG1;

— `string B` для сохранения результата промежуточной подстановки для RG1 в строку для осуществления поиска в файле с таблицей второй подстановки (генерация Bi);

— `ifstream fffin(sh)` для открытия файла с таблицей второй подстановки (генерация Bi);

— `int o5` счётчик для определения номера строки вхождения результата промежуточной подстановки для RG1 в таблице второй подстановки (генерация Bi);

— `ifstream fin5(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив значений ключа второй подстановки (генерация Bi) для RG1;

— `int Per1` для осуществления процедуры сдвига RG1;

— `int Per2` для осуществления процедуры сдвига RG1;

— `string POD11` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG2;

— `string POD22` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG2;

— `string POD33` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG2;

— `string POD44` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG2;

— `string POD55` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG2;

— `string POD66` для осуществления поиска в файле с таблицей первой подстановки для реализации подстановки для RG2;

— `ifstream gfin1(t)` для открытия файла с таблицей первой подстановки для определения 1 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o11` счётчик для определения 1 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream gfin2(t)` для открытия файла с таблицей первой подстановки для определения 2 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o22` счётчик для определения 2 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream gfin3(t)` для открытия файла с таблицей первой подстановки для определения 3 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o33` счётчик для определения 3 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream gfin4(t)` для открытия файла с таблицей первой подстановки для определения 4 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o44` счётчик для определения 4 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream gfin5(t)` для открытия файла с таблицей первой подстановки для определения 5 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o55` счётчик для определения 5 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream gfin6(t)` для открытия файла с таблицей первой подстановки для определения 6 номера строки вхождения отобранных символов в таблице первой подстановки;

— `int o66` счётчик для определения 6 номера строки вхождения отобранных символов в таблице первой подстановки;

— `ifstream gin1(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 1x значений ключа промежуточной подстановки для RG2;

— `ifstream gin2(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 2x значений ключа промежуточной подстановки для RG2;

— `ifstream gin3(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 3x значений ключа промежуточной подстановки для RG2;

— `ifstream gin4(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 4x значений ключа промежуточной подстановки для RG2;

— `ifstream gin5(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 5x значений ключа промежуточной подстановки для RG2;

— `ifstream gin6(p)` для открытия файла, содержащего ключ шифрования, для вывода в массив 6x значений ключа промежуточной подстановки для RG2;

— `string Y` для сохранения результата промежуточной подстановки для RG2;

— `ifstream gffin(yh)` для открытия файла с таблицей второй подстановки (генерация Yi);

— `int o77` счётчик для определения номера строки вхождения результата промежуточной подстановки для RG2 в таблице второй подстановки (генерация Yi);

— `ifstream gin7(p)` открытие файла, содержащего ключ шифрования, для вывода в массив значений ключа второй подстановки (генерация Bi) для RG1;

— `int Yk` для хранения результата генерации гаммы-шифра;

— `int Per5` для сложения по модулю 2 гаммы-шифра с пиксельным кодом файла;

— `int Per6` для сложения по модулю 2 гаммы-шифра с пиксельным кодом файла.

Работа с файлами осуществлялась с помощью потоков типа `fsteam`. Для чтения использовалась функция `fread()`, для записи – `fwrite()`.

Ввод с клавиатуры значений от пользователя осуществлялся с помощью стандартного потока ввода `cin`, вывод сообщений на экран осуществлялся с помощью стандартного потока вывода `count` (см. рисунок 7).

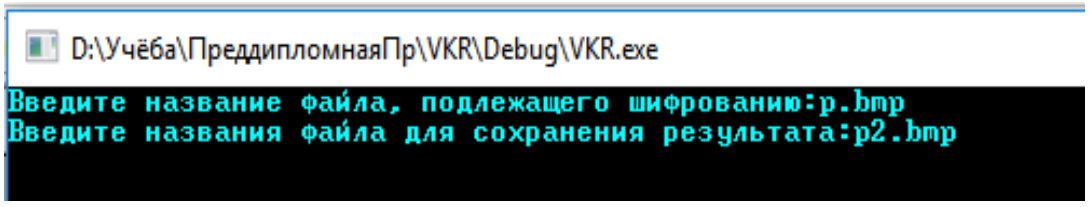

Рисунок 7 — Демонстрация стандартных потоков ввода и вывода

В режиме работы шифра каждый сгенерированный байт гаммы шифра складывается по модулю 2 с байтом пиксельных данных, хранящихся в файле, подающегося на вход.

Ключ шифрования и таблицы подстановки хранятся во внешних файлах.

Программа выдаёт сообщение об ошибке, если не были успешно открыты файлы, переданные в программу шифрования изображений (см. рисунок 8).

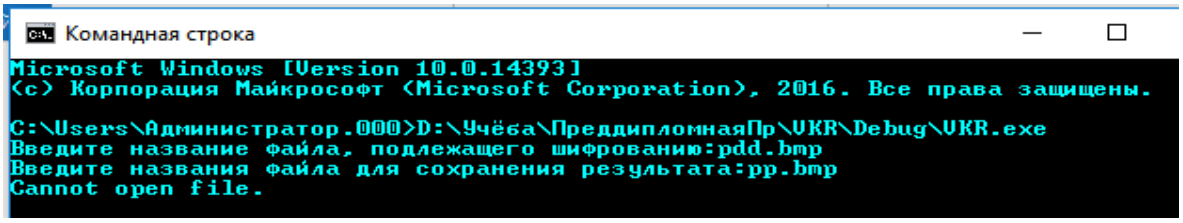

Рисунок 8 — Демонстрация выдачи сообщения об ошибки программы при неудачном открытии введённого файла

Программа осуществляет проверку на правильность ввода ключа шифрования, а именно на длину ключа и на отсутствие в ключе неверных символов.

При несоответствии ключа выдаётся сообщение об ошибке ввода ключа (см. рисунок 9). Пример несоответствия ключа представлен на рисунке 10.

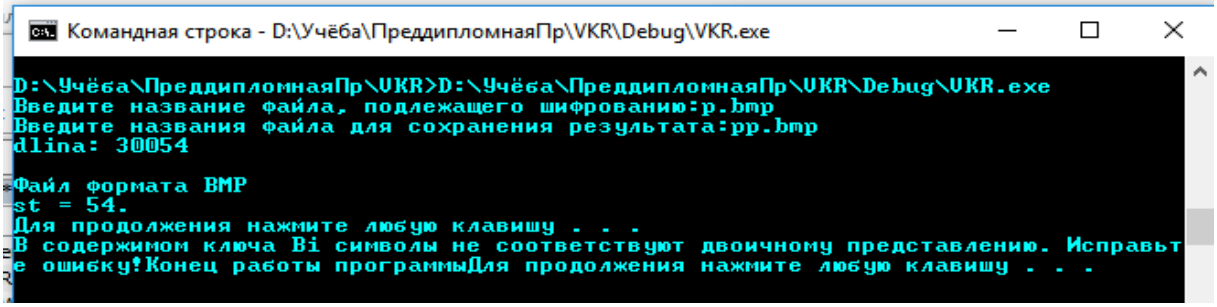

Рисунок 9 — Демонстрация выдачи сообщения об ошибки программы при неверном вводе ключа шифрования

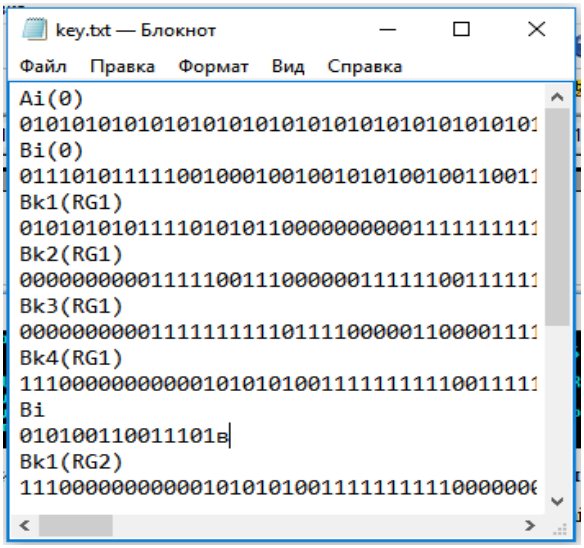

Рисунок 10 — Демонстрация неверного ввода ключа шифрования

В программе были реализованы следующие недекларированные возможности:

– программа вырабатывает периодическую гамму шифра для файлов формата BMP;

– программа производит инверсию бит файла вместо его шифрования, если размер изображения не превышает 20 килобайт;

– программа добавляет ключ к файлу после процедуры шифрования, если изображение черно-белое.

Для файлов формата BMP период гаммы шифра был выбран равным 6000 байт. Демонстрация шифрования файлов формата BMP представлена на рисунке 11. Код файлов представлен на рисунках 12 и 13.

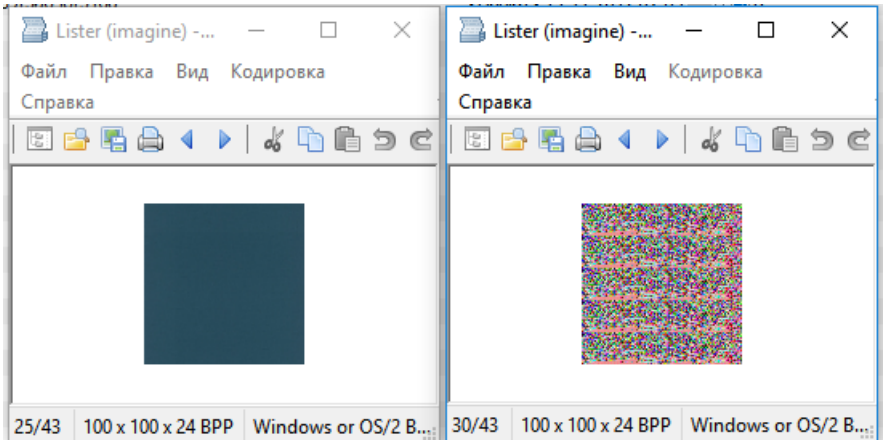

Рисунок 11 — Демонстрация шифрования файлов формата BMP

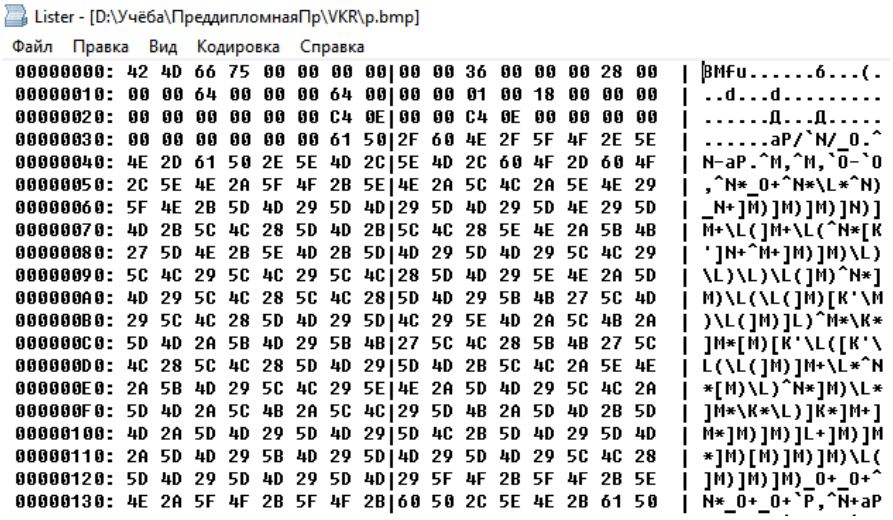

Рисунок 12 — Демонстрация шифрования файлов формата BMP (код изначального файла)

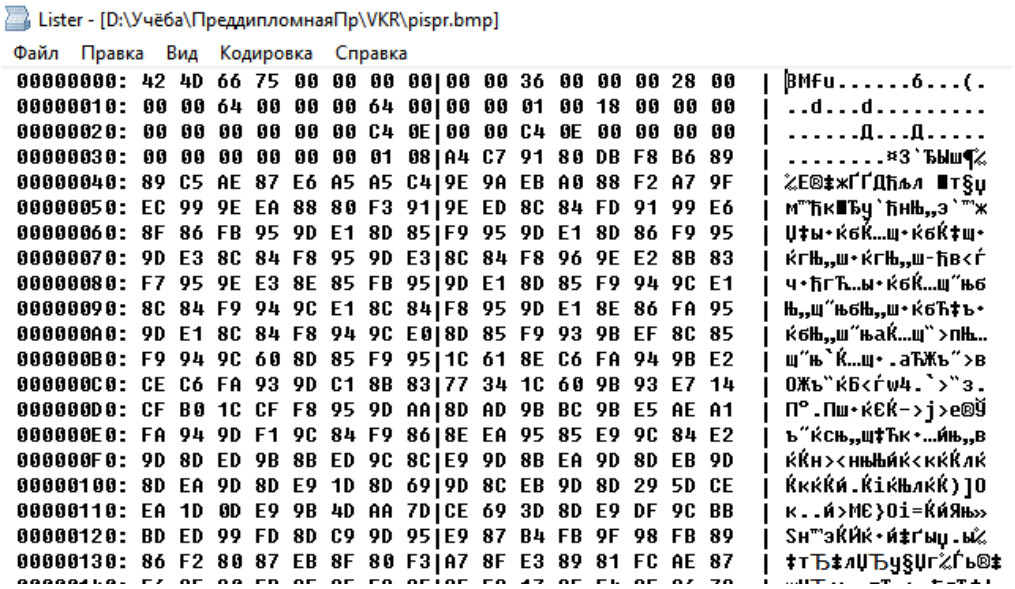

Рисунок 13 — Демонстрация шифрования файлов формата BMP (код зашифрованного файла)

Демонстрация шифрования файлов, размер которых не превышает 20 килобайт, представлена на рисунке 14.

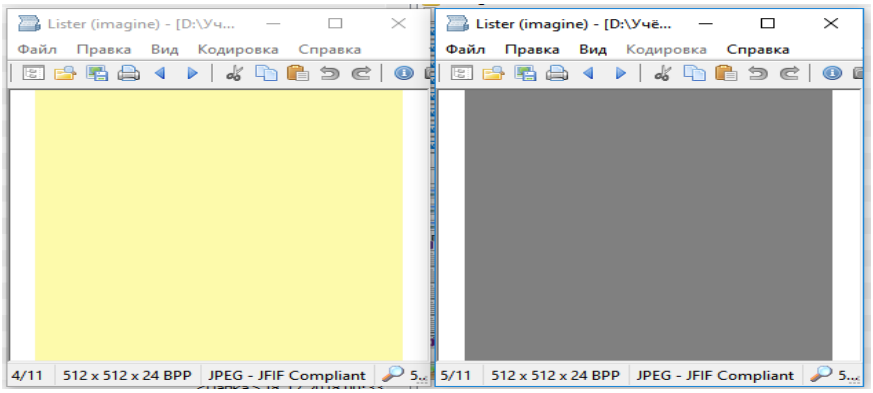

Рисунок 14 — Демонстрация шифрования файлов, размер которых не превышает 20 килобайт

Для реализации функции размер кода файла определялся из значения, определённого в заголовке. Код файлов представлен на рисунках 15 и 16.

Для определения черно-белых изображений в коде файла вычислялось значение использованного количества бит на пиксель при кодировании изображения формата BMP.

Если значение было равным 1, изображение признавалось монохромным и обрабатывалось как черно-белое. Если значение не было равно 1, происходил

анализ байт на соответствие значений кода пикселей «00» (черный цвет) и «FF» (белый цвет). Если все значения кодов пикселей подходили под это условие, изображение обрабатывалось как черно-белое.

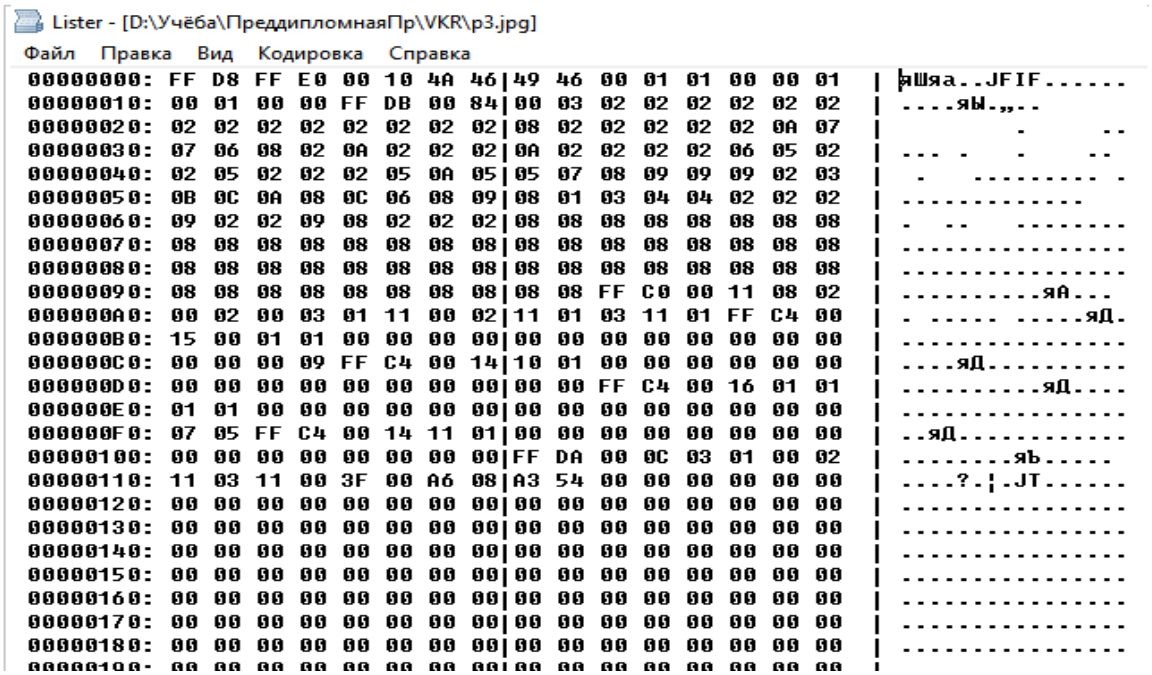

Рисунок 15 — Демонстрация шифрования файлов, размер которых не превышает 20 килобайт (код изначального файла)

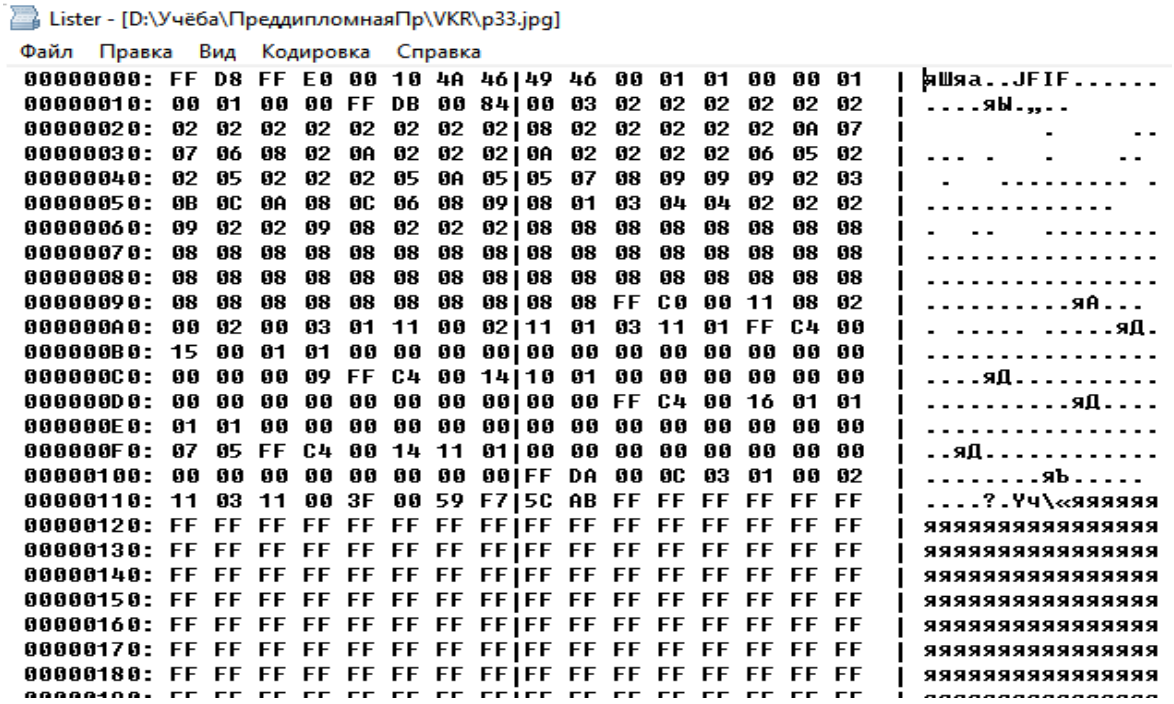

Рисунок 16 — Демонстрация шифрования файлов, размер которых не превышает 20 килобайт (код шифрованного файла)

Пример результата выполнения процедуры шифрования монохромного изображения представлен на рисунке 17. Код файлов представлен на рисунках 18 и 19.

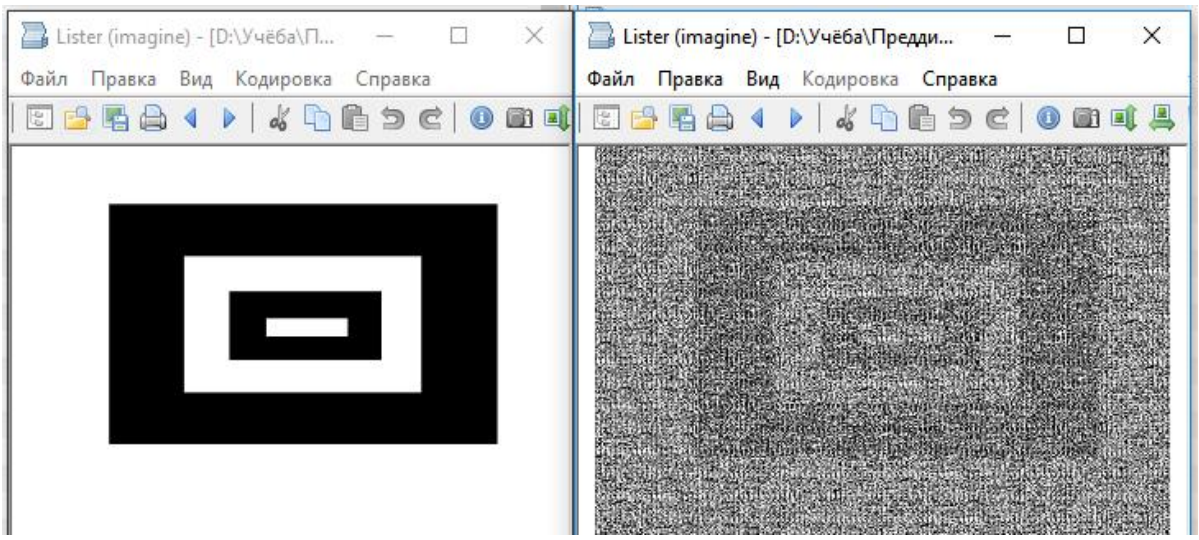

Рисунок 17 — Демонстрация шифрования монохромных изображений

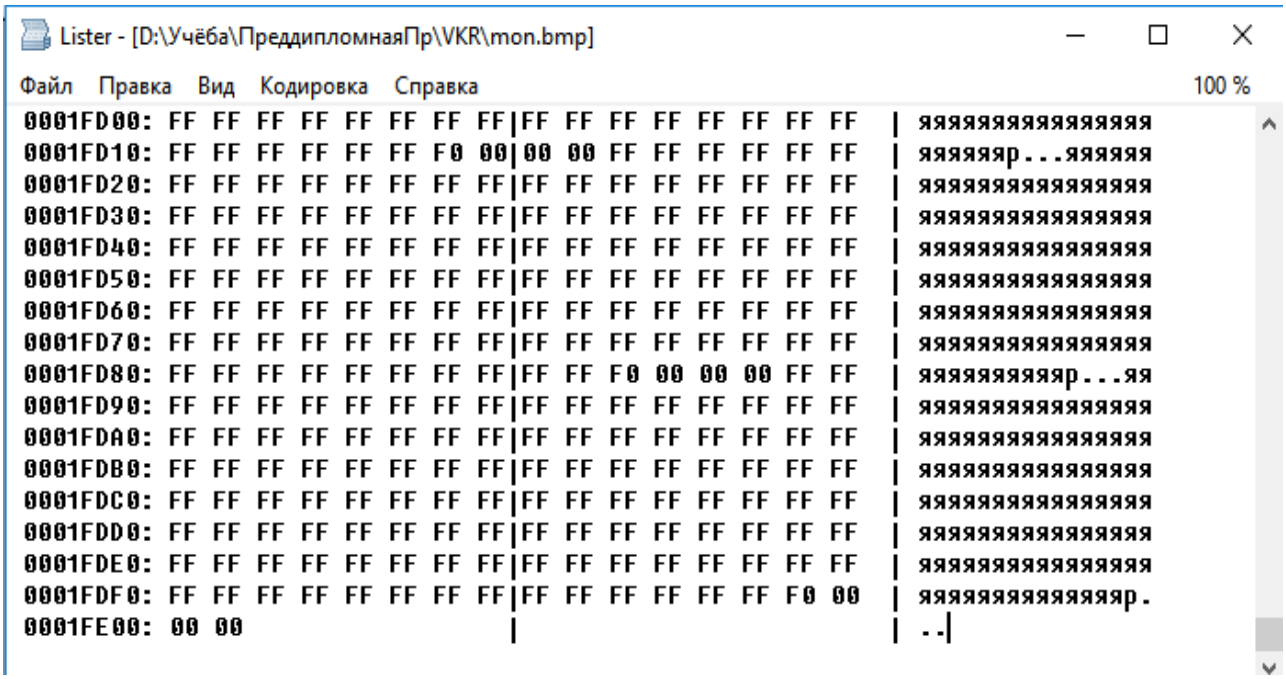

Рисунок 18 — Демонстрация шифрования монохромных изображений (код изначального файла (конец кода))

Результат зашифрования чёрно-белого изображения с количеством бит на пиксель, равным 32, представлен на рисунке 20.

Код файлов представлен на рисунках 21 и 22.

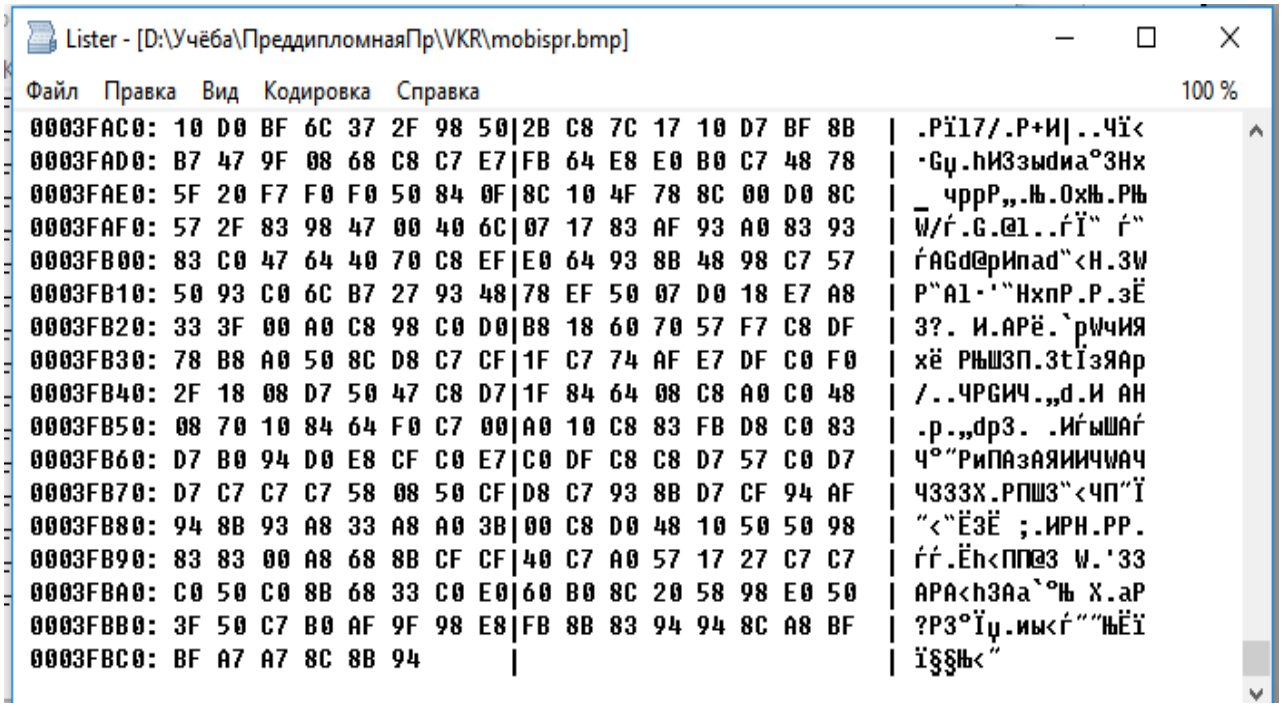

Рисунок 19 — Демонстрация шифрования монохромных изображений (код зашифрованного файла (конец кода))

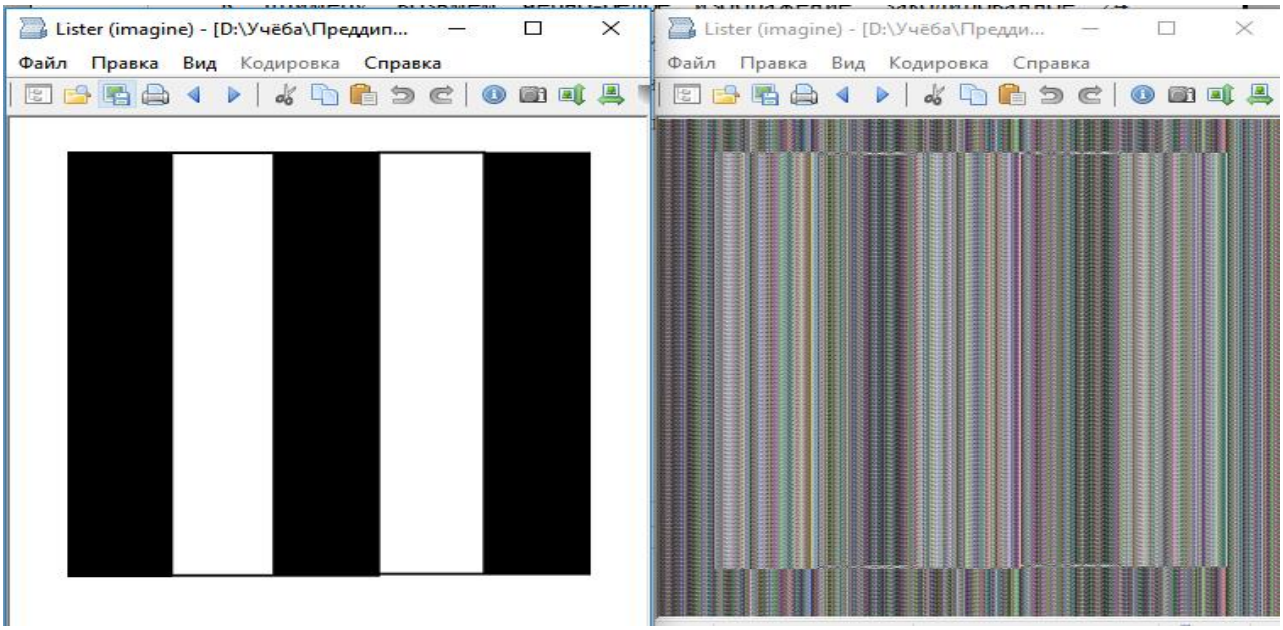

Рисунок 20 — Демонстрация шифрования чёрно-белых изображений

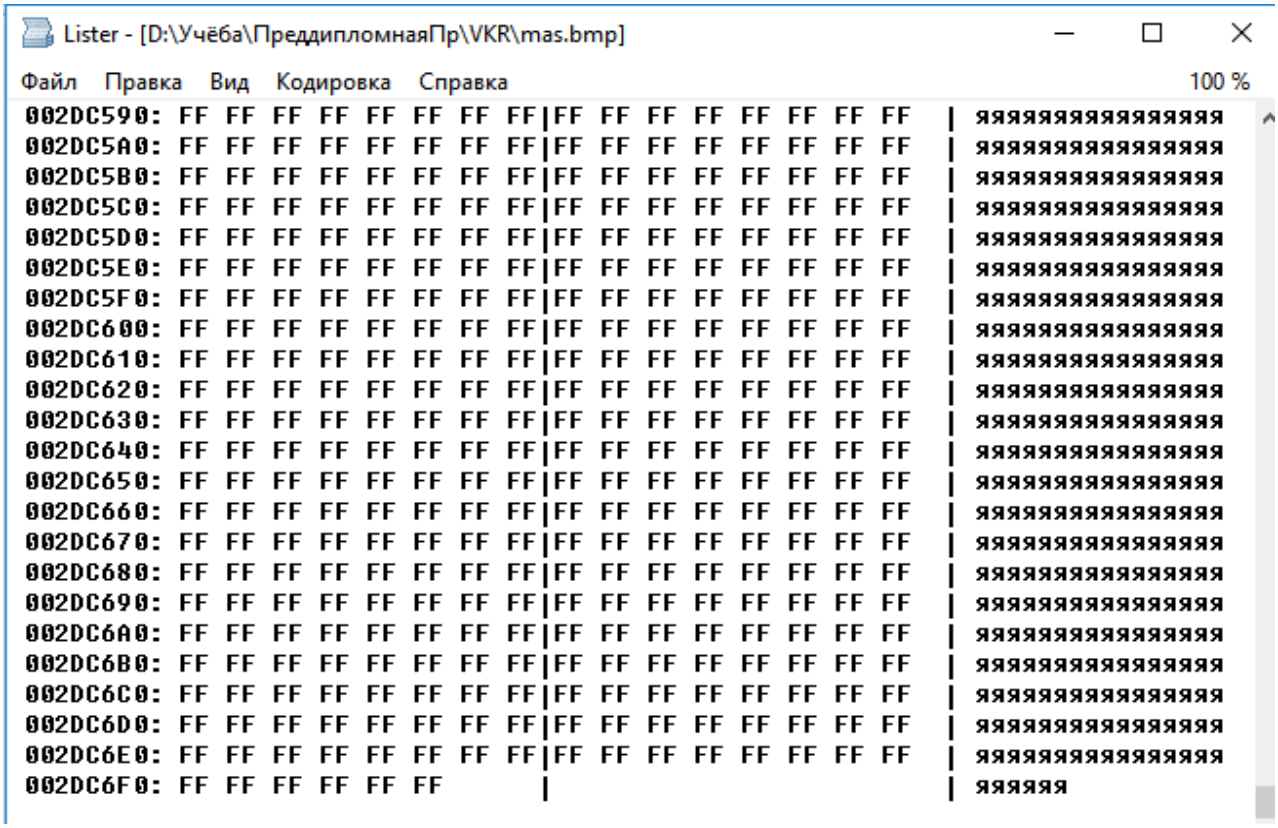

Рисунок 21 — Демонстрация шифрования чёрно-белых изображений (код изначального файла (конец кода))

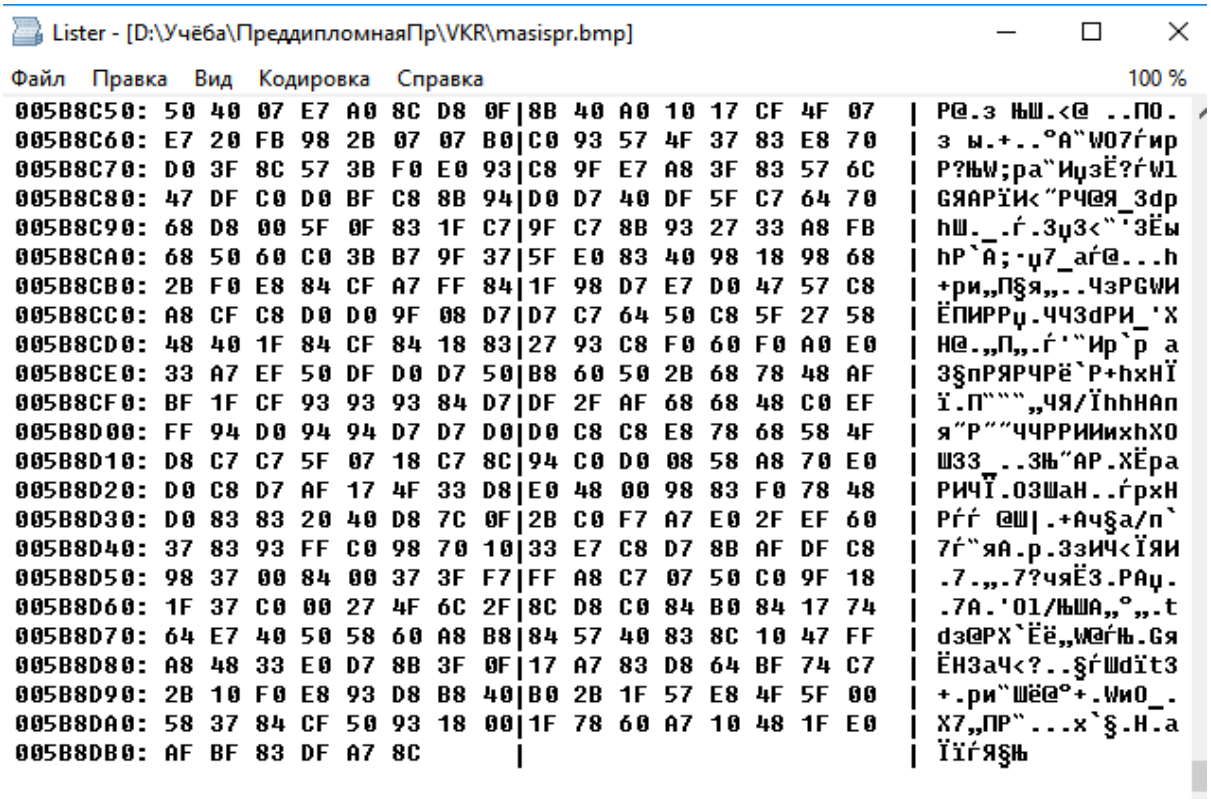

Рисунок 22 — Демонстрация шифрования чёрно-белых изображений (код зашифрованного файла (конец кода))

В результате выполнения данного этапа задания на преддипломную практику была разработана программа шифрования изображений, соответствующая требованиям, описанным в техническом задании на выпускную квалификационную работу.

6 Руководство пользователя программы

6.1 Назначение программы

6.1.1 Функциональное назначение программы.

Функциональным назначением программы является работа с файлами, введенными пользователем, содержащими изображения формата BMP и JPG. Первый введённый файл зашифровывается методом гаммирования с использованием вводимых в программу текстовых файлов с ключом и таблицами подстановки. Вывод результата шифрования изображения сохраняется во второй введённый файл путём его перезаписи. Если первый файл не был найден программой (не существует), выводиться предупредительное сообщение. Также сообщение об ошибке выводится, если ключ шифрования изображения не имеет необходимого размера и содержит в себе недопустимые символы. Также, если введённый файл не является форматом JPG или BMP, выдаётся сообщение об ошибке.

6.1.2 Эксплуатационное назначение программы.

Программа предназначена для использования на персональных компьютерах с операционной системой Windows версии не ниже Windows 2000.

6.1.3 Состав функций.

Шифрование методом гаммирования вводимых в программу файлов и вывод сообщения об ошибке в случаях, если файл, надлежащий зашифрованию, не был найден и ключ шифрования введён не правильно.

6.2 Условия выполнения программы

6.2.1 Минимальный состав аппаратных средств.

Аппаратными средствами является ПЭВМ с процессорами типа Intel Core i3 и выше. 512 МБайт видеопамяти, свободное дисковое пространство не менее 50 Мбайт, оперативная память не менее 2048 Мбайт.

6.2.2 Минимальный состав программных средств.

Операционная система Windows версии не ниже Windows 7.

6.2.3 Требования к персоналу (пользователю).

Требованиями к персоналу являются начальные навыки работы с компьютером с установленной на него операционной системой Windows версии не ниже Windows 7.

6.3 Выполнение программы

6.3.1 Загрузка и запуск программы.

Запуск программы осуществляется двойным кликом левой кнопки мыши по файлу VKR.exe.

6.3.2 Выполнение программы.

Во время выполнения программы нужно ввести два файла для зашифрования изображения: первый — для обработки, второй — для сохранения результата.

6.3.3 Завершение работы программы.

По завершению работы программы на монитор выводится сообщение об успешном завершении процедуры шифрования и размер обработанного изображения в байтах.

6.4 Сообщения оператору

В процессе работы оператору выводятся сообщения:

— «Ошибка открытия файла» - запрос на проверку существования файла;

— «Длина ключа ... не совпадает с должной. Исправьте ошибку!» - запрос на проверку и исправление вводимого ключа шифрования;

— «В содержимом ключа Ai(0) символы не соответствуют двоичному представлению. Исправьте ошибку!» - запрос на проверку и исправление вводимого ключа шифрования;

— «Введите название файла, подлежащего шифрованию:» - запрос на ввод пути файла, подлежащего зашифрованию;

— «Введите названия файла для сохранения результата:» - запрос на ввод пути файла для сохранения в нём результатов шифрования;

— «Файл формата JPG» - пояснение формата принятого файла;

— «Файл формата BMP» - пояснение формата принятого файла;

— «Неподдерживаемый формат файла! Укажите путь другого файла:» - запрос на ввод файла другого формата;

— «Файл успешно защифрован!» - уведомление об успешном выполнении зашифрования изображения;

— «Конец работы программы» - уведомление о конце работы программы.

7 Руководство по поиску недекларированных возможностей в разработанной программе шифрования

7.1 НДВ № 1: программа добавляет ключ к файлу после процедуры шифрования, если изображение черно-белое

Для обнаружения ошибок программы шифрования необходимо подготовить изображения для исследования, отличающиеся друг от друга различными параметрами: изображения малого и большого объёма, различного типа кодирования - количества бит на пиксель, различного содержания, форматов JPG, BMP и других, для проверки обработки ошибок.

НДВ добавления ключа к файлу после процедуры шифрования, если изображение черно-белое, достаточно легко определить, если сравнивать параметры исходных и зашифрованных файлов.

НДВ проявится, если проанализировать исходное и зашифрованное изображения файлов, закодированных одним битом на пиксель, а также файлов, код пикселей которых равен «00» (черный цвет) и «FF» (белый цвет).

К примеру, возьмём черно-белое изображение, закодированное 24 разрядами на пиксель (см. рисунок 23).

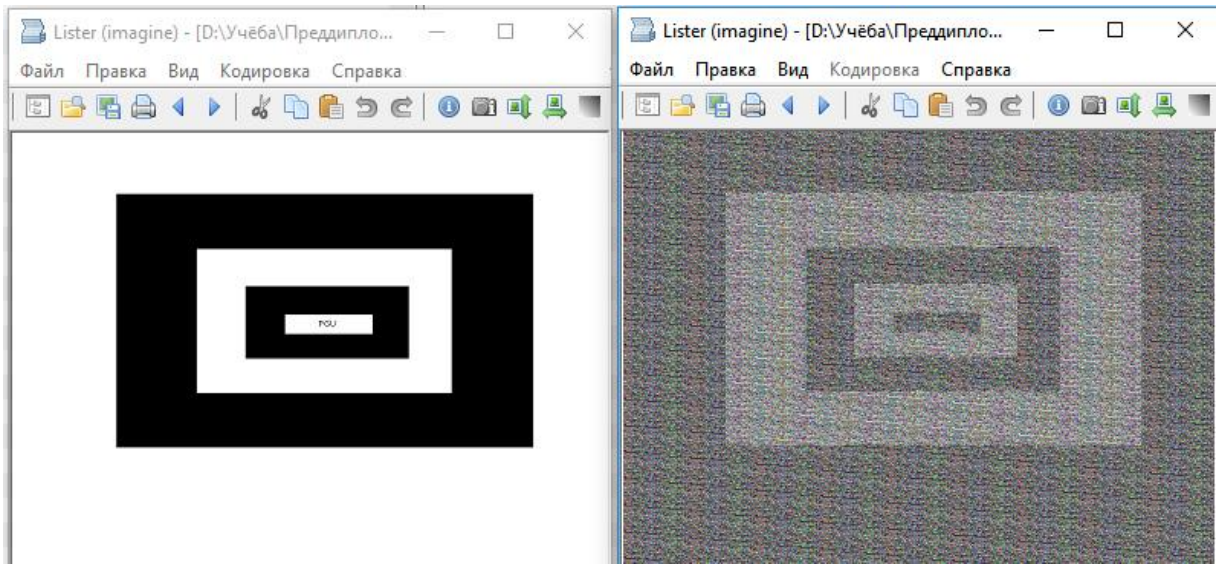

Рисунок 23 — Демонстрация шифрования чёрно-белых изображений

При сравнении размеров изображений становится ясным, что зашифрование изображения произошло не правильно, так как обработанный файл стал больше на 3 Кбайта. Проверяем код файлов. Исходный код файла представлен на рисунке 24, зашифрованного — на рисунке 25.

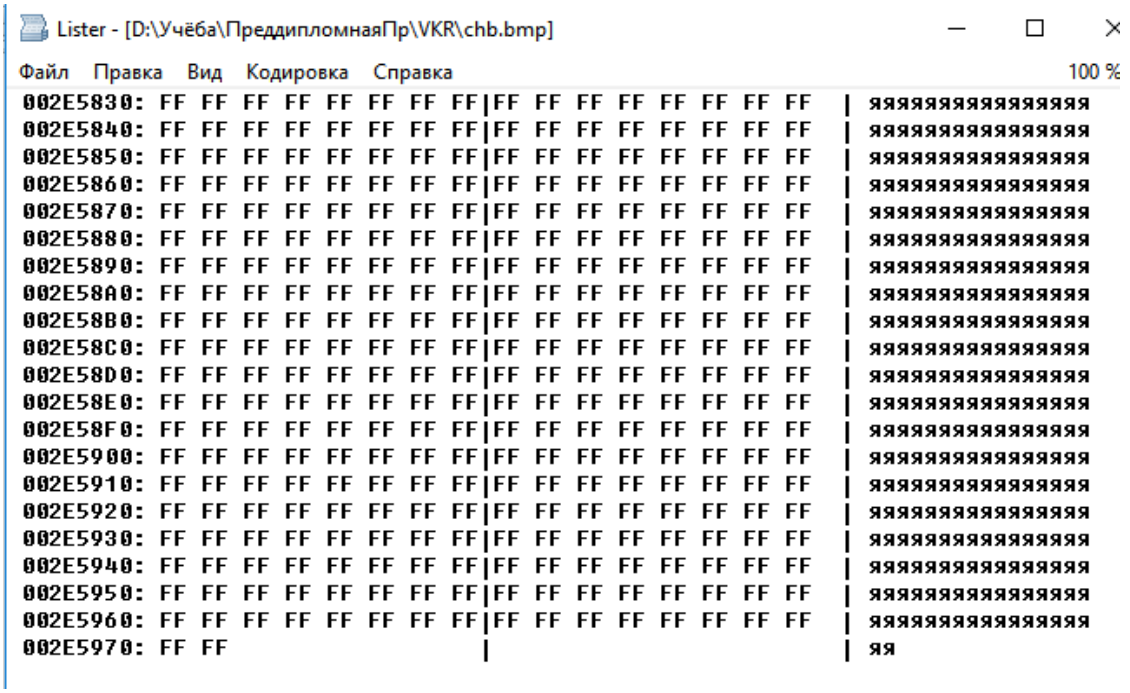

Рисунок 24 — Демонстрация шифрования чёрно-белых изображений (код изначального файла (конец кода))

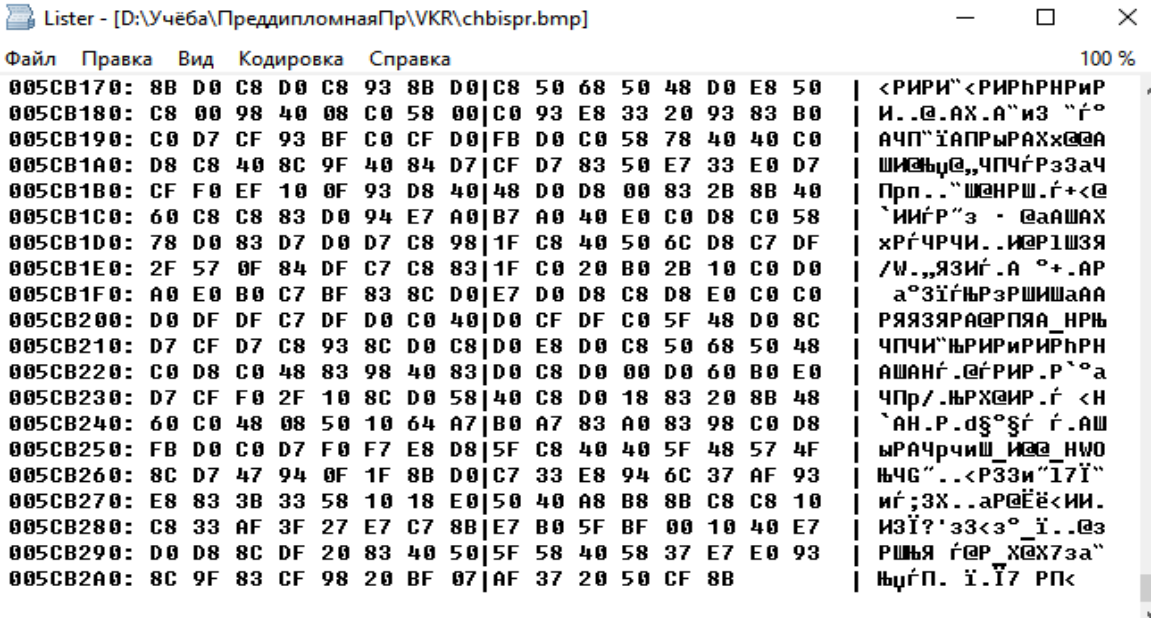

Рисунок 25 — Демонстрация шифрования чёрно-белых изображений (код зашифрованного файла (конец кода))

Из представленных изображений видно, что в зашифрованном изображении в конце кода добавляется дополнительный код, при том равный по количеству числу кодов пикселей исходного файла, не относящийся к пиксельным данным.

Если наложить этот дополнительный код на пиксельный код зашифрованного изображения, станет ясно, что к зашифрованному изображению добавляется гамма-шифра.

7.2 НДВ № 2: программа производит инверсию бит файла вместо его шифрования, если размер изображения не превышает 20 килобайт

Данную НДВ не всегда можно выявить наглядно, просто сопоставив друг с другом исходный и зашифрованный файлы. Поэтому следует воспользоваться динамическим анализом.

Для определения того, что программа производит инверсию бит файла вместо его шифрования, необходимо построить гистограммы вероятности появления байт в исходном и зашифрованном файлах.

В результате проведения анализа полученных гистограмм получится, что распределение вероятности появления байт в исходном изображении будет сопоставимым с распределением зашифрованного изображения.

При правильном же шифровании в обработанном изображении распределение вероятности появления байт должно стремиться к равномерному.

Для анализа использовалось изображение, приведённое на рисунке 26.

Исходный код файла представлен на рисунке 27, зашифрованном — на рисунке 28.

Построенные графики гистограмм исходного и зашифрованного файлов представлены на рисунках 29 и 30 соответственно.

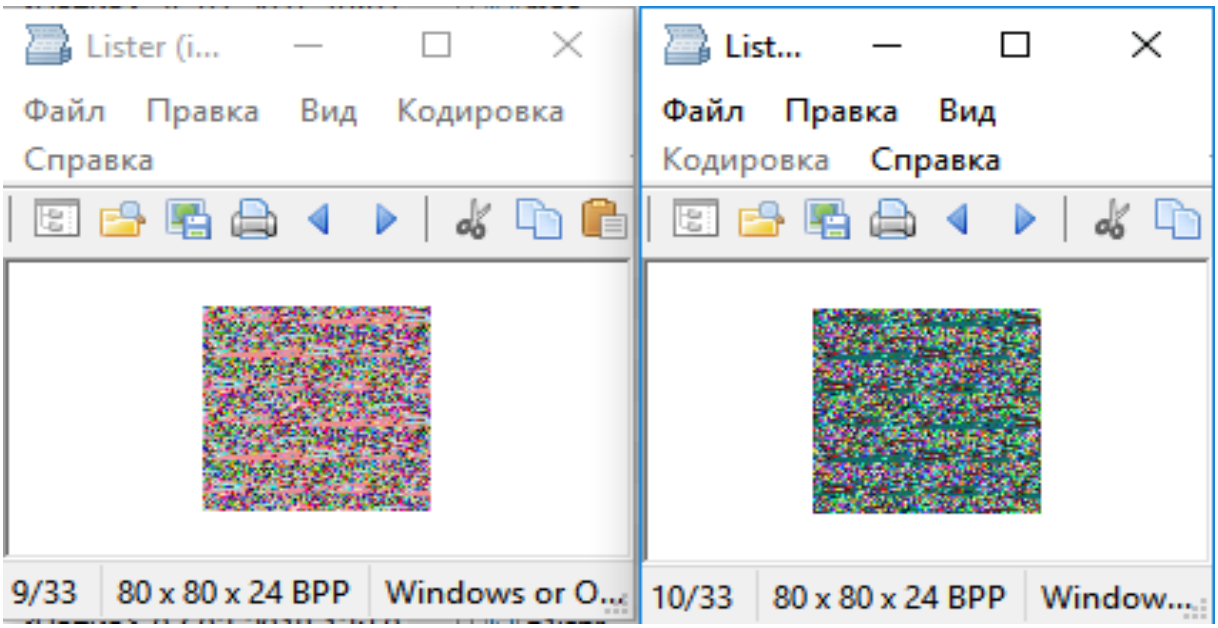

Рисунок 26 — Демонстрация шифрования изображений, меньших 20 Кбайт

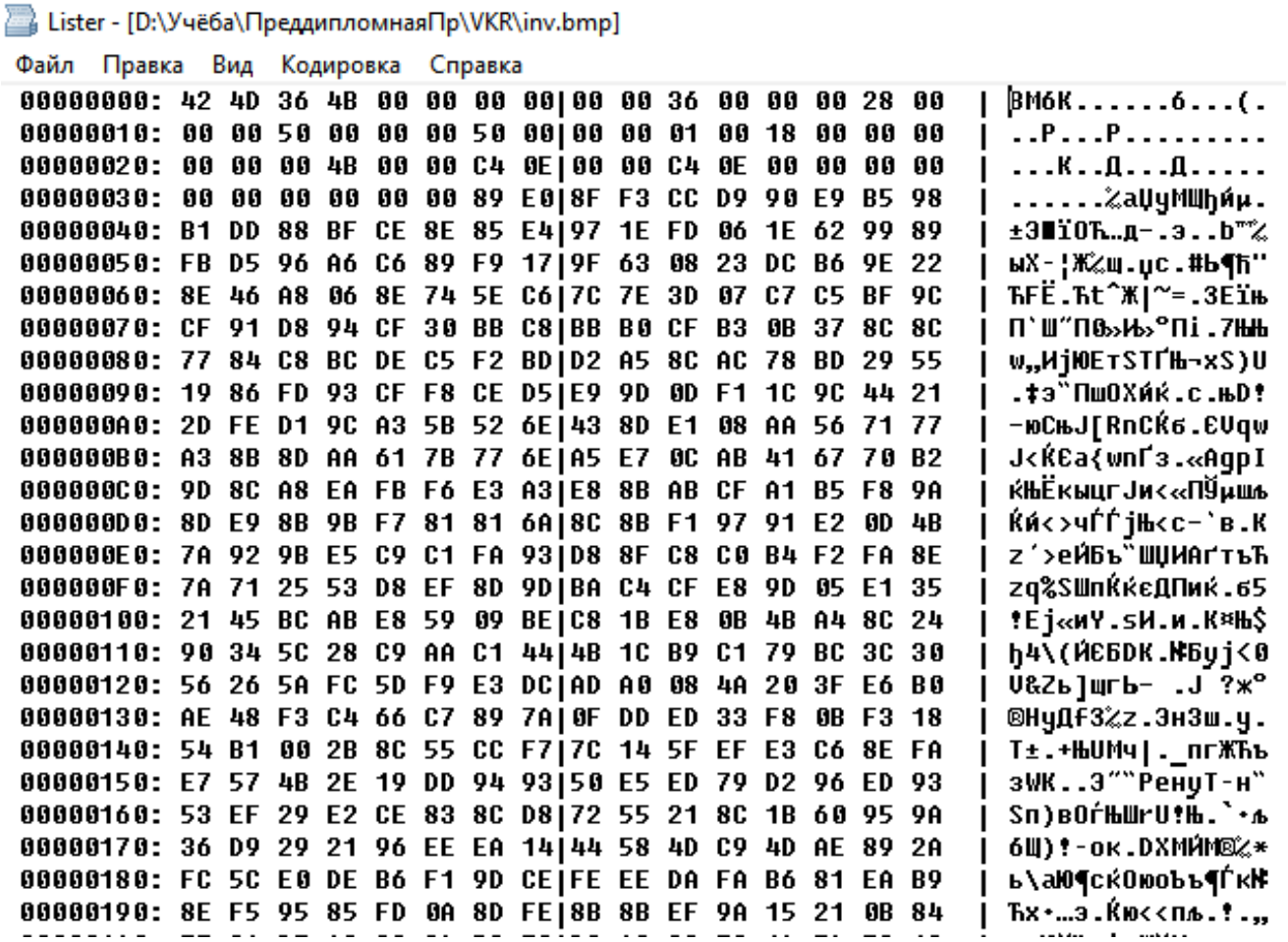

Рисунок 27 — Демонстрация шифрования изображений, меньших 20 Кбайт (код изначального файла)

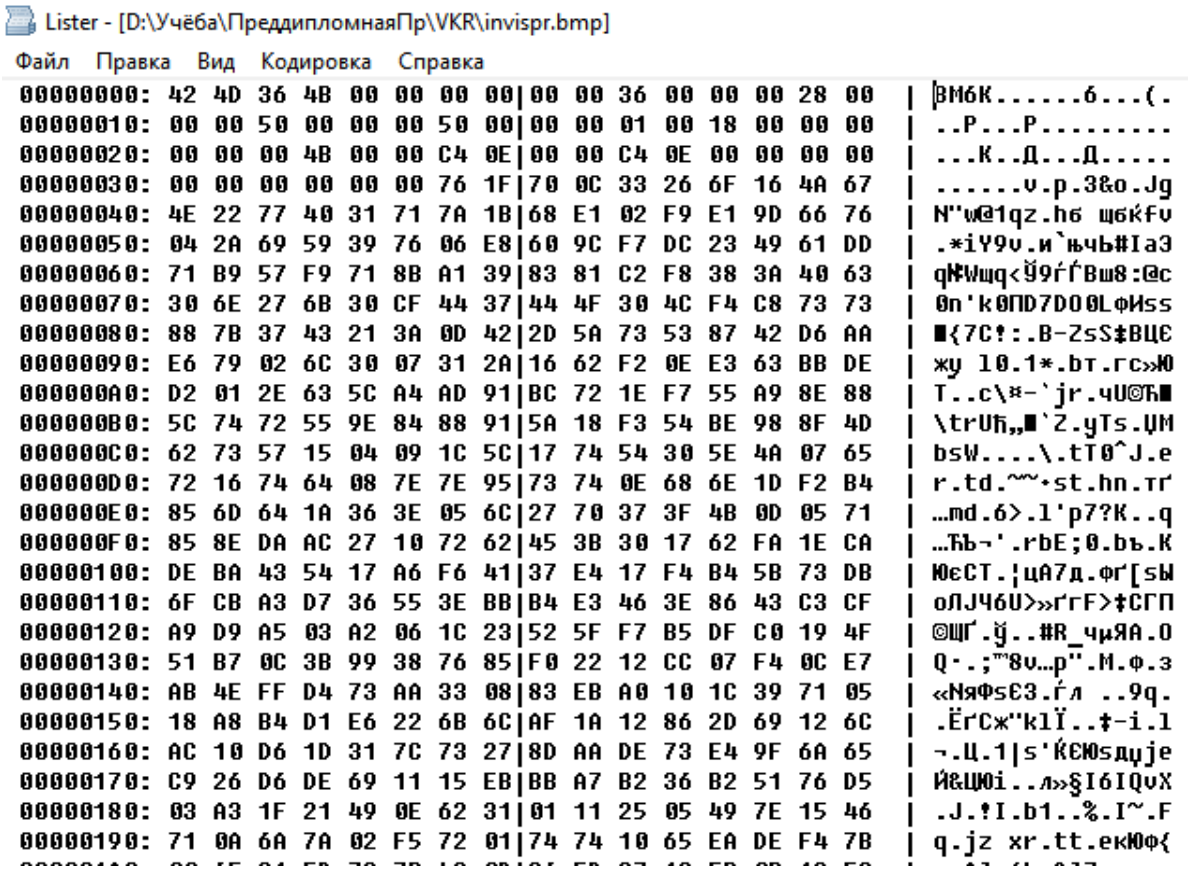

Рисунок 28 — Демонстрация шифрования изображений, меньших 20 Кбайт (код зашифрованного файла)

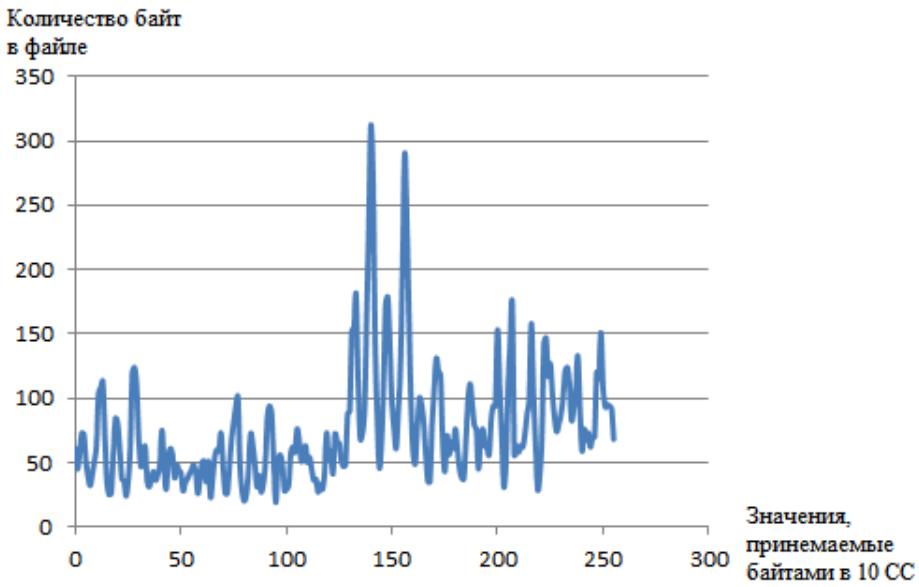

Рисунок 29 — Гистограмма вероятности появления байт в исходном файле

На полученных графиках по оси X размещены значения, которые могут принимать байты файла, а по оси Y – количество встречаемых этих байтов в файле.

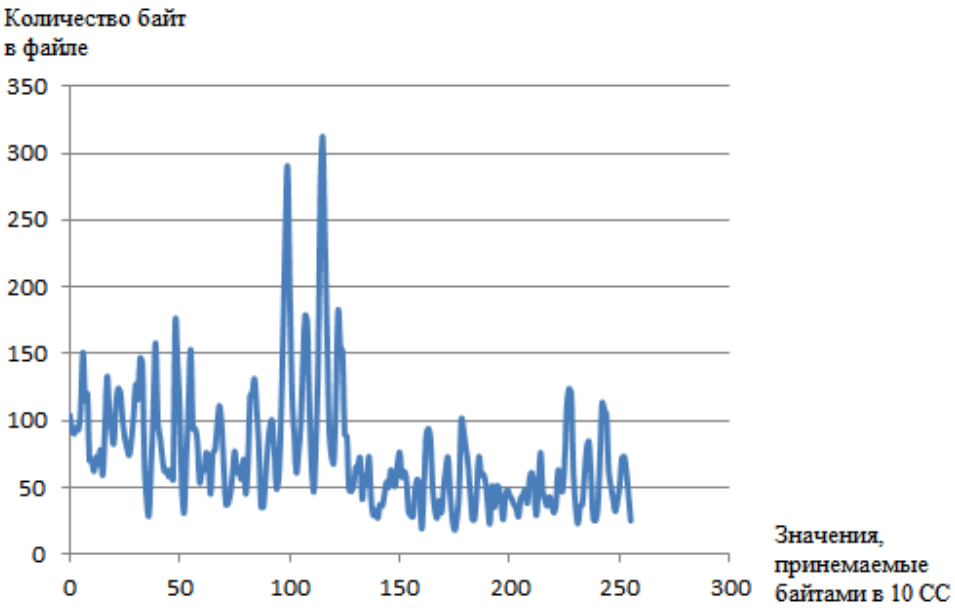

Рисунок 30 — Гистограмма вероятности появления байт в зашифрованном файле

Проанализировав полученные гистограммы, можно отметить, что функция, представленная на рисунке 30, является перевёрнутой по оси X функцией, изображенной на рисунке 29.

Из этого можно сделать вывод, что изображение, представленное на рисунке 26, зашифровано инверсией бит.

7.3 НДВ № 3: программа вырабатывает периодическую гамму шифра для файлов формата BMP

Данную НДВ тоже не всегда можно выявить наглядно, просто сопоставив друг с другом исходный и зашифрованный файлы. Динамический анализ с помощью построения гистограмм вероятности появления байт тоже не поможет выявить данную НДВ.

Для выявления НДВ, состоящей в выработке периодической гаммы шифра для файлов формата BMP, необходимо построить для исходного и зашифрованного файлов функции автокорреляции.

Для анализа использовалось изображение, представленное на рисунке 31.

Исходный код файла представлен на рисунке 32, зашифрованного — на рисунке 33.

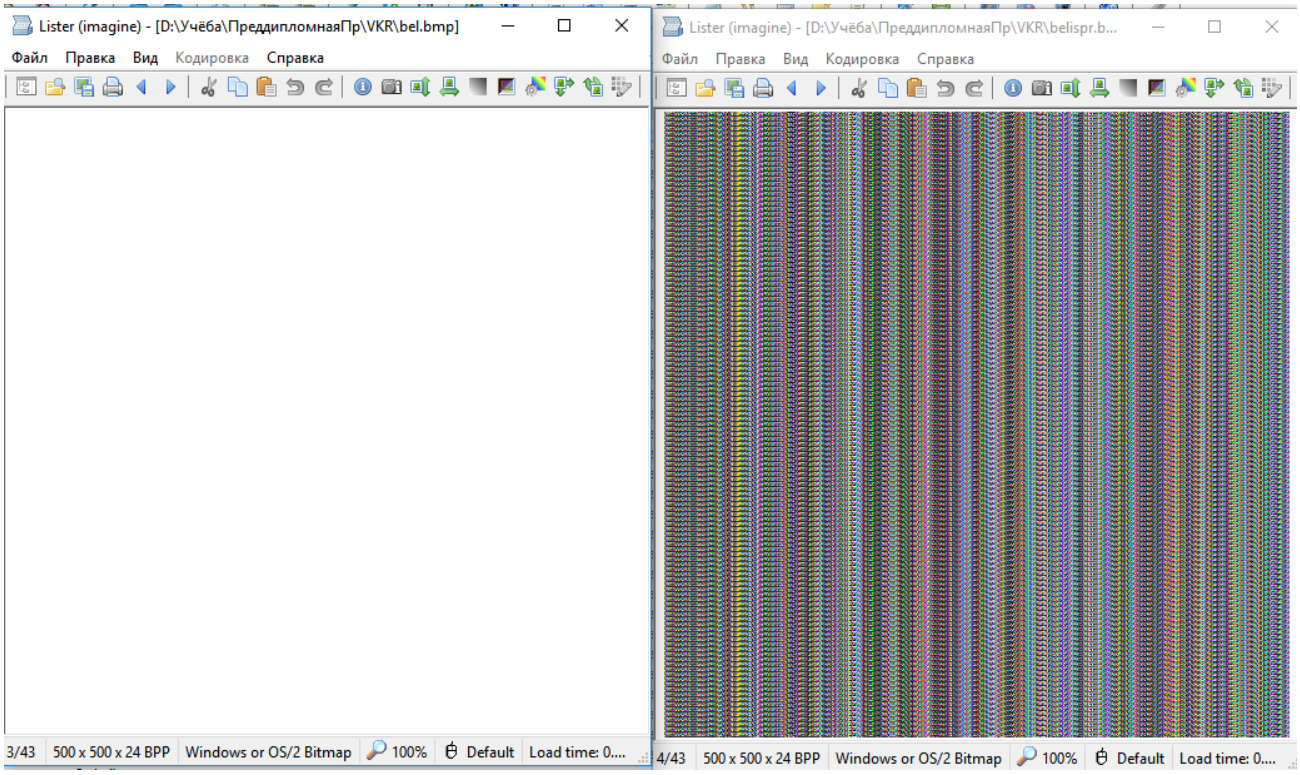

Рисунок 31 — Демонстрация шифрования изображений формата BMP

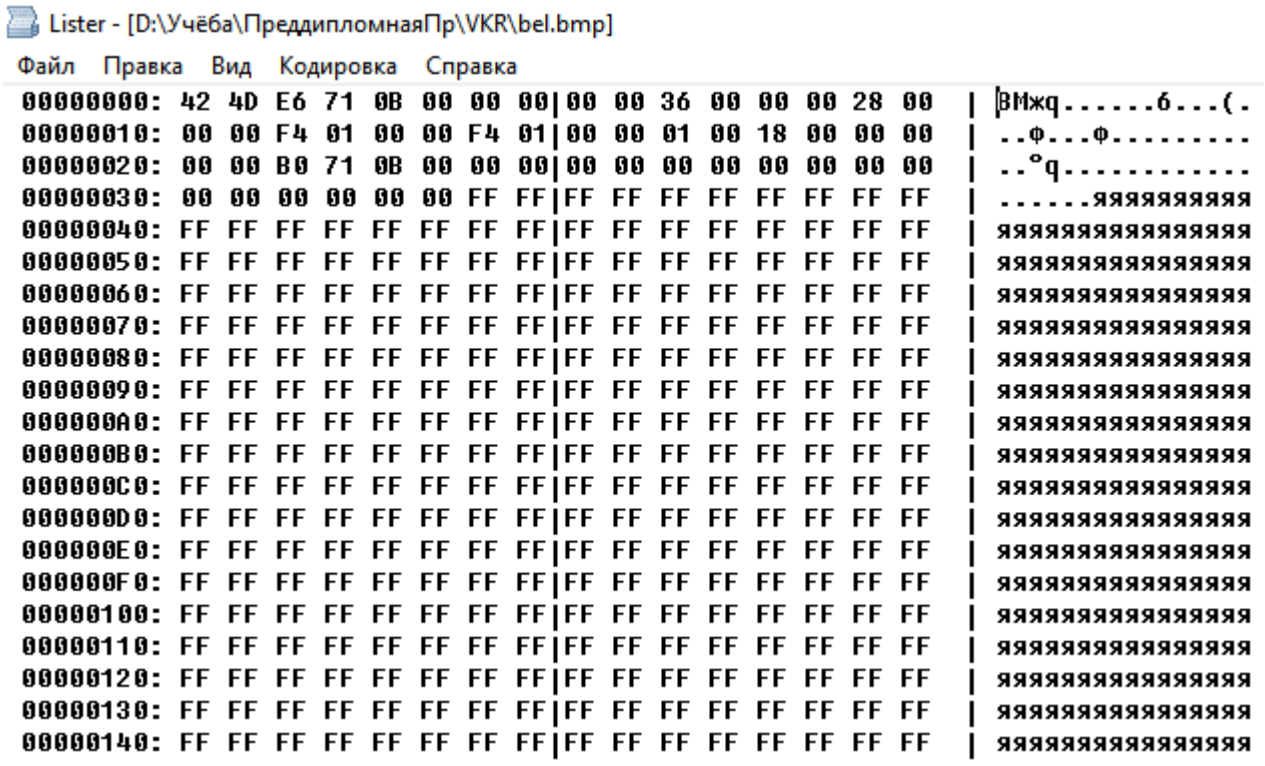

Рисунок 32 — Демонстрация шифрования изображений формата BMP (код изначального файла)

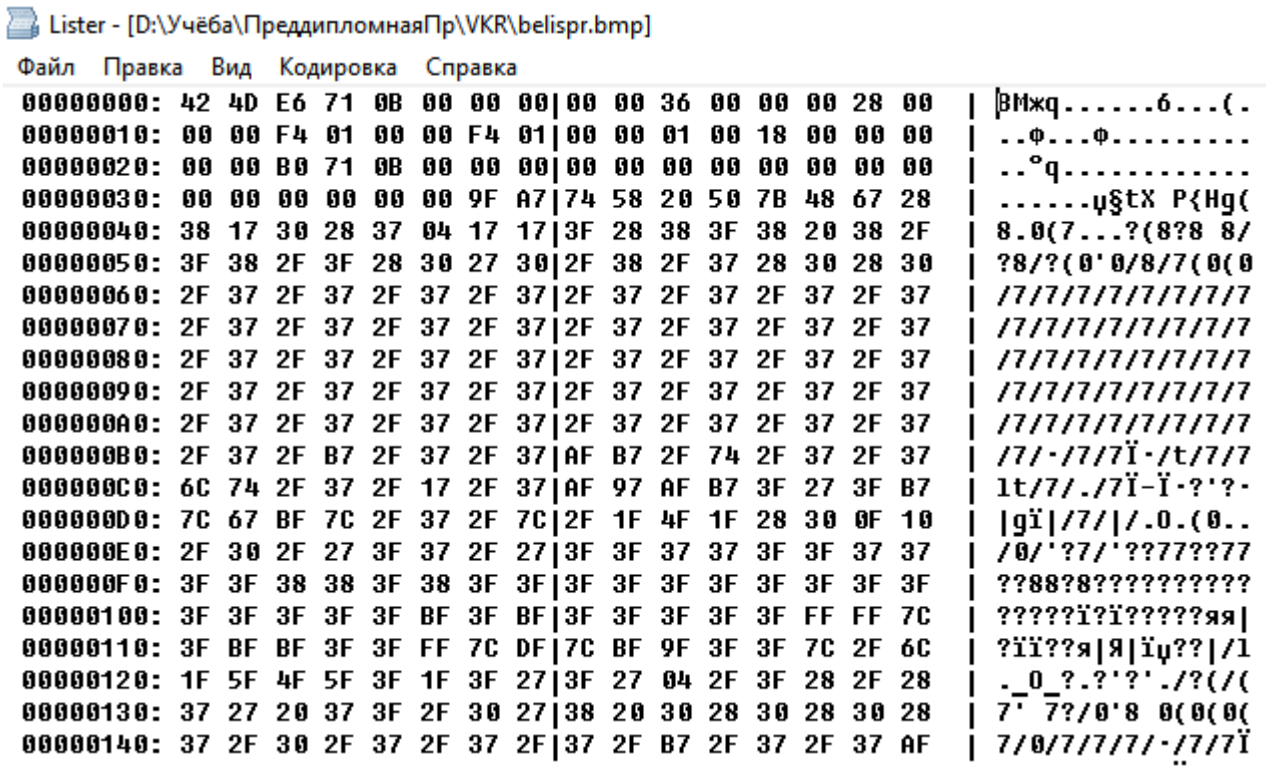

Рисунок 33 — Демонстрация шифрования изображений формата BMP (код зашифрованного файла)

В статистике автокорреляция случайного процесса описывает корреляцию между значениями процесса в различные моменты времени.

Пусть X_t - значение случайного процесса в момент времени t (t может быть вещественным, если процесс непрерывный, или целым, если процесс дискретный) [26].

Если X_t имеет среднее значение μ_t и дисперсию $\sigma^2{}_t$, то автокорреляция X_t определяется следующим образом:

$$R(t,s) = \frac{\mathrm{E}[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s},$$

где E – это математическое ожидание.

Если же это выражение корректно, то его значение лежит в интервале от минус 1 до 1, причем 1 оно принимает в случае полного совпадения, а 0 - в случае, если корреляции не наблюдается.

При помощи программы вычисления функции автокорреляции, был построен график этой функции по исходному коду исследуемого изображения (см. рисунок 34) и по его зашифрованному виду (см. рисунок 35).

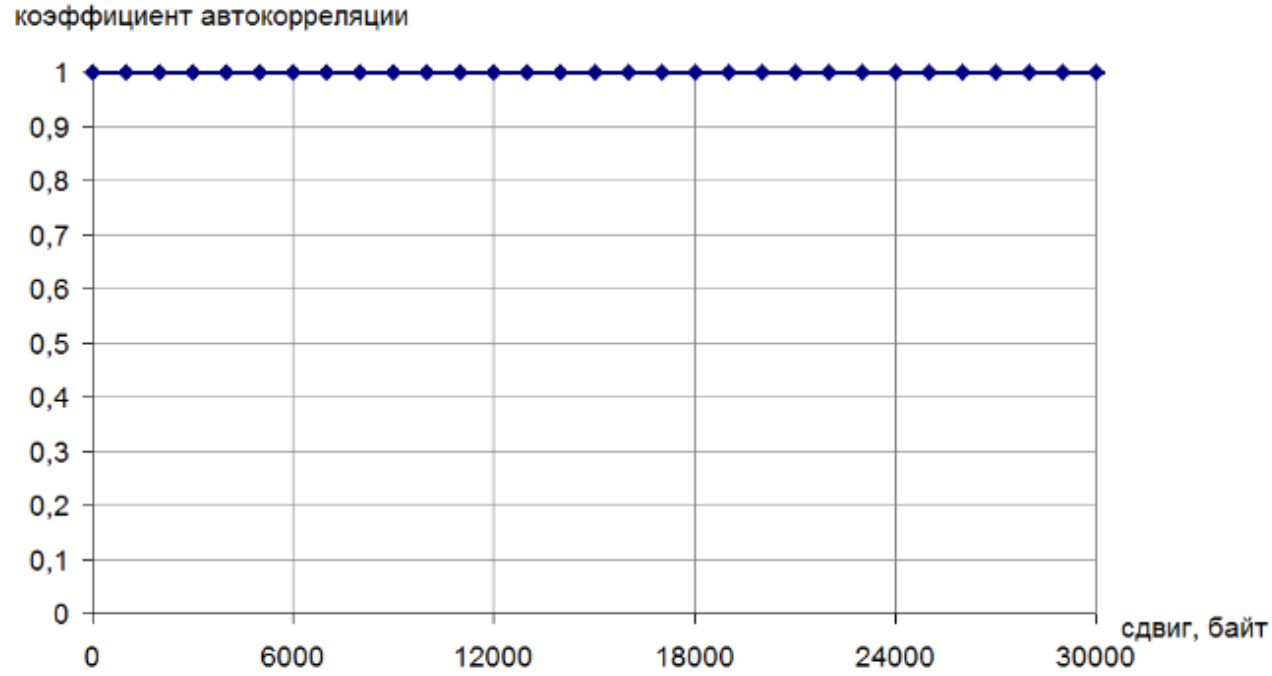

Рисунок 34 — Функция автокорреляции для исходного файла

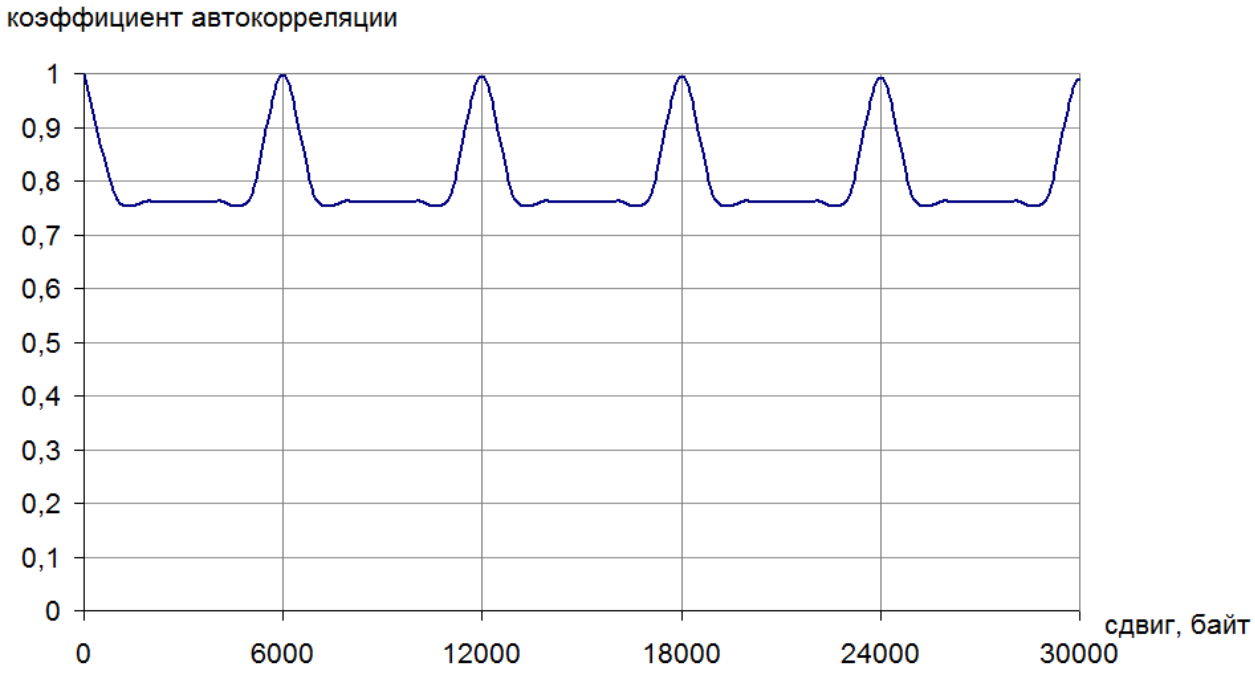

Рисунок 35 — Функция автокорреляции для зашифрованного файла

Наличие на графике периодичности поведения функции автокорреляции - признак повторяемости байтов кода файла. В данном примере период функции равен 6000 байт.

Таким образом, проанализировав полученные графики функций, можно сделать вывод, что файл с зашифрованным изображением закодирован с использованием периодической гаммы шифра.

Для данных НДВ были описаны методы динамического анализа для поиска ошибок в ходе программы. Более точный и развёрнутый результат анализа может дать статический анализ кода. Однако не всегда есть возможность дизассемблировать код или же есть код самой программы шифрования.

8 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

Человеческий фактор занимает особое место в информационной безопасности. В условиях обработки защищаемой информации необходимо учитывать психофизиологические особенности участников трудового процесса [23]. Психофизиологические основы безопасности базируются на психологии и физиологии человека.

Психическое состояние человека — это относительно устойчивая структурная организация всех компонентов психики, выполняющая функцию активного взаимодействия человека с внешней средой, представленной в данный момент конкретной ситуацией [24]. Психические состояния человека отличаются разнообразием и временным характером, определяют особенности психической деятельности в конкретный момент и могут положительно или отрицательно сказываться на течении всех психических процессов. В процессе деятельности реакция организма на внешние изменения не остается постоянной. Организм стремится приспособиться к изменяющимся условиям деятельности, преодолеть трудности и опасности.

Стресс проявляется во всеобщем адаптационном синдроме как необходимая и полезная реакция организма на резкое увеличение его общей внешней нагрузки. Он состоит в целом ряде физиологических сдвигов в организме, способствующих повышению его энергетических возможностей и успешности выполнения сложных и опасных действий.

Поэтому сам по себе стресс является не только целесообразной защитной реакцией человеческого организма, но и механизмом, содействующим успеху трудовой деятельности в условиях помех, трудностей и опасностей, что особенно важно в области обеспечения информационной безопасности бизнес-процессов организации.

Стресс оказывает положительное влияние на результаты труда лишь до тех пор, пока он не превысил определенного критического уровня. При превышении же этого уровня в организме развивается так называемый процесс гипермобилизации, который влечет за собой нарушение механизмов саморегуляции и ухудшение результатов деятельности, вплоть до ее срыва. Гипермобилизация организма приводит к чрезмерным формам психического состояния, которые называются дистрессом или запредельными формами. Можно выделить два типа запредельного психического напряжения - тормозной и возбудимый.

Тормозной тип характеризуется скованностью и замедленностью движений. Специалист не способен с прежней ловкостью производить профессиональные действия. Снижается скорость ответных реакций. Замедляется мыслительный процесс, ухудшается воспоминание, проявляются рассеянность и другие отрицательные признаки, несвойственные данному человеку в спокойном состоянии.

Возбудимый тип проявляется гиперактивностью, многословностью, дрожанием рук и голоса. Операторы совершают многочисленные, не диктуемые конкретной потребностью действия. Они проверяют состояние приборов, поправляют одежду, растирают руки, в общении с окружающими они обнаруживают раздражительность, вспыльчивость, несвойственную им резкость, грубость, обидчивость.

Длительные психические напряжения и особенно их запредельные формы ведут к выраженным состояниям утомления.

Влияние даже однократных сильных стрессорных воздействий на организм и постоянное нервно-эмоциональное напряжение могут привести к дестабилизации функций нервной и сердечно-сосудистой систем, что, в свою очередь, ведёт к возникновению психосоматической патологии [25]. Отрицательное воздействие на организм усиливается при суммарном влиянии

нескольких факторов риска, когда они воздействуют одновременно и принимают хронический характер.

Организация контроля за психическим состоянием работников необходима в области обеспечения защиты информации в связи с возможностью появления у специалистов особых психических состояний, который не являются постоянным свойством личности, но, возникая спонтанно или под влиянием внешних факторов, существенно изменяют работоспособность человека.

В условиях большой ответственности за надлежащее исполнение возложенных на специалиста задач, в потенциальных условиях повышенного риска специалист по информационной безопасности должен быть в состоянии адекватно реагировать на события информационной безопасности.

К объективным факторам работы, влияющим на состояние специалиста, относят повышенную ответственность за надлежащее исполнение возложенных задач, возраст, пол, состояние здоровья, общую профессиональную нагрузку, отдых, в том числе активный.

К субъективным факторам относятся знания, профессиональные способности, мотивация к рабочей деятельности, работоспособность, нервно-психическая устойчивость, темп рабочей деятельности, утомляемость, психофизические возможности, особенности характера, темперамент, коммуникабельность, способность адаптироваться к социальным условиям в рабочей среде.

Рабочий процесс требует напряжения основных психических функций, таких как память (долговременная и оперативная), внимание (особенно его концентрация и устойчивость), восприятие, оперативное мышление. Кроме того, рабочий процесс сопровождается наличием стрессовых ситуаций (события и инциденты информационной безопасности).

Для поддержания здоровья и работоспособности на соответствующем уровне одинаково необходимы четыре элемента: физическая нагрузка,

рациональное питание, закаливание и умение отдыхать. При отсутствии достаточной дозы ежедневных мышечных движений происходят нежелательные и существенные изменения функционального состояния мозга и сенсорных систем.

Среди мероприятий, направленных на повышение умственной работоспособности, на преодоление и профилактику психоэмоционального и функционального состояния можно рекомендовать следующие:

— систематическое обучение специалиста по защите информации в рамках повышения квалификации в течение года;

— ритмичная и системная организация труда;

— постоянное поддержание интереса с помощью особых поощрений и актуальности поставленных задач;

— совершенствование межличностных отношений между сотрудниками;

— организация рационального режима труда, питания, сна и отдыха;

— включение двигательной активности и физической культуры;

— прохождение специалистом по защите информации обучения методам самоконтроля состояния организма с целью выявления отклонений от нормы, своевременной корректировки и устранения этих отклонений средствами профилактики.

Примером двигательной активности и физической культуры могут послужить малые формы физической культуры в режиме рабочего дня. К ним относятся: утренняя гигиеническая гимнастика, физкультурная пауза, физкультурные микропаузы с использованием физических упражнений.

Утренняя гигиеническая гимнастика является наименее сложной, но достаточно эффективной формой для ускорения включения сотрудника в режим продуктивного трудового дня. Уровень современных знаний позволяет утверждать, что среди самых разнообразных средств и методов занятий физическими упражнениями утренняя гигиеническая гимнастика – одно из

ценных в оздоровлении организма и повышения его работоспособности. Она создаёт в организме «рабочую установку», усиливает ток крови и лимфы во всех частях тела и учащает дыхание, что активизирует обмен веществ и быстро удаляет продукты распада, накопившиеся за ночь, ликвидирует неблагоприятные последствия гиподинамии и подготавливает нервную систему, двигательный аппарат и внутренние органы к предстоящей ответственной работе.

Физкультурная пауза является действенной и доступной формой, призванной решать задачу обеспечения активного отдыха специалиста и повышения его работоспособности. Для сохранения высокой умственной работоспособности в течение всего рабочего дня целесообразно к концу третьего – четвертого часа «сидячей» работы использовать «малые формы» активного отдыха, так как к этому времени наступает утомление от большого умственного напряжения и недостатка кислорода в помещении. Как результат, снижается внимание, активность, интерес к делам.

Микропаузы в рабочей деятельности специалиста по защите информации с использованием физических упражнений полезны в связи с тем, что в условиях труда в силу разнообразных факторов возникают состояния отвлечения от выполняемой работы. Чаще это обусловлено усталостью в условиях ограничения активности скелетной мускулатуры, монотонным характером выполняемой работы. При снижении внимания и наступающего утомления рекомендуется проведение физкультурных минуток – малой формы активного отдыха преимущественно локального характера длительностью от 1 до 2 мин. Она уменьшает утомление отдельных частей тела (кистей, шеи, спины, ног), или систем и органов (нервной, зрительного анализатора, голосового аппарата и др.). Оказывая общее стимулирующее влияние на организм, физкультурные минутки улучшают его функциональное состояние, являются в полном смысле экспресс-методом восстановления, незаменимы в условиях длительной работы без регламентированных перерывов для отдыха.

Положительный эффект достигается лишь при многократном выполнении разных по содержанию физкультурных минуток в течение дня.

Физкультурная микропауза – ещё одна малая форма активного отдыха, получившая в последнее время широкое распространение. Обычно используют одно упражнение, отдельное движение или специальный приём, повторенный от 4 до 6 раз. Применяют разнообразные дыхательные упражнения, приёмы самомассажа, холодовые воздействия на лицо и голову. Продолжительность физкультурной микропаузы составляет от 20 до 30 с, но после её выполнения у человека быстро наступает чувство облегчения. Сложность овладения этой формой активного отдыха заключается в том, что человек должен хорошо разбираться, что и когда ему лучше применять из арсенала средств восстановления.

Малые формы активного отдыха хороши тем, что на выполнение упражнений затрачивается минимум времени, а эффект облегчения достигается сразу, так как средства и приемы их просты и доступны каждому. Они успешно сочетаются с другими формами отдыха или применяются отдельно, причем в большинстве рабочих условий.

Хорошее психофизическое развитие специалиста по защите информации повышает трудовые возможности человека, их качество, позволяет успешно и адекватно реагировать на возникающие события и инциденты информационной безопасности, переносить высокие психические и физические нагрузки, эффективно функционировать в обычных и экстремальных условиях.

Заключение

Контроль программного кода и выявление НДВ – это та область, где особенно важно превентивное воздействие. Не расследование инцидента после его реализации и выявление «сработавших» программных закладок, а предотвращение самого инцидента, так как последствия нарушения работы программ, кражи или искажения данных могут оказаться фатальными.

В рамках выполнения выпускной квалификационной работы было разработано программное средство для изучения методов поиска недекларированных возможностей при функциональном тестировании программного обеспечения. Данная программа предназначена для выполнения обучаемыми курсового проекта по дисциплине «Разработка и эксплуатация защищённых автоматизированных систем».

На первом этапе выполнения работы были проанализированы основные разновидности недекларированных возможностей. Было выявлено, что НДВ могут применяться не только для шпионажа, нарушения работоспособности компьютеров и оборудования коммерческих и общественных организаций, но и для вывода из строя АСУ ТП критически важных объектов, управления техникой военного или двойного назначения.

На втором этапе выполнения работы были выявлены угрозы ИБ, возникающие в случае реализации НДВ. Было определено, что при наличии НДВ в информационной системе для неё могут быть актуальны угрозы 1-го и 2-го типа, что приводит к необходимости обеспечения, в зависимости от деятельности организации, 1, 2 или 3 уровня защищённости данных при их обработке в информационной системе.

На третьем этапе выполнения работы были сформулированы основные методы поиска НДВ при функциональном тестировании программных средств и при анализе исходного кода программных средств. Было определено, что методы статического и динамического анализа ПО основываются на

формальном подходе и приближают перспективное решение проблемы, связанное с доказательством разрешимости множества НДВ. Кроме того, они позволяют создать легко применяемые средства анализа, независящие от анализируемых программ.

На четвёртом этапе выполнения работы был разработан алгоритм разработанной программы шифрования изображений. По алгоритму производится шифрование пиксельного кода BMP и JPG файлов методом гаммирования с использованием внешних ключа шифрования и таблиц подстановки.

На пятом этапе выполнения работы была реализована по разработанному алгоритму программа шифрования изображений на языке программирования Си++. Программа осуществляет проверку на ввод ключа шифрования и выдаёт сообщение об ошибке, если ключ шифрования введён не правильно. Также в программе были реализованы НДВ с целью последующего описания методов поиска НДВ в программном средстве.

На шестом этапе выполнения работы было разработано руководство пользователя программы. Были описаны способы и условия работы с программой для осуществления процедуры шифрования изображений.

На седьмом этапе выполнения работы было разработано руководство по поиску НДВ в разработанной программе шифрования. Были описаны методы статического и динамического анализа в целях поиска реализованных НДВ в программном средстве. Были промоделированы действия обучаемых при выполнении курсового проекта по дисциплине «Разработка и эксплуатация защищённых автоматизированных систем».

На восьмом этапе выполнения работы были выявлены вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Были определены методы и средства физической культуры, снижающие их воздействие, как рекомендации к

применению в организации мер по регулированию психофизиологического состояния специалиста.

Таким образом, все пункты технического задания выполнены полностью.

Список используемых источников

1 ГОСТ Р 50922-2006. Защита информации. Основные термины и определения [Электронный ресурс] - URL: http://docs.cntd.ru/document/gost-r-50922-2006 (дата обращения — 04.06.20);

2 Академик. Словарь бизнес-терминов. [Электронный ресурс] - URL: https://dic.academic.ru/dic.nsf/business/17665 (дата обращения — 10.02.20);

3 ГОСТ Р 51275-2006. Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положения [Электронный ресурс]: – URL: http://docs.cntd.ru/document/gost-r-51275-2006 , (дата обращения — 10.02.20);

4 ГОСТ Р ИСО/МЭК 18028-1-2008 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Сетевая безопасность информационных технологий. Часть 1. Менеджмент сетевой безопасности [Электронный ресурс] - URL: http://docs.cntd.ru/document/1200081733 (дата обращения — 04.06.20);

5 ГОСТ Р 52863-2007 Защита информации. Автоматизированные системы в защищенном исполнении. Испытания на устойчивость к преднамеренным силовым электромагнитным воздействиям. Общие требования [Электронный ресурс] - URL: http://docs.cntd.ru/document/1200065692 (дата обращения — 04.06.20);

6 ГОСТ 28806-90. Качество программных средств. Термины и определения [Электронный ресурс] - URL: http://docs.cntd.ru/document/gost-28806-90 (дата обращения — 01.06.20);

7 ГОСТ Р 53114-2008. Обеспечение информационной безопасности в организации. Основные термины и определения [Электронный ресурс] - URL: http://docs.cntd.ru/document/gost-r-53114-2008 (дата обращения — 01.06.20);

8 АО «Лаборатория Касперского». Backdoor [Электронный ресурс] – URL: https://encyclopedia.kaspersky.ru/knowledge/backdoor/, (дата обращения — 10.02.20);

9 TADVISER. Государство. Бизнес. ИТ. Руткит (Rootkit) [Электронный ресурс] - URL: http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A0%D1%83%D1%82%D0%BA%D0%B8%D1%82_(Rootkit), (дата обращения — 11.02.20);

10 АО «Лаборатория Касперского». Что такое Кейлоггер? [Электронный ресурс] – URL: https://www.kaspersky.ru/blog/chto-takoe-keylogger/700/, (дата обращения — 11.02.20);

11 Руководящий документ «Защита от несанкционированного доступа к информации Часть 1. Программное обеспечение средств защиты информации" [Электронный ресурс]: – URL: https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty/114-spetsialnye-normativnye-dokumenty/382-rukovodyashchij-dokument-prikaz-predsedatelya-gostekhkomissii-rossii-ot-4-iyunya-1999-g-n-114 (дата обращения — 12.02.20);

12 Постановление Правительства Российской Федерации от 1 ноября 2012 г. N 1119 г. Москва "Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных" [Электронный ресурс] – URL: https://rg.ru/2012/11/07/pers-dannye-dok.html, (дата обращения — 14.02.20);

13 ООО «АМ Медиа». Недекларированные возможности [Электронный ресурс]: – URL: https://www.anti-malware.ru/threats/undeclared-capabilities, (дата обращения — 12.02.20);

14 Мстандарт. Контроль отсутствия недекларированных возможностей программного обеспечения [Электронный ресурс] – URL: https://www.mstandard.ru/certification/04/, (дата обращения — 14.02.20);

15 А. Марков, С. Миронов, В. Цирлов. Выявление уязвимостей в программном коде [Электронный ресурс] – URL: https://www.osp.ru/os/2005/12/380655/, (дата обращения – 16.02.20);

16 Темнов, О.Д. Анализ и исследование методов и средств обнаружения недекларированных возможностей [Электронный ресурс] – URL: https://cyberleninka.ru/article/n/analiz-i-issledovanie-metodov-i-sredstv-obnaruzheniya-nedeklarirovannyh-vozmozhnostey , (дата обращения – 18.04.20);

17 Осовецкий, Л. Г. Технология выявления недекларированных возможностей при сертификации промышленного программного обеспечения [Электронный ресурс]: - URL: https://cyberleninka.ru/article/n/tehnologiya-vyyavleniya-nedeklarirovannyh-vozmozhnostey-pri-sertifikatsii-promyshlennogo-programmnogo-obespecheniya-po/viewer, (дата обращения — 16.05.20);

18 ООО «АМ Медиа». Недекларированные возможности [Электронный ресурс]: – URL: https://www.anti-malware.ru/threats/undeclared-capabilities, (дата обращения — 01.06.20);

19 BMP (Bitmap Picture) Национальная библиотека им. Н. Э. Баумана [Электронный ресурс] – URL: https://ru.bmstu.wiki/BMP_(Bitmap_Picture) , (дата обращения – 07.06.20);

20 Формат данных BMP (bitmap) [Электронный ресурс] - URL: http://gamesmaker.ru/programming/graphics/format-dannyh-bmp-bitmap (дата последнего обращения — 29.05.20);

21 JPG [Электронный ресурс]: – URL: http://www.codenet.ru/progr/formt/jpeg_13.php, (дата обращения — 01.06.20).

22 Основы программирования на языках Си и C++ [Электронный ресурс] –URL: http://cppstudio.com/ , (дата обращения — 10.05.20);

23 В. Девисилов. Журнал «Охрана труда и техника безопасности на промышленных предприятиях». Психологические основы безопасности труда [Электронный ресурс] – URL: http://coyt.org/index.php/news/233-psikhofiziologicheskie-osnovy-bezopasnosti-truda , (дата обращения – 07.06.20);

24 А.А. Сухачев. «Охрана труда в строительстве». Учебник для среднего профессионального образования. Издательство КноРус, 2011. Психофизиологические основы безопасности труда [Электронный ресурс] – URL: http://www.kiout.ru/info/publish/15014 , (дата обращения – 07.06.20);

25 «Физическое воспитание студентов в техническом вузе» Учебное пособие В.Ф. Кошелев О.Ю. Малозёмов Ю.Г. Бердникова А.В. Минаев С.И. Филимонова [Электронный ресурс] – URL: https://core.ac.uk/download/pdf/42049582.pdf , (дата обращения – 07.06.20);

26 Автокорреляционная функция [Электронный ресурс] – URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%90%D0%B2%D1%82%D0%BE%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D1%8F , (дата обращения – 07.06.20);

27 ГОСТ Р ИСО/МЭК ТО 18044-2007 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Менеджмент инцидентов информационной безопасности [Электронный ресурс] – URL: http://docs.cntd.ru/document/1200068822 , (дата обращения – 07.06.20);

28 Ю.А. Есьман, Н.А. Егорова, «Обзор методов поиска недекларированных возможностей в программных средствах», сборник VII Ежегодной всероссийской межвузовской научно - практической конференции "Информационные технологии в науке и образовании. Проблемы и перспективы" (ВМНПК – 2020) [Электронный ресурс] – URL: https://yadi.sk/i/t4dQdKwG4lTAOg , (дата обращения – 07.06.20).

Приложение А

(обязательное)

Алгоритм шифрования файлов

Рисунок А.1 - Алгоритм работы функции `main()`. Лист 1

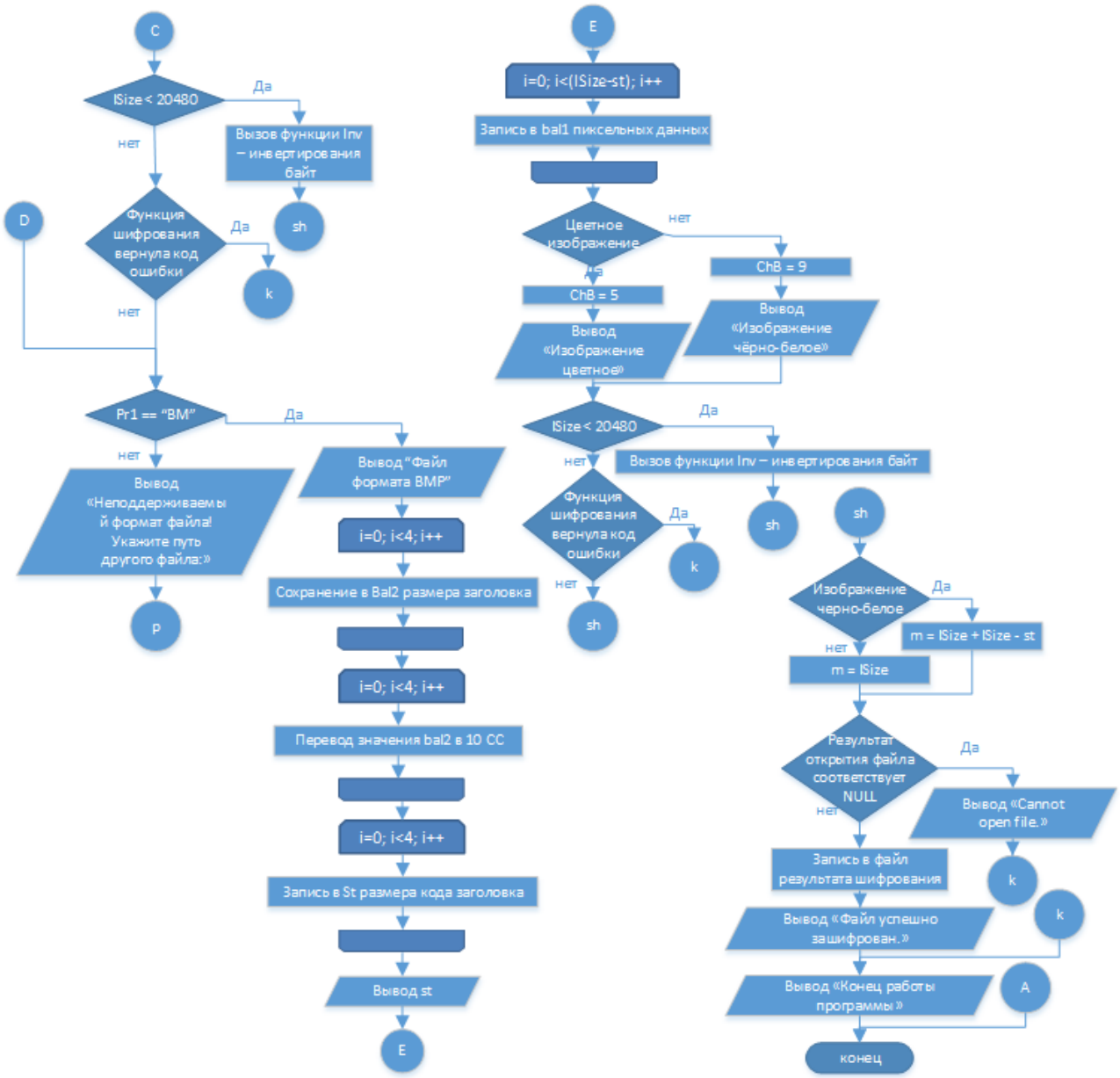

Рисунок А.1 - Алгоритм работы функции `main()`. Лист 2

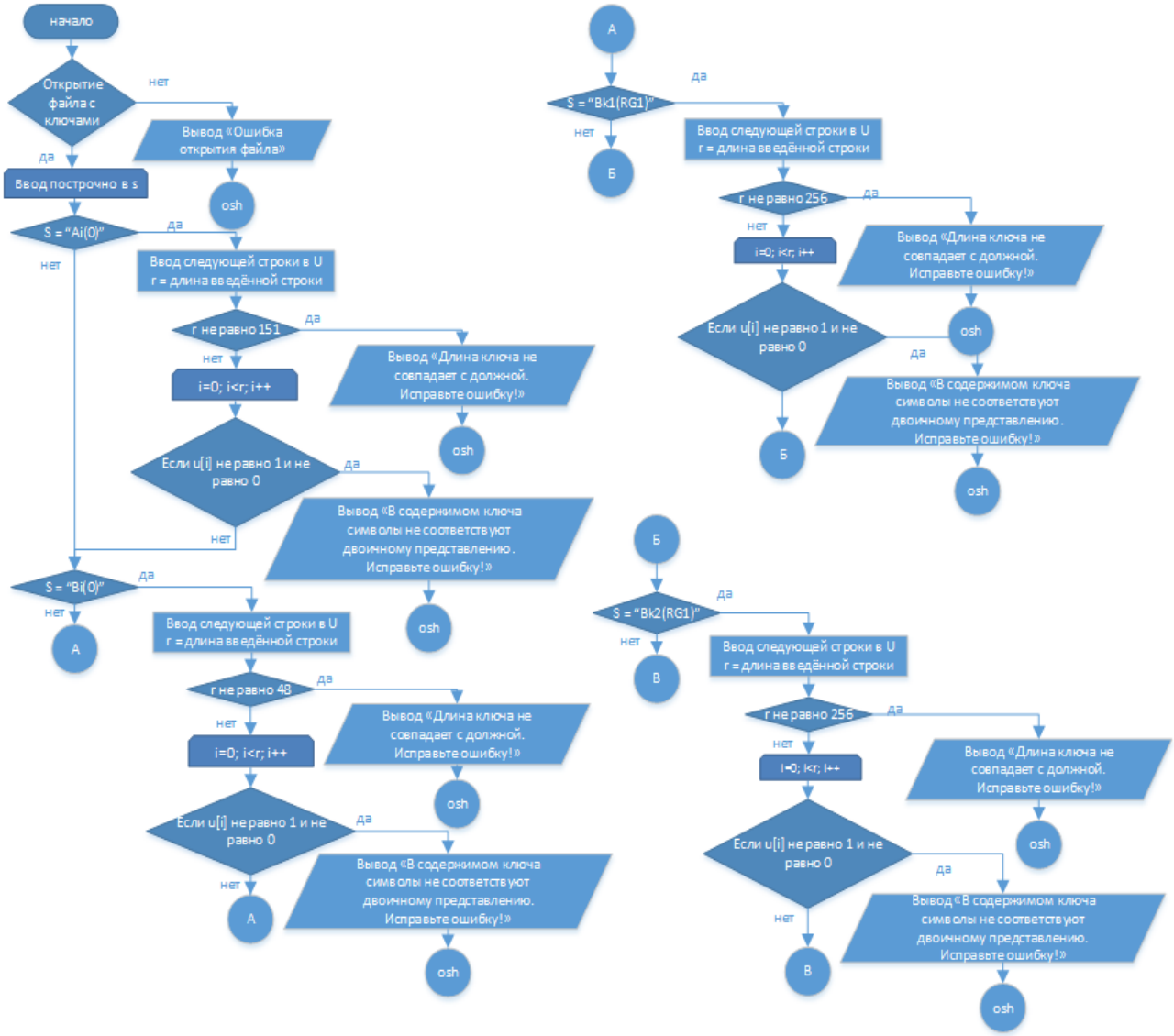

Рисунок А.2 - Алгоритм работы функции `int Pr_K()`. Лист 1

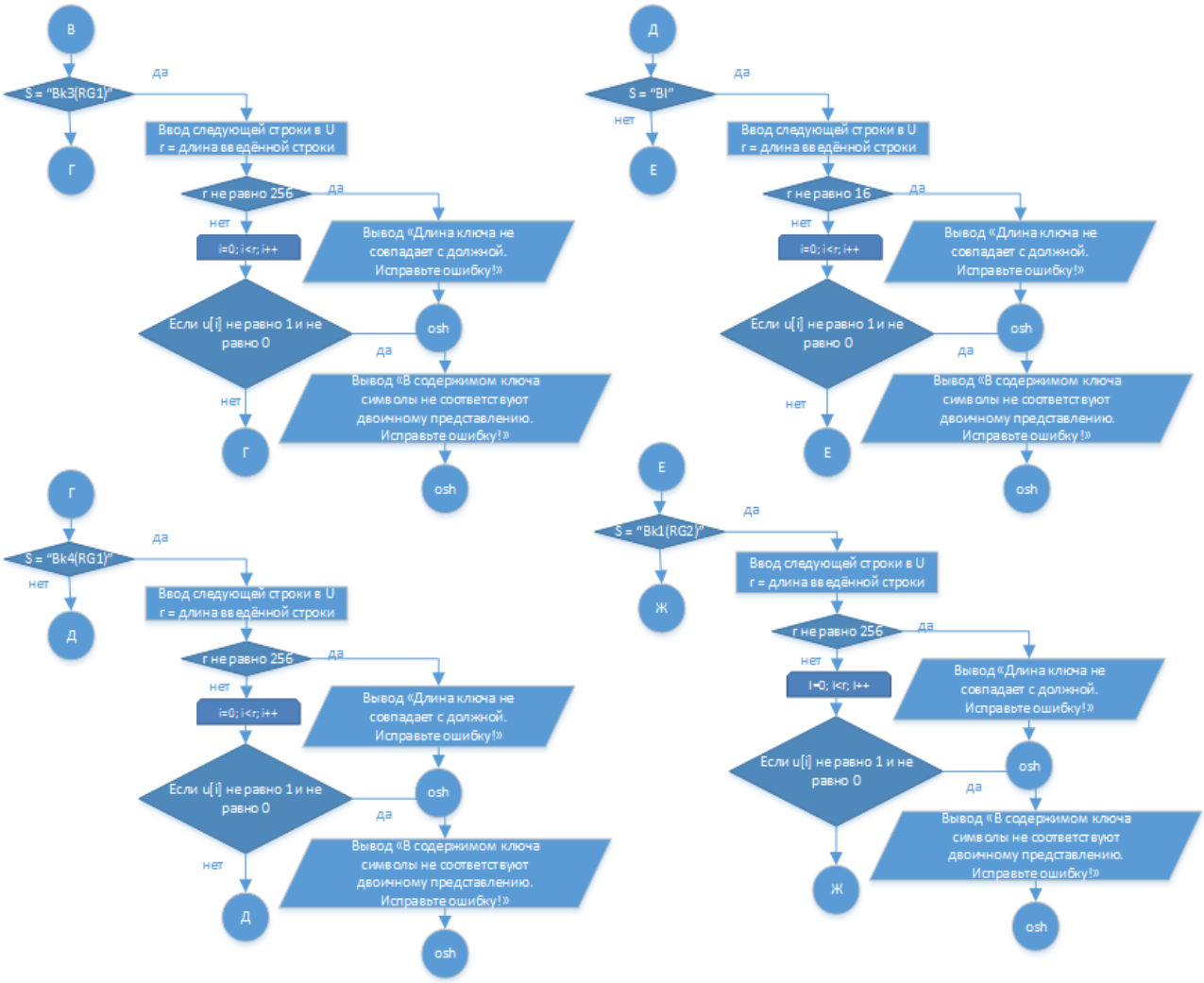

Рисунок А.2 - Алгоритм работы функции `int Pr_K()`. Лист 2

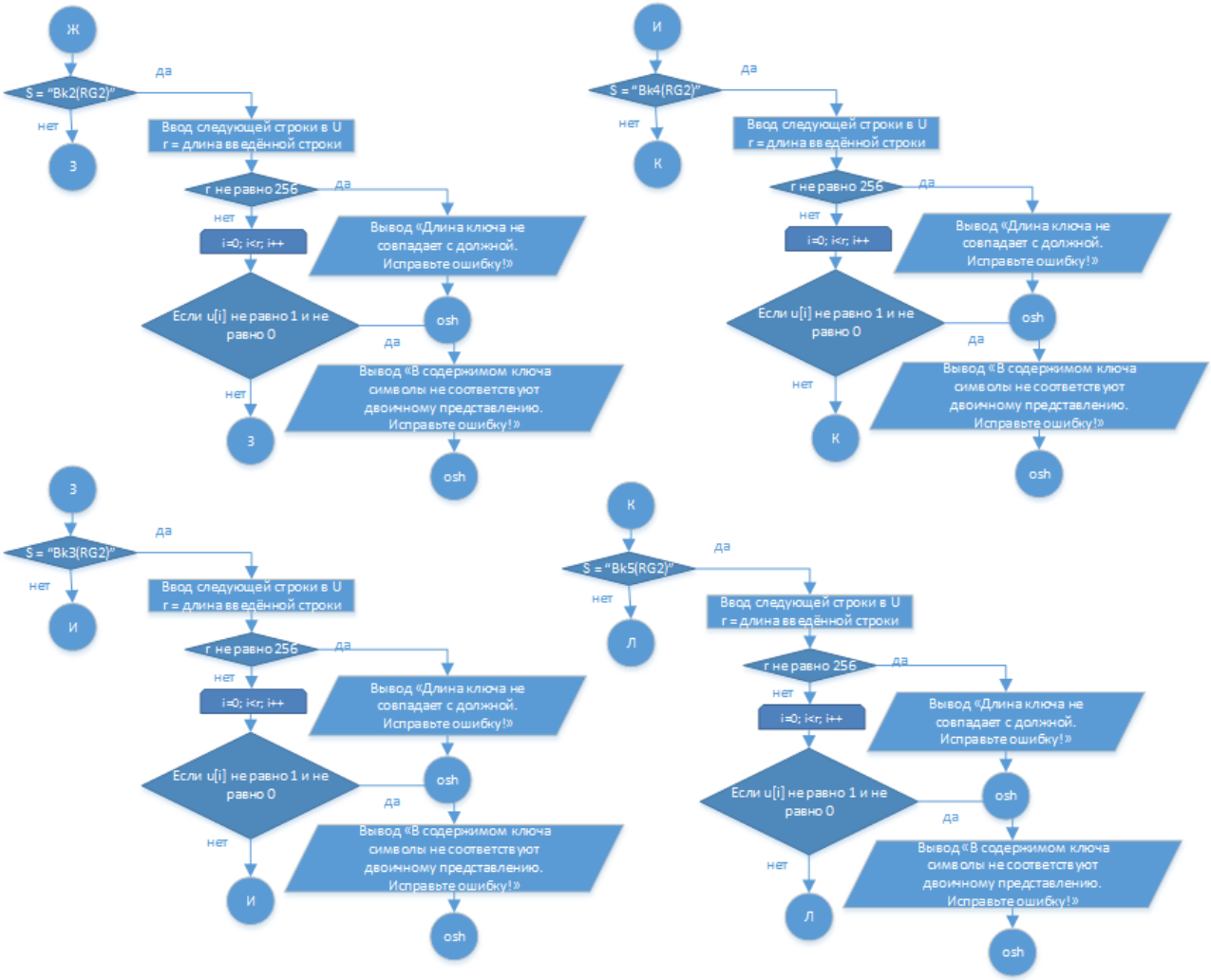

Рисунок А.2 - Алгоритм работы функции `int Pr_K()`. Лист 3

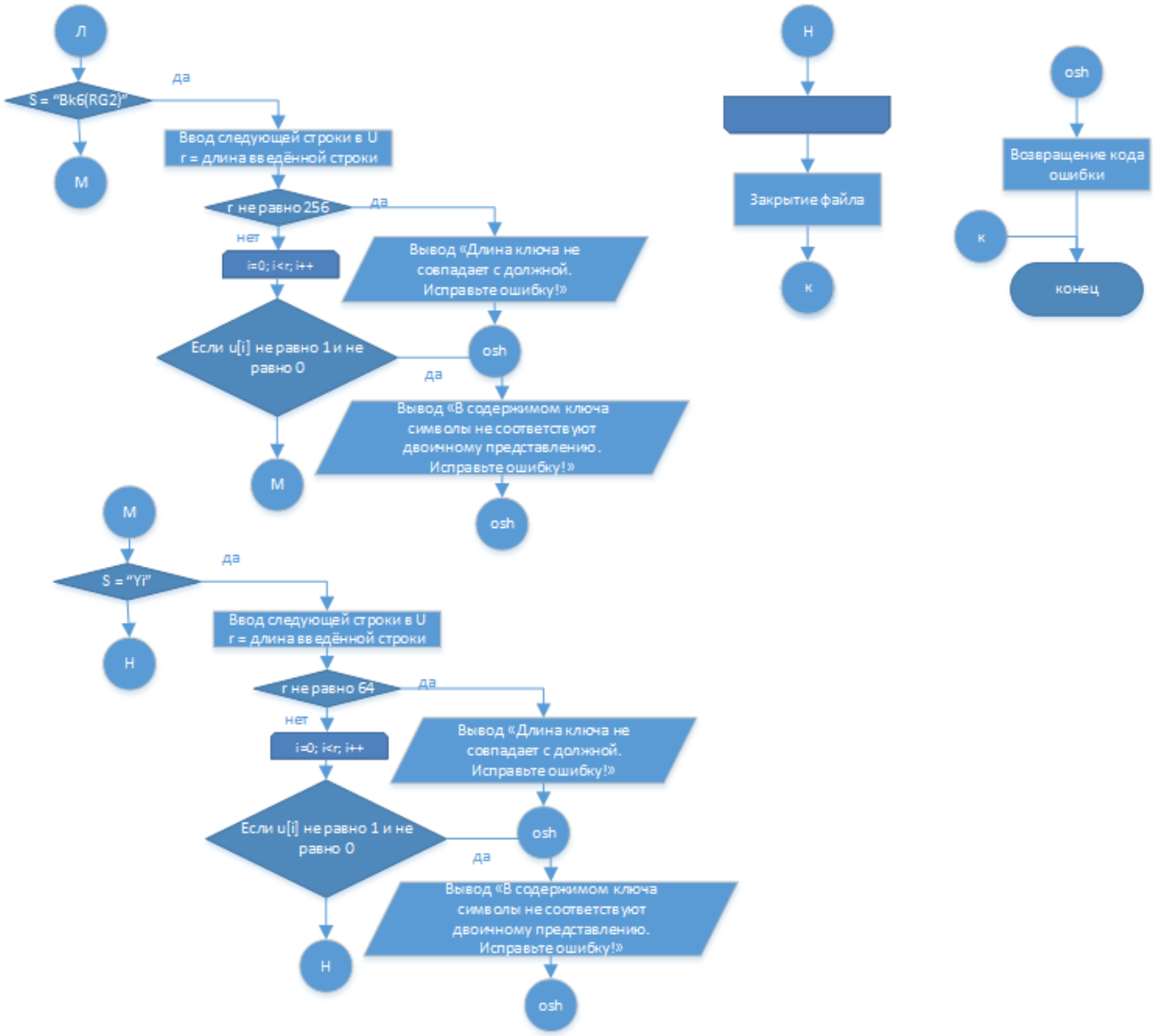

Рисунок А.2 - Алгоритм работы функции `int Pr_K()`. Лист 4

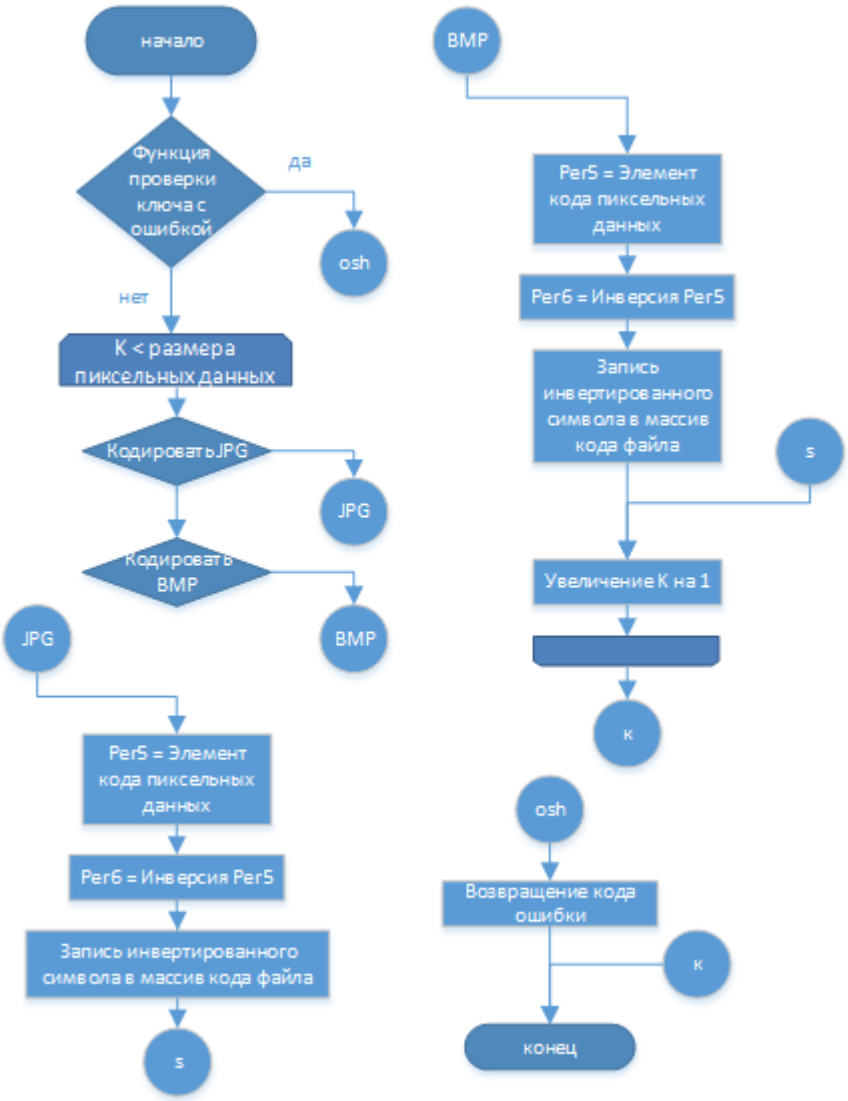

Рисунок А.3 - Алгоритм работы функции `int inv()`

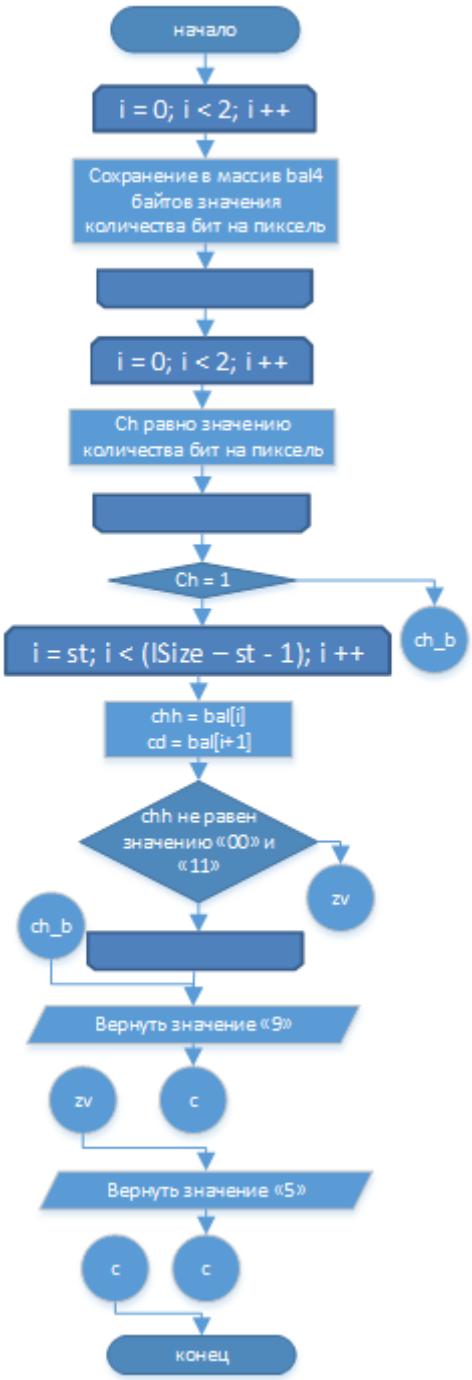

Рисунок А.4 - Алгоритм работы функции `int Pr_ChB()`

Приложение Б

(обязательное)

Алгоритм выработки гаммы шифра

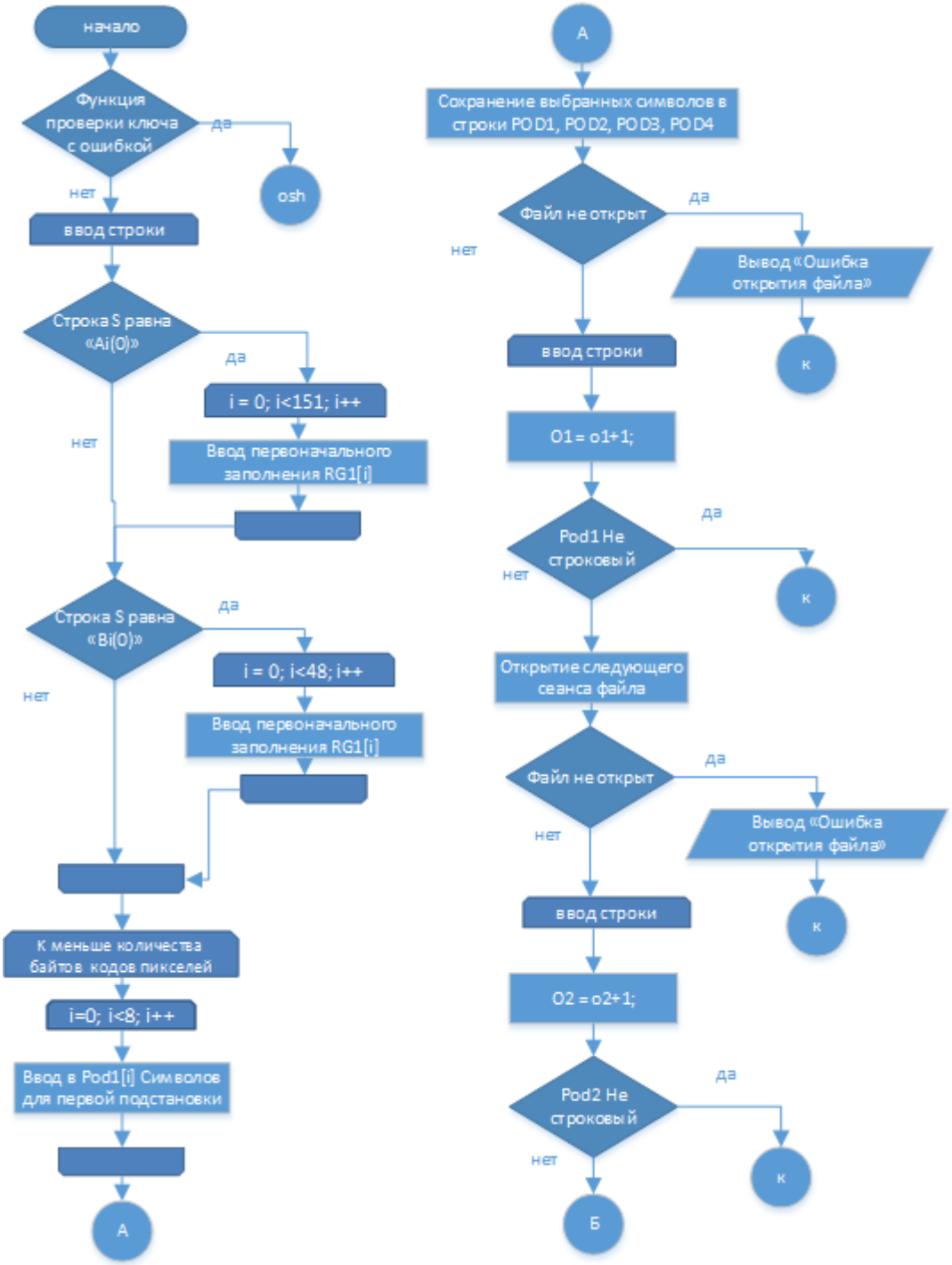

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 1

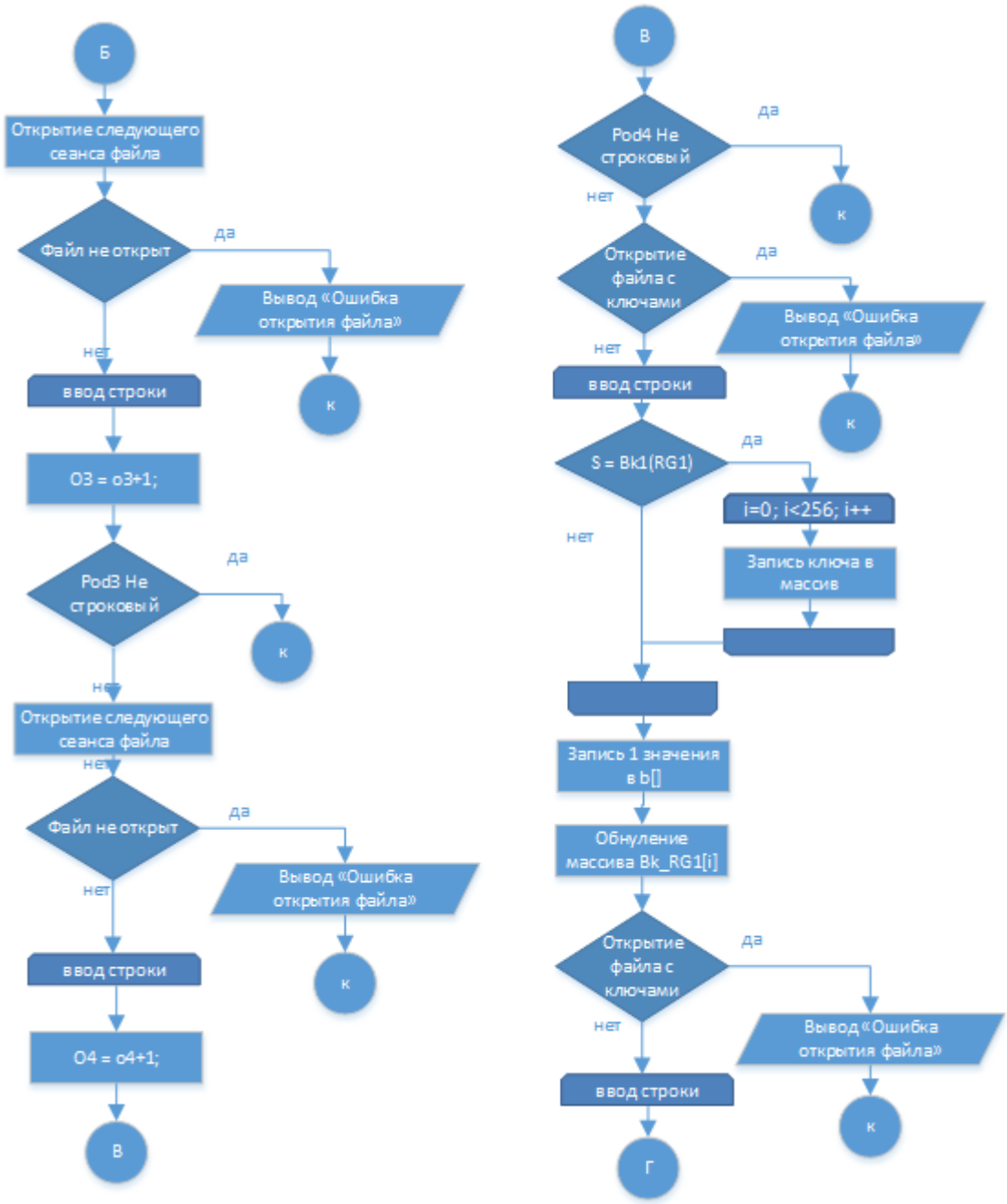

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 2

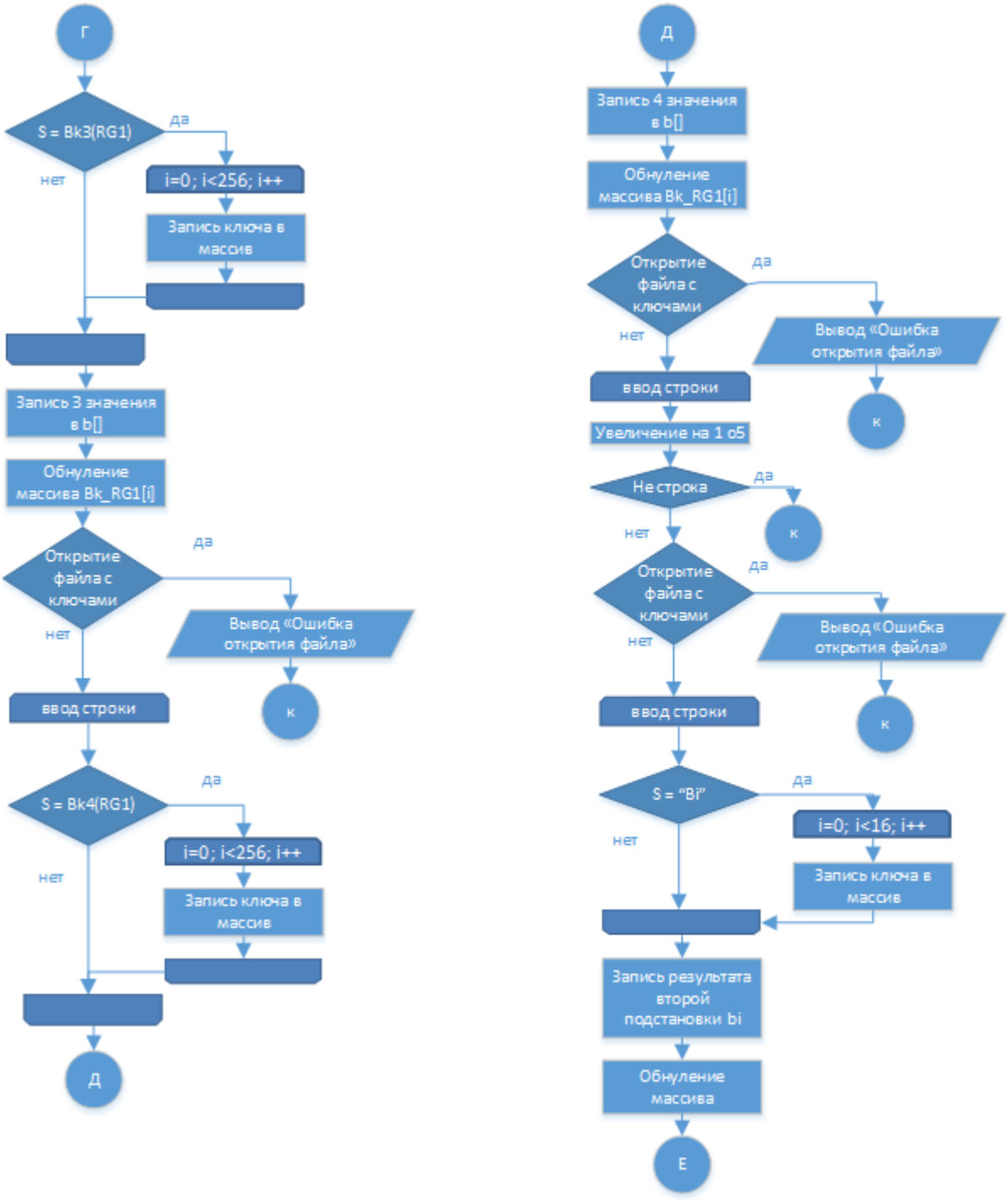

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 3

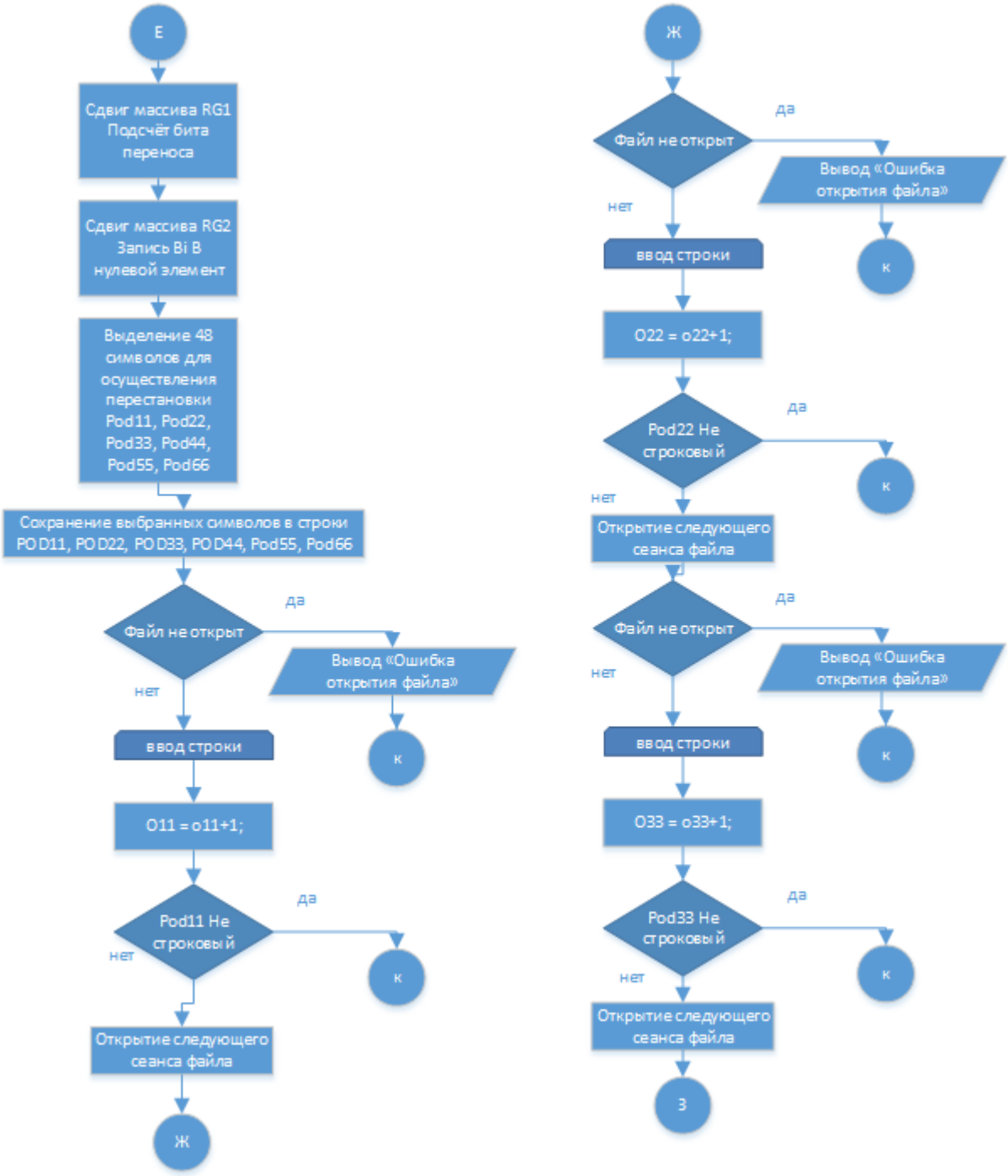

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 4

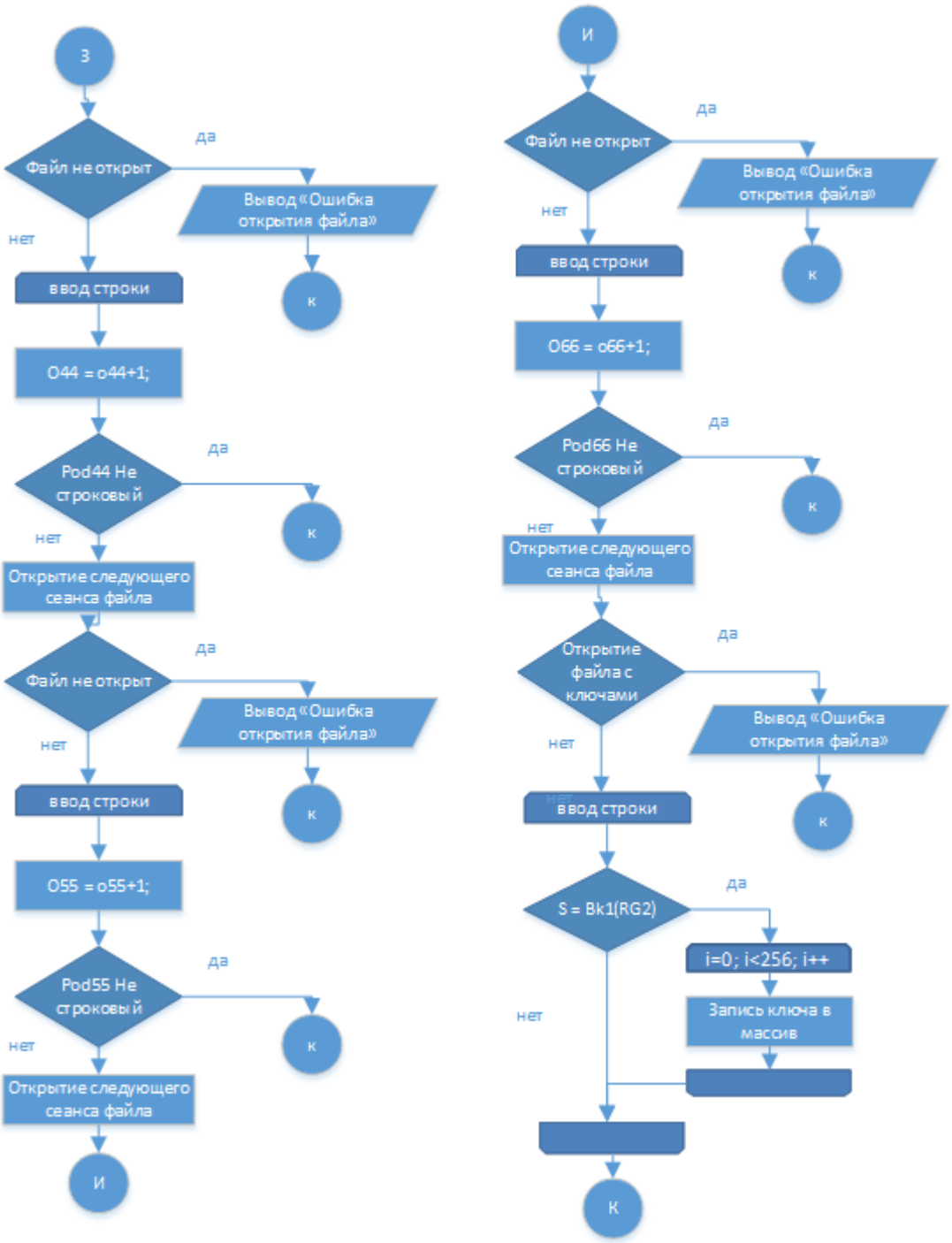

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 5

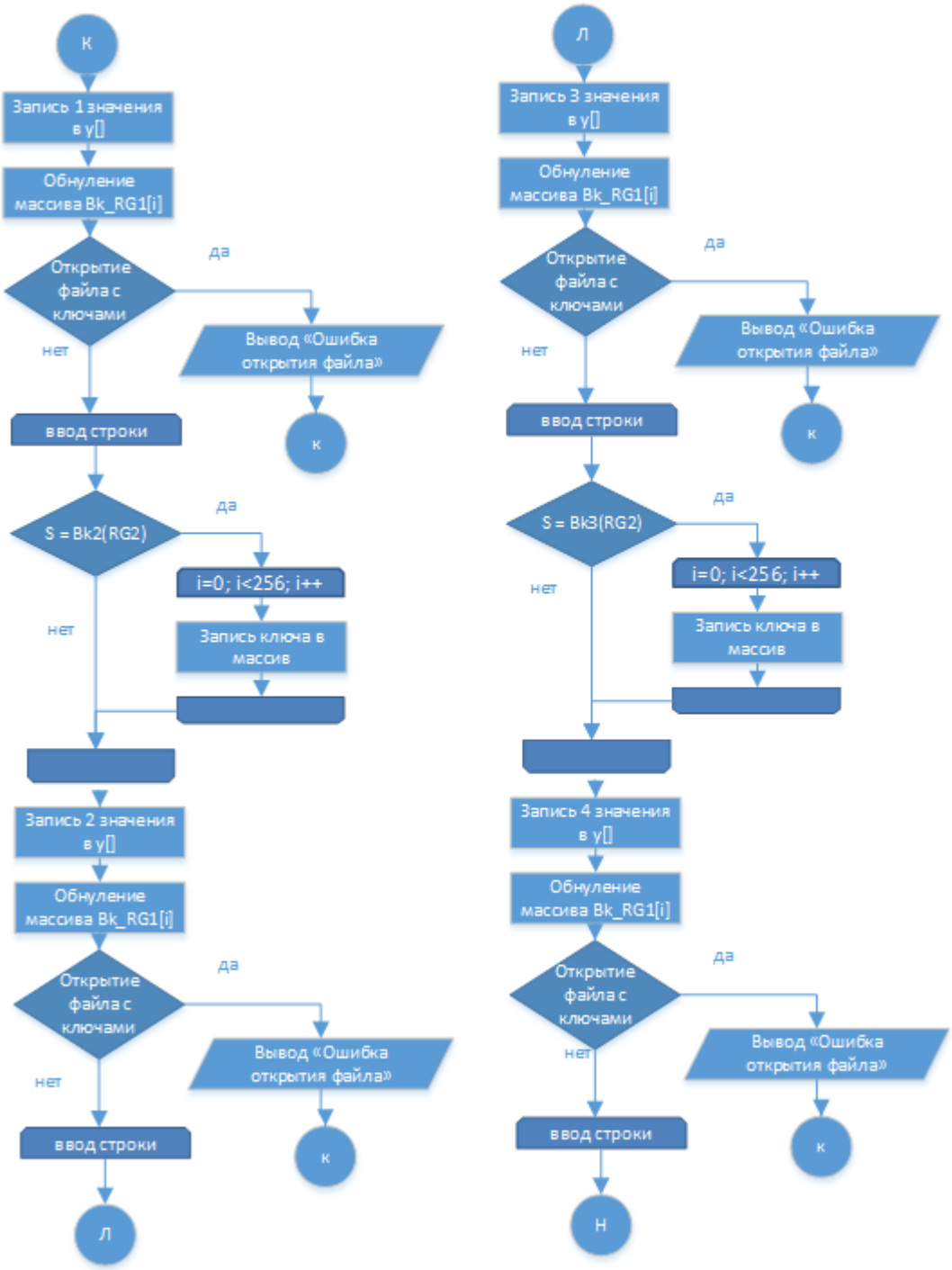

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 6

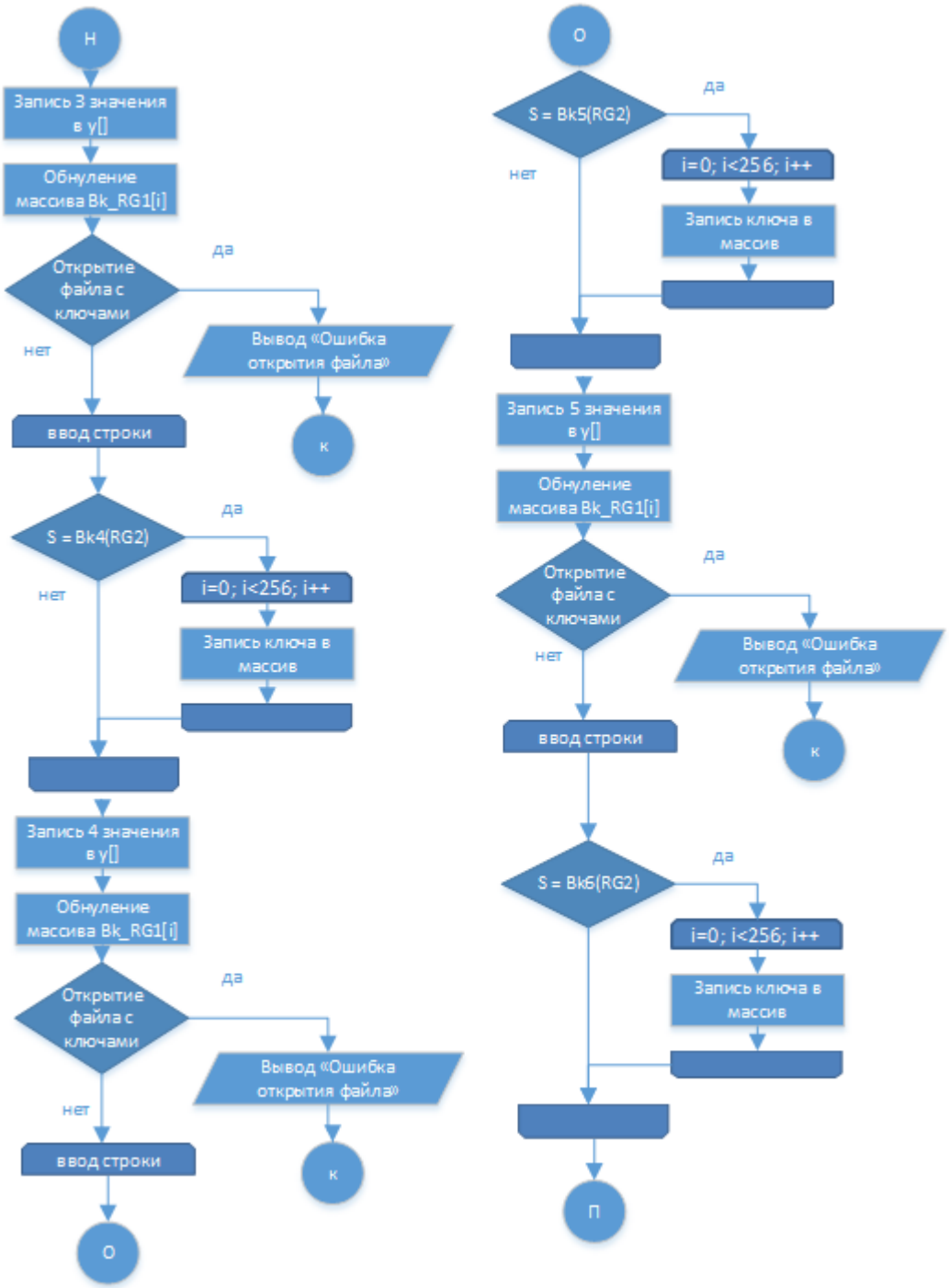

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 7

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 8

Рисунок Б.1 - Алгоритм работы функции `int sh()`. Лист 9

Приложение В

(обязательное)

Код программы шифрования изображений

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <clocale>
#include <iomanip>
#include <stdio.h>
#include <stdlib.h>
#pragma warning(disable : 4996)
using namespace std;

int Pr_ChB(char* bal, int* bal1, long  lSize, int st)
{
     string p = "key.txt";
     int i;
     int bal4[2];
     for (i = 0; i < 2; i++)
          bal4[i] = bal[i + 28];
     bal4[1] = bal4[1] * 256;
     int ch = 0;
     for (i = 0; i < 2; i++)
          ch += bal4[i];
     if (ch == 1) goto ch_b;
     for (i = st; i < (lSize - st - 1); i++)
     {
          int chh = 0;
          chh = bal[i];
          if ((chh != 0) && (chh != 15) && (chh != (-16)) && (chh
!= (-1))) goto zvet;
     }
ch_b:
     return(9);
     goto konez;
zvet:
     return(5);
     goto konez;
konez:
     {}
}

int Pr_K()
{
     string p = "key.txt";
     string s, u;
```

```
     int r, i;
     ifstream fin(p);
     if (!fin.is_open())
     {
          cout << "Ошибка открытия файла." << endl;
          goto oshibka;
     }
     while (fin >> s)
     {
          if (s == "Ai(0)")
          {
               fin >> u;
               r = u.length();
               if (r != 151)
               {
                    cout << "Длина ключа Ai(0) не совпадает с
должной. Исправьте ошибку!";
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Ai(0)
символы не соответствуют двоичному представлению. Исправьте
ошибку!";
                         goto oshibka;
                    }
               }
          }
          if (s == "Bi(0)")
          {
               fin >> u;
               r = u.length();
               if (r != 48)
               {
                    cout << "Длина ключа Bi(0) не совпадает с
должной. Исправьте ошибку!";
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Bi(0) символы
не соответствуют двоичному представлению. Исправьте ошибку!";
                         goto oshibka;
                    }
               }
          }
          if (s == "Bk1(RG1)")
```

```
          {
               fin >> u;
               r = u.length();
               if (r != 256)
               {
                    cout << "Длина ключа Bk1(RG1) не совпадает с
должной. Исправьте ошибку!";
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Bk1(RG1)
символы не соответствуют двоичному представлению. Исправьте
ошибку!";
                         goto oshibka;
                    }
               }
          }
          if (s == "Bk2(RG1)")
          {
               fin >> u;
               r = u.length();
               if (r != 256)
               {
                    cout << "Длина ключа Bk2(RG1) не совпадает с
должной. Исправьте ошибку!";
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Bk2(RG1)
символы не соответствуют двоичному представлению. Исправьте
ошибку!";
                         goto oshibka;
                    }
               }
          }
          if (s == "Bk3(RG1)")
          {
               fin >> u;
               r = u.length();
               if (r != 256)
               {
                    cout << "Длина ключа Bk3(RG1) не совпадает с
должной. Исправьте ошибку!";
                    goto oshibka;
               }
```

```
                for (i = 0; i < r; i++)
                {
                     if ((u[i] != '0') && (u[i] != '1'))
                     {
                          cout << "В содержимом ключа Bk3(RG1) символы не соответствуют двоичному представлению. Исправьте ошибку!";
                          goto oshibka;
                     }
                }
           }
           if (s == "Bk4(RG1)")
           {
                fin >> u;
                r = u.length();
                if (r != 256)
                {
                     cout << "Длина ключа Bk4(RG1) не совпадает с должной. Исправьте ошибку!";
                     goto oshibka;
                }
                for (i = 0; i < r; i++)
                {
                     if ((u[i] != '0') && (u[i] != '1'))
                     {
                          cout << "В содержимом ключа Bk4(RG1) символы не соответствуют двоичному представлению. Исправьте ошибку!";
                          goto oshibka;
                     }
                }
           }
           if (s == "Bi")
           {
                fin >> u;
                r = u.length();
                if (r != 16)
                {
                     cout << "Длина ключа Bi не совпадает с должной. Исправьте ошибку!";
                     goto oshibka;
                }
                for (i = 0; i < r; i++)
                {
                     if ((u[i] != '0') && (u[i] != '1'))
                     {
                          cout << "В содержимом ключа Bi символы не соответствуют двоичному представлению. Исправьте ошибку!";
                          goto oshibka;
                     }
                }
```

```
          }
          if (s == "Bk1(RG2)")
          {
               fin >> u;
               r = u.length();
               if (r != 256)
               {
                    cout << "Длина ключа Bk1(RG2) не совпадает с
должной. Исправьте ошибку!";
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Bk1(RG2)
символы не соответствуют двоичному представлению. Исправьте
ошибку!";
                         goto oshibka;
                    }
               }
          }
          if (s == "Bk2(RG2)")
          {
               fin >> u;
               r = u.length();
               if (r != 256)
               {
                    cout << "Длина ключа Bk2(RG2) не совпадает с
должной. Исправьте ошибку!";
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Bk2(RG2)
символы не соответствуют двоичному представлению. Исправьте
ошибку!";
                         goto oshibka;
                    }
               }
          }
          if (s == "Bk3(RG2)")
          {
               fin >> u;
               r = u.length();
               if (r != 256)
               {
                    cout << "Длина ключа Bk3(RG2) не совпадает с
должной. Исправьте ошибку!";
```

```
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Bk3(RG2) символы не соответствуют двоичному представлению. Исправьте ошибку!";
                         goto oshibka;
                    }
               }
          }
          if (s == "Bk4(RG2)")
          {
               fin >> u;
               r = u.length();
               if (r != 256)
               {
                    cout << "Длина ключа Bk4(RG2) не совпадает с должной. Исправьте ошибку!";
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Bk4(RG2) символы не соответствуют двоичному представлению. Исправьте ошибку!";
                         goto oshibka;
                    }
               }
          }
          if (s == "Bk5(RG2)")
          {
               fin >> u;
               r = u.length();
               if (r != 256)
               {
                    cout << "Длина ключа Bk5(RG2) не совпадает с должной. Исправьте ошибку!";
                    goto oshibka;
               }
               for (i = 0; i < r; i++)
               {
                    if ((u[i] != '0') && (u[i] != '1'))
                    {
                         cout << "В содержимом ключа Bk5(RG2) символы не соответствуют двоичному представлению. Исправьте ошибку!";
```

```
                    goto oshibka;
                }
            }
        }
        if (s == "Bk6(RG2)")
        {
            fin >> u;
            r = u.length();
            if (r != 256)
            {
                cout << "Длина ключа Bk6(RG2) не совпадает с должной. Исправьте ошибку!";
                goto oshibka;
            }
            for (i = 0; i < r; i++)
            {
                if ((u[i] != '0') && (u[i] != '1'))
                {
                    cout << "В содержимом ключа Bk6(RG2) символы не соответствуют двоичному представлению. Исправьте ошибку!";
                    goto oshibka;
                }
            }
        }
        if (s == "Yi")
        {
            fin >> u;
            r = u.length();
            if (r != 64)
            {
                cout << "Длина ключа Yi не совпадает с должной. Исправьте ошибку!";
                goto oshibka;
            }
            for (i = 0; i < r; i++)
            {
                if ((u[i] != '0') && (u[i] != '1'))
                {
                    cout << "В содержимом ключа Yi символы не соответствуют двоичному представлению. Исправьте ошибку!";
                    goto oshibka;
                }
            }
        }
    }
    fin.close();
    return(0);
    goto konez;
oshibka:
    return(4);
```

```
konez:
     {}
}

int inv(char* bal, int* bal1, long lSize, int st, int f)
{
     int k = 0;
     if (Pr_K() == 4) //проверка правильности ключа
          goto Oshibka;
     while (k < (lSize - st)) //цикл генерации гаммы шифра
     {
          char Per5, Per6;
          if (f == 2) goto sh_JPG;
          if (f == 1) goto sh_BMP;
sh_JPG:
          Per5 = bal1[k];
          Per6 = ~(Per5);
          bal1[k] = Per6;
          bal[k + st] = bal1[k];
          goto sled;
sh_BMP:
          Per5 = bal1[k];
          Per6 = ~(Per5);
          bal1[k] = Per6;
          bal[k + st] = bal1[k];
sled:
          k += 1;
     }
     return(0);
     goto konez;
Oshibka:
     return(5);
konez:
     {}
}

int sh(char * bal, int * bal1, long lSize, int st, int f, int ChB)
{
     int i, j, k = 0;
     string p = "key.txt"; //путь файла с ключами
     string t = "tab.txt"; //путь файла с таблицей первой
подстановки
     string sh = "tab16.txt"; //путь файла с таблицей второй
подстановки (генерация Bi)
     string yh = "tab32.txt"; //путь файла с таблицей второй
подстановки (генерация Yi)
     char RG1[151], RG2[48]; //регистры сдвига RG1 и RG2
     char Per1RG1, Per2RG1; //для реализации процедуры сдвига
регистра RG1
     char Pod1[8], Pod2[8], Pod3[8], Pod4[8]; //массивы для
реализации подстановки для RG1
```

```
     char Bk_RG1[256]; //массив для хранения ключа подстановки для
реализации подстановки для RG1
     char b[4]; //массив хранения промежуточного результата
перестановки для RG1
     char y[6], Yi[8];//массив хранения промежуточного результата
перестановки для RG2
     char bi; //переменная результата перестановки для RG1
     char Pod11[8], Pod22[8], Pod33[8], Pod44[8], Pod55[8],
Pod66[8];//массивы для реализации подстановки для RG2
     ifstream fin(p); //открытие файла для первоначального
заполнения регистров сдвига RG1 и RG2
     string s;
     if (Pr_K() == 4) //проверка правильности ключа
          goto Oshibka;
     while (fin >> s)
     {
          if (s == "Ai(0)")
          {
               for (i = 0; i < 151; i++)
               fin >> RG1[i];
          }
          if (s == "Bi(0)")
          {
               for (i = 0; i < 48; i++)
               fin >> RG2[i];
          }
     }//ввод первоначального заполнения регистров сдвига RG1 и RG2
     fin.close();
     while (k < (lSize-st)) //цикл генерации гаммы шифра
     {
          for (i = 0; i < 8; i++) //выделение 32 символов для
осуществления подстановки для RG1
               Pod1[i] = RG1[i + 119];
          for (i = 0; i < 8; i++)
               Pod2[i] = RG1[i + 127];
          for (i = 0; i < 8; i++)
               Pod3[i] = RG1[i + 135];
          for (i = 0; i < 8; i++)
               Pod4[i] = RG1[i + 143];
          string POD1(Pod1, sizeof(Pod1)); //сохранение выбранных
символов в строки для осуществления поиска в файле с таблицей
первой подстановки
          string POD2(Pod2, sizeof(Pod2));
          string POD3(Pod3, sizeof(Pod3));
          string POD4(Pod4, sizeof(Pod4));
          ifstream ffin1(t); //открытие файла с таблицей первой
подстановки
          if (!ffin1.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
```

```
          } //проверка открытия файла
          int o1 = 0; //счётчик для определения 1 номера строки
вхождения отобранных символов в таблице первой подстановки
          while (ffin1 >> s)
          {
               o1 = o1 + 1;
               if (s.find(POD1) != string::npos)
               {
                    break;
               }
          }
          ffin1.close();
          ifstream ffin2(t);
          if (!ffin2.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          int o2 = 0; //счётчик для оперделения 2 номера строки
вхождения отобранных символов в таблице первой подстановки
          while (ffin2 >> s)
          {
               o2 = o2 + 1;
               if (s.find(POD2) != string::npos)
               {
                    break;
               }
          }
          ffin2.close();
          ifstream ffin3(t);
          if (!ffin3.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          int o3 = 0; //счётчик для оперделения 3 номера строки
вхождения отобранных символов в таблице первой подстановки
          while (ffin3 >> s)
          {
               o3 = o3 + 1;
               if (s.find(POD3) != string::npos)
               {
                    break;
               }
          }
          ffin3.close();
          ifstream ffin4(t);
          if (!ffin4.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
```

```
          }//проверка открытия файла
          int o4 = 0; //счётчик для оперделения 4 номера строки
вхождения отобранных символов в таблице первой подстановки
          while (ffin4 >> s)
          {
               o4 = o4 + 1;
               if (s.find(POD4) != string::npos)
               {
                    break;
               }
          }
          ffin4.close();
          ifstream fin1(p); //открытие файла с ключами
          if (!fin1.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (fin1 >> s) //вывод в массив 1х значений ключа
промежуточной подстановки для RG1
          {
               if (s == "Bk1(RG1)")
               {
                    for (i = 0; i < 256; i++)
                    fin1 >> Bk_RG1[i];
                    break;
               }
          }
          fin1.close();
          b[0] = Bk_RG1[o1 - 1]; //запись 1 значения результата
промежуточной подстановки для RG1
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          ifstream fin2(p);//открытие файла с ключами
          if (!fin2.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (fin2 >> s)//вывод в массив 2х значений ключа
промежуточной подстановки для RG1
          {
               if (s == "Bk2(RG1)")
               {
                    for (i = 0; i < 256; i++)
                    fin2 >> Bk_RG1[i];
                    break;
               }
          }
          fin2.close();
```

```
          b[1] = Bk_RG1[o2 - 1];//запись 2 значения результата
промежуточной подстановки для RG1
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          ifstream fin3(p);//открытие файла с ключами
          if (!fin3.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (fin3 >> s)//вывод в массив 3х значений ключа
промежуточной подстановки для RG1
          {
               if (s == "Bk3(RG1)")
               {
                    for (i = 0; i < 256; i++)
                    fin3 >> Bk_RG1[i];
                    break;
               }
          }
          fin3.close();
          b[2] = Bk_RG1[o3 - 1];//запись 3 значения результата
промежуточной подстановки для RG1
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          ifstream fin4(p);//открытие файла с ключами
          if (!fin4.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (fin4 >> s)//вывод в массив 4х значений ключа
промежуточной подстановки для RG1
          {
               if (s == "Bk4(RG1)")
               {
                    for (i = 0; i < 256; i++)
                    fin4 >> Bk_RG1[i];
                    break;
               }
          }
          fin4.close();
          b[3] = Bk_RG1[o4 - 1];//запись 4 значения результата
промежуточной подстановки для RG1
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          string    B(b,    sizeof(b));//сохранение    результата
промежуточной подстановки для RG1 в строку для осуществления
поиска в файле с таблицей второй подстановки (генерация Bi)
          ifstream fffin(sh);//открытие файла с таблицей второй
подстановки (генерация Bi)
```

```
          if (!fffin.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          int o5 = 0;//счётчик для оперделения номера строки
вхождения результата промежуточной подстановки для RG1 в таблице
второй подстановки (генерация Bi)
          while (fffin >> s)
          {
               o5 = o5 + 1;
               if (s.find(B) != string::npos)
               {
                    break;
               }
          }
          fffin.close();
          ifstream fin5(p);//открытие файла с ключами
          if (!fin5.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (fin5 >> s)//вывод в массив значений ключа второй
подстановки (генерация Bi) для RG1
          {
               if (s == "Bi")
               {
                    for (i = 0; i < 16; i++)
                    fin5 >> Bk_RG1[i];
                    break;
               }
          }
          fin5.close();
          bi = Bk_RG1[o5 - 1];//сохранение результата второй
подстановки (генерация Bi) для RG1
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          Per1RG1 = RG1[150];//осуществление процедуры сдвига для
RG1
          for (i = 150; i > 0; i--)
               RG1[i] = RG1[i - 1];
          Per2RG1 = RG1[83];
          int Per1 = Per1RG1 - '0';
          int Per2 = Per2RG1 - '0';
          RG1[0] = (Per1 ^ Per2) + '0';
          for (i = 47; i > 0; i--)//осуществление процедуры сдвига
для RG2
               RG2[i] = RG2[i - 1];
          RG2[0] = bi;
```

```
          for (i = 0; i < 8; i++)//выделение 48 символов для
осуществления подстановки для RG2
               Pod11[i] = RG2[i];
          for (i = 0; i < 8; i++)
               Pod22[i] = RG2[i + 8];
          for (i = 0; i < 8; i++)
               Pod33[i] = RG2[i + 16];
          for (i = 0; i < 8; i++)
               Pod44[i] = RG2[i + 24];
          for (i = 0; i < 8; i++)
               Pod55[i] = RG2[i + 32];
          for (i = 0; i < 8; i++)
               Pod66[i] = RG2[i + 40];
          string POD11(Pod11, sizeof(Pod11));//сохранение выбранных
символов в строки для осуществления поиска в файле с таблицей
первой подстановки
          string POD22(Pod22, sizeof(Pod22));
          string POD33(Pod33, sizeof(Pod33));
          string POD44(Pod44, sizeof(Pod44));
          string POD55(Pod55, sizeof(Pod55));
          string POD66(Pod66, sizeof(Pod66));
          ifstream gfin1(t);//открытие файла с таблицей первой
подстановки
          if (!gfin1.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          int o11 = 0;//счётчик для определения 1 номера строки
вхождения отобранных символов в таблице первой подстановки
          while (gfin1 >> s)
          {
               o11 = o11 + 1;
               if (s.find(POD11) != string::npos)
               {
                    break;
               }
          }
          gfin1.close();
          ifstream gfin2(t);
          if (!gfin2.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          int o22 = 0;//счётчик для оперделения 2 номера строки
вхождения отобранных символов в таблице первой подстановки
          while (gfin2 >> s)
          {
               o22 = o22 + 1;
               if (s.find(POD22) != string::npos)
```

```
            {
                break;
            }
        }
        gfin2.close();
        ifstream gfin3(t);
        if (!gfin3.is_open())
        {
            cout << "Ошибка открытия файла." << endl;
            break;
        }//проверка открытия файла
        int o33 = 0;//счётчик для оперделения 3 номера строки
вхождения отобранных символов в таблице первой подстановки
        while (gfin3 >> s)
        {
            o33 = o33 + 1;
            if (s.find(POD33) != string::npos)
            {
                break;
            }
        }
        gfin3.close();
        ifstream gfin4(t);
        if (!gfin4.is_open())
        {
            cout << "Ошибка открытия файла." << endl;
            break;
        }//проверка открытия файла
        int o44 = 0;//счётчик для оперделения 4 номера строки
вхождения отобранных символов в таблице первой подстановки
        while (gfin4 >> s)
        {
            o44 = o44 + 1;
            if (s.find(POD44) != string::npos)
            {
                break;
            }
        }
        gfin4.close();
        ifstream gfin5(t);
        if (!gfin5.is_open())
        {
            cout << "Ошибка открытия файла." << endl;
            break;
        }//проверка открытия файла
        int o55 = 0;//счётчик для оперделения 5 номера строки
вхождения отобранных символов в таблице первой подстановки
        while (gfin5 >> s)
        {
            o55 = o55 + 1;
            if (s.find(POD55) != string::npos)
```

```
            {
                break;
            }
        }
        gfin5.close();
        ifstream gfin6(t);
        if (!gfin6.is_open())
        {
            cout << "Ошибка открытия файла." << endl;
            break;
        }//проверка открытия файла
        int o66 = 0;//счётчик для оперделения 6 номера строки
вхождения отобранных символов в таблице первой подстановки
        while (gfin6 >> s)
        {
            o66 = o66 + 1;
            if (s.find(POD66) != string::npos)
            {
                break;
            }
        }
        gfin6.close();
        ifstream gin1(p); //открытие файла с ключами
        if (!gin1.is_open())
        {
            cout << "Ошибка открытия файла." << endl;
            break;
        }//проверка открытия файла
        while (gin1 >> s) //вывод в массив 1х значений ключа
промежуточной подстановки для RG2
        {
            if (s == "Bk1(RG2)")
            {
                for (i = 0; i < 256; i++)
                    gin1 >> Bk_RG1[i];
                break;
            }
        }
        gin1.close();
        y[0] = Bk_RG1[o11 - 1]; //запись 1 значения результата
промежуточной подстановки для RG2
        for (i = 0; i < 256; i++)
            Bk_RG1[i] = 0;
        ifstream gin2(p);//открытие файла с ключами
        if (!gin2.is_open())
        {
            cout << "Ошибка открытия файла." << endl;
            break;
        }//проверка открытия файла
        while (gin2 >> s)//вывод в массив 2х значений ключа
промежуточной подстановки для RG2
```

```
          {
               if (s == "Bk2(RG2)")
               {
                    for (i = 0; i < 256; i++)
                         gin2 >> Bk_RG1[i];
                    break;
               }
          }
          gin2.close();
          y[1] = Bk_RG1[o22 - 1];//запись 2 значения результата
промежуточной подстановки для RG2
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          ifstream gin3(p);//открытие файла с ключами
          if (!gin3.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (gin3 >> s)//вывод в массив 3х значений ключа
промежуточной подстановки для RG2
          {
               if (s == "Bk3(RG2)")
               {
                    for (i = 0; i < 256; i++)
                         gin3 >> Bk_RG1[i];
                    break;
               }
          }
          gin3.close();
          y[2] = Bk_RG1[o33 - 1];//запись 3 значения результата
промежуточной подстановки для RG2
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          ifstream gin4(p);//открытие файла с ключами
          if (!gin4.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (gin4 >> s)//вывод в массив 4х значений ключа
промежуточной подстановки для RG2
          {
               if (s == "Bk4(RG2)")
               {
                    for (i = 0; i < 256; i++)
                         gin4 >> Bk_RG1[i];
                    break;
               }
          }
          gin4.close();
```

```
          y[3] = Bk_RG1[o44 - 1];//запись 4 значения результата
промежуточной подстановки для RG2
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          ifstream gin5(p);//открытие файла с ключами
          if (!gin5.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (gin5 >> s)//вывод в массив 5х значений ключа
промежуточной подстановки для RG2
          {
               if (s == "Bk5(RG2)")
               {
                    for (i = 0; i < 256; i++)
                         gin5 >> Bk_RG1[i];
                    break;
               }
          }
          gin5.close();
          y[4] = Bk_RG1[o55 - 1];//запись 5 значения результата
промежуточной подстановки для RG2
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          ifstream gin6(p);//открытие файла с ключами
          if (!gin6.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (gin6 >> s)//вывод в массив 6х значений ключа
промежуточной подстановки для RG2
          {
               if (s == "Bk6(RG2)")
               {
                    for (i = 0; i < 256; i++)
                         gin6 >> Bk_RG1[i];
                    break;
               }
          }
          gin6.close();
          y[5] = Bk_RG1[o66 - 1];//запись 6 значения результата
промежуточной подстановки для RG2
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          string    Y(y,    sizeof(y));//сохранение    результата
промежуточной подстановки для RG2 в строку для осуществления
поиска в файле с таблицей второй подстановки (генерация Yi)
          ifstream gffin(yh);//открытие файла с таблицей второй
подстановки (генерация Yi)
```

```
          if (!gffin.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          int o77 = 0;//счётчик для оперделения номера строки
вхождения результата промежуточной подстановки для RG2 в таблице
второй подстановки (генерация Yi)
          while (gffin >> s)
          {
               o77 = o77 + 1;
               if (s.find(Y) != string::npos)
               {
                    break;
               }
          }
          gffin.close();
          ifstream gin7(p);//открытие файла с ключами
          if (!gin7.is_open())
          {
               cout << "Ошибка открытия файла." << endl;
               break;
          }//проверка открытия файла
          while (gin7 >> s)//вывод в массив значений ключа второй
подстановки (генерация Bi) для RG1
          {
               if (s == "Yi")
               {
                    for (i = 0; i < 64; i++)
                         gin7 >> Bk_RG1[i];
                    break;
               }
          }
          gin7.close();
          Yi[6] = Bk_RG1[o77 - 1];//сохранение результата второй
подстановки (генерация Yi) для RG2
          Yi[7] = Bk_RG1[o77 - 1];//сохранение результата второй
подстановки (генерация Yi) для RG2
          for (i = 0; i < 6; i++)
               Yi[i] = y[i];
          for (i = 0; i < 256; i++)
               Bk_RG1[i] = 0;
          int Yk = 0;
          for (int i = 0; i < 8; i++)
          {
               Yk *= 2;
               Yk += Yi[i] - '0';   // '0'-'0' равно 0, '1'-'0'
Равно 1
          }
          int Per5, Per6;
          if (f == 2) goto sh_JPG;
```

```
          if (f == 1) goto sh_BMP;
sh_JPG:
          Per5 = bal1[k];
          Per6 = Per5 ^ Yk;
          bal1[k] = Per6;
          bal[k + st] = bal1[k];
          goto sled_Y;
sh_BMP:
          if (k < 6000)
          {
               for (j = 0; (j+k) < (lSize - st); j++)
               {
                    Per5 = bal1[k+j];
                    Per6 = Per5 ^ Yk;
                    bal1[k+j] = Per6;
                    bal[k + st+j] = bal1[k+j];
                    Per5 = 0;
                    Per6 = 0;
                    if (ChB == 9)
                    {
                         bal[k+j + lSize] = Yk;
                    }
                    j += 5999;
               }
          }
          else k = (lSize - st);
sled_Y:
          cout << endl << "k = "<<k;
          k = k + 1;//счётчик количества генерируемых гамм ключа
шифрования
     }
     if (k < (lSize - st)) goto Oshibka;
          return(0);
     goto konez;
Oshibka:
     return(5);
konez:
     {}
}

int main()
{
     setlocale(0, "");
     FILE* fp, * fp1;
     int i, j;
     int st = 0;
     long lSize;
     char p[20];
Povtor:
     cout << "Введите название файла, подлежащего шифрованию:";
     cin>>p;
```

```
     char d[20];
     cout << "Введите названия файла для сохранения результата:";
     cin >> d;
     if ((fp = fopen(p, "rb")) == NULL)
     {
          printf("Cannot open file.\n");
          exit(1);
     }//проверка открытия файла
     fseek(fp, 0, SEEK_END);//подсчёт необходимого размера массива
     lSize = ftell(fp);
     rewind(fp);
     cout << "Объём в байтах: " << lSize << endl;//вывод
необходимого размера массива
     int m = lSize + lSize - st;
     char* bal = new char[m];//создание массива необходимого
размера
     for (i = 0; i < (m); i++)
          bal[i] = 0;
     if (fread(bal, sizeof(char), lSize, fp) != lSize) //чтение
файла в массив
     {
          if (feof(fp))
               printf("Premature end of file.");
          else printf("File read error.");
     }
     fclose(fp);
     string Pr1;
     for (i = 0; i < 2; i++)
          Pr1 += bal[i];
     int ChB = 5;
     if (Pr1 == "яШ") //обработка файла формата JPG
     {
          cout << endl << "Файл формата JPG";
          int q;
          string SOS;
          j = 2;
          while (j < (lSize - 4))
          {
               for (i = j; i < (j + 2); i++)
                    SOS += bal[i];
               if (SOS == "яЪ")
               {
                    for (i = j; i < (j + 2); i++)
                         st += bal[i + 2];
                    q = j;
                    break;
               }
               SOS = "";
               j = j + 1;
          }
          st = st + q + 2;
```

```
          int* bal1 = new int[lSize - st-2];
          for (i = 0; i < (lSize - st-2); i++)//массив пиксельных
данных в 10 СС
               bal1[i] = bal[st + i];
          if (lSize < 20480)
          {
               inv(bal, bal1, lSize - 2, st, 2);
               goto shifr;
          }
          if (sh(bal, bal1, lSize-2, st, 2, 5)==5) goto conez;
     }
     else if (Pr1 == "BM")//обработка файла формата BMP
     {
          cout << endl << "Файл формата BMP";
          int bal2[4];
          for (i = 0; i < 4; i++)//получение положения пиксельных
данных относительно начала структуры файла
               bal2[i]= bal[i+10];
          for (i = 1; i < 4; i++)
               bal2[i] = bal2[i] * 256 * i;
          for (i = 0; i < 4; i++)
               st += bal2[i];
          int* bal1 = new int[lSize - st];
          for (i = 0; i < (lSize - st); i++)//массив пиксельных
данных в 10 СС
               bal1[i] = bal[st + i];
          if (Pr_ChB(bal, bal1, lSize, st) == 5)
          {
               ChB = 5;
               cout << endl << "Изображение цветное";
          }
          else
          {
               ChB = 9;
               cout << endl << "Изображение ч/б";
          }
          system("pause");
          if (lSize < 20480)
          {
               inv(bal, bal1, lSize, st, 1);
               goto shifr;
          }
          if (sh(bal, bal1, lSize, st, 1, ChB) == 5)
               goto conez;
          goto shifr;
     }
     Else
     {
          cout << endl << "Неподдерживаемый формат файла! Укажите
путь другого файла:";
          goto Povtor;
```

```
     }
shifr:
     if (ChB == 9)
          m = lSize + lSize - st;
     else
          m = lSize;
     if ((fp1 = fopen(d, "wb")) == NULL)
     {
          printf("Cannot open file.\n");
          goto conez;
     }//проверка открытия файла
     fwrite(bal, sizeof(char), m, fp1);
     fclose(fp1);
     cout << endl<<"Файл успешно защифрован!";
conez:
     cout << "Конец работы программы";
     delete[] bal;
     system("pause");
     return 0;
}
```