МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

И.о. зав. кафедрой «КБИС»

_____ О.А. Сафарьян
(подпись)

«___» _____ 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(дипломная работа)

Тема: «РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА ДЛЯ УНИЧТОЖЕНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ДАННЫХ В ОС WINDOWS»

Специальность 10.05.01 Компьютерная безопасность

Специализация Математические методы защиты информации

Обозначение ВКР 10.05.01.990000.000 Группа ВКБ61

Обучающийся _____ В.В. Следков
подпись, дата

Руководитель ВКР _____ доцент Н.Н. Язвинская
подпись, дата

Нормоконтроль ВКР _____ доцент Р.В. Егорова
подпись, дата

Ростов-на-Дону

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

И.о. зав. кафедрой «КБИС»
_____ О.А. Сафарьян
(подпись)
«___» _____ 2024 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

Тема: «РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА ДЛЯ УНИЧТОЖЕНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ДАННЫХ В ОС WINDOWS»

Обучающийся Следков Владислав Валерьевич

Обозначение ВКР 10.05.01.990000.000 Группа ВКБ61

Тема утверждена приказом по ДГТУ от _____ № _____

Срок представления ВКР к защите « » февраля 2024 г.

Исходные данные для выполнения выпускной квалификационной работы:

Федеральный закон от 27 июля 2006 г. N 149-ФЗ «Об информации, информационных технологиях и о защите информации».
Указ Президента РФ от 6 марта 1997 г. N 188 «Об утверждении перечня сведений конфиденциального характера».
ГОСТ Р 50739-95 «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования».
Руководящий документ Гостехкомиссии России «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации».
Материалы исследований, проводимых в ходе преддипломной практики.

Содержание выпускной квалификационной работы:

Введение:
Во введении необходимо: изложить актуальность выбранной темы, обозначить объект и предмет исследования, цель и задачи выпускной квалификационной работы, теоретическую и практическую значимость работы, структуру работы.

Наименование и краткое содержание разделов:
1 Теоретические сведения об особенностях хранения и удаления информации с накопителей различных типов.
Рассмотреть физическую и логическую структуры накопителей различных типов – твердотельных и жёстких дисков. Описать особенности хранения и удаления информации с накопителей различных типов. Привести методы уничтожения данных, подходящие для каждого из рассматриваемых типов накопителей.
2 Обзор существующих алгоритмов уничтожения данных.
Провести обзор наиболее популярных существующих алгоритмов уничтожения данных, применимых для накопителей на жестких магнитных дисках. Привести область применения, дату разработки, текущий статус, описать алгоритмы.
3 Обзор и анализ программных средств для безвозвратного удаления информации в ОС Windows.
Провести обзор и анализ наиболее популярных программ для безвозвратного удаления информации в операционной системе (ОС) Windows. Описать условия распространения программного обеспечения (ПО), информацию о разработчике, дату разработки, текущий статус поддержки ПО, поддерживаемые операционные системы, привести функционал программных средств и подробно рассмотреть поддерживаемые алгоритмы уничтожения данных. Выделить преимущества и недостатки каждого из рассматриваемых программных средств.
4 Алгоритмическая реализация.
Привести блок-схемы и описание алгоритмов работы программного средства.
5 Программная реализация.
Обосновать выбор средств разработки и языка программирования. Описать модули, классы и методы разрабатываемого программного средства. Разработать графический интерфейс, описать возможные входные и выходные данные. Привести пошаговый пример использования программного средства, а также проверку его работы.

Заключение:
Основные выводы о проделанной работы, оценка достижения цели.

Перечень графического и иллюстративного материалов:
Презентация выпускной квалификационной работы на тему «Разработка программного средства для уничтожения пользовательских данных в ОС Windows»

Руководитель ВКР _____ доцент Н.Н. Язвинская
подпись, дата

Задание принял к исполнению _____ В.В. Следков
подпись, дата

Аннотация

В данной выпускной квалификационной работе было разработано программное средство для уничтожения пользовательских данных в ОС Windows. Указанное программное средство устанавливается как приложение в ОС Windows, после чего становится возможным его использование. В рамках данной работы были рассмотрены особенности хранения и удаления информации с накопителей различных типов, а также существующие алгоритмы уничтожения данных. Проведен обзор и анализ программных средств для безвозвратного удаления информации в ОС Windows. Структурно в виде блок-схем представлены детали разработанного программного средства. Приведено описание работы программного средства, в результате которого можно сделать вывод о его применимости.

Объем работы – 82, количество иллюстраций – 30, приложений – 5, использовано информационных ресурсов – 17.

Annotation

In this final qualifying work, a software tool was developed for destroying user data in Windows OS. The specified software tool is installed as an application in Windows OS, after which it becomes possible to use it. As part of this work, the features of storing and deleting information from various types of drives, as well as existing data destruction algorithms, were examined. A review and analysis of software tools for permanently deleting information in Windows OS was carried out. Structurally, the details of the developed software are presented in the form of block diagrams. A description of the operation of the software is given, as a result of which one can draw a conclusion about its applicability.

Number of pages – 82, number of illustrations – 30, applications – 5, used information resources – 17.

Содержание

Введение

Вопрос безопасности и доступности данных на цифровых носителях с каждым годом становится все актуальнее. Информационные технологии развиваются постоянно, а с этим растет и необходимость в умелом обращении с данными. По оценкам International Data Corporation, к 2025 году глобальная сфера данных вырастет до 175 зеттабайт. С ростом объема данных также растет риск их кражи и утечки. Это означает, что безопасность данных имеет первостепенное значение, как для обычных пользователей, так и для компаний и государственных структур, и ее нельзя упускать из виду на любом этапе обработки данных. Если о безопасности данных не позаботиться, последствия могут быть тяжелыми, как с точки зрения юридических и финансовых издержек, так и с точки зрения репутационного ущерба. Во избежание таких ситуаций, необходимо предпринимать меры, нацеленные на то, чтобы устройства хранения данных не стали потенциальными источниками утечек информации в течение их жизненного цикла, когда они переходят из рук в руки или достигают конца своего срока службы. Многие случайные утечки данных происходят, когда пользователи считают, что они удалили все конфиденциальные данные, но на самом деле их можно восстановить с помощью технических и криминалистических методов [1]. Таким образом, сказанное выше обосновывает актуальность разработки.

Целью выпускной квалификационной работы является разработка программного средства для уничтожения пользовательских данных в ОС Windows. Цель определила следующие задачи:

– проанализировать особенности хранения и безвозвратного удаления информации с накопителей различных типов;

– провести обзор существующих алгоритмов уничтожения данных;

– выполнить обзор и анализ программных средств для безвозвратного удаления информации в ОС Windows;

– разработать алгоритм работы программного средства;

– выполнить программную реализацию.

Объектом исследования является процесс безвозвратного удаления данных с накопителей различных типов в ОС Windows.

Предметом исследования являются методы безвозвратного удаления пользовательских данных.

Методологическая основа работы включает следующие научные методы: анализ, дедукция, сравнение и описание.

1 Теоретические сведения об особенностях хранения и удаления информации с накопителей различных типов

На данный момент все носители информации делятся на энергозависимые и энергонезависимые. Разница между ними заключается в том, что энергозависимые, в отличие от энергонезависимых, не способны сохранять данные при отсутствии электроэнергии. Именно поэтому в рамках данной работы будут рассматриваться энергонезависимые носители.

Удаление информации с помощью обычных средств операционной системы (ОС) не приводит к бесследному удалению информации, наоборот, удаленная информация остается в том же самом виде на носителе до тех пор, пока память не будет перезаписана и может быть восстановлена с помощью различных алгоритмов и определенных программ, которые можно найти в открытом доступе. В связи с этим, встает вопрос удаления информации таким образом, чтобы её невозможно было восстановить, используя какие-либо алгоритмы, при этом, сохраняя работоспособность носителя, на котором информация была сохранена.

1.1 Особенности жёстких дисков

Описывая особенности хранения и удаления информации с носителей данных типа «жёсткий диск», стоит рассмотреть структуру данного типа накопителей. HDD – накопитель на жёстких магнитных дисках (жёсткий диск) – устройство хранения информации, основанное на принципе магнитной записи. Информация в них записывается на жёсткие пластины. Физическая структура жёсткого диска представлена на рисунке 1.1. С целью адресации пространство поверхности пластин диска делится на дорожки – концентрические кольцевые области. Каждая дорожка делится на равные отрезки – секторы. Сектор – минимальная адресуемая единица хранения информации на дисковых запоминающих устройствах.

Рисунок 1.1 – Физическая структура жёсткого диска

Для более эффективного использования места на диске файловая система может объединять секторы в кластеры. Кластер представляет собой логическую единицу хранения данных в таблице размещения файлов, которая объединяет группу секторов. Например, на дисках с размером секторов в 512 байт, 512-байтный кластер содержит один сектор, тогда как 4-килобайтный кластер содержит восемь секторов. Как правило, это наименьшее место на диске, которое может быть выделено для хранения файла [2]. Логическая структура жёсткого диска представлена на рисунке 1.2.

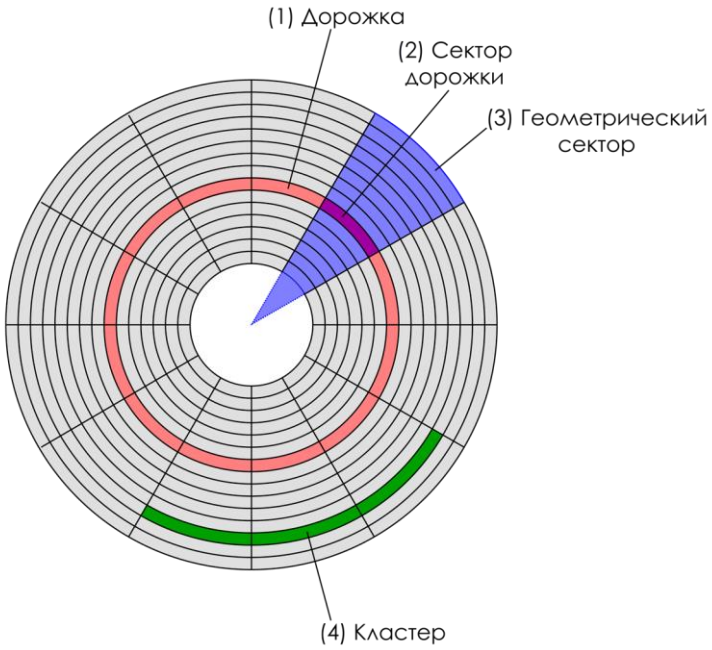


Рисунок 1.2 – Логическая структура жёсткого диска

Как было отмечено ранее, простое удаление информации с помощью стандартных средств операционной системы не приводит к полному удалению данных, а лишь удаляет прямые указатели на секторы диска с данными, что делает возможным их восстановление с помощью обычных программных инструментов. Для обеспечения более надежного удаления информации необходимо либо перезаписывать области памяти, занимаемые удаляемыми данными, используя специальные алгоритмы уничтожения информации, либо применять более радикальные методы, такие как размагничивание или физическое уничтожение накопителя.

1.2 Особенности твердотельных накопителей

В отличие от магнитных дисков, данные на которых записываются более или менее последовательно, в SSD накопителях данные разбиваются на блоки, которые записываются на нескольких чипах флэш-памяти NAND параллельно. Но даже в рамках одной микросхемы данные не сохраняются в линейной последовательности. Связано это с механизмом переадресации блоков.

Переадресация физических блоков NAND была введена для борьбы с преждевременным износом ячеек. Стандартная адресация блоков приводила бы к быстрому выходу из строя часто перезаписываемых ячеек, что могло привести к потере данных. Чтобы бороться с неравномерным износом, производители применяют сложные схемы балансирования нагрузки. Контроллер диска отслеживает количество записей в каждую физическую ячейку и подменяет адреса ячеек так, чтобы запись происходила в ячейки с наименьшим износом. Таким образом, запись блока данных по некоторому "физическому" адресу фактически может быть осуществлена в любую ячейку на любой из микросхем памяти [3].

Ячейка – базовый элемент хранения данных в SSD. В зависимости от типа памяти, установленной в SSD, одна ячейка может содержать один, два, три или четыре бита информации. Помимо ограниченного ресурса записи отдельных

ячеек, у NAND-памяти, используемой в SSD, есть ещё одна особенность. Запись в ячейку занимает больше времени, чем чтение, но повторная запись в уже записанную ячейку будет медленнее в разы. Связано это с тем, что для записи новых данных необходимо использовать пустую ячейку. Если ячейка уже содержит данные, то перед её перезаписью данные нужно удалить, полностью очистив ячейку. Процесс очистки ячейки весьма медленный, к тому же невозможно очистить только одну ячейку.

В отличие от магнитных накопителей, где минимальной адресуемой единицей хранения информации является сектор, в памяти NAND используются "страницы", которые объединяются в "блоки". Страница представляет собой группу байтов и как правило имеет следующий размер в байтах: 528, 2112, 4320, 8640, 9216, 18592 и т.д. Страница – всегда минимальный объем байт, который возможно считать за один такт. Даже если требуется прочитать всего один байт информации, всё равно будет считана целая страница, включая нужный байт. Однако если чтение происходит постранично, то записать данные можно только в целый блок. Блок – это группа страниц. Обычно размер блока составляет 64, 128, 256 или 512 страниц. Контроллеры SSD считывают данные страницами, а записывают и стирают блоками [3]. Логическая структура твердотельного накопителя представлена на рисунке 1.3.

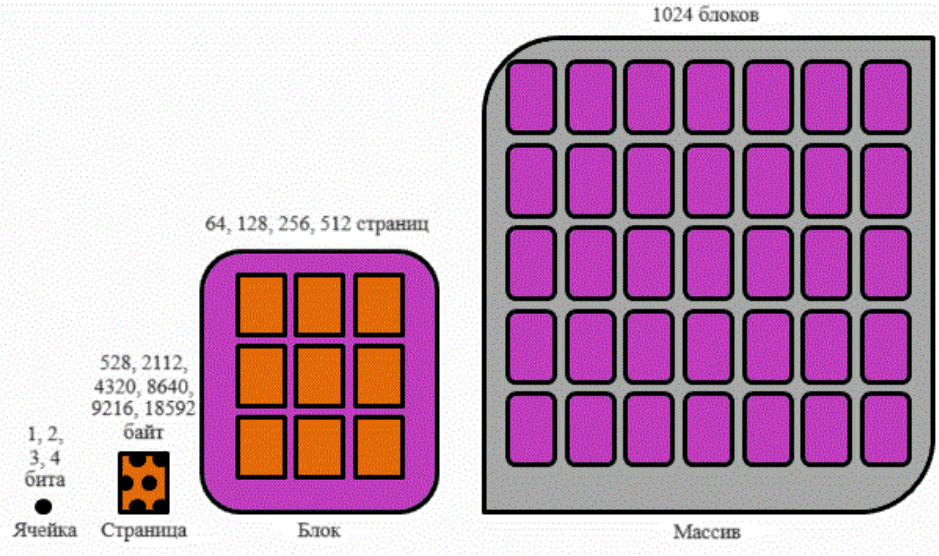


Рисунок 1.3 – Логическая структура твердотельного накопителя

Как было отмечено ранее, из-за ограниченного количества циклов записи в память NAND, контроллер SSD стремится минимизировать повторное использование блоков. При записи новых данных контроллер SSD использует блоки с наименьшим количеством перезаписей. На практике это означает то, что в процессе записи новой порции данных в блок с определённым адресом контроллер SSD осуществит моментальную подмену адресов: нужный «физический» адрес будет назначен другому, незанятому блоку, а ранее записанный блок получит другой адрес или вовсе уйдёт в неадресуемый резервный пул. Контроллер пометит блок как неиспользуемый для последующей очистки, тем самым подготовив блок к записи новых данных.

Фактическая ёмкость любого SSD накопителя как минимум на 5-10% больше, чем указано в спецификациях. Эта дополнительная память используется в качестве резервного пула, где хранятся запасные ячейки для замены поврежденных блоков и ускорения процесса записи. В резервной неадресуемой области могут остаться блоки с актуальными данными. Однако доступ к этим блокам через стандартные команды SATA невозможен, поэтому восстановление информации из этой области также невозможно [3].

Вследствие рассмотренных особенностей хранения данных на твердотельных накопителях применение специальных алгоритмов уничтожения информации нецелесообразно. При попытке перезаписать область памяти, отведенную для удаляемых данных, контроллер SSD осуществит подмену адресов – новые данные запишутся в другой блок с «подмененным» адресом, а блок с информацией, подлежащей удалению, будет помечен неиспользуемым и останется неизменённым.

Проблема с уничтожением данных на SSD-накопителях решается с помощью механизма фоновой очистки данных, известного как Trim. Этот механизм работает совместно с операционной системой. Когда пользователь удаляет файл, форматирует диск или создает новый раздел, операционная система передает контроллеру SSD информацию о том, какие ячейки больше не

содержат полезных данных и могут быть очищены. В результате работы Trim сама операционная система не перезаписывает эти блоки и не стирает информацию. Она лишь передает контроллеру массив адресов ячеек, которые больше не содержат полезной информации. С этого момента контроллер может начать фоновый процесс удаления данных из этих ячеек. Дальнейшая работа контроллера SSD не зависит от действий пользователя или операционной системы: алгоритмы контроллера начнут очистку ненужных блоков и продолжат её, даже если SSD будет извлечен из компьютера и установлен в другой. Не поможет и специализированное устройство для блокировки записи. Механизм Trim поддерживается на уровне операционной системы начиная с Windows 7 и только при соблюдении ряда условий: диск должен быть подключен напрямую (через интерфейсы SATA или NVMe), в качестве файловой системы тома должен быть NTFS, Trim должны поддерживать драйверы и BIOS компьютера [4].

Вывод

Таким образом, для безвозвратного удаления данных с жёстких дисков необходимо либо перезаписывать области памяти, занимаемые удаляемыми данными, используя специальные алгоритмы уничтожения информации, либо применять более радикальные методы, такие как размагничивание или физическое уничтожение накопителя. В случае твердотельных накопителей следует использовать механизм Trim либо более радикальный метод – физическое уничтожение накопителя.

2 Обзор существующих алгоритмов уничтожения данных

Уничтожение данных – последовательность операций, предназначенных для осуществления программными или аппаратными средствами необратимого удаления данных, в том числе остаточной информации. Как правило, уничтожение данных используется государственными учреждениями, прочими специализированными структурами и предприятиями в целях сохранения какого-либо из видов тайн. Алгоритмы уничтожения информации стандартизированы, во многих государствах изданы национальные стандарты, нормы и правила, регламентирующие использование программных средств для уничтожения информации и описывающие механизмы их реализации. Существует широкий спектр доступных программных средств безопасного уничтожения данных, в том числе программы с открытым исходным кодом. Уничтожение данных используется также в средствах программного шифрования информации для безопасного удаления временных файлов и уничтожения исходных, поскольку, используя классическое удаление, существует возможность восстановления исходного файла.

Все программные реализации алгоритмов уничтожения данных основаны на простейших операциях записи, тем самым происходит многократная перезапись информации в секторах жесткого диска ложными данными. В зависимости от алгоритма, для этой цели могут использоваться псевдослучайные числа либо фиксированные значения. Существующие алгоритмы могут содержать от одного до 35 и более циклов перезаписи, а также предоставлять возможность выбора количества циклов. Теоретически, простейшим методом уничтожения исходного файла является его полная перезапись байтами #FF, #00 либо другими произвольными байтами, что делает невозможным восстановление информации с помощью стандартных программных средств, доступных пользователю. Однако существует вероятность восстановления исходной информации при использовании специализированных аппаратных средств, которые анализируют поверхность магнитных и других носителей информации и позволяют восстановить данные на основе остаточной намагниченности или других показателей. В связи с этим и были разработаны специальные алгоритмы уничтожения данных [5].

Как было отмечено ранее, описываемые алгоритмы применимы для накопителей типа «жёсткий диск». В случае работы с твердотельными накопителями, данные алгоритмы не актуальны по причине особенностей хранения и удаления данных на носителях данного типа.

Рассмотрим детально некоторые алгоритмы уничтожения данных.

В Российской Федерации существует два документа, описывающие методы безвозвратного удаления данных – ГОСТ Р 50739-95 «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования» и руководящий документ Гостехкомиссии России «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации». В ГОСТ не описан конкретный алгоритм удаления, содержится лишь следующая формулировка: «Очистка должна производиться путём записи маскирующей информации в память при её освобождении (перераспределении)» [6].

Согласно руководящему документу Гостехкомиссии России:

- «Очистка осуществляется двукратной произвольной записью в освобождаемую область памяти, ранее использованную для хранения защищаемых данных (файлов)» – для автоматизированных систем (АС) с классами защищённости 3А, 2А;
- «Очистка осуществляется однократной произвольной записью в освобождаемую область памяти, ранее использованную для хранения защищаемых данных (файлов)» – для АС с классом защищённости 1Г;
- «Очистка осуществляется двукратной произвольной записью в любую освобождаемую область памяти, использованную для хранения защищаемой информации» – для АС с классами защищённости 1В, 1Б;
- «Очистка осуществляется двукратной произвольной записью в любую освобождаемую область памяти, в которой содержалась защищаемая информация» – для АС с классом защищённости 1А [7].

На данный момент, оба документа имеют действующий статус.

NZSIT 402 – стандарт программного уничтожения данных, принятый Службой безопасности правительственных коммуникаций Новой Зеландии в 2008 году. Применяется для несекретных данных, действителен в настоящее время. Данный алгоритм предусматривает один проход – заполнение области памяти случайными данными [8].

VSITR – стандарт программного уничтожения данных, принятый Федеральным управлением по информационной безопасности Германии в 1999 году. Применяется для несекретных данных, в настоящее время недействителен. Данный алгоритм предусматривает 7 проходов:

- заполнение области памяти нулями (0x00);
- заполнение области памяти единицами (0xFF);
- заполнение области памяти нулями (0x00);
- заполнение области памяти единицами (0xFF);
- заполнение области памяти нулями (0x00);
- заполнение области памяти единицами (0xFF);
- заполнение области памяти байтами 0xAA [9].

DoD 5220.22-M – стандарт программного уничтожения данных, принятый Министерством обороны США в 1995 году. Использовался в армии США, в настоящее время недействителен. Данный алгоритм имеет 2 модификации: DoD 5220.22-M (E) и DoD 5220.22-M (ECE).

Модификация E предусматривает 3 прохода:

- заполнение области памяти нулями (0x00);
- заполнение области памяти единицами (0xFF);
- заполнение области памяти случайными данными.

Модификация ECE предусматривает 7 проходов:

- заполнение области памяти нулями (0x00);
- заполнение области памяти единицами (0xFF);
- заполнение области памяти случайными данными;
- заполнение области памяти случайными данными;

– заполнение области памяти нулями (0x00);
– заполнение области памяти единицами (0xFF);
– заполнение области памяти случайными данными [9].

NCSC-TG-025 – стандарт программного уничтожения данных, принятый Центром национальной компьютерной безопасности США в 1991 году. Использовался в Агентстве национальной безопасности США, в настоящее время недействителен.

AFSSI-5020 (Air Force System Security Instruction 5020) – стандарт программного уничтожения данных, определённый в инструкции по безопасности Военно-воздушных сил США в 1996 году. Неизвестно, действителен ли данный стандарт в настоящее время.

NAVSO P-5239-26 – стандарт программного уничтожения данных, принятый Военно-морским министерством США в 1993 году. Неизвестно, действителен ли данный стандарт в настоящее время.

CSEC ITSG-06 – стандарт программного уничтожения данных, принятый Центром безопасности коммуникаций Канады в 2006 году. Применяется для несекретных данных, действителен в настоящее время.

HMG IS5 – стандарт программного уничтожения данных, определённый Группой безопасности электронных коммуникаций Великобритании. Неизвестно, действителен ли данный стандарт в настоящее время.

Реализация данных пяти алгоритмов предусматривает 3 прохода:
– заполнение области памяти нулями (0x00);
– заполнение области памяти единицами (0xFF);
– заполнение области памяти случайными данными [9].

AR 380-19 – стандарт программного уничтожения данных, определённый в армейском постановлении 380-19, опубликованном Армией США в 1998 году. Использовался Армией США, в настоящее время недействителен. Данный алгоритм предусматривает 3 прохода:
– заполнение области памяти случайными данными;
– заполнение области памяти единицами (0xFF);
– заполнение области памяти нулями (0x00) [9].

RCMP TSSIT OPS-II – стандарт программного уничтожения данных, определённый Королевской канадской конной полицией. В настоящее время недействителен, вместо него в Канаде используется ранее рассмотренный стандарт CSEC ITSG-06. Данный алгоритм предусматривает 7 проходов:

– заполнение области памяти нулями (0x00);
– заполнение области памяти единицами (0xFF);
– заполнение области памяти нулями (0x00);
– заполнение области памяти единицами (0xFF);
– заполнение области памяти нулями (0x00);
– заполнение области памяти единицами (0xFF);
– заполнение области памяти случайными данными [9].

Алгоритм Роя Пфицнера. Создатель этого метода, заявлял, что информацию можно будет восстановить, если перезаписать её всего 20 раз, и что перезаписи области памяти случайными символами более 30 раз должно быть достаточно. Однако вопрос о том, насколько это верно, остается предметом споров. Неизвестно, применяется ли данный алгоритм где-либо в настоящее время. Существует 2 модификации данного алгоритма: с 7 и 33 проходами, в обеих модификациях на каждом проходе осуществляется заполнение области памяти случайными данными [9].

Алгоритм Брюса Шнайера. Описан в книге «Прикладная криптография. Протоколы, алгоритмы и исходный код на С», изданной в 1996 году. Неизвестно, применяется ли данный алгоритм где-либо в настоящее время. Данный алгоритм предусматривает 7 проходов:

– заполнение области памяти единицами (0xFF);
– заполнение области памяти нулями (0x00);
– заполнение области памяти случайными данными;
– заполнение области памяти случайными данными;
– заполнение области памяти случайными данными;
– заполнение области памяти случайными данными;
– заполнение области памяти случайными данными [9].

Алгоритм Питера Гутмана. Разработан в 1996 году. Жёсткие диски, существовавшие в то время, использовали методы кодирования, отличные от современных. Выбор проходов предполагает, что пользователь не знает механизм кодирования, используемый диском, и потому включает в себя проходы, разработанные специально для трех различных типов приводов. Большинство из них были адаптированы для дисков, закодированных по схемам MFM и RLL. Однако современные диски уже не используют эти методы кодирования, что делает многие проходы метода Гутмана излишними. Как заявлял Питер Гутман в своей статье от 2007 года, «На самом деле, нет смысла проводить полную 35-проходную перезапись для каждого диска, поскольку она нацелена на сочетание сценариев с участием всех трех типов технологий кодирования, которые охватывают все более чем 30-летние MFM методы. Если вы используете диск, который использует технологии кодирования PRML/EPRML, вам не нужно выполнять все 35 проходов, нужно выполнить только определенные. Лучшее, что вы можете сделать для любого современного PRML/EPRML привода – это несколько проходов очистки случайными данными» [10]. Несмотря на это, данный алгоритм можно применять и для современных жёстких дисков, хоть это и будет избыточно и не совсем эффективно. Данный алгоритм предусматривает 35 проходов:

- четыре прохода: заполнение области памяти случайными данными;
- заполнение области памяти байтами 0x55;
- заполнение области памяти байтами 0xAA;
- заполнение области памяти байтами 0x92 0x49 0x24;
- заполнение области памяти байтами 0x49 0x24 0x92;
- заполнение области памяти байтами 0x24 0x92 0x49;
- заполнение области памяти нулями (0x00);
- заполнение области памяти байтами 0x11;
- заполнение области памяти байтами 0x22;
- заполнение области памяти байтами 0x33;
- заполнение области памяти байтами 0x44;

– заполнение области памяти байтами 0x55;
– заполнение области памяти байтами 0x66;
– заполнение области памяти байтами 0x77;
– заполнение области памяти байтами 0x88;
– заполнение области памяти байтами 0x99;
– заполнение области памяти байтами 0xAA;
– заполнение области памяти байтами 0xBB;
– заполнение области памяти байтами 0xCC;
– заполнение области памяти байтами 0xDD;
– заполнение области памяти байтами 0xEE;
– заполнение области памяти единицами (0xFF);
– заполнение области памяти байтами 0x92 0x49 0x24;
– заполнение области памяти байтами 0x49 0x24 0x92;
– заполнение области памяти байтами 0x24 0x92 0x49;
– заполнение области памяти байтами 0x6D 0xB6 0xDB;
– заполнение области памяти байтами 0xB6 0xDB 0x6D;
– заполнение области памяти байтами 0xDB 0x6D 0xB6;
– четыре прохода: заполнение области памяти случайными данными.

Вывод

Таким образом, в данной главе были детально рассмотрены некоторые существующие алгоритмы уничтожения данных. Все рассмотренные алгоритмы будут применены в разрабатываемом программном средстве Delete File Tool.

3 Обзор и анализ программных средств для безвозвратного удаления информации в ОС Windows

3.1 CCleaner

CCleaner – программное средство с закрытым исходным кодом, разработанное британской частной фирмой Piriform Limited в 2003 году. В настоящее время поддерживается разработчиками, актуальная версия 6.17.10746 была выпущена 18 октября 2023 года. Является условно-бесплатной утилитой, имеет как бесплатную версию, так и платную – с расширенным функционалом. Программа работает в ОС семейства Microsoft Windows, в частности, на 32-х и 64-х разрядных версиях Windows 7, 8, 8.1, 10 и 11, а также в операционной системе macOS [11]. Главное окно утилиты представлено на рисунке 3.1.

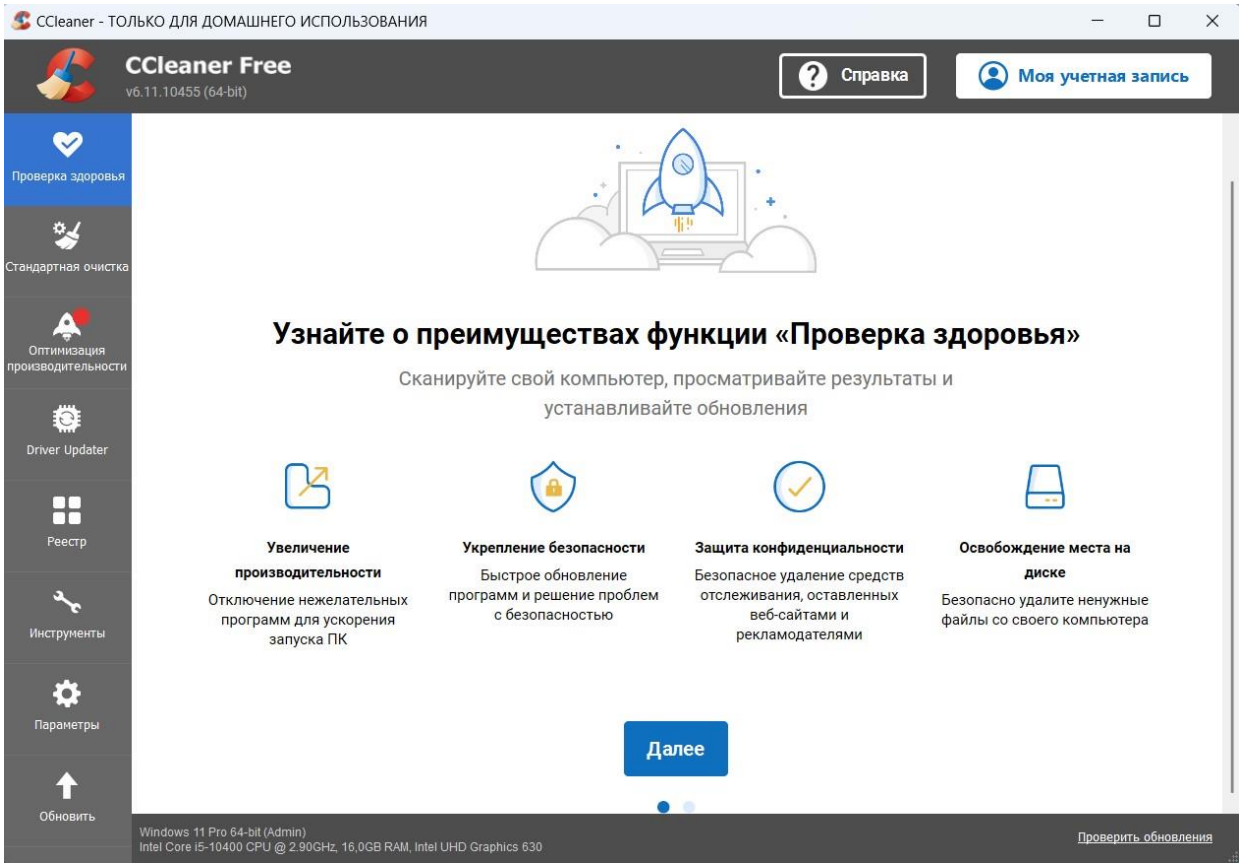


Рисунок 3.1 – Главное окно программы CCleaner

Программное средство обладает следующим функционалом:

1 Очистка данных браузеров Internet Explorer, Google Chrome, Opera, Mozilla Firefox, Safari: временных файлов, истории посещений, cookies, истории загрузок, истории автозаполнения форм, сохранённых паролей.
2 Очистка системы: временных файлов, корзины, последних документов, буфера обмена, файлов журнала, кэша DNS, дампа памяти, отчетов об ошибках, фрагментов файлов CHKDSK.
3 Очистка операционной системы после работы с популярными приложениями, такими как: Adobe Photoshop, Adobe Acrobat, Microsoft Office, Notepad++, BitTorrent, uTorrent, Winamp, Media Player Classic, DAEMONN Tools, WinRAR, WordPad, Paint и другими.
4 Сканирование и поиск ошибок в системном реестре Windows. Программа проверяет неверные расширения файлов, элементы управления ActiveX, ClassID, ProgID, деинсталляторы, общие DLL, шрифты, ссылки файлов помощи, пути приложений, иконки, неправильные ярлыки.
5 Стирание дисков – безопасное удаление всего содержимого или свободного места на диске.

В рамках данной работы интерес представляет последний пункт. Программа CCleaner предоставляет возможность безопасного удаления всего содержимого диска или только свободного места. В данной утилите доступны 4 алгоритма уничтожения данных под названиями: «Простая перезапись (1 проход)», «Продвинутая перезапись (3 прохода)», «Сложная перезапись (7 проходов)» и «Самая сложная перезапись (35 проходов)». Окно программного средства с режимом стирания дисков представлено на рисунке 3.2. В более ранних версиях утилиты методы имели другие названия, а именно: «Простая перезапись (1 проход)», «DOD 5220.22-M (3 прохода)», «NSA (7 проходов)» и «Гутманн (35 проходов)». Окно ранней версии приложения представлено на рисунке 3.3.

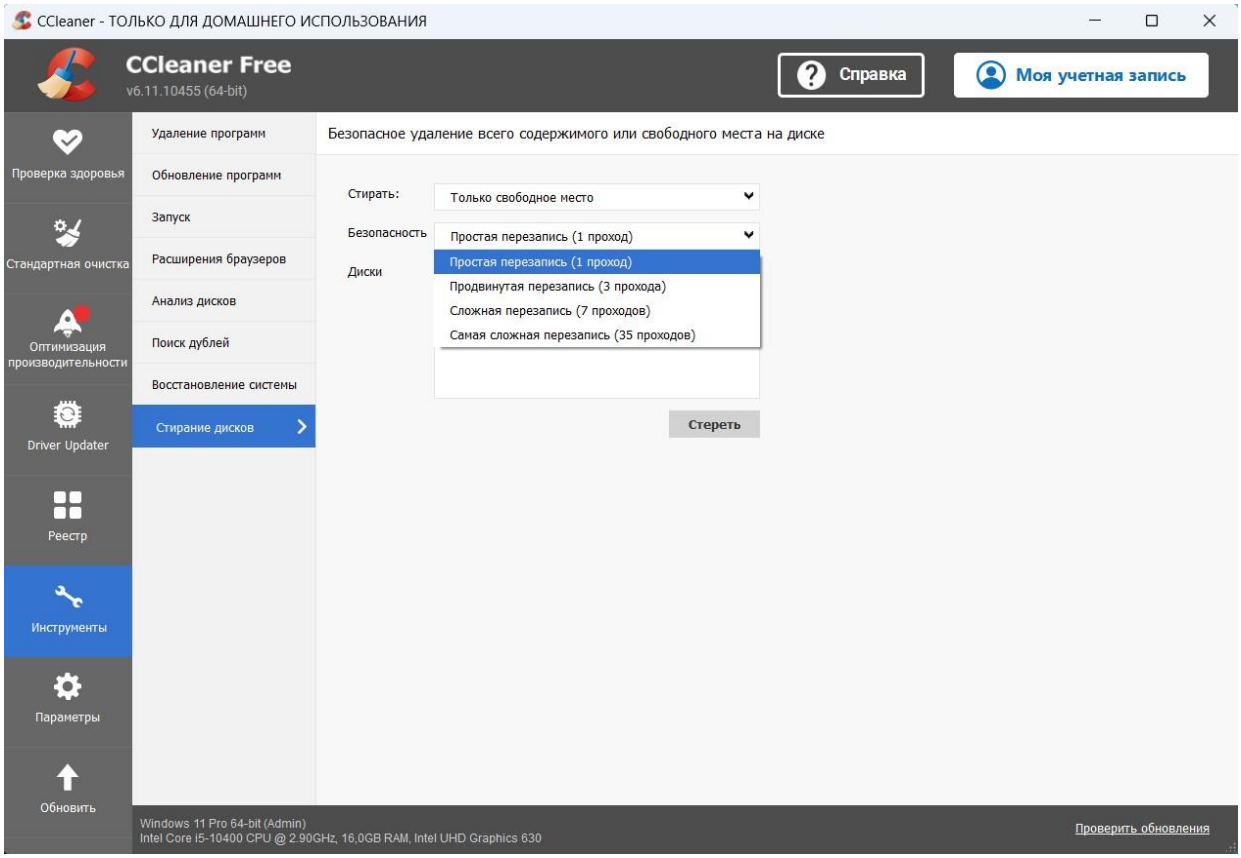


Рисунок 3.2 – Режим стирания дисков в программе CCleaner

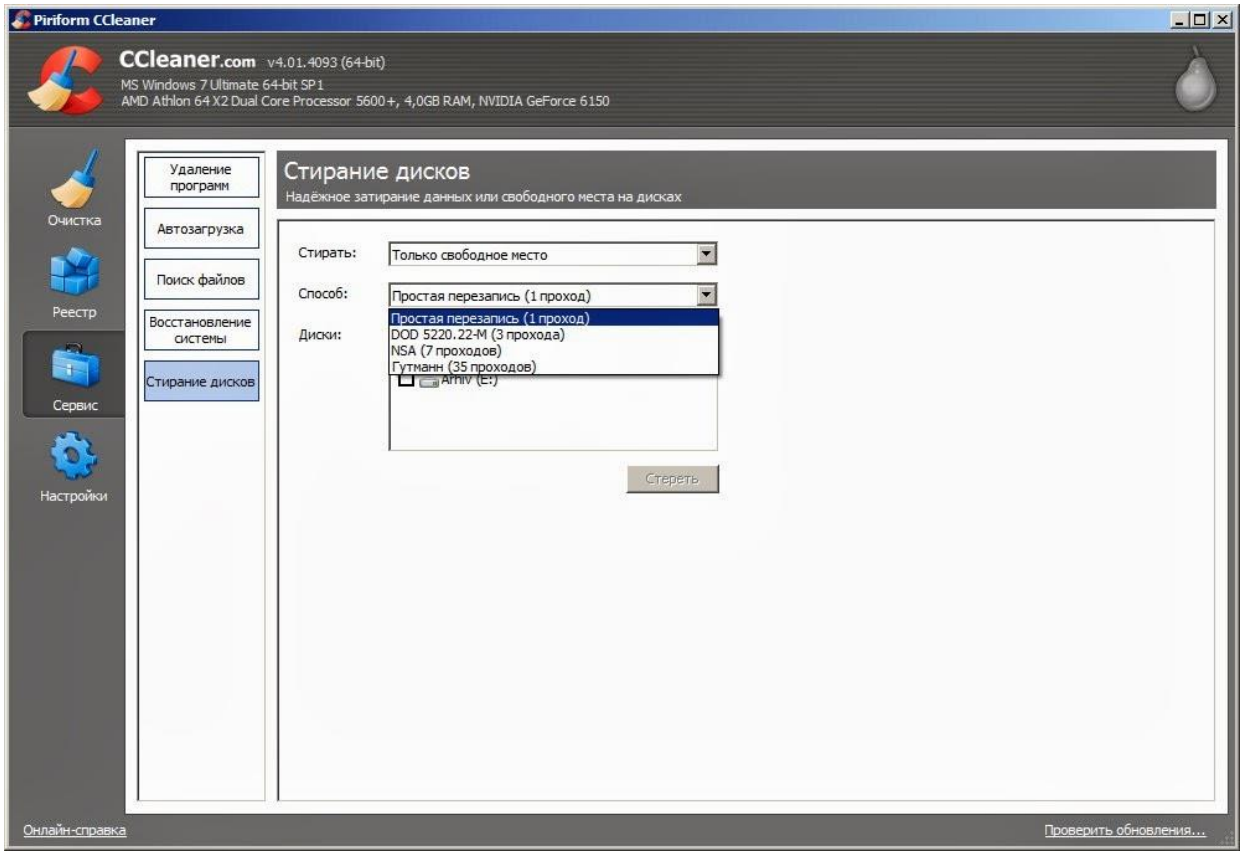


Рисунок 3.3 – Ранняя версия утилиты CCleaner

К преимуществам программы CCleaner можно отнести:

– доступность функции стирания дисков в бесплатной версии программного средства;
– мультиязычность – в утилите предусмотрен выбор из 60 предустановленных языков;
– богатый функционал приложения – помимо безвозвратного удаления информации, есть множество других полезных функций;
– поддержка программного средства разработчиками – частые обновления расширяют функционал утилиты и повышают скорость её работы;
– интуитивно понятный интерфейс приложения.

К недостаткам данного программного средства можно отнести:

– невозможность выборочного удаления файлов – доступно лишь полное стирание диска либо стирание свободного пространства;
– довольно небольшой набор доступных методов безвозвратного удаления информации, состоящий всего из 4 алгоритмов.

3.2 Eraser

Eraser – бесплатное программное средство для безвозвратного удаления информации, исходный код которого распространяется под лицензией GNU General Public License. Последняя версия программы была выпущена 5 октября 2021 года. Eraser предназначен для работы в ОС семейства Microsoft Windows. Актуальная версия программы может быть запущена на 32-х и 64-х разрядных версиях Windows Server 2003 (SP2), Server 2008, Server 2012-2022, XP (SP3), Vista, 7, 8, 10, 11. Eraser версии 5.7 поддерживает работу в ОС Windows 98, ME, NT и 2000 [12]. Единственной функцией программы является безвозвратное удаление информации, другой функционал не предусмотрен. Главное окно программного средства представлено на рисунке 3.4.

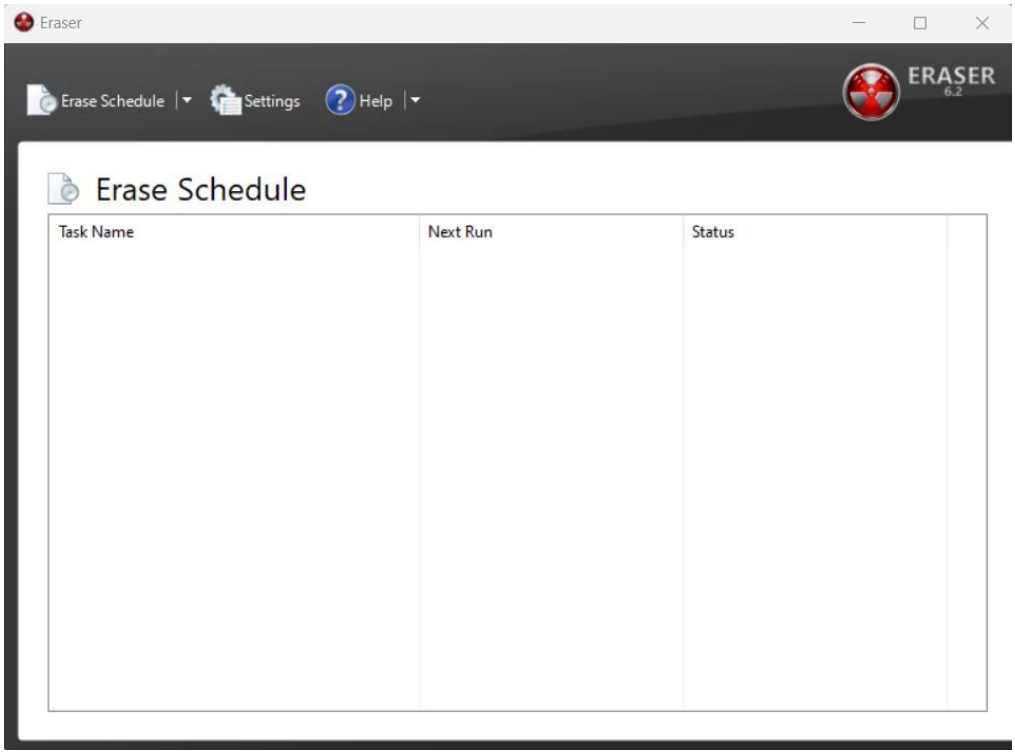


Рисунок 3.4 – Главное окно программы Eraser

Для уничтожения данных предусмотрено 13 алгоритмов: DoD 5220.22-M (модификация E, 3 прохода), DoD 5220.22-M (модификация ECE, 7 проходов), алгоритм Гутмана (35 проходов), RCMP TSSIT OPS-II (7 проходов), алгоритм Шнайера (7 проходов), VSITR (7 проходов), HMG IS5 (1 проход), HMG IS5 (3 прохода), AFSSI-5020 (3 прохода), AR 380-19 (3 прохода), ГОСТ Р 50739-95 (2 прохода), заполнение области памяти псевдослучайными данными, удаление первых и последних 16 Кбайт файла. Окно программного средства с выбором алгоритма уничтожения информации представлено на рисунке 3.5.

К преимуществам программы Eraser можно отнести:

- распространение на бесплатной основе;
- возможность затирания как отдельных файлов и папок, так и свободного места, и всей информации на накопителе;
- обширный набор доступных методов безвозвратного удаления информации, состоящий из 13 алгоритмов;
- возможность создавать запланированные задачи по удалению выбранных данных, например, после перезагрузки или в определенные дни.

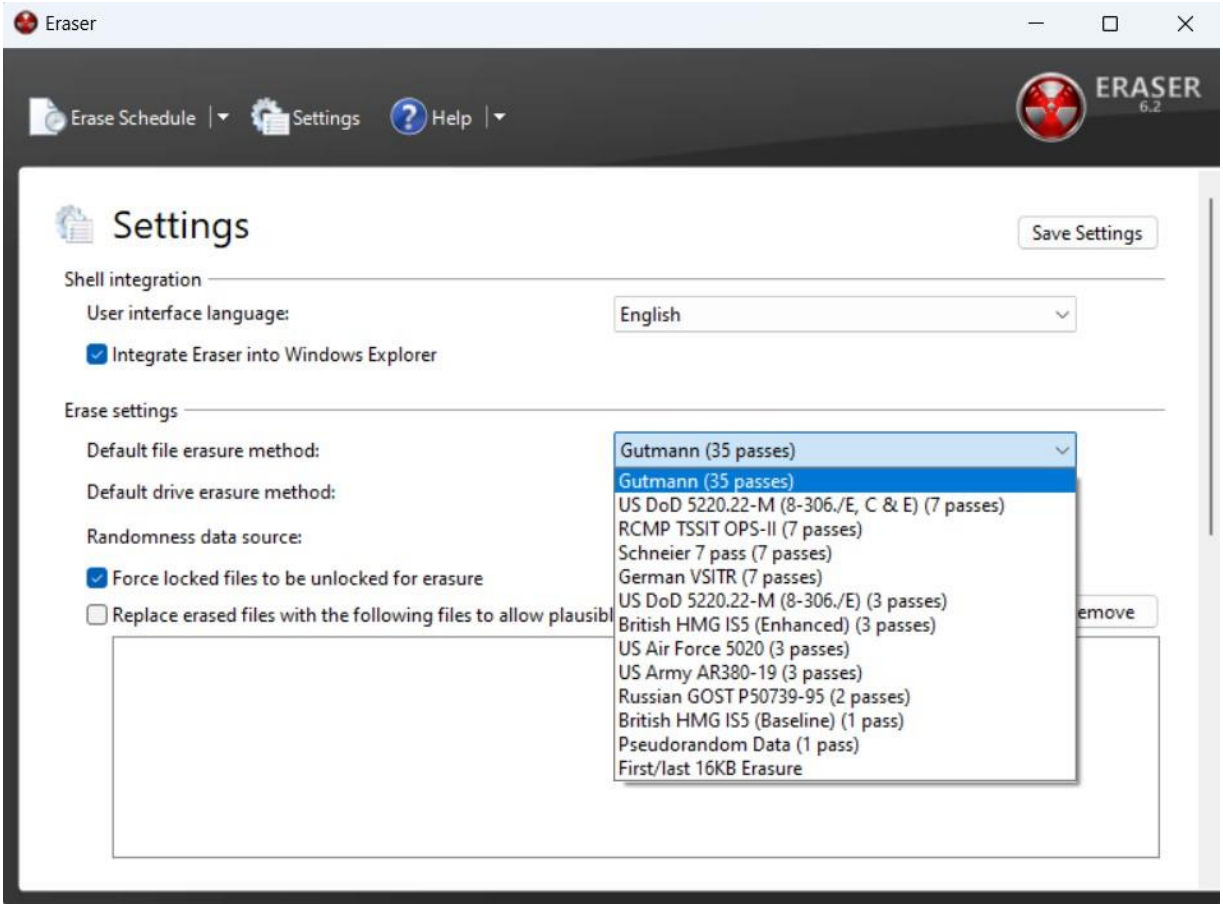


Рисунок 3.5 – Окно программы Eraser с выбором алгоритма удаления данных

К недостаткам данного программного средства можно отнести:

– довольно непростой интерфейс для интуитивного понимания;

– отсутствие возможности выбора языка – доступен только английский.

3.3 File Shredder

File Shredder – бесплатное программное средство для безвозвратного удаления информации, исходный код которого распространяется под лицензией GNU General Public License. File Shredder предназначен для работы в ОС семейства Microsoft Windows, в частности, на 32-х и 64-х разрядных версиях Windows NT, 2000, XP, Server 2003, Vista, 7, 8, 10 и 11 [13]. Единственной функцией программы является безвозвратное удаление информации, другой функционал не предусмотрен. Главное окно программного средства представлено на рисунке 3.6.

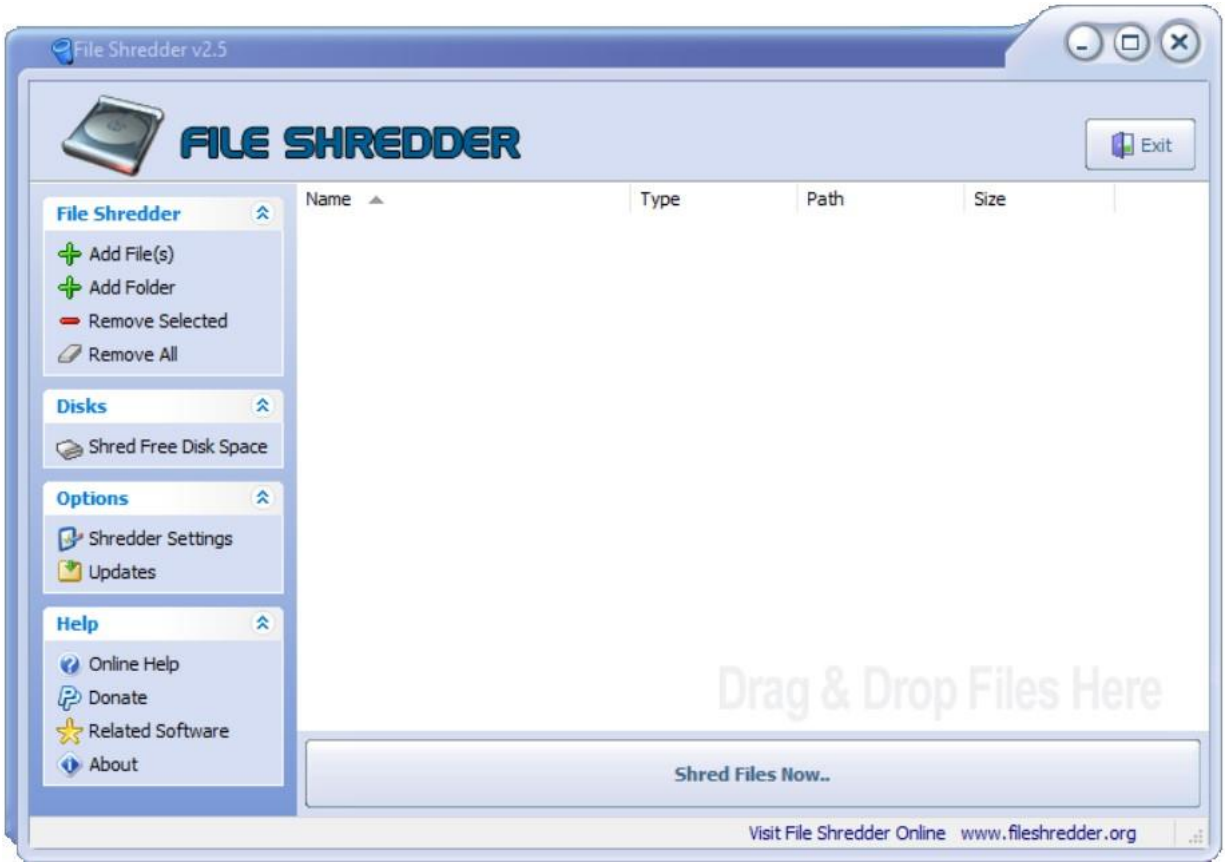


Рисунок 3.6 – Главное окно программы File Shredder

Для уничтожения данных предусмотрено 5 методов: «Simple One Pass», «Simple Two Pass», «DoD 5220.22-M», «Secure erasing algorithm with 7 passes» и «Gutmann algorithm 35 passes». Окно программы с выбором алгоритма уничтожения данных представлено на рисунке 3.7.

К преимуществам программы File Shredder можно отнести:

– распространение на бесплатной основе;
– возможность затирания как отдельных файлов и папок, так и свободного места на накопителе;
– интуитивно понятный интерфейс приложения.

К недостаткам данного программного средства можно отнести:

– довольно небольшой набор доступных методов безвозвратного удаления информации, состоящий всего из 5 алгоритмов;
– отсутствие возможности выбора языка – доступен только английский.

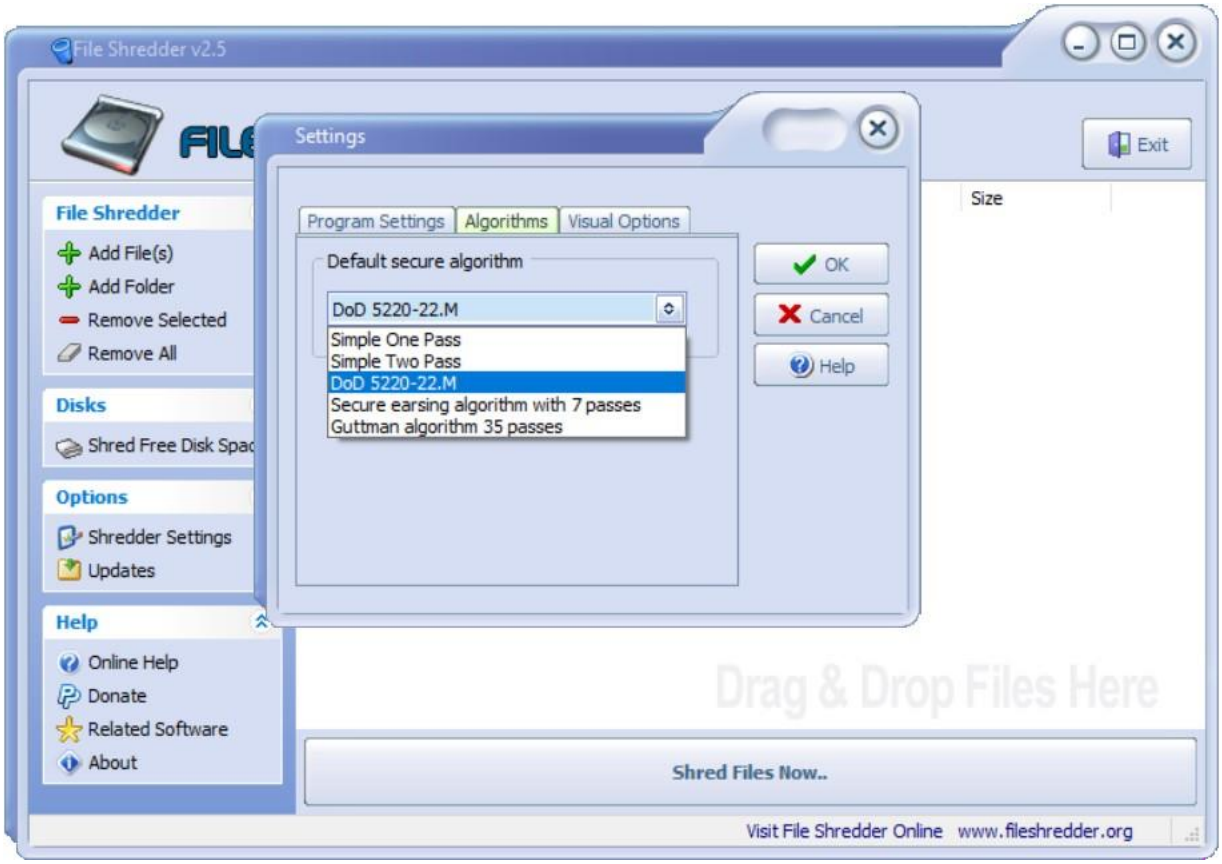


Рисунок 3.7 – Окно программы File Shredder с выбором алгоритма удаления данных

Вывод

Таким образом, в результате проведенного обзора программных средств для безвозвратного удаления информации в ОС Windows были выявлены основные преимущества и недостатки рассматриваемых программ, которые будут учитываться при разработке собственного программного средства.

4 Алгоритмическая реализация

Для разработки программного средства была сконструирована блок-схема алгоритма его работы, она представлена на рисунке 4.1.

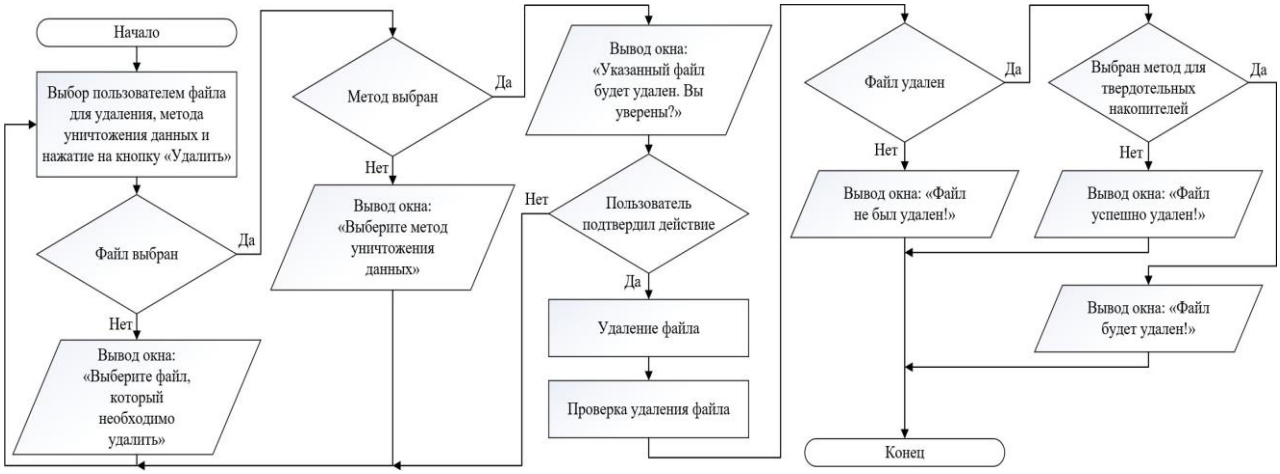


Рисунок 4.1 – Блок-схема алгоритма работы программного средства

Алгоритм состоит в следующем: после установки программного средства в меню «Пуск» и на рабочем столе создаются ярлыки для запуска программы. При открытии этих ярлыков запускается программное средство. Сначала пользователю необходимо выбрать файл, который необходимо удалить, и метод уничтожения данных. В случае если пользователь не выбрал что-то из вышеперечисленного, должно появиться всплывающее окно с соответствующим содержанием. Далее пользователю необходимо нажать на кнопку для удаления файла. После нажатия на кнопку должно появиться всплывающее окно с просьбой подтвердить действие. С согласия пользователя должен запуститься алгоритм удаления файла и затем алгоритм проверки факта удаления. В случае если по каким-либо причинам файл не был удален, должно появиться соответствующее всплывающее окно. В случае если файл был удален и выбранный метод был не для твердотельных накопителей, должно появиться всплывающее окно с сообщением о том, что файл был успешно удален. В случае же если был выбран метод для твердотельных накопителей, и проверка

установила, что файл был удален, должно появиться всплывающее окно с сообщением о том, что файл будет удален. На этом алгоритм работы программного средства завершается.

Алгоритмы удаления файла и проверки факта удаления будут рассмотрены отдельно, блок-схема данных алгоритмов представлена на рисунке 4.2.

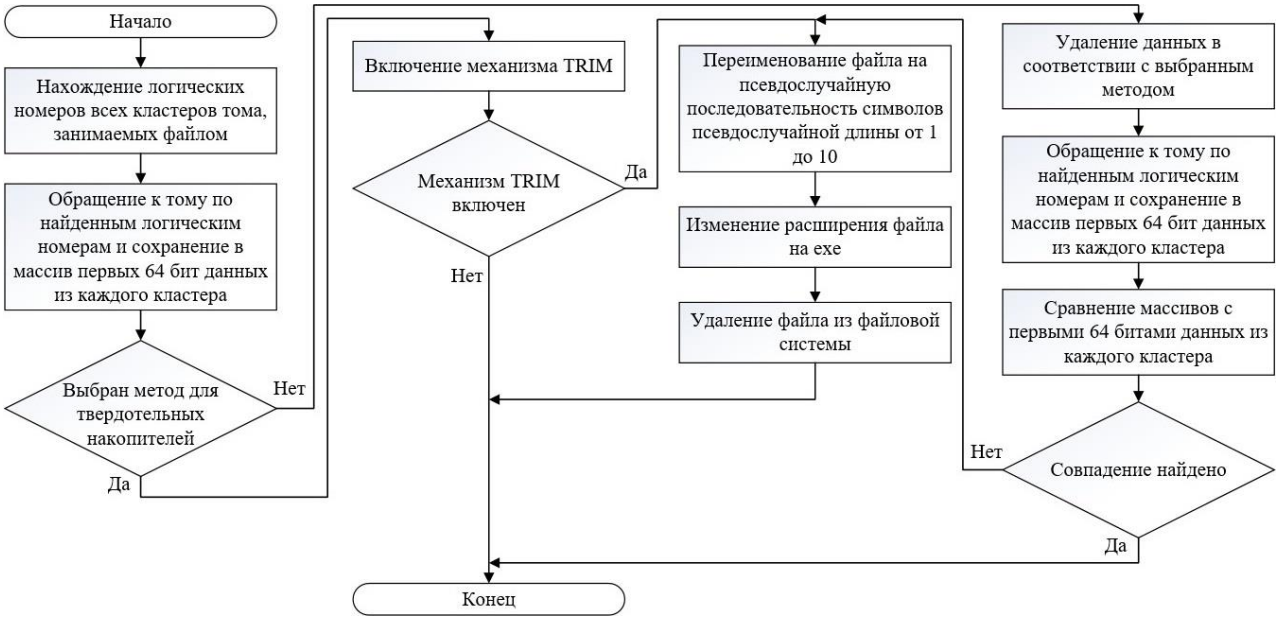


Рисунок 4.2 – Блок-схема алгоритмов удаления файлов и проверки факта удаления

Суть данных алгоритмов состоит в следующем: сначала находятся логические номера всех кластеров тома, занимаемых выбранным для удаления файлом. Далее происходит обращение к тому и первые 64 бита каждого кластера, занимаемого файлом, сохраняются в массив. Если пользователем был выбран метод для твердотельных накопителей, то программное средство включает в операционной системе механизм TRIM. В случае если данный механизм включить не удалось, алгоритм возвращает результат о том, что файл не был удален. В противном случае имя файла меняется на псевдослучайную последовательность символов псевдослучайной длины от 1 до 10, расширение файла меняется на .exe, файл удаляется из файловой системы с последующим автоматическим удалением записи из главной файловой таблицы NTFS и алгоритм возвращает результат о том, что файл был удален. В случае если пользователем был выбран метод не для твердотельных накопителей, то происходит

уничтожение данных с накопителя в соответствии с выбранным методом. Далее происходит обращение к тому, запись первых 64 бит каждого кластера, занимаемого файлом, в массив и сравнение данного массива с ранее сформированным массивом. Если находится совпадение хоть одного элемента, алгоритм возвращает результат о том, что файл не был удален. Если совпадений найдено не было, имя файла меняется на псевдослучайную последовательность символов псевдослучайной длины от 1 до 10, расширение файла меняется на .exe, файл удаляется из файловой системы с последующим автоматическим удалением записи из главной файловой таблицы NTFS и алгоритм возвращает результат о том, что файл был удален.

Отдельно следует сказать о том, что вследствие особенностей хранения и удаления информации с твердотельных накопителей, применение для них алгоритмов уничтожения данных нецелесообразно. Для данных накопителей существует уже разработанный механизм очистки ячеек памяти, называемый TRIM. Разработанное программное средство включает в операционной системе данный механизм и удаляет файл из файловой системы. Накопитель получает информацию от операционной системы о том, что блоки данных, содержащие удаляемый файл, не несут полезной нагрузки и их можно очистить. Очистку твердотельного накопителя осуществляет его контроллер, вследствие этого невозможно узнать через какой промежуток времени данные будут уничтожены. Именно поэтому программное средство в случае выбора пользователем метода уничтожения данных для твердотельных накопителей, успешного включения механизма TRIM и удаления файла из файловой системы, выдаст информацию о том, что файл будет удален.

Вывод

Таким образом, в данной главе были приведены блок-схемы и описание алгоритмов работы, которые являются основой для написания эффективного кода и построения правильной архитектуры программного средства.

5 Программная реализация

5.1 Средства разработки

Для написания исходного кода программного средства был использован язык программирования C++.

C++ – высокоуровневый, компилируемый, статически типизированный язык программирования общего назначения, созданный датским программистом Бьёрном Страуструпом в 1985 году. Изначально выпускался как расширение для языка программирования C, с тех пор значительно расширился – на текущий момент имеет объектно-ориентированные, универсальные и функциональные возможности в дополнение к средствам низкоуровневого манипулирования памятью. Этот язык широко используется в различных отраслях, включая игровую индустрию, финансовый сектор и другие. Он применяется для создания операционных систем, прикладных программ, драйверов устройств, приложений для встраиваемых систем, серверов, компьютерных игр, веб-сервисов. C++ обладает обширной стандартной библиотекой, которая включает в себя распространенные контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. Этот язык программирования сочетает свойства как высокоуровневых, так и низкоуровневых языков.

К преимуществам C++ можно отнести:

- переносимость: возможность компиляции и запуска одного и того же кода на различных операционных системах без необходимости его изменения;
- мультипарадигменность: поддержка различных парадигм программирования, таких как процедурное, объектно-ориентированное и обобщенное программирование;
- полный контроль над управлением памятью, что обеспечивает эффективное использование памяти и высокую скорость работы;

– поддержку низкоуровневых операций;
– поддержку шаблонов, что позволяет создавать универсальный и модульный код;
– строгую проверку типов, которая помогает выявлять ошибки на этапе компиляции и упрощает отладку кода [14].

В качестве среды разработки был использован Visual Studio Code.

Visual Studio Code (VS Code) – это бесплатный многоязычный редактор исходного кода, разработанный Microsoft для операционных систем Windows, Linux и macOS. Редактор был представлен на конференции Build 29 апреля 2015 года, а первая версия была выпущена 14 ноября 2015 года. VS Code основан на фреймворке Electron и использует веб-редактор Monaco, разработанный для Visual Studio Online. Visual Studio Code поддерживает широкий спектр языков программирования, включая Java, C++, Python, CSS, Go и Dockerfile. Среди возможностей VS Code: поддержка отладки кода, подсветка синтаксиса, автодополнение IntelliSense, рефакторинг кода, инструменты работы с Git, линтинг, многокурсорное редактирование, подсказки по параметрам и другие функции редактирования. Пользователи имеют возможность настраивать темы оформления, сочетания клавиш и устанавливать расширения, добавляющие функциональность [15].

Для разработки графического интерфейса был использован фреймворк Qt.

Qt – это фреймворк для разработки графических пользовательских интерфейсов и кроссплатформенных приложений, которые работают на различных программных и аппаратных платформах, таких как Linux, Windows, macOS, Android или встраиваемых системах путём простой компиляции кода с небольшими изменениями или без изменений исходного кода. Фреймворк был выпущен компанией TrollTech 20 мая 1995 года под названием Qt 0.90, в настоящее время компания переименовалась в The Qt Company. Qt доступен как по коммерческой лицензии, так и по лицензиям с открытым исходным кодом GPL 2.0, GPL 3.0, LGPL 3.0. Qt поддерживает создание приложений с графическим интерфейсом, а также программ без него, например, инструменты командной строки и консольные приложения для серверов. Qt

поддерживает различные компиляторы и имеет обширную поддержку интернационализации. Библиотека Qt разделена на модули, предназначенные для работы с базами данных, графикой, XML, аудио, видео и прочим. Каждый модуль предоставляет определенный функционал для разработки приложений. Кроме библиотеки и ее модулей, Qt содержит дополнительное программное обеспечение, утилиты, справочники и внутренние языки:

1 Qt Creator – это интегрированная среда разработки, предназначенная для написания, компиляции, тестирования и отладки кода на различных операционных системах.
2 Qt Assistant – обширная библиотека документации и справочник, позволяющий открывать и читать документы в формате QCH.
3 Qt Quick – инструмент для создания интерфейсов с использованием языка QML, удобный для разработки мобильных приложений и игр.
4 Qt Designer – инструмент для быстрого создания графических интерфейсов, поставляемый вместе с фреймворком.
5 QML – язык для создания интерфейсов от команды Qt, основанный на среде JavaScript. В Qt реализована полная поддержка QML, а сам язык встроен в инструмент Qt Quick.
6 Qt Linguist – инструмент для локализации приложений на различные языки [16].

Для создания установщика программного средства был использован Inno Setup.

Inno Setup – это система создания инсталляторов для программ под Windows с открытым исходным кодом. Была разработана Джорданом Расселом и Мартейном Лааном, первая версия была выпущена в 1997 году. Inno Setup является бесплатным программным обеспечением, в том числе для коммерческого использования. Ключевыми особенностями программы являются:

– расширенная поддержка установки 64-битных приложений в 64-битных версиях Windows;
– поддержка всех версий Windows с 2006 года;

- возможность удаления установленного программного обеспечения;
- настраиваемые типы установки;
- возможность создания ярлыков в различных местах;
- создание записей в реестре и .ini-файлов;
- поддержка создания многоязычных инсталляторов программ;
- возможность установки пароля и шифрования инсталляторов;
- поддержка установки и удаления с цифровой подписью;
- полный исходный код доступен на GitHub;
- все функции полностью документированы;
- используется Microsoft Visual Studio Code и Embarcardero Delphi [17].

5.2 Структура программного средства

Программное средство структурно состоит из:

1 Модуля DelFile, предназначенного для: сохранения первых 64 бит данных из каждого кластера, занимаемого файлом; непосредственного уничтожения данных с накопителя путем использования выбранного пользователем метода; проверки успешности выполненной операции по уничтожению данных; переименования файла на псевдослучайную последовательность символов псевдослучайной длины от 1 до 10; изменения расширения файла на .exe; выравнивания размера файла до значения, кратного размеру кластера; удаления файла из файловой системы NTFS с последующим автоматическим стиранием записи из главной файловой таблицы MFT.

2 Модуля GetClusters, предназначенного для поиска логических номеров всех кластеров тома, занимаемых выбранным для удаления файлом.

3 Модуля TRIMstatus, отвечающего за выполнение команды включения в операционной системе механизма TRIM, предназначенного для корректного уничтожения данных с твердотельных накопителей, а также за проверку результата выполнения данной команды.

4 Классов и методов фреймворка Qt, отвечающих за реализацию пользовательского графического интерфейса. Помимо главного окна программного средства, предусмотрены различные всплывающие окна, отвечающие за выбор файла, подтверждение действия пользователем и информирующие о результате работы программы.

5.3 Разработка графического интерфейса

Главное окно программного средства содержит:

– надпись с указанием пути к удаляемому файлу и кнопку выбора файла;
– блок с выбором метода уничтожения данных;
– кнопку удаления файла.

При наведении курсором на любой из представленных методов уничтожения данных появляется подсказка с кратким описанием выбранного метода.

Общий вид графического интерфейса представлен на рисунке 5.1.

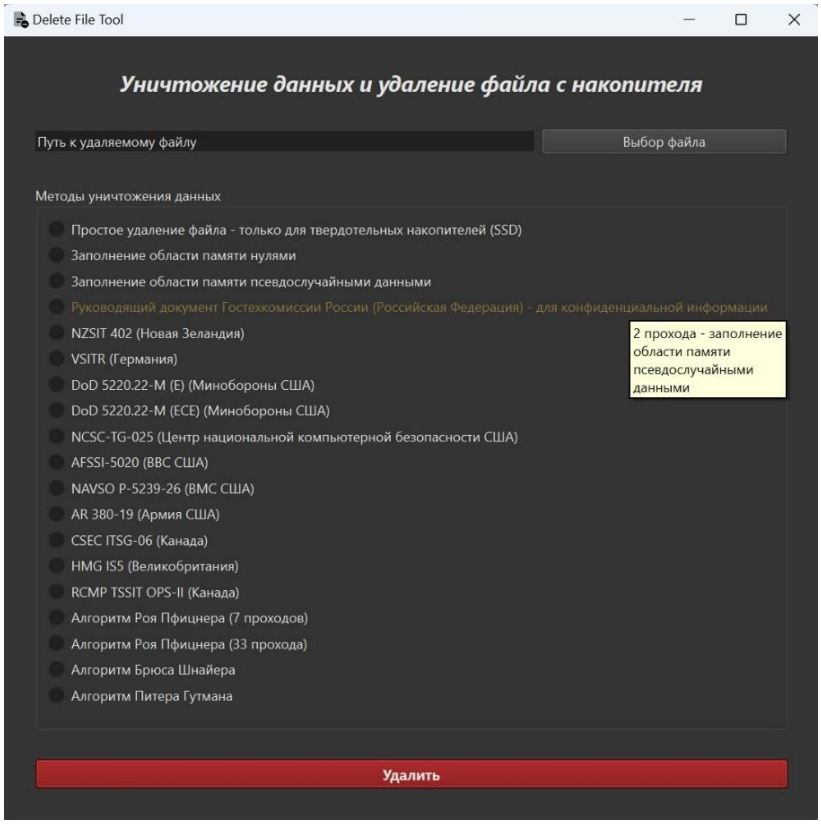


Рисунок 5.1 – Общий вид графического интерфейса

При нажатии на кнопку «Выбор файла» появляется всплывающее окно, в котором пользователь может выбрать файл, который необходимо удалить. Данное окно представлено на рисунке 5.2.

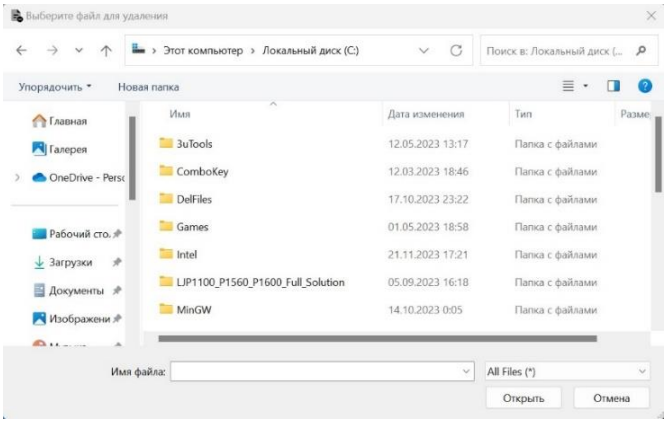


Рисунок 5.2 – Окно выбора файла

В случае если пользователь не выбрал файл и нажал кнопку «Удалить», появится всплывающее окно с просьбой выбрать файл. Данное окно представлено на рисунке 5.3.

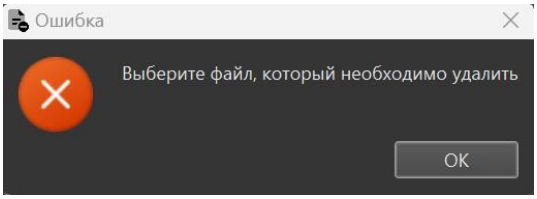


Рисунок 5.3 – Окно с просьбой выбрать файл

Если пользователь выбрал файл, но не выбрал метод уничтожения данных и нажал кнопку «Удалить», появится всплывающее окно с просьбой выбрать метод. Данное окно представлено на рисунке 5.4.

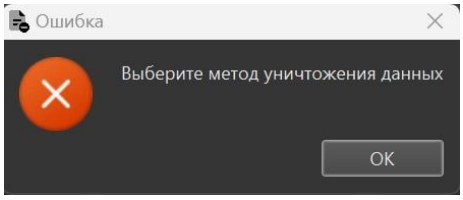


Рисунок 5.4 – Окно с просьбой выбрать метод

Если же пользователь выбрал файл, метод и нажал на кнопку «Удалить», появляется всплывающее окно с просьбой подтвердить действие. Данное окно представлено на рисунке 5.5.

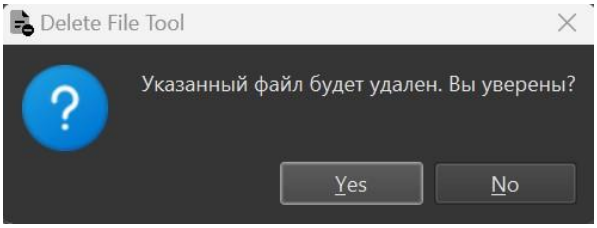


Рисунок 5.5 – Окно с просьбой подтвердить удаление файла

В случае если пользователь выбрал метод уничтожения данных для твердотельного накопителя, и алгоритмы работы модулей удаления и проверки удаления файла завершились успешно, появляется всплывающее окно с информацией о том, что файл будет удален. Данное окно представлено на рисунке 5.6.

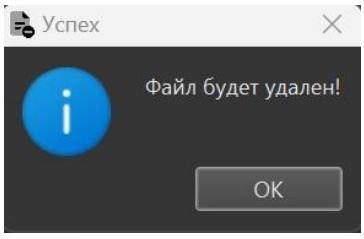


Рисунок 5.6 – Окно успешного удаления файла с твердотельного накопителя

Если же пользователь выбрал метод уничтожения данных для жёсткого диска, и алгоритмы работы модулей удаления и проверки удаления файла завершились успешно, появляется всплывающее окно с информацией о том, что файл был успешно удален. Данное окно представлено на рисунке 5.7.

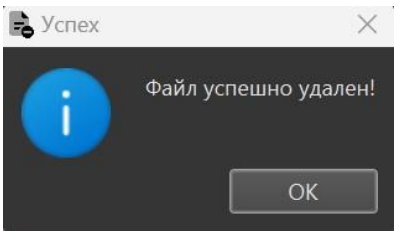


Рисунок 5.7 – Окно успешного удаления файла с жёсткого диска

В случае если модуль проверки удаления файла вернул информацию о том, что файл удалён не был, появляется соответствующее всплывающее окно. Данное окно представлено на рисунке 5.8.

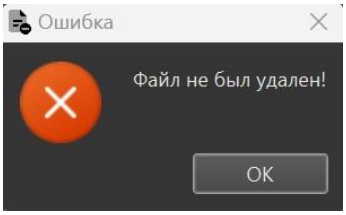


Рисунок 5.8 – Окно с неудачным результатом удаления файла

5.4 Описание работы программного средства

Для взаимодействия с программным средством необходима его предварительная установка в операционной системе. Окно установки представлено на рисунке 5.9.

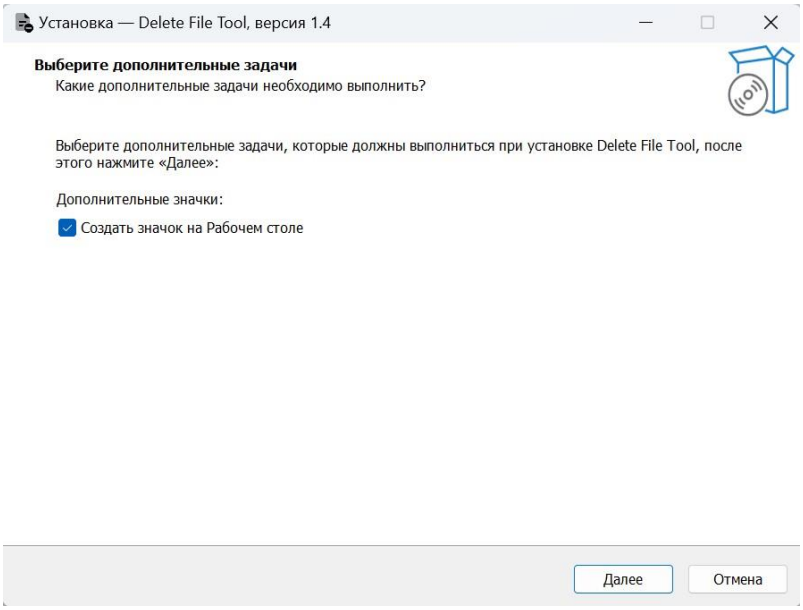


Рисунок 5.9 – Окно установки разработанного программного средства

Установленное программное средство необходимо запустить путём открытия ярлыка на рабочем столе либо в меню «Пуск». Окно запущенного программного средства представлено на рисунке 5.10.

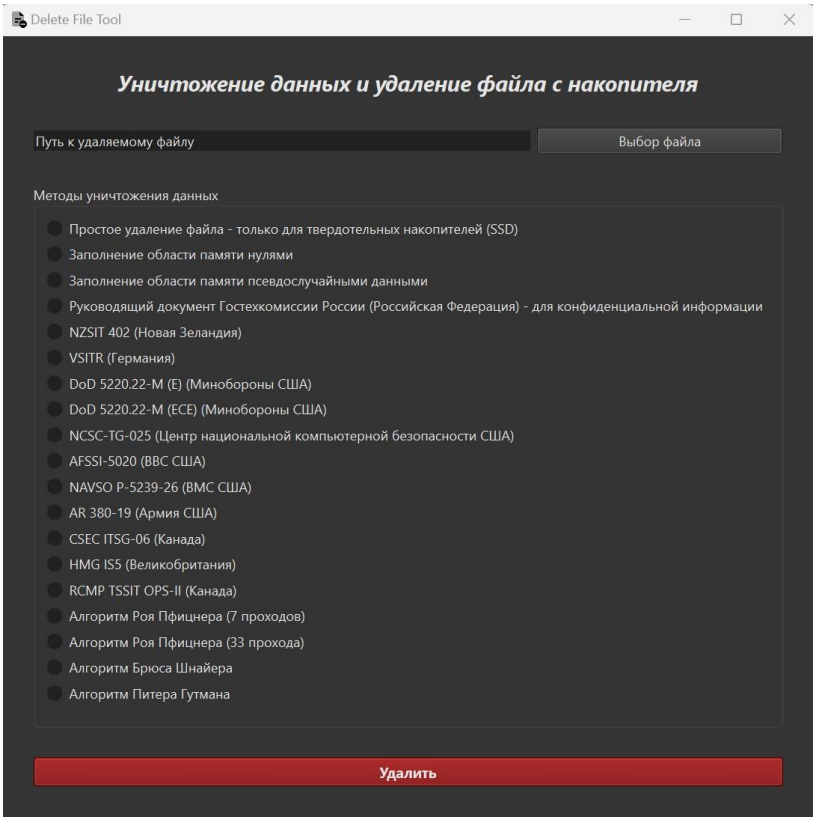


Рисунок 5.10 – Окно разработанного программного средства

После этого следует выбрать файл, который необходимо удалить. Для этого необходимо нажать на кнопку «Выбор файла» и выбрать необходимый файл. Окно выбора файла представлено на рисунке 5.11.

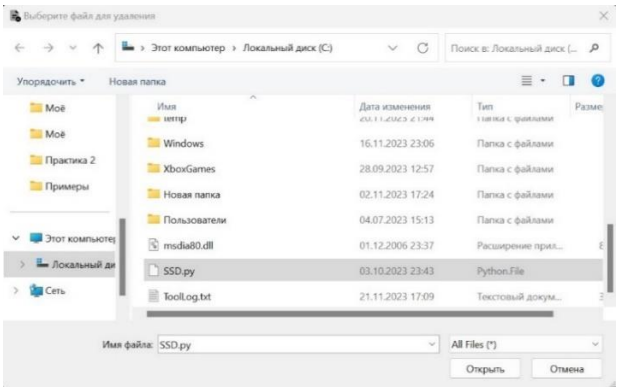


Рисунок 5.11 – Окно выбора файла

Затем необходимо выбрать метод уничтожения данных в главном окне разработанного программного средства. Окно программного средства с выбранными файлом и методом уничтожения данных представлено на рисунке 5.12.

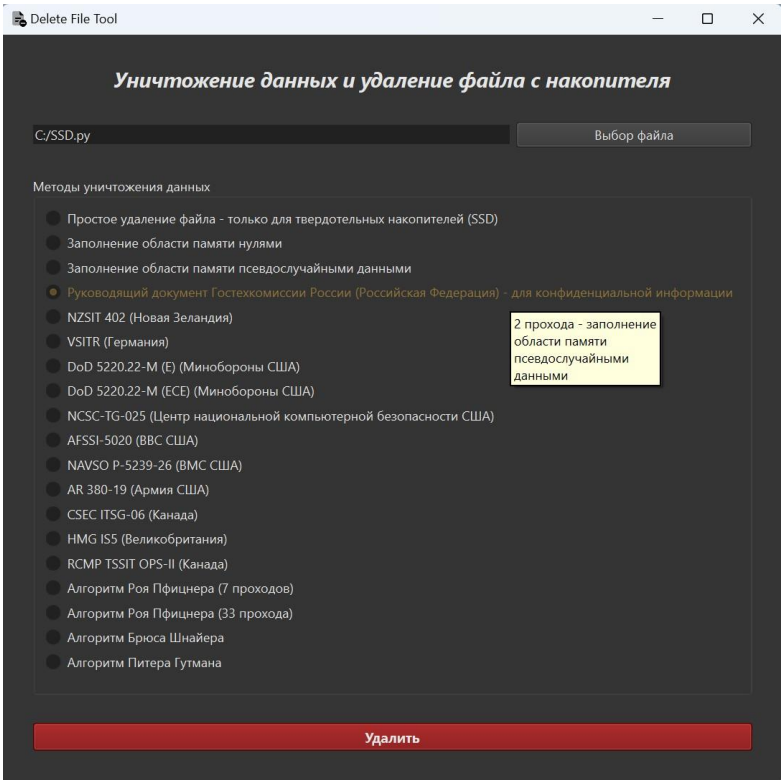


Рисунок 5.12 – Окно программного средства с выбранным файлом и методом уничтожения

Перед удалением зафиксируем информацию о выбранном файле с помощью программы «Active@ Disk Editor». Запись в главной файловой таблице MFT представлена на рисунке 5.13.

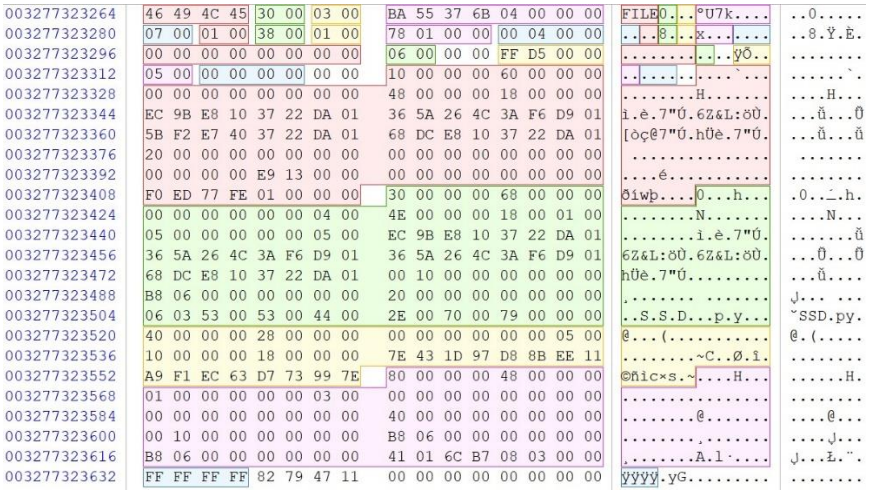


Рисунок 5.13 – Запись о выбранном файле в главной файловой таблице

Содержимое файла, хранящееся на накопителе, представлено на рисунке 5.14.

```
208498245632  69 6D 70 6F 72 74 20 73  75 62 70 72 6F 63 65 73  import subproces  ........
208498245648  73 0D 0A 69 6D 70 6F 72  74 20 73 79 73 0D 0A 0D  s..import sys...  ........
208498245664  0A 64 65 66 20 54 72 69  6D 5F 63 68 65 63 6B 28  .def Trim_check(  ........
208498245680  29 3A 0D 0A 20 20 20 20  69 73 54 72 69 6D 4F 4E  ):..    isTrimON  ..††....
208498245696  20 3D 20 73 75 62 70 72  6F 63 65 73 73 2E 72 75   = subprocess.ru  ........
208498245712  6E 28 22 66 73 75 74 69  6C 20 62 65 68 61 76 69  n("fsutil behavi  ........
208498245728  6F 72 20 71 75 65 72 79  20 64 69 73 61 62 6C 65  or query disable  ........
208498245744  64 65 6C 65 74 65 6E 6F  74 69 66 79 22 2C 20 73  deletenotify", s  ........
208498245760  74 64 6F 75 74 3D 73 75  62 70 72 6F 63 65 73 73  tdout=subprocess  ........
208498245776  2E 50 49 50 45 2C 20 74  65 78 74 3D 54 72 75 65  .PIPE, text=True  ........
208498245792  29 0D 0A 20 20 20 20 69  73 54 72 69 6D 4E 54 46  )..    isTrimNTF  ..†.....
208498245808  53 20 3D 20 69 73 54 72  69 6D 4F 4E 2E 73 74 64  S = isTrimON.std  .?......
208498245824  6F 75 74 5B 69 73 54 72  69 6D 4F 4E 2E 73 74 64  out[isTrimON.std  ........
208498245840  6F 75 74 2E 66 69 6E 64  28 22 3D 22 29 2B 32 5D  out.find("=")+2]  ........
208498245856  0D 0A 20 20 20 20 69 73  54 72 69 6D 52 65 46 53  ..    isTrimReFS  .††.....
208498245872  20 3D 20 69 73 54 72 69  6D 4F 4E 2E 73 74 64 6F   = isTrimON.stdo  ........
208498245888  75 74 5B 69 73 54 72 69  6D 4F 4E 2E 73 74 64 6F  ut[isTrimON.stdo  ........
208498245904  75 74 2E 72 66 69 6E 64  28 22 3D 22 29 2B 32 5D  ut.rfind("=")+2]  ........
208498245920  0D 0A 20 20 20 20 72 65  74 75 72 6E 28 69 73 54  ..    return(isT  .††.....
208498245936  72 69 6D 4E 54 46 53 2C  20 69 73 54 72 69 6D 52  rimNTFS, isTrimR  ........
208498245952  65 46 53 29 0D 0A 0D 0A  64 65 66 20 45 6E 61 62  eFS)....def Enab  ........
208498245968  6C 65 5F 54 52 49 4D 28  4E 54 46 53 2C 20 52 65  le_TRIM(NTFS, Re  ........
208498245984  46 53 29 3A 0D 0A 20 20  20 20 44 69 73 6B 44 65  FS):..    DiskDe  ...††...
208498246000  66 69 6E 69 74 69 6F 6E  20 3D 20 73 75 62 70 72  finition = subpr  ........
208498246016  6F 63 65 73 73 2E 72 75  6E 28 22 77 69 6E 73 61  ocess.run("winsa  ........
208498246032  74 20 64 69 73 6B 66 6F  72 6D 61 6C 22 2C 20 73  t diskformal", s  ........
208498246048  74 64 6F 75 74 3D 73 75  62 70 72 6F 63 65 73 73  tdout=subprocess  ........
208498246064  2E 50 49 50 45 2C 20 74  65 78 74 3D 54 72 75 65  .PIPE, text=True  ........
208498246080  29 0D 0A 20 20 20 20 69  66 20 4E 54 46 53 20 3D  )..    if NTFS =  ..†.....
208498246096  3D 20 27 31 27 3A 0D 0A  20 20 20 20 20 20 20 20  = '1':..          ?...††††
208498246112  4E 54 46 53 4F 4E 20 3D  20 73 75 62 70 72 6F 63  NTFSON = subproc  ........
208498246128  65 73 73 2E 72 75 6E 28  22 66 73 75 74 69 6C 20  ess.run("fsutil   ........
```

Рисунок 5.14 – Содержимое файла

Далее необходимо нажать кнопку «Удалить» в разработанном программном средстве, после чего появится всплывающее окно с просьбой подтвердить действие. Данное окно представлено на рисунке 5.15.

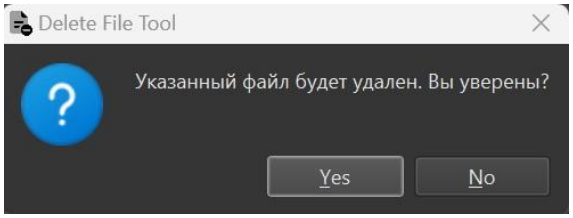


Рисунок 5.15 – Всплывающее окно с просьбой подтвердить удаление файла

После нажатия на кнопку «Yes» появится новое всплывающее окно, информирующее о результате работы программного средства. Данное окно представлено на рисунке 5.16.

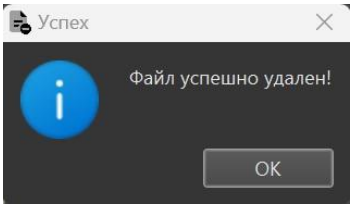


Рисунок 5.16 – Всплывающее окно с результатом работы программного средства

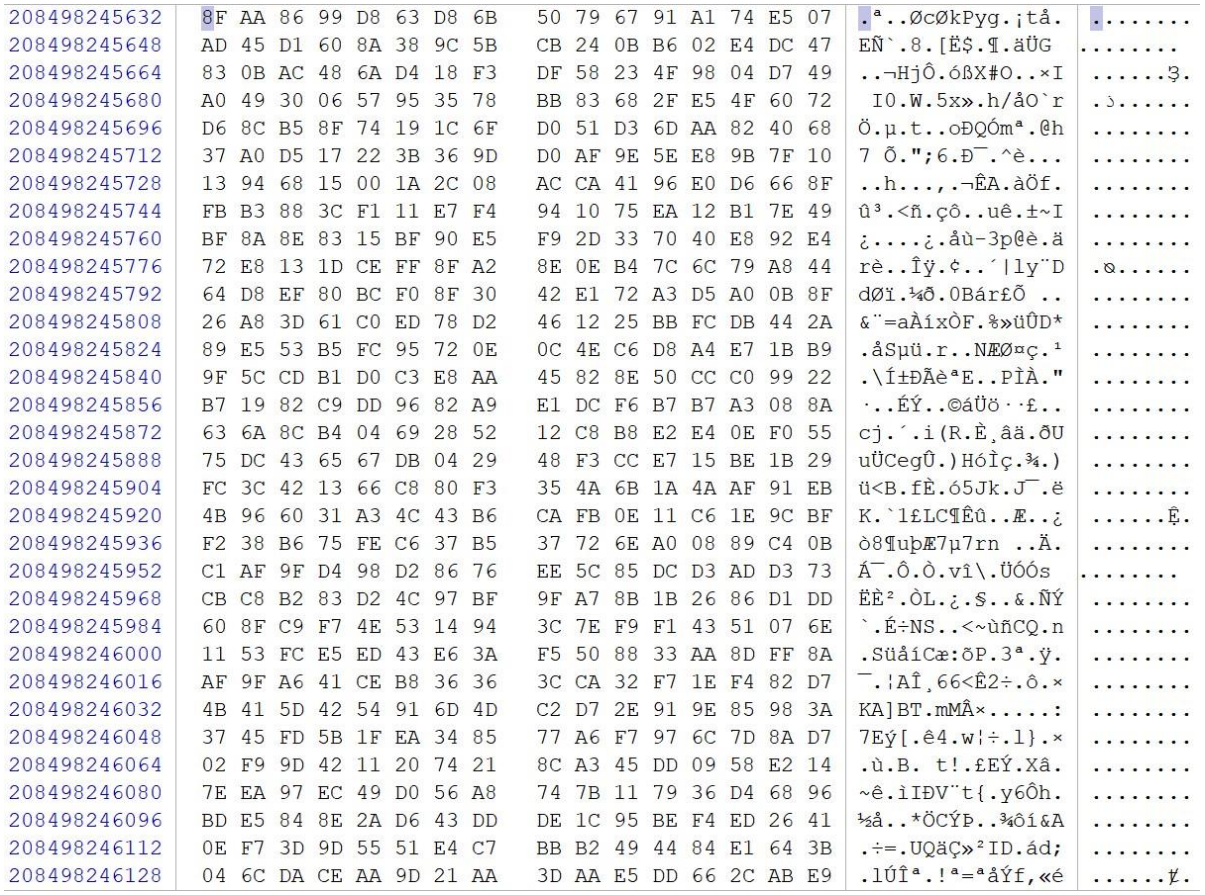
Исходя из информации, представленной на всплывающем окне, выбранный файл был успешно удален. В целях проверки работы программного средства снова используем программу «Active@ Disk Editor». С помощью нее проинспектируем записи на накопителе, содержащиеся по адресам, ранее использовавшимся для хранения данных о файле, который был удален. Запись в главной файловой таблице представлена на рисунке 5.17.

003277323264	46	49	4C	45	30	00	03	00	30	CA	44	6B	04	00	00	00	FILE0...0ÊDk....	..0.....
003277323280	0D	00	02	00	38	00	01	00	F0	01	00	00	00	04	00	00	....8...ð.....	..8.ǰ.È.
003277323296	00	00	00	00	00	00	00	00	07	00	00	00	FF	D5	00	00	............ÿÕ..	....
003277323312	03	00	00	00	00	00	00	00	10	00	00	00	60	00	00	00	............`...	......`.
003277323328	00	00	00	00	00	00	00	00	48	00	00	00	18	00	00	00	........H.....	....H...
003277323344	05	E6	0D	D6	37	22	DA	01	3D	D0	1C	D6	37	22	DA	01	.æ.Ö7"Ú.=Ð.Ö7"Ú.	...ǔ...ǔ
003277323360	CF	94	16	DB	37	22	DA	01	C7	A2	A7	D6	37	22	DA	01	Ï..Û7"Ú.Ç¢§Ö7"Ú.	...ǔ...ǔ
003277323376	20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	.....
003277323392	00	00	00	00	BC	11	00	00	00	00	00	00	00	00	00	00	....¼.....	.....
003277323408	C0	B6	7A	FE	01	00	00	00	30	00	00	00	68	00	00	00	À¶zþ....0...h...	.0..¸.h.
003277323424	00	00	00	00	00	00	02	00	50	00	00	00	18	00	01	00	........P.....	....P...
003277323440	D3	18	00	00	00	00	46	01	05	E6	0D	D6	37	22	DA	01	Ó.....F..æ.Ö7"Ú.	...ņ...ǔ
003277323456	8A	DF	19	D6	37	22	DA	01	8A	DF	19	D6	37	22	DA	01	.ß.Ö7"Ú..ß.Ö7"Ú.	...ǔ...ǔ
003277323472	8A	DF	19	D6	37	22	DA	01	00	00	00	00	00	00	00	00	.ß.Ö7"Ú.........	...ǔ....
003277323488	00	00	00	00	00	00	00	00	20	00	00	00	00	00	00	00	....	....
003277323504	07	01	32	00	2E	00	38	00	2E	00	6A	00	70	00	67	00	..2...8...j.p.g.	ć2.8.jpg
003277323520	30	00	00	00	78	00	00	00	00	00	00	00	00	00	03	00	0...x...........	0.x.....
003277323536	5A	00	00	00	18	00	01	00	D3	18	00	00	00	00	46	01	Z.......Ó.....F.	Z......ņ
003277323552	05	E6	0D	D6	37	22	DA	01	8A	DF	19	D6	37	22	DA	01	.æ.Ö7"Ú..ß.Ö7"Ú.	...ǔ...ǔ
003277323568	8A	DF	19	D6	37	22	DA	01	8A	DF	19	D6	37	22	DA	01	.ß.Ö7"Ú..ß.Ö7"Ú.	...ǔ...ǔ
003277323584	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	.....
003277323600	20	00	00	00	00	00	00	00	0C	02	32	00	38	00	36	00	.....2.8.6.	...Ò286
003277323616	39	00	34	00	33	00	7E	00	31	00	2E	00	4A	00	50	00	9.4.3.~.1...J.P.	943~1.JP
003277323632	47	00	00	00	00	00	00	00	40	00	00	00	28	00	00	00	G.......@...(...	G...@.(.
003277323648	00	00	00	00	00	00	06	00	10	00	00	00	18	00	00	00	.....	.....
003277323664	E0	43	1D	97	D8	8B	EE	11	A9	F1	EC	63	D7	73	99	7E	àC..Ø.î.©ñic×s.~	.....
003277323680	80	00	00	00	48	00	00	00	01	00	00	00	00	00	05	00	....H.....	..H.....
003277323696	00	00	00	00	00	00	00	00	02	00	00	00	00	00	00	00	.....	.....
003277323712	40	00	00	00	00	00	00	00	00	30	00	00	00	00	00	00	@.....0.....	@.....
003277323728	54	20	00	00	00	00	00	00	54	20	00	00	00	00	00	00	T .....T	.....
003277323744	31	03	BD	30	0A	00	00	00	FF	FF	FF	FF	82	79	47	11	1.½0....ÿÿÿÿ.yG.	.....
003277323760	00	00	00	00	00	00	00	00	00	00	00	00	00	00	03	00	.....	.....

Рисунок 5.17 – Запись в главной файловой таблице

При сравнении записи в главной файловой таблице до удаления файла, представленной на рисунке 5.13, и записи после удаления файла, представленной на рисунке 5.17, можно сделать вывод о том, что запись об удаляемом файле в главной файловой таблице MFT была удалена и по тому же адресу уже располагается информация о другом файле.

Данные, хранящиеся на накопителе, представлены на рисунке 5.18.



208498245632	8F AA 86 99 D8 63 D8 6B	50 79 67 91 A1 74 E5 07	.ª..ØcØkPyg.¡tå.
208498245648	AD 45 D1 60 8A 38 9C 5B	CB 24 0B B6 02 E4 DC 47	EÑ`.8.[Ë$.¶.äÜG
208498245664	83 0B AC 48 6A D4 18 F3	DF 58 23 4F 98 04 D7 49	..¬HjÔ.óßX#O..×I
208498245680	A0 49 30 06 57 95 35 78	BB 83 68 2F E5 4F 60 72	I0.W.5x».h/åO`r
208498245696	D6 8C B5 8F 74 19 1C 6F	D0 51 D3 6D AA 82 40 68	Ö.µ.t..oÐQÓmª.@h
208498245712	37 A0 D5 17 22 3B 36 9D	D0 AF 9E 5E E8 9B 7F 10	7 Õ.";6.Ð¯.^è...
208498245728	13 94 68 15 00 1A 2C 08	AC CA 41 96 E0 D6 66 8F	..h...,.¬ÊA.àÖf.
208498245744	FB B3 88 3C F1 11 E7 F4	94 10 75 EA 12 B1 7E 49	û³.<ñ.çô..uê.±~I
208498245760	BF 8A 8E 83 15 BF 90 E5	F9 2D 33 70 40 E8 92 E4	¿....¿.åù-3p@è.ä
208498245776	72 E8 13 1D CE FF 8F A2	8E 0E B4 7C 6C 79 A8 44	rè..Îÿ.¢..´|ly¨D
208498245792	64 D8 EF 80 BC F0 8F 30	42 E1 72 A3 D5 A0 0B 8F	dØï.¼ð.0Bár£Õ ..
208498245808	26 A8 3D 61 C0 ED 78 D2	46 12 25 BB FC DB 44 2A	&¨=aÀíxÒF.%»üÛD*
208498245824	89 E5 53 B5 FC 95 72 0E	0C 4E C6 D8 A4 E7 1B B9	.åSµü.r..NÆØ¤ç.¹
208498245840	9F 5C CD B1 D0 C3 E8 AA	45 82 8E 50 CC C0 99 22	.\Í±ÐÃèªE..PÌÀ."
208498245856	B7 19 82 C9 DD 96 82 A9	E1 DC F6 B7 B7 A3 08 8A	·..ÉÝ..©áÜö··£..
208498245872	63 6A 8C B4 04 69 28 52	12 C8 B8 E2 E4 0E F0 55	cj.´.i(R.È¸âä.ðU
208498245888	75 DC 43 65 67 DB 04 29	48 F3 CC E7 15 BE 1B 29	uÜCegÛ.)HóÌç.¾.)
208498245904	FC 3C 42 13 66 C8 80 F3	35 4A 6B 1A 4A AF 91 EB	ü<B.fÈ.ó5Jk.J¯.ë
208498245920	4B 96 60 31 A3 4C 43 B6	CA FB 0E 11 C6 1E 9C BF	K.`1£LC¶Êû..Æ..¿
208498245936	F2 38 B6 75 FE C6 37 B5	37 72 6E A0 08 89 C4 0B	ò8¶uþÆ7µ7rn ..Ä.
208498245952	C1 AF 9F D4 98 D2 86 76	EE 5C 85 DC D3 AD D3 73	Á¯.Ô.Ò.vî\.ÜÓÓs
208498245968	CB C8 B2 83 D2 4C 97 BF	9F A7 8B 1B 26 86 D1 DD	ËÈ².ÒL.¿.§..&.ÑÝ
208498245984	60 8F C9 F7 4E 53 14 94	3C 7E F9 F1 43 51 07 6E	`.É÷NS..<~ùñCQ.n
208498246000	11 53 FC E5 ED 43 E6 3A	F5 50 88 33 AA 8D FF 8A	.Süåí Cæ:õP.3ª.ÿ.
208498246016	AF 9F A6 41 CE B8 36 36	3C CA 32 F7 1E F4 82 D7	¯.¦AÎ¸66<Ê2÷.ô.×
208498246032	4B 41 5D 42 54 91 6D 4D	C2 D7 2E 91 9E 85 98 3A	KA]BT.mMÂ×.....:
208498246048	37 45 FD 5B 1F EA 34 85	77 A6 F7 97 6C 7D 8A D7	7Eý[.ê4.w¦÷.l}.×
208498246064	02 F9 9D 42 11 20 74 21	8C A3 45 DD 09 58 E2 14	.ù.B. t!.£EÝ.Xâ.
208498246080	7E EA 97 EC 49 D0 56 A8	74 7B 11 79 36 D4 68 96	~ê.ìIÐV¨t{.y6Ôh.
208498246096	BD E5 84 8E 2A D6 43 DD	DE 1C 95 BE F4 ED 26 41	½å..*ÖCÝÞ..¾ôí&A
208498246112	0E F7 3D 9D 55 51 E4 C7	BB B2 49 44 84 E1 64 3B	.÷=.UQäÇ»²ID.ád;
208498246128	04 6C DA CE AA 9D 21 AA	3D AA E5 DD 66 2C AB E9	.lÚÎª.!ª=ªåÝf,«é

Рисунок 5.18 – Данные, хранящиеся на накопителе

При сравнении данных файла, хранящихся на накопителе до удаления и представленных на рисунке 5.14, и данных, хранящихся на накопителе после удаления файла и представленных на рисунке 5.18, можно сделать вывод о том, что данные были уничтожены в соответствии с выбранным пользователем методом, в данном случае это алгоритм, описанный в Руководящем документе Гостехкомиссии России.

Вывод

Таким образом, было разработано программное средство для ОС Windows, обладающее следующим функционалом: уничтожение пользовательских данных; проверка успешности уничтожения данных; переименование файла, содержащего уничтожаемые данные, на последовательность псевдослучайной длины от 1 до 10 псевдослучайных символов; изменение расширения файла на exe; увеличение размера файла до значения, кратного размеру кластера; удаление файла.

Заключение

В результате проведенной работы было разработано программное средство, уничтожающее пользовательские данные в ОС Windows. Программа представлена в виде приложения для ОС Windows, для использования которого необходима его предварительная установка в операционной системе. Разработанное программное средство предлагает на выбор пользователя 19 методов уничтожения данных, включая один для конфиденциальной информации, и может применяться как физическими лицами, так и сотрудниками предприятий, организаций, компаний. Программа поддерживает уничтожение данных как с жестких дисков, так и с твердотельных накопителей. Помимо уничтожения данных, разработанное программное средство имеет дополнительный функционал, позволяющий проверить успешность проведенного уничтожения, изменить имя файла, содержащего уничтожаемые данные, его расширение и размер, а также удалить сам файл в целях затруднения программного восстановления данных.

Для достижения цели работы были выполнены следующие задачи:

– проанализированы особенности хранения и безвозвратного удаления информации с накопителей различных типов;

– проведён обзор существующих алгоритмов уничтожения данных;

– выполнен обзор и анализ программных средств для безвозвратного удаления информации в ОС Windows;

– разработан алгоритм работы программного средства;

– выполнена программная реализация.

Таким образом, поставленная цель работы была достигнута, программное средство для уничтожения пользовательских данных в ОС Windows было разработано. Из чего следует, что выполнение поставленных задач в ходе работы было информативным и корректным.

Перечень использованных информационных ресурсов

1 Чандна, С. Всё, что Вам нужно знать об удалении данных = Everything You Need To Know About Data Erasure / С. Чандна. – Текст : электронный // BitRaser : [сайт]. – 2022. – 13 июня. – URL: https://www.bitraser.com/article/what-is-data-erasure.php (дата обращения: 06.11.2023).

2 Основы жёсткого диска = Hard Disk Drive Basics. – Текст : электронный // NTFS : [сайт]. – URL: https://www.ntfs.com/hard-disk-basics.htm (дата обращения: 06.11.2023).

3 Афонин, О. Жизнь после Trim: как восстановить удалённые данные с накопителей SSD / О. Афонин. – Текст : электронный // ЭлкомСофт : [сайт]. – 2018. – 21 декабря. – URL: https://blog.elcomsoft.ru/2018/12/zhizn-posle-trim-kak-vosstanovit-udalyonnyie-dannyie-s-nakopiteley-ssd/ (дата обращения: 06.11.2023).

4 Афонин, О. Стратегии гарантированного уничтожения данных и санация накопителей / О. Афонин. – Текст : электронный // Interface : [сайт]. – 2018. – 3 сентября. – URL: https://www.interface.ru/home.asp?artId=39862 (дата обращения: 06.11.2023).

5 Вэй, М. Надёжное удаление данных с твердотельных накопителей на базе флэш-памяти = Reliably Erasing Data From Flash-Based Solid State Drives / М. Вэй, Л.М. Групп, Ф.Е. Спада, С. Суонсон. – Текст : электронный // USENIX : [сайт]. – 2011. – 16 февраля. – URL: https://www.usenix.org/legacy/events/fast11/tech/full_papers/Wei.pdf (дата обращения: 06.11.2023).

6 ГОСТ Р 50739-95. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования : государственный стандарт Российской Федерации : издание официальное : принят и введён в действие Постановлением Госстандарта России от 9 февраля 1995 г. № 49 : введён впервые : дата введения 1996-01-01. – Москва : Стандартинформ, 2006. – с. 8. – Текст : непосредственный.

7 Руководящий документ. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации : утверждён решением председателя Государственной технической комиссии при Президенте Российской Федерации от 30 марта 1992 г. – Текст : электронный // ФСТЭК России : [сайт]. – URL: https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/rukovodyashchij-dokument-ot-30-marta-1992-g-3 (дата обращения: 06.11.2023).

8 Список методов очистки данных = List of Data Sanitization Methods. – Текст : электронный // Macrorit : [сайт]. – URL: https://macrorit.com/wipe-hard-drive/list-data-sanitization-methods.html (дата обращения: 06.11.2023).

9 Фишер, Т. Методы очистки данных. Список программных методов уничтожения данных = Data Sanitization Methods. A list of software based data destruction methods / Т. Фишер. – Текст : электронный // Lifewire : [сайт]. – 2023. – 20 сентября. – URL: https://www.lifewire.com/data-sanitization-methods-2626133 (дата обращения: 06.11.2023).

10 Гутман, П. Безопасное удаление данных из магнитной и твердотельной памяти = Secure Deletion of Data from Magnetic and Solid-State Memory / П. Гутман. – Текст : электронный // University of Auckland : [сайт]. – 1996. – 22 июля. – URL: https://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html (дата обращения: 06.11.2023).

11 CCleaner : официальный сайт. – Бельгия. – Обновляется в течение суток. – URL: https://www.ccleaner.com/ru-ru/ccleaner (дата обращения: 06.11.2023). – Текст : электронный.

12 Eraser : официальный сайт. – 2020. – URL: https://eraser.heidi.ie/ (дата обращения: 06.11.2023). – Текст : электронный.

13 File Shredder : официальный сайт. – 2007. – URL: https://www.fileshredder.org/ (дата обращения: 06.11.2023). – Текст : электронный.

14 Деронич, В. Полное руководство по C++: преимущества и недостатки = A Comprehensive Guide to C++: Advantages and Disadvantages / В. Деронич. – Текст : электронный // Pangea : [сайт]. – 2023. – 30 января. – URL: https://pangea.ai/blog/languages/a-comprehensive-guide-to-c-advantages-and-disadvantages (дата обращения: 04.12.2023).

15 Хеллер, М. Что такое Visual Studio Code? Расширяемый редактор кода от Microsoft = What is Visual Studio Code? Microsoft’s extensible code editor / М. Хеллер. – Текст : электронный // InfoWorld : [сайт]. – 2022. – 8 июля. – URL: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html (дата обращения: 04.12.2023).

16 Qt : официальный сайт. – Канада. – Обновляется в течение суток. – URL: https://www.qt.io/ (дата обращения: 04.12.2023). – Текст : электронный.

17 Inno Setup : официальный сайт. – 2022. – URL: https://jrsoftware.org/isinfo.php (дата обращения: 04.12.2023). – Текст : электронный.

Приложение А

Техническое задание

"СОГЛАСОВАНО"
Доц. каф. «КБИС»,
канд. техн. наук
_____ Н.Н. Язвинская
«___» _____ 202_ г.

"УТВЕРЖДЕНО"
И.о. зав. кафедрой «КБИС»,
канд. техн. наук
_____ О.А. Сафарьян
«___» _____ 202_ г.

А.1 Введение

А.1.1 Наименование программы

Наименование темы разработки: «Программное средство для уничтожения пользовательских данных в ОС Windows».

Наименование программного средства: «Delete File Tool».

А.1.2 Область применения

Областью применения программного средства являются персональные компьютеры (ПК) с 64-разрядными версиями ОС Windows 10 или Windows 11. Программа предназначена для использования физическими лицами, сотрудниками предприятий, организаций, компаний для уничтожения пользовательских данных, не составляющих государственную тайну в соответствии с законодательством Российской Федерации.

А.2 Основания для разработки

Разработка ведется на основании учебного плана подготовки специалистов по направлению 10.05.01 «Компьютерная безопасность» и приказа об утверждении тем выпускных квалификационных работ, утвержденного Донским государственным техническим университетом.

А.3 Назначение разработки

А.3.1 Функциональное назначение

Функциональным назначением программного средства является уничтожение данных в ОС Windows и проверка отсутствия удаляемых данных на накопителе после их уничтожения.

А.3.2 Эксплуатационное назначение

Программное средство предназначено для эксплуатации физическими лицами, сотрудниками предприятий, организаций, компаний, которым необходимо удалить пользовательские данные, не составляющие государственную тайну в соответствии с законодательством Российской Федерации.

А.4 Требования к программе

А.4.1 Требования к функциональным характеристикам

Программное средство должно выполнять следующий набор функций:

– уничтожение пользовательских данных;

– проверка отсутствия удаляемых данных на накопителе после их уничтожения;
– переименование удаляемого файла на последовательность псевдослучайной длины от 1 до 10 псевдослучайных символов;
– изменение расширения исходного файла на .exe;
– увеличение размера файла до значения, кратного размеру кластера;
– удаление файла из файловой системы.

Источником входных данных для программного средства служат действия пользователя.

Выходные данные из программного средства предоставляются пользователю на всплывающих окнах.

А.4.2 Требования к надежности

Надежность функционирования программного средства обеспечивается корректным функционированием операционной системы, ее своевременным обновлением, а также невыполнением других задач параллельно работе программного средства. Программное средство должно проверять на корректность входные данные, а также выводить корректные данные или сообщения об ошибке. Время восстановления после отказа работы программного средства не должно превышать время, необходимое для перезагрузки операционной системы.

А.4.3 Условия эксплуатации

Климатические условия эксплуатации, при которых должны предоставляться заданные характеристики, обязаны удовлетворять требованиям, предъявляемым к техническим устройствам, на которых запущено программное средство. Обслуживание не требуется. Для запуска

программного средства и взаимодействия с ним необходим один человек. Требуемая квалификация пользователя – оператор ЭВМ, который ознакомился с руководством оператора и обладает базовыми навыками эксплуатации компьютерной техники.

А.4.4 Требования к составу и параметрам технических средств

Программное средство должно работать на персональном компьютере, обладающим следующими минимальными техническими характеристиками:

- двухъядерный 64-битный процессор с частотой 1 гигагерц;
- 4 ГБ оперативной памяти;
- 64 ГБ свободного места на накопителе;
- видеокарта, совместимая с DirectX 12 и WDDM 2.X;
- 9-дюймовый дисплей с разрешением 1366x768.

А.4.5 Требования к информационной и программной совместимости

Разработанное программное средство предназначено для использования в 64-разрядных версиях ОС Windows 10 и Windows 11. Базовый язык программирования – C++. Среда разработки – Visual Studio Code.

А.4.6. Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

А.4.7 Требования к транспортированию и хранению

Исходный код программного средства должен храниться на веб-сервисе GitHub.

А.4.8 Специальные требования

Специальные требования не предъявляются.

А.5 Требования к программной документации

Программная документация должна включать следующие документы:

- "Разработка программного средства для уничтожения пользовательских данных в ОС Windows". Техническое задание (ГОСТ 19.201-78);
- "Разработка программного средства для уничтожения пользовательских данных в ОС Windows". Пояснительная записка (ГОСТ 19.404-79);
- "Разработка программного средства для уничтожения пользовательских данных в ОС Windows". Руководство системного программиста (ГОСТ 19.503-79);
- "Разработка программного средства для уничтожения пользовательских данных в ОС Windows". Руководство программиста (ГОСТ 19.504-79);
- "Разработка программного средства для уничтожения пользовательских данных в ОС Windows". Руководство оператора (ГОСТ 19.505-79);
- "Разработка программного средства для уничтожения пользовательских данных в ОС Windows". Текст программы (ГОСТ 19.401-78).

А.6 Технико-экономические показатели

В рамках данной работы расчет технико-экономических показателей не предусмотрен.

А.7 Стадии и этапы разработки

К стадиям разработки данного программного средства относятся:

- системный анализ;
- проектирование;
- подготовка оборудования;
- программная реализация, разработка рабочего проекта;
- отладка программного средства и исправление недостатков.

А.8 Порядок контроля и приемки

Контроль разработки осуществляется на основе испытаний отладочных примеров. Примеры должны демонстрировать правильность работы используемых в программном средстве структур и алгоритмов в различных ситуациях, которые могут возникнуть при выполнении программы. При этом проверяется выполнение всех функций программы и полнота документации.

Прием программного средства будет утвержден при его корректной работе при различных входных данных в соответствии с пунктом А.4.1 данного документа и при предоставлении полной документации, указанной в пункте А.5 данного технического задания.

Разработчик: Следков Владислав Валерьевич

Дата начала разработки: «___» _____ 2023 г.

Приложение Б

Руководство системного программиста

Б.1 Общие сведения о программе

Программное средство «Delete File Tool» осуществляет уничтожение пользовательских данных в операционной системе (ОС) Windows. Предназначено для физических лиц, сотрудников предприятий, организаций, компаний, которым необходимо удалить пользовательские данные, не составляющие государственную тайну в соответствии с законодательством Российской Федерации, использующих персональные компьютеры (ПК) с ОС Windows.

К функциям программы относятся уничтожение пользовательских данных, удаление файлов и проверка отсутствия удаляемых данных на накопителе после их уничтожения.

Для корректной работы программного средства требуется ПК с 64-разрядной версией ОС Windows 10 или Windows 11. Также необходимо использовать твердотельные накопители либо накопители на жёстких магнитных дисках, подключенные посредством интерфейсов SATA или NVMe, и использующие NTFS в качестве файловой системы. Вследствие особенностей файловой системы NTFS, могут быть удалены только те файлы, размер которых «на диске» не равен 0. Программное средство необходимо запускать от имени администратора.

Б.2 Структура программы

Программное средство состоит из: ядра программы; графического интерфейса; модуля получения номеров кластеров, занимаемых файлом; модуля включения и проверки механизма TRIM; модуля уничтожения данных посредством различных алгоритмов; модуля проверки выполненного удаления данных.

Б.3 Настройка программы

Программное средство «Delete File Tool» не требует каких-либо настроек на условия конкретного применения, состав технических и программных средств.

Б.4 Проверка программы

Для проверки работоспособности программного средства необходимо убедиться в том, что при вызове программы осуществилась ее загрузка, и отсутствуют сообщения операционной системы о невозможности выполнения программы.

Б.5 Дополнительные возможности

К дополнительным возможностям программного средства относятся изменение расширения исходного файла на .exe; переименование удаляемого файла на последовательность псевдослучайной длины от 1 до 10 псевдослучайных символов; увеличение размера файла до значения, кратного размеру кластера; непосредственное удаление файла из файловой системы.

Б.6 Сообщения системному программисту

Для системного программиста в данном программном средстве не предусмотрены какие-либо дополнительные сообщения.

Приложение В

Руководство программиста

В.1 Назначение и условия применения программы

Программное средство «Delete File Tool» осуществляет уничтожение пользовательских данных в операционной системе (ОС) Windows. Предназначено для физических лиц, сотрудников предприятий, организаций, компаний, которым необходимо удалить пользовательские данные, не составляющие государственную тайну в соответствии с законодательством Российской Федерации, использующих персональные компьютеры (ПК) с ОС Windows.

К функциям программы относятся уничтожение пользовательских данных, удаление файлов и проверка отсутствия удаляемых данных на накопителе после их уничтожения.

Для корректной работы программного средства требуется ПК с 64-разрядной версией ОС Windows 10 или Windows 11. Также необходимо использовать твердотельные накопители либо накопители на жёстких магнитных дисках, подключенные посредством интерфейсов SATA или NVMe, и использующие NTFS в качестве файловой системы. Вследствие особенностей файловой системы NTFS, могут быть удалены только те файлы, размер которых «на диске» не равен 0.

В.2 Характеристика программы

Программное средство уничтожает пользовательские данные в операционной системе Windows, производит проверку отсутствия удаляемых данных на накопителе после их уничтожения, изменяет расширение исходного файла на .exe, переименовывает удаляемый файл на последовательность псевдослучайной длины от 1 до 10 псевдослучайных символов, выравнивает размер файла до значения, кратного размеру кластера, и удаляет файл из файловой системы.

При удалении файла программа вызывает окно с подтверждением действия; при успешном удалении файла вызывается окно с информацией об успешной попытке удаления, в противном случае – соответствующее окно.

Программное средство состоит из: ядра программы; графического интерфейса; модуля получения номеров кластеров, занимаемых файлом; модуля включения и проверки механизма TRIM; модуля уничтожения данных посредством различных методов; модуля проверки выполненного удаления данных.

В.3 Обращение к программе

Для обращения к программному средству необходима его предварительная установка в операционной системе. После успешной инсталляции на рабочем столе и в меню «Пуск» будут созданы ярлыки с названием «Delete File Tool». Запуск программного средства осуществляется путем открытия данных ярлыков от имени администратора.

В.4 Входные и выходные данные

В качестве входных данных программа получает путь к файлу, который необходимо удалить, и метод уничтожения данных, выбранный из списка доступных.

Выходными данными являются вызываемые всплывающие окна с результатом удаления файла – успешным или неудачным. Также в случае выбора пользователем метода для твердотельных накопителей, в папке с программным средством создается текстовый файл с результатом выполнения консольной команды, после чего из данного файла считываются данные, и он удаляется.

В.5 Сообщения

Для программиста в данном программном средстве не предусмотрены какие-либо дополнительные сообщения.

Приложение Г

Руководство оператора

Г.1 Назначение программы

Программное средство «Delete File Tool» осуществляет уничтожение пользовательских данных в операционной системе (ОС) Windows. Предназначено для физических лиц, сотрудников предприятий, организаций, компаний, которым необходимо удалить пользовательские данные, не составляющие государственную тайну в соответствии с законодательством Российской Федерации, использующих персональные компьютеры (ПК) с ОС Windows.

К функциям программы относятся уничтожение пользовательских данных, удаление файлов и проверка отсутствия удаляемых данных на накопителе после их уничтожения.

Г.2 Условия выполнения программы

Для корректной работы программного средства требуется ПК с 64-разрядной версией ОС Windows 10 или Windows 11. Также необходимо использовать твердотельные накопители либо накопители на жёстких магнитных дисках, подключенные посредством интерфейсов SATA или NVMe, и использующие NTFS в качестве файловой системы. Вследствие особенностей файловой системы NTFS, могут быть удалены только те файлы, размер которых «на диске» не равен 0.

Г.3 Выполнение программы

Для обращения к программному средству необходима его предварительная установка в операционной системе. После успешной

инсталляции на рабочем столе и в меню «Пуск» будут созданы ярлыки с названием «Delete File Tool». Запуск программного средства осуществляется путем открытия данных ярлыков от имени администратора.

После запуска программы следует нажать кнопку «Выбор файла» и выбрать файл, который необходимо удалить. Далее необходимо выбрать один из доступных методов уничтожения данных. По нажатию кнопки «Удалить» будет вызвано окно с просьбой подтвердить действие. При успешном удалении файла будет вызвано окно с сообщением «Файл успешно удален!» либо «Файл будет удален!», в противном случае – с сообщением «Файл не был удален!».

Г.4 Сообщения оператору

Сообщения оператору в данном программном средстве представлены в виде всплывающих окон. При успешном удалении файла вызывается окно с сообщением «Файл успешно удален!» либо «Файл будет удален!», в зависимости от выбранного метода уничтожения данных, в противном случае – окно с сообщением «Файл не был удален!». В случае если не был выбран файл, будет вызвано окно с сообщением «Выберите файл, который необходимо удалить», если не был выбран метод – вызывается окно с сообщением «Выберите метод уничтожения данных».

Приложение Д

Листинг программы

1 Листинг Д.1 – Основной исходный файл программы main.cpp

```
#include "mainwindow.h"
#include <QApplication>
#include <QStyleFactory>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    a.setStyle(QStyleFactory::create("Fusion"));
    QPalette darkPalette;
    darkPalette.setColor(QPalette::Window,   QColor(53,   53,
53));
    darkPalette.setColor(QPalette::WindowText,    QColor(230,
230, 230));
    darkPalette.setColor(QPalette::Base, QColor(33, 33, 33));
    darkPalette.setColor(QPalette::Text,   QColor(230,   230,
230));
    darkPalette.setColor(QPalette::Button,   QColor(53,   53,
53));
    darkPalette.setColor(QPalette::ButtonText,    QColor(230,
230, 230));
    darkPalette.setColor(QPalette::Highlight,     QColor(230,
230, 230));
    darkPalette.setColor(QPalette::HighlightedText,
QColor(53, 53, 53));
    darkPalette.setColor(QPalette::PlaceholderText,
QColor(230, 230, 230));
    a.setPalette(darkPalette);
    MainWindow w;
    w.show();
    return a.exec();
}
```

2 Листинг Д.2 – Заголовочный файл главного окна mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QFileDialog>
#include <QMessageBox>
#include <windows.h>
#include <winioctl.h>
```

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <fstream>
using namespace std;

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    bool TRIMstatus();

    string SelectedMethod();

    ULONGLONG    *GetClusters(PCHAR    lpFileName,    ULONG
ClusterSize, ULONG *ClCount, ULONG *FileSize);

    BOOL DelFile(PCHAR lpSrcName, string Type);

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

3 Листинг Д.3 – Файл с исходным кодом главного окна mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
```

```
MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    QString str = QFileDialog::getOpenFileName(this, "Выберите
файл для удаления", "C:\\");
    ui->lineEdit->setText(str);
}

void MainWindow::on_pushButton_2_clicked()
{
    QString PathToFile = ui->lineEdit->text();
    if(PathToFile != "" and SelectedMethod() != "")
    {
        QMessageBox::StandardButton          sure          =
QMessageBox::question(this, "Delete File Tool", "Указанный файл
будет удален. Вы уверены?", QMessageBox::Yes | QMessageBox::No);
        if(sure == QMessageBox::Yes)
        {
            bool                  isDeleted                  =
MainWindow::DelFile(PathToFile.toLocal8Bit().data(),
SelectedMethod());
            if(SelectedMethod() == "SSD" and isDeleted)
            {
                QMessageBox::information(this, "Успех", "Файл
будет удален!");
                ui->lineEdit->setText("");
            }
            else if(SelectedMethod() != "SSD" and isDeleted)
            {
                QMessageBox::information(this, "Успех", "Файл
успешно удален!");
                ui->lineEdit->setText("");
            }
            else
            {
                QMessageBox::critical(this,  "Ошибка",  "Файл
не был удален!");
            }
        }
    }
    else if(PathToFile == "")
    {
        QMessageBox::critical(this, "Ошибка", "Выберите файл,
который необходимо удалить");
    }
    else
    {
```

```
            QMessageBox::critical(this, "Ошибка", "Выберите метод
уничтожения данных");
        }
    }

    bool MainWindow::TRIMstatus()
    {
        system("fsutil behavior set disabledeletenotify NTFS 0 >
1.txt");
        string NTFS;
        ifstream in("1.txt");
        getline (in, NTFS);
        in.close();
        remove("1.txt");
        if (NTFS[27] == '0')
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    string MainWindow::SelectedMethod()
    {
        if(ui->t1SSD->isChecked())
        {
            return "SSD";
        }
        else if(ui->t2Nulls->isChecked())
        {
            return "Nulls";
        }
        else if(ui->t3Randoms->isChecked())
        {
            return "Randoms";
        }
        else if(ui->t4GOST->isChecked())
        {
            return "GOST";
        }
        else if(ui->t5NZSIT402->isChecked())
        {
            return "NZSIT402";
        }
        else if(ui->t6VSITR->isChecked())
        {
            return "VSITR";
        }
        else if(ui->t7DODmE->isChecked())
        {
```

```
        return "DOD5220.22-mE";
    }
    else if(ui->t8DODmECE->isChecked())
    {
        return "DOD5220.22-mECE";
    }
    else if(ui->t9NCSC->isChecked())
    {
        return "NCSC-TG-025";
    }
    else if(ui->t10AFSSI5020->isChecked())
    {
        return "AFSSI-5020";
    }
    else if(ui->t11Navso->isChecked())
    {
        return "NavsoP-5239-26";
    }
    else if(ui->t12AR38019->isChecked())
    {
        return "AR380-19";
    }
    else if(ui->t13CSEC->isChecked())
    {
        return "CSEC-ITSG-06";
    }
    else if(ui->t14HMGis5->isChecked())
    {
        return "HMGis5";
    }
    else if(ui->t15RCMP->isChecked())
    {
        return "RCMPtssitOPS-II";
    }
    else if(ui->t16Pfitzner7->isChecked())
    {
        return "Pfitzner7";
    }
    else if(ui->t17Pfitzner33->isChecked())
    {
        return "Pfitzner33";
    }
    else if(ui->t18Schneier->isChecked())
    {
        return "Schneier";
    }
    else if(ui->t19Gutmann->isChecked())
    {
        return "Gutmann";
    }
    else
    {
```

```
            return "";
        }
    }

    ULONGLONG*  MainWindow::GetClusters(PCHAR  file_name,  ULONG
size_of_cluster, ULONG *clusters_count, ULONG *file_size){
        HANDLE                          file_handle;
        ULONG                           output_buffer_size;
        ULONG                            output_bytes, k, cn_count,
m;
        ULONGLONG                       *clusters = NULL;
        LARGE_INTEGER                   previous_VCN, Lcn;
        STARTING_VCN_INPUT_BUFFER       input_buffer;
        PRETRIEVAL_POINTERS_BUFFER      output_buffer;

        file_handle=CreateFileA(file_name,  FILE_READ_ATTRIBUTES,
FILE_SHARE_READ  |  FILE_SHARE_WRITE  |  FILE_SHARE_DELETE,  NULL,
OPEN_EXISTING, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
            *file_size=GetFileSize(file_handle, NULL);

output_buffer_size=sizeof(RETRIEVAL_POINTERS_BUFFER)+(*file_size/
size_of_cluster)*sizeof(output_buffer->Extents);

output_buffer=(PRETRIEVAL_POINTERS_BUFFER)malloc(output_buffer_si
ze);
            input_buffer.StartingVcn.QuadPart=0;
            if                      (DeviceIoControl(file_handle,
FSCTL_GET_RETRIEVAL_POINTERS, &input_buffer, sizeof(input_buffer),
output_buffer, output_buffer_size, &output_bytes, NULL)){
                *clusters_count=(*file_size+size_of_cluster-
1)/size_of_cluster;
                clusters=(ULONGLONG
*)malloc(*clusters_count*sizeof(ULONGLONG));
                previous_VCN=output_buffer->StartingVcn;
                for (m=0, k=0; m<output_buffer->ExtentCount; m++){
                    Lcn=output_buffer->Extents[m].Lcn;
                    for          (cn_count=(ULONG)(output_buffer-
>Extents[m].NextVcn.QuadPart-previous_VCN.QuadPart);
                          cn_count;        cn_count--,        k++,
Lcn.QuadPart++) clusters[k]=Lcn.QuadPart;
                    previous_VCN=output_buffer-
>Extents[m].NextVcn;
                }
            }
            free(output_buffer);
            CloseHandle(file_handle);
        }
        return clusters;
    }
```

```
     BOOL MainWindow::DelFile(PCHAR source_name, string type){
       ULONG         size_of_cluster, block_size;
       ULONGLONG     *clusters;
       ULONG              clusters_count,  file_size,  file_size_2,
output_bytes;
       HANDLE        drive_handle, file_handle;
       ULONG         sectors_per_cluster, bytes_per_cluster, m, j;
       PVOID         buffer;
       LARGE_INTEGER offset;
       CHAR          drive[7];
       BOOL          isDeleted = true;
       CONST CHAR    *new_name;
       string        new_name_2, new_name_3;
       size_t        found;
       string                                          alphabet     =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
       int           i;

       srand (time(0));
       drive[0]=source_name[0];
       drive[1]=':';
       drive[2]=0;
       GetDiskFreeSpaceA(drive,              &sectors_per_cluster,
&bytes_per_cluster, NULL, NULL);
       size_of_cluster=sectors_per_cluster * bytes_per_cluster;
       file_handle=CreateFileA(source_name,     GENERIC_READ     |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, 0);
       if(file_handle == INVALID_HANDLE_VALUE)
       {
             isDeleted = false;
             return isDeleted;
       }
       CloseHandle(file_handle);
       clusters=MainWindow::GetClusters(source_name,
size_of_cluster, &clusters_count, &file_size);
       ULONGLONG                          PartSign[clusters_count],
PartSign2[clusters_count];
       CHAR          str[size_of_cluster+2];
       if (clusters){
         drive[0]='\\';
         drive[1]='\\';
         drive[2]='.';
         drive[3]='\\';
         drive[4]=source_name[0];
         drive[5]=':';
         drive[6]=0;
         drive_handle    =    CreateFileA(drive,    GENERIC_READ,
FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, 0);
         if (drive_handle!=INVALID_HANDLE_VALUE){
           buffer=malloc(size_of_cluster);
           file_size_2 = file_size;
```

```
            for    (m=0;    m<clusters_count;    m++,    file_size_2-
=block_size){
              offset.QuadPart=size_of_cluster*clusters[m];
              SetFilePointer(drive_handle,          offset.LowPart,
&offset.HighPart, FILE_BEGIN);
              ReadFile(drive_handle,    buffer,    size_of_cluster,
&output_bytes, NULL);
              block_size=file_size_2 < size_of_cluster ? file_size_2
: size_of_cluster;
              PartSign[m] = *(ULONGLONG*) buffer;
            }
            free(buffer);
          }
          CloseHandle(drive_handle);
        }
        else{
          isDeleted = false;
          return isDeleted;
        }

        if (type == "Nulls"){
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 0, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
        }

        if (type == "Randoms" or type == "NZSIT402"){
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              for (j=0; j<size_of_cluster; j++){
                  str[j] = rand()%256;
              }
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
        }

        if (type == "GOST"){
          for (i=0; i<2; i++){
```

```
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                  for (j=0; j<size_of_cluster; j++){
                      str[j] = rand()%256;
                  }
                  WriteFile(file_handle,   &str,   size_of_cluster,
NULL, NULL);
              }
            }
            CloseHandle(file_handle);
          }
        }

        if (type == "VSITR"){
          for (j=0; j<3; j++){
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                  memset(str, 0, size_of_cluster);
                  WriteFile(file_handle,   &str,   size_of_cluster,
NULL, NULL);
              }
            }
            CloseHandle(file_handle);
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                  memset(str, 255, size_of_cluster);
                  WriteFile(file_handle,   &str,   size_of_cluster,
NULL, NULL);
              }
            }
            CloseHandle(file_handle);
          }
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 170, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
```

```
        }

        if (type == "DOD5220.22-mE" or type == "NCSC-TG-025" or type
== "AFSSI-5020" or type == "NavsoP-5239-26" or type == "CSEC-ITSG-
06" or type == "HMGis5"){
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 0, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 255, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              for (j=0; j<size_of_cluster; j++){
                  str[j] = rand()%256;
              }
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
        }

        if (type == "DOD5220.22-mECE"){
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 0, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
```

```
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 255, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          for (i=0; i<2; i++){
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                  for (j=0; j<size_of_cluster; j++){
                      str[j] = rand()%256;
                  }
                  WriteFile(file_handle,   &str,   size_of_cluster,
NULL, NULL);
              }
            }
            CloseHandle(file_handle);
          }
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 0, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 255, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
```

```
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              for (j=0; j<size_of_cluster; j++){
                str[j] = rand()%256;
              }
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
        }

        if (type == "RCMPtssitOPS-II"){
          for (j=0; j<3; j++){
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                memset(str, 0, size_of_cluster);
                WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
              }
            }
            CloseHandle(file_handle);
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                memset(str, 255, size_of_cluster);
                WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
              }
            }
            CloseHandle(file_handle);
          }
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              for (j=0; j<size_of_cluster; j++){
                str[j] = rand()%256;
              }
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
```

```
          CloseHandle(file_handle);
        }

        if (type == "AR380-19"){
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              for (j=0; j<size_of_cluster; j++){
                str[j] = rand()%256;
              }
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 255, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 0, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
        }

        if (type == "Pfitzner7"){
          for (i=0; i<7; i++){
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                for (j=0; j<size_of_cluster; j++){
                      str[j] = rand()%256;
                }
```

```
                WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
              }
            }
            CloseHandle(file_handle);
          }
        }

        if (type == "Pfitzner33"){
          for (i=0; i<33; i++){
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                for (j=0; j<size_of_cluster; j++){
                      str[j] = rand()%256;
                }
                WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
              }
            }
            CloseHandle(file_handle);
          }
        }

        if (type == "Schneier"){
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 255, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 0, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          for (i=0; i<5; i++){
```

```
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                for (j=0; j<size_of_cluster; j++){
                      str[j] = rand()%256;
                }
                WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
              }
            }
            CloseHandle(file_handle);
          }
        }

        if (type == "Gutmann"){
          for (i=0; i<4; i++){
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                for (j=0; j<size_of_cluster; j++){
                      str[j] = rand()%256;
                }
                WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
              }
            }
            CloseHandle(file_handle);
          }
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 85, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
              memset(str, 170, size_of_cluster);
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
```

```
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for (m=0; m<size_of_cluster; m+=3){
              str[m] = 146;
              str[m+1] = 73;
              str[m+2] = 36;
            }
            for(m=0; m<clusters_count; m++){
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for (m=0; m<size_of_cluster; m+=3){
              str[m] = 73;
              str[m+1] = 36;
              str[m+2] = 146;
            }
            for(m=0; m<clusters_count; m++){
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for (m=0; m<size_of_cluster; m+=3){
              str[m] = 36;
              str[m+1] = 146;
              str[m+2] = 73;
            }
            for(m=0; m<clusters_count; m++){
              WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
            }
          }
          CloseHandle(file_handle);
          file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
          if (file_handle!=INVALID_HANDLE_VALUE){
            for(m=0; m<clusters_count; m++){
```

```
            memset(str, 0, size_of_cluster);
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 17, size_of_cluster);
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 34, size_of_cluster);
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 51, size_of_cluster);
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 68, size_of_cluster);
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
```

```
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 85, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 102, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 119, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 136, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 153, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
```

```
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 170, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 187, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 204, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 221, size_of_cluster);
            WriteFile(file_handle,  &str,  size_of_cluster,  NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
```

```
            memset(str, 238, size_of_cluster);
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for(m=0; m<clusters_count; m++){
            memset(str, 255, size_of_cluster);
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for (m=0; m<size_of_cluster; m+=3){
            str[m] = 146;
            str[m+1] = 73;
            str[m+2] = 36;
          }
          for(m=0; m<clusters_count; m++){
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for (m=0; m<size_of_cluster; m+=3){
            str[m] = 73;
            str[m+1] = 36;
            str[m+2] = 146;
          }
          for(m=0; m<clusters_count; m++){
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for (m=0; m<size_of_cluster; m+=3){
```

```
            str[m] = 36;
            str[m+1] = 146;
            str[m+2] = 73;
          }
          for(m=0; m<clusters_count; m++){
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for (m=0; m<size_of_cluster; m+=3){
            str[m] = 109;
            str[m+1] = 182;
            str[m+2] = 219;
          }
          for(m=0; m<clusters_count; m++){
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for (m=0; m<size_of_cluster; m+=3){
            str[m] = 182;
            str[m+1] = 219;
            str[m+2] = 109;
          }
          for(m=0; m<clusters_count; m++){
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
          }
        }
        CloseHandle(file_handle);
        file_handle=CreateFileA(source_name,    GENERIC_READ    |
GENERIC_WRITE,    FILE_SHARE_DELETE    |    FILE_SHARE_READ    |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
        if (file_handle!=INVALID_HANDLE_VALUE){
          for (m=0; m<size_of_cluster; m+=3){
            str[m] = 219;
            str[m+1] = 109;
            str[m+2] = 182;
          }
          for(m=0; m<clusters_count; m++){
            WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
```

```
            }
          }
          CloseHandle(file_handle);
          for (i=0; i<4; i++){
            file_handle=CreateFileA(source_name,   GENERIC_READ   |
GENERIC_WRITE,     FILE_SHARE_DELETE     |     FILE_SHARE_READ     |
FILE_SHARE_WRITE, NULL, OPEN_ALWAYS, 0, 0);
            if (file_handle!=INVALID_HANDLE_VALUE){
              for(m=0; m<clusters_count; m++){
                for (j=0; j<size_of_cluster; j++){
                      str[j] = rand()%256;
                }
                WriteFile(file_handle, &str, size_of_cluster, NULL,
NULL);
              }
            }
            CloseHandle(file_handle);
          }
        }

        drive_handle     =     CreateFileA(drive,     GENERIC_READ,
FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, 0);
        if (drive_handle!=INVALID_HANDLE_VALUE){
            buffer=malloc(size_of_cluster);
            file_size_2 = file_size;
            block_size = 0;
            for    (m=0;    m<clusters_count;    m++,    file_size_2-
=block_size){
              offset.QuadPart=size_of_cluster*clusters[m];
              SetFilePointer(drive_handle,          offset.LowPart,
&offset.HighPart, FILE_BEGIN);
              ReadFile(drive_handle,    buffer,    size_of_cluster,
&output_bytes, NULL);
              block_size=file_size_2 < size_of_cluster ? file_size_2
: size_of_cluster;
              PartSign2[m] = *(ULONGLONG*) buffer;
            }
            free(buffer);
        }
        CloseHandle(drive_handle);

        for (m=0; m<clusters_count; m++){
          if (PartSign[m] == PartSign2[m]){
              isDeleted = false;
          }
          PartSign[m] = 0;
          PartSign2[m] = 0;
        };

        if (type == "SSD"){
          if (MainWindow::TRIMstatus()){
            isDeleted = true;
```

```
        }
      }

      free(clusters);
      if (isDeleted){
        new_name_2 = source_name;
        found = new_name_2.find_last_of("/\\");
        for (int j=0; j<((rand()%10)+1); j++){
            new_name_3 += alphabet[rand()%62];
        };
        new_name_2 = new_name_2.substr(0,found) + "/" + new_name_3
+ ".exe";
        new_name = new_name_2.c_str();
        rename(source_name, new_name);
        remove(new_name);
      }
      return isDeleted;
    }
```

4 Листинг Д.4 – Файл проекта DFT.pro

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++17

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

win32:RC_FILE = resources.rc

qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

5 Листинг Д.5 – Файл коллекции ресурсов resources.rc

```
IDI_ICON1 ICON DISCARDABLE "icon.ico"
```