Оглавление

[ВВЕДЕНИЕ 3](#_Toc138278879)

[ГЛАВА 1 ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc138278880)

[1.1 Способы безвозвратного физического уничтожения информации 5](#_Toc138278881)

[1.2 Способы безвозвратного программного уничтожения информации 7](#_Toc138278882)

[1.3 Теоретические сведения о хранении файлов 8](#_Toc138278883)

[ГЛАВА 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРИЛОЖЕНИЯ 11](#_Toc138278884)

[1.1 Обзор алгоритмов безвозвратного программного уничтожения информации 11](#_Toc138278885)

[1.2 Выбор языка для разработки программного средства 13](#_Toc138278886)

[1.3 Разработка интерфейса программы 13](#_Toc138278887)

[1.4 Реализация стирания файлов 14](#_Toc138278888)

[ГЛАВА 3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРИЛОЖЕНИЯ 16](#_Toc138278889)

[ЗАКЛЮЧЕНИЕ 17](#_Toc138278890)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc138278891)

[ПРИЛОЖЕНИЕ А 19](#_Toc138278892)

# ВВЕДЕНИЕ

Благодаря научно-техническому прогрессу контролировать данные с каждым годом становится все сложнее. И дело отнюдь не только в развитии сетевых технологий. Если раньше, чтобы быть уверенным в собственной безопасности достаточно было сжечь несколько писем пикантного содержания, то сегодня удаляя компрометирующие вас документы с жесткого диска или SSD нельзя быть на 100% уверенным, что их не удастся восстановить.

Безвозвратное удаление данных – это уничтожению данных с жесткого диска или внешних носителях без возможности восстановления с помощью аппаратных средств или программного обеспечения. Процедура уничтожения данных необходима в случае утилизации поврежденного диска, его продажи или передачи другому пользователю для сохранения личной и секретной информации. При удалении файла операционная система стирает только указатель на этот файл в памяти, а пространство жесткого считается свободным. Пока не выполнится запись другой информации в этот отсек, прежние данные можно восстановить.

Навсегда удалить данные можно двумя способами: физическим и посредством специальной программы. Физическое уничтожение данных заключается в применении устройств на магнитных носителях. Жесткий диск после процедуры размагничивания становится непригодным для последующего использования. Если предполагается дальнейшая эксплуатация диска, то выбирается программа по удалению данных. Процесс очистки файлов и разделов осуществляется в ходе загрузки компьютера после того, как прогрузится операционная система. Защита от восстановления удаленных данных обеспечивается путем удаления всего файла (не только заголовка) и многократной перезаписи диска случайным набором байтов (шумом). В результате чего данные становится невозможно восстановить по оставшимся частям или по остаточной намагниченности. Необходимо помнить, что очищающей программой нельзя воспользоваться при поврежденном диске. Программы по удалению данных способны стирать информацию как с отдельных разделов, так и с целого диска. Программное обеспечение для уничтожения информации с компьютера простое в установке и использовании, есть много бесплатных решений, отлично справляющихся с возложенной задачей.

В данной программе сделана попытка объединить только самые нужные для пользователя функции по работе с файлами в наглядном и простом виде.

Целью курсового проекта является безвозвратное удаление файлов на языке Delphi.

Задачи:

1. Изучить материал, который необходим для выполнения работы
2. Разработать программу
3. Протестировать программу

Для выполнения поставленной задачи, необходимо современное оборудование, которое позволит комфортно работать. Программа разрабатывалась на операционной системе Windows 10, в среде разработки Embarcadero RAD Studio 11.

Объект исследования: использование среды Embarcadero RAD Studio 11, возможностей языка Delphi и его библиотек для решения поставленной задачи.

Предмет исследования: безвозвратное удаление файлов без возможности восстановления.

# ГЛАВА 1 ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Способы безвозвратного физического уничтожения информации

В сочетании с предварительным софтверным удалением файлов, физическое уничтожение накопителя полностью исключает вероятность восстановления данных с HDD даже с применением спецсредств. Способов расправиться с жестким диском существует немало, однако все их можно разделить на три обширные группы:

**Механическое уничтожение**

Нас интересует сугубо уничтожение магнитных пластин, так как DRAM-буфер, в котором кэшируются данные, очищается автоматически после обесточивания HDD, а единственное ПЗУ, имеющееся у винчестера, используется лишь для хранения на нем микропрограммы. Сразу оговоримся, что бить по жесткому диску кувалдой практически бесполезно: да, вы сможете уничтожить текстолитовые платы, разбить корпус (его пластиковую часть) и серьезно повредить блины, однако у профессионала все еще останутся шансы на восстановление информации. Чтобы достичь нужного результата, можно прибегнуть к услугам специализированного шредера.

Главная проблема таких устройств заключается в том, что они чрезвычайно дороги и занимают уйму места. Например, даже за самый простой агрегат уровня Ideal 0101 HDP придется выложить более 200 тысяч рублей, а ведь все, что умеет такой «малыш» с габаритными размерами 850 x 495 x 470 мм и весом 63 кг — проделывать в жестких дисках сквозные отверстия, что хотя и выводит винчестеры из строя, не может являться 100% гарантией уничтожения данных.

**Термическое уничтожение**

По сравнению с механической переработкой этот метод имеет одно весьма интересное преимущество: выбрав его, вы можете не беспокоиться о предварительном стирании данных с помощью специализированных утилит. Давайте вспомним определение ферромагнетика. Ферромагнитные материалы представляют собой кристаллические соединения, способные сохранять намагниченность в условиях отсутствия внешнего магнитного поля при температуре ниже точки Кюри. По достижении последней ферромагнетик утрачивает свои свойства, так что вся записанная информация оказывается гарантировано удалена.

Далее — простая математика:

* точка Кюри ферромагнитного покрытия на основе оксида хрома, которое используется в жестких дисках, составляет около 386 градусов Кельвина, то есть 113°С;
* температура плавления алюминия, из которого состоят сами пластины, составляет 660°С;
* температура пламени пропан-бутановой горелки, которую можно приобрести в любом строительном магазине, достигает 1300°С.

Плюсы такого подхода — ультрабюджетность и возможность «обработки» накопителя в домашних условиях, минусы — длительность процедуры и пожароопасность.

**Химическое уничтожение**

Смесь концентрированных азотной и соляной кислот, взятых в соотношении 1:3, более известная под названием «царская водка», способна без труда растворить жесткий диск даже целиком, не оставив и следа ни от сохраненных данных, ни от самого накопителя. Впрочем, HNO3 неплохо справляется с покрытием блинов даже соло.

Все перечисленные нами методики достаточно эффективны, хотя и не лишены ряда недостатков. Во-первых, подобные манипуляции потенциально травмоопасны, особенно в условиях нехватки времени: если в спешке можно серьезно повредить руку даже шлифовальной машинкой, то что уж говорить о кислоте или горелке? Во-вторых, вам потребуется закупить специальное оборудование, а для работы с кислотой так и вовсе иметь мини-лабораторию с хорошей вытяжкой. В-третьих, они мало подходят для ситуаций, когда «за вами уже выехали»: любой из приведенных методов занимает достаточно времени, часть которого придется потратить на разборку винчестера.

Но что, если проявить толику смекалки, заранее выкрутить все болты, удерживающие крышку HDD, и посадить ее на скотч, чтобы в случае чего быстро сорвать и извлечь блины? Хотя это действительно поможет сэкономить драгоценное время, здравой подобную идею не назовешь.

Если речь идет о «воздушном» накопителе (к таковым относятся модели емкостью до 6 ТБ включительно), вскрытие устройства негативно скажется на его надежности. Пускай такие винчестеры и не являются герметичными, даже они собираются в стерильных условиях, а барометрическое отверстие, присутствующее в корпусе каждого диска, и предназначенное для выравнивания давления, надежно защищено микрофильтром, задерживающим влагу и мелкодисперсные частицы. Кстати, для большей надежности внутрь HDD устанавливается еще и фильтр рециркуляции, обеспечивающий очистку воздуха в гермозоне во время работы устройства.

После вскрытия диска внутрь гермозоны неизбежно станет просачиваться пыль, что сперва приведет к появлению множественных ошибок чтения/записи, обусловленных нагревом сенсоров под действием силы трения, возникающей, когда движущаяся на огромной скорости пыль станет задевать их поверхность, а затем и к полному выходу винчестера из строя, поскольку мелкодисперсные частицы, подобно абразиву, будут повреждать слой ферромагнетика и стачивать сами головки чтения/записи. Ведь не стоит забывать, что размеры последних у современных жестких дисков не превышают 120 нанометров, а зазор между сенсорами и поверхностью блина составляет всего 12–15 нм, вследствие чего винчестеры чрезвычайно уязвимы.

## 1.2 Способы безвозвратного программного уничтожения информации

При удалении файлов через стандартный менеджер якобы стертая информация на самом деле никуда не девается. Все, что при этом происходит — присвоение соответствующим секторам статуса свободных, чтобы в будущем они были перезаписаны. Однако до тех пор, пока этого не произошло, файлы можно без особого труда восстановить при помощи специализированных программ.

Вывод напрашивается сам собой: для того, чтобы гарантированно удалить тот или иной файл, нам следует инициировать его принудительную перезапись. Для этих целей существует немало утилит, однако проще всего воспользоваться штатными средствами вашей операционной системы.

**Windows**

Начиная с Windows Vista операционные системы Microsoft по умолчанию однократно перезаписывают содержимое выбранного раздела нулями во время его полного форматирования. Количество циклов перезаписи можно задать и самостоятельно, однако для этого придется прибегнуть к услугам командной строки:

format d: /fs:NTFS /p:10

В данном примере мы форматируем том D в файловую систему NTFS, при этом даем операционной системе команду предварительно перезаписать весь раздел нулями 10 раз, используя параметр /p («passes»), задающий количество проходов. Перезаписать конкретный файл или каталог прицельно Windows, к сожалению, не позволяет. Впрочем, рассматривать это как серьезный недостаток нельзя: если уж и хранить конфиденциальную информацию, которую в какой-то момент может понадобится уничтожить, на ПК, то делать это следует, как минимум, на специально выделенном для этого разделе, а лучше всего — на отдельном накопителе, который можно будет быстро очистить непосредственно из вашей ОС.

**Linux**

Операционные системы семейства Linux традиционно предлагают своим пользователям куда больше простора. Так, например, если вам понадобилось быстро и надежно удалить один или несколько файлов, вы можете воспользоваться консольной утилитой shred:

shred -u -n 40 /private/file.txt

В этом примере мы перезаписали файл file.txt случайными данными 40 раз подряд (параметр -n, по умолчанию же shred совершает 25 проходов) и затем удалили его (параметр -u, поскольку по умолчанию shred не удаляет файлы, а сразу перезаписывает). Также утилита поддерживает удаление файлов по маске (например, \*txt) или обработку сразу нескольких файлов подряд в рамках одной команды — для этого необходимо указать ведущие к ним абсолютные пути, разделенные пробелами:

shred -u -n 40 /private/file.txt /private/file2.txt

Если же вы хотите дополнительно скрыть и сам факт «зачистки», то можете воспользоваться параметром -z:

shred -u -z -n 40 /private/file.txt

Единственный недостаток shred — данная утилита не умеет работать с директориями. Поэтому если вам требуется уничтожить содержимое целого каталога с вложенными папками, стоит воспользоваться wipe:

wipe -rf /private/\*

В данном примере мы отдали команду рекурсивно (-r) удалить все, что находится в папке private, а также отключили запрос подтверждения удаления папок и файлов (-f), которое wipe выводит по умолчанию. Среди интересных дополнительных опций, поддерживаемых утилитой, стоит выделить принудительное выполнение chmod для каталогов и файлов, защищенных от перезаписи (-c), работу в тихом режиме (-s) и возможность заставить wipe следовать символическим ссылкам (-D, конфликтует с -r).

**Mac OS**

В версиях операционной системы с 10.4 по 10.10 вы могли инициировать перезапись файлов, переместив их в корзину, а затем выбрав в Finder опцию «Очистить корзину необратимо», однако впоследствии Apple от нее отказалась. Вплоть до версии 10.11 для безопасного удаления данных можно было воспользоваться консольной командой srm:

srm -rm /private/

В данном примере мы рекурсивно удаляем содержимое папки private (-r), перезаписывая каждый файл случайными данными 7 раз подряд (-m). Также можно выбрать однократную перезапись файлов (-s) или же не указывать этот параметр вовсе — в этом случае утилита выполнит 35 проходов.

В актуальных версиях Mac OS инициировать безопасное удаление файлов с жесткого диска можно с помощью Дисковой утилиты (для этого необходимо нажать на кнопку «Параметры безопасности» и задать количество циклов перезаписи), либо через терминал, используя конструкцию вида:

rm -P /private/file.txt

Параметр -P является специфичным для Mac OS и сообщает утилите, что удаляемые файлы необходимо трижды перезаписать.

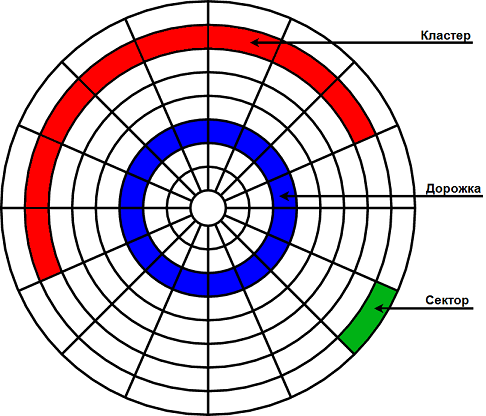
По умолчанию однократную перезапись файлов предлагает лишь встроенная в ОС Windows утилита format, тогда как все прочие средства безопасного удаления файлов совершают от 3 до нескольких десятков прогонов за раз. Аналогичным образом обстоят дела с бесплатным и коммерческим софтом от сторонних разработчиков.

## 1.3 Теоретические сведения о хранении файлов

В упрощенном виде жесткий диск можно представить как круг, который поделен радиальными дорожками. Каждая дорожка, свою очередь, делится на сектора. Сектора могут объединяться в кластеры. Минимальная адресуемая единица — сектор, его размер обычно составляет 512 байт. Схема расположения секторов представлена на рисунке 1.

Рисунок 1.

Схема расположения секторов

[](https://club.directum.ru/uploads/images/41-0.png_d9482d09-fedc-4590-a6f6-d5a635ac0cac.png)

Файл на жестком диске хранится «по частям». Т.е. если размер файла больше, чем размер сектора, то тело файла распределяется по свободным секторам диска. А информация о размещении файла на диске записывается в специальный раздел диска — таблицу размещения файлов.

При удалении файла тело файла остается на диске (в большинстве файловых систем). В таблице размещения файлов соответствующий файл помечается как удаленный, а занятые им сектора становятся «свободными» для записи. Но само тело файла в этих секторах остается. Если эти сектора ничем не занять, то удаленный файл вполне можно восстановить. Этот принцип лежит в основе работы множества утилит восстановления удаленных файлов.

Но даже если тело удаленного файла перезаписать, например, «нулями». То по-прежнему существует возможность восстановления информации из-за эффекта остаточной намагниченности.

Допустим, логические «1» и «0» характеризуются аналоговыми уровнями намагниченности +10 и -10 единиц, соответственно. То есть если головка диска считывает уровень +10, то это «1», если -10, то это «0» — происходит преобразование аналогового сигнала в цифровой.  
Аналоговый сигнал будет далек от «идеальных» +10 и -10. На самом деле у него будет какой-то разброс, к примеру от +8,5 до +11,5 – это будет «1». А от -11 до -9 – «0». Разброс будет обусловлен многими факторами, в том числе и эффектом остаточной намагниченности.

Рассмотрим пример. Какой-то бит информации закодирован как «1» и с него считывается +9,8 единиц. Затем этот бит перезаписали в «0». С него стало считываться не ровно -10, а допустим -9 единиц. Разница аналогового сигнала в +1 единицу от «идеального» значения может говорить о том, что ранее этот бит скорее всего имел противоположное значение. Таким образом, можно попытаться восстановить информацию, конечно, если для этого есть соответствующий инструмент. Но чем плотнее друг к другу находятся биты информации на диске, тем менее вероятно восстановление.

Для уменьшения или исключения вероятности восстановления информации были разработаны алгоритмы безвозвратного удаления информации. Суть всех алгоритмов схожа — нужно перезаписать участки, на которых находился файл, некоторой информацией несколько раз.

В процессе исследования предметной области были рассмотрены способы безвозвратного физического уничтожения информации и способы безвозвратного программного уничтожения информации, а также теоретические сведения о хранении файлов, способы определения содержимого диска после обычного стирания файлов различными операционными системами.

# ГЛАВА 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРИЛОЖЕНИЯ

## 1.1 Обзор алгоритмов безвозвратного программного уничтожения информации

**Американский стандарт DoD (Department of Defence) 5220.22-M**

Существуют несколько редакций алгоритма.

В одной из редакций используется три прохода по файлу. В первый раз файл перезаписывается нулевыми битами, второй – единичными битами, третий - случайной последовательностью из нулей и единиц. А, например, редакция "**Е"** предусматривает тоже три прохода по телу файла, но другой алгоритм:

* первый проход - сохраняются случайным образом выбранные символы в каждым байте каждого сектора;
* второй проход - записываются инвертированные данные (нулевой бит заменяется единицей, единичный - нулем);
* третий проход - записывается случайная последовательность.

В редакции "**ЕСЕ"** используется уже семь проходов, где чередуются проходы, использующиеся в редакции "**Е"**.

**Российский стандарт ГОСТ P50739-95**

ГОСТ Р 50739-95 определяет, что стирание производится путем записи в нее маскирующей информации, но количество циклов и содержание маскирующей информации не регламентируется. Однако многие считают, что ГОСТ использует два прохода: на первом файл затирается последовательностью из нулевых бит, а на втором — случайной последовательностью.

**Британский стандарт HMG IS5**

Есть две версии алгоритма. Первая просто перезаписывает файл последовательностью нулей. Вторая, трехпроходовая версия, аналогична одной из редакций американского стандарта DoD – сначала файл перезаписывается нулями, потом единицами и в завершение — случайной последовательностью.

**Немецкий стандарт VSItR**

Семипроходовый алгоритм. Сначала выполняются два прохода с перезаписыванием файла нулевыми и единичными последовательностями. Затем эти два прохода повторяются еще два раза. И седьмой проход из случайной последовательности.

**Метод Брюса Шнайера**

Снова семипроходовый алгоритм.

* Первый проход — перезапись файла последовательностью нулевых бит.
* Второй проход —перезапись файла последовательностью единичных бит.
* Проходы с 3 по 7 — перезапись случайной последовательностью.

Достаточно надежный алгоритм, хотя, сам Шнайдер как-то заявил:

«Последние исследования с помощью туннельных микроскопов показали, что даже этого может быть недостаточно».

**Метод Гутманна**

Алгоритм состоит из 35 проходов. Считается самым параноидальным и надежным. И одновременно самым медленным.

В первые четыре прохода записываются случайно выбранные символы в каждый байт каждого сектора, с 5 по 31 проход происходит запись определенной последовательности символов из [специальной таблицы](http://ru.wikipedia.org/wiki/Метод_Гутмана), в последние 4 прохода снова записываются случайно выбранные символы.

Таблица 1

Проходы стирания методом Гутмана

|  |  |  |
| --- | --- | --- |
| 5 | 01010101 01010101 01010101 | 55 55 55 |
| 6 | 10101010 10101010 10101010 | AA AA AA |
| 7 | 10010010 01001001 00100100 | 92 49 24 |
| 8 | 01001001 00100100 10010010 | 49 24 92 |
| 9 | 00100100 10010010 01001001 | 24 92 49 |
| 10 | 00000000 00000000 00000000 | 00 00 00 |
| 11 | 00010001 00010001 00010001 | 11 11 11 |
| 12 | 00100010 00100010 00100010 | 22 22 22 |
| 13 | 00110011 00110011 00110011 | 33 33 33 |
| 14 | 01000100 01000100 01000100 | 44 44 44 |
| 15 | 01010101 01010101 01010101 | 55 55 55 |
| 16 | 01100110 01100110 01100110 | 66 66 66 |
| 17 | 01110111 01110111 01110111 | 77 77 77 |
| 18 | 10001000 10001000 10001000 | 88 88 88 |
| 19 | 10011001 10011001 10011001 | 99 99 99 |
| 20 | 10101010 10101010 10101010 | AA AA AA |
| 21 | 10111011 10111011 10111011 | BB BB BB |
| 22 | 11001100 11001100 11001100 | CC CC CC |
| 23 | 11011101 11011101 11011101 | DD DD DD |
| 24 | 11101110 11101110 11101110 | EE EE EE |
| 25 | 11111111 11111111 11111111 | FF FF FF |
| 26 | 10010010 01001001 00100100 | 92 49 24 |
| 27 | 01001001 00100100 10010010 | 49 24 92 |
| 28 | 00100100 10010010 01001001 | 24 92 49 |
| 29 | 01101101 10110110 11011011 | 6D B6 DB |
| 30 | 10110110 11011011 01101101 | B6 DB 6D |
| 31 | 11011011 01101101 10110110 | DB 6D B6 |

Таблица для перезаписи данных в проходах с 5 по 31 была рассчитана разработчиками алгоритма Питером Гутманом и Коллином Пламбом. Данные в таблице являются «универсальными» для дисков, в которых применяются методы аналого-цифрового преобразования последних лет.

Сам Гутманн про использование своего метода говорит:

«На самом деле, нет смысла проводить полную 35-проходную перезапись для каждого диска, поскольку она нацелена на сочетание сценариев с участием всех трех типов кодирования технологии, которая охватывает все более чем 30-летние MFM методы. Если вы используете диск, который использует технологии кодирования Х, вам не нужно выполнять все 35 проходов, нужно выполнить только определенные».

Другими словами, если точно известно, какой метод кодирования используется в конкретном устройстве хранения, то нужно использовать лишь определенные проходы, подходящие для этого метода. Но часто используются все 35 проходов.

## 1.2 Выбор языка для разработки программного средства

Delphi — это современное продолжение языка Turbo Pascal. На Delphi можно писать программы для всех операционных систем, включая мобильные, а также для всех популярных типов процессоров — x86 и ARM.

Чтобы оценить прогресс в развитии языка, сравните среду разработки Turbo Pascal и Delphi на рисунках 2 и 3.

Рисунок 2

Интерфейс Turbo Pascal

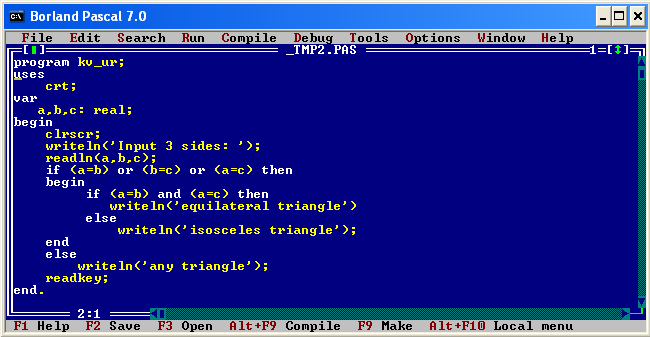
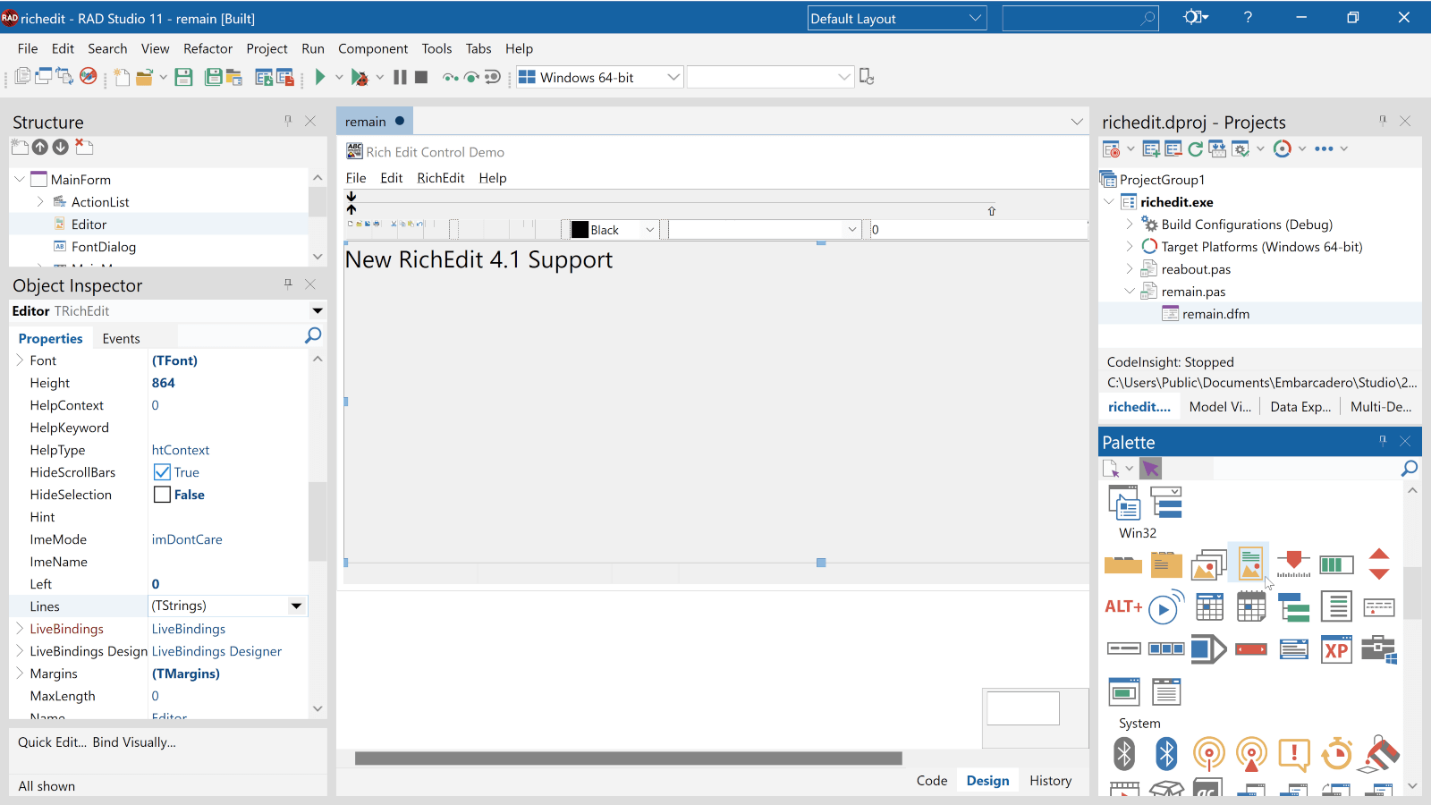


Рисунок 3

Интерфейс Delphi



Delphi — это высокоуровневый язык программирования со строгой статической типизацией и объектно-ориентированным подходом:

* Высокоуровневый — это значит, что в нём используется не машинный код, а команды, удобные для программиста.
* Строгая типизация означает, что в Delphi не получится присвоить строке числовую переменную, как в JavaScript. Получить число из строки тоже не выйдет — нужно будет использовать команды приведения типов.
* Статичная типизация: если программист объявил переменную X как число, то работать с этой переменной можно будет только как с числом на протяжении всей программы. Это надёжнее, чем когда в языке есть возможность менять типы по ходу работы.
* Объектно-ориентированный — то есть язык поддерживает объекты и даёт возможность использовать все их свойства: инкапсуляцию, наследование и полиморфизм.

Главная мощь Delphi — интегрированная среда разработки. Это похоже на конструктор сайтов, когда мы выбираем подходящие элементы и перетаскиваем их на страницу. В Delphi всё работает точно так же: есть панель компонентов, сгруппированных по типам и возможностям, а программист перетаскивает их на интерфейс программы.

У каждого компонента есть свойства и события — разработчик их настраивает в отдельной панели или прямо в коде. Например, можно сделать так:

1. Найти на панели компонент «Кнопка».
2. Перетащить его в нужное место в интерфейсе будущей программы.
3. В панели свойств поменять название, цвет кнопки, шрифт надписи и размеры.
4. В той же панели привязать к кнопке событие onClick, которое сработает при нажатии мыши на кнопку.
5. Прописать в коде, что будет выполняться внутри этого события, например, появление окна справки или вызов другой функции.

Получается, что разработчик может одновременно и писать код, и конструировать интерфейс программы, привязывая к нему нужные события.

Так как в основе Delphi лежит простой Паскаль, то изучить Delphi с нуля новичку гораздо проще, чем Java или C#. Освоиться в языке можно за пару дней, а уже через неделю можно сделать полноценное приложение.

Ещё Delphi — это кроссплатформенный язык. Это значит, что один и тот же код можно использовать для создания программ, например, на Windows, Linux и MacOS. Такой подход ускоряет разработку и делает её дешевле, но повышает требования к квалификации программистов.

В плюс можно отнести и визуальный редактор — программисту можно брать уже готовые компоненты и собирать из них нужный интерфейс, а не описывать его детально вручную в коде. Если каких-то компонентов не хватает, их можно скачать либо с официального сайта разработчиков, либо сделать самому.

Минусы у Delphi тоже есть:

* Из-за поддержки полной обратной совместимости в Delphi много легаси-компонентов и решений. Они иногда затрудняют разработку и усложняют код.
* Сложно разобраться во всех доступных компонентах — по умолчанию их доступно около 200, и хорошо бы знать, за что они отвечают и как работают.

Сейчас программами на Delphi пользуются в основном банки, бухгалтерский сектор, организации систем учёта и госкомпании. Это связано с тем, что в Delphi легко создать простой интерфейс для работы с большим количеством данных — он будет выглядеть примерно, как интерфейс старого 1С, но гарантированно будет работать в любых условиях.

Большинство вакансий разработчиков Delphi — это как раз поддержка и оптимизация старого кода. Такого кода и таких компаний много, переписать всё с нуля дорого, поэтому спрос на Delphi-программистов будет ещё долго. Но в количественном отношении программистов на JavaScript, C или Python требуется гораздо больше, чем на Delphi.

С другой стороны, современные возможности Delphi позволяют ей пользоваться как основной системой для разработки программ для любой архитектуры и ОС. Вот несколько примеров известных программ, написанных на Delphi:

* Skype (до того, как его купила Microsoft);
* почтовая программа The Bat!
* Guitar Pro (до шестой версии);
* AIMP — популярный медиаплеер для Windows;
* Inno Setup — программа-установщик для Windows, один из самых популярных сейчас установщиков.

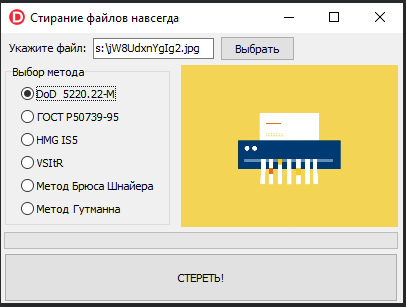
## 1.3 Разработка интерфейса программы

Интерфейс программы разработан в IDE Embarcadero RAD Studio. Поскольку функций у приложения немного, то выглядеть он будет легковесным, простым и интуитивно понятным, а само приложение будет быстрее запускаться за счет малого количества необходимых для работы окон.

Расположим на главной форме основные элементы управления: кнопки собственно стирания, выбора файла с полем для отображения имени стираемого файла, переключатели способа стирания и поясняющие надписи. Интерфейс приложения представлен на рисунке 4.

Рисунок 4

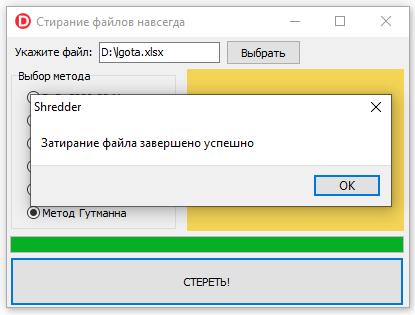
Окно программы



Для отображения процесса стирания размещен указатель процесса выполнения. По окончании стирания выводится окно с результатом выполнения операции. Внешний вид окна представлен на рисунке 5.

Рисунок 5

Сообщение о завершении операции



## 1.4 Реализация стирания файлов

Каждый из 6 способов стирания файлов выполнен в виде отдельной процедуры, полный код программы приводится в приложении А.

Кратко рассмотрим первый метод стирания файла методом **DoD (Department of Defence) 5220.22-M.**

**В процедуре инициализируется генератор случайных чисел путем загрузки зерна в виде текущего времени. На следующем этапе файловая переменная связывается с именем файла и открывается. Настраивается полоса отображающая ход выполнения процесса, и в цикле по каждому байту файла производится запись значения согласно алгоритму стирания. После выполнения необходимого количества проходов записи необходимых значений, файл закрывается и выводится сообщение о завергении стирания файла.**

procedure TForm1.Method1;

var

f: file of byte;

Size: Longint;

i: Longint;

c: byte;

begin

{

три прохода по файлу.

В первый раз файл перезаписывается нулевыми битами,

второй – единичными битами,

третий - случайной последовательностью из нулей и единиц.

}

Randomize;

AssignFile(f, Edit1.Text);

ProgressBar1.min := 0;

Reset(f);

Size := Filesize(f);

ProgressBar1.Max := 3;

ProgressBar1.Position := 0;

c := 0;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := 1;

Application.ProcessMessages;

c := 255;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := 2;

Application.ProcessMessages;

For i := 0 to Size - 1 do

begin

seek(f, i);

c := Random(255);

write(f, c);

end;

ProgressBar1.Position := 3;

Application.ProcessMessages;

CloseFile(f);

end;

В процессе проектирования программного средства было произведено исследование алгоритмов безвозвратного программного уничтожения информации, выбран язык программирования, разработан интерфейс программного средства, и реализована задача безвозвратного стирания файлов выбранными алгоритмами.

# ГЛАВА 3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРИЛОЖЕНИЯ

Десктопные приложения – это полнофункциональные программы, которые работают вне зависимости от других приложений и требуют наличие оператора. Для их работы необходимы достаточные аппаратные ресурсы компьютера, само приложение и набор функций для работы с приложением.

Такие приложения размещаются на компьютере пользователя. Они не требуют для работы подключение к интернету, взаимодействуют с пользователем посредством стандартного интерфейса, имеют более высокое быстродействие, зависят от используемой операционной системы и требуют установку на каждый компьютер пользователя, желающего работать с данным приложением. Все программы, которые установлены у нас на компьютерах, являются desktop-приложениями. Так как мы имеем доступ к системным файлам программы, данный тип приложений более уязвим, и полностью зависит от действий пользователя.

Особенности тестирования десктопных приложений виды тестов

Виды тестирования, которые необходимо проводить на десктопных приложениях помимо основных (функционального, GUI, юзабилити и т.д) также имеют свои особенности:

* тестирование инсталляции
* тестирование обновления
* тестирование деинсталляции

Выполняя тестирование установки, проверяется:

1. Запускается ли программа после установки
2. Расположение программы в файловой системе по умолчанию
3. Расположение программы в файловой системе если путь сохранения изменен пользователем
4. Наличие ярлыков на рабочем столе
5. Есть ли установленный компонент в меню Пуск  > Программы
6. При установке обратить внимание на издателя
7. Установка программы для текущего пользователя/для всех пользователей компьютера
8. Установка пользователем с правами админа
9. Установка пользователем без прав админа

Для выполнения тестирования приложения необходимо производить запуск приложения с использованием всех заявленных функций и добиться того, чтобы не возникало ошибок времени выполнения, а результаты выполнения соответствовали документации.

В процессе тестирования программного средства путем многократного запуска приложения и исправления замеченных недостатков и несоответствий была достигнута уверенно бесперебойная работа программы без ошибок при различных вариантах входных данных и комбинации последовательности действий.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была проведена аналитическая работа, по результатам которой было выявлены различные способы безвозвратного уничтожения информации на диске компьютера, а также теоретические сведения о хранении файлов, способы определения содержимого диска после обычного стирания файлов различными операционными системами.

В процессе проектирования программного средства было произведено исследование алгоритмов безвозвратного программного уничтожения информации, выбран язык программирования, разработан интерфейс программного средства, и реализована задача безвозвратного стирания файлов выбранными алгоритмами.

В процессе тестирования программного средства путем многократного запуска приложения и исправления замеченных недостатков и несоответствий была достигнута уверенно бесперебойная работа программы без ошибок при различных вариантах входных данных и комбинации последовательности действий.

Выполнены задачи, поставленные в курсовой работе:

1. Изучен материал, который необходим для выполнения работы;
2. Разработана программа;
3. Протестирована программа;
4. Получен опыт разработки программного обеспечения для работы с файлами;
5. Приобретены знания о способах безвозвратного удаления файлов и вариантах их восстановления.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дятченко Д.А. Методические указания к выполнению курсовой работы по дисциплине «Программирование и алгоритмические языки»– Северодонецк, 2005.
2. Зуев Е.А. Программирование на языке Turbo Pascal 6.0,7.0. – М.: Радио и связь, Веста, 1993.
3. Иванов В.И. Проектирование и разработка приложений в среде Delphi. М.: Издательский центр "Академия", 2012. 368 с.
4. Кузнецов А. А. Создание установщика приложений на языке Delphi. [Электронный ресурс] // Хабр. Режим доступа: https://habr.com/ru/post/84600/
5. Петров В.Д. Разработка многопользовательских приложений с помощью языка Delphi. СПб.: БХВ- Петербург, 2014. 448 с.
6. Фаронов В.В. Turbo Pascal 7.0. Начальный курс. - М.: Нолидж, 2000.
7. Фаронов В.В. «DELPHI. Программирование на языке высокого уровня». – Питер, 2005.

# ПРИЛОЖЕНИЕ А

unit UMain;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Buttons,

Vcl.ExtCtrls, GIFImg,

Vcl.ComCtrls;

type

TForm1 = class(TForm)

Label1: TLabel;

GroupBox1: TGroupBox;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

RadioButton3: TRadioButton;

RadioButton4: TRadioButton;

RadioButton5: TRadioButton;

RadioButton6: TRadioButton;

Edit1: TEdit;

OpenDialog1: TOpenDialog;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

Image1: TImage;

Timer1: TTimer;

ProgressBar1: TProgressBar;

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure Method1;

procedure Method2;

procedure Method3;

procedure Method4;

procedure Method5;

procedure Method6;

private

Gif: TGifImage;

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

// ----------------------------

procedure TForm1.Method1;

var

f: file of byte;

Size: Longint;

i: Longint;

c: byte;

begin

{

три прохода по файлу.

В первый раз файл перезаписывается нулевыми битами,

второй – единичными битами,

третий - случайной последовательностью из нулей и единиц.

}

Randomize;

AssignFile(f, Edit1.Text);

ProgressBar1.min := 0;

Reset(f);

Size := Filesize(f);

ProgressBar1.Max := 3;

ProgressBar1.Position := 0;

c := 0;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := 1;

Application.ProcessMessages;

c := 255;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := 2;

Application.ProcessMessages;

For i := 0 to Size - 1 do

begin

seek(f, i);

c := Random(255);

write(f, c);

end;

ProgressBar1.Position := 3;

Application.ProcessMessages;

CloseFile(f);

end;

procedure TForm1.Method2;

var

f: file of byte;

Size: Longint;

i: Longint;

c: byte;

begin

{ ГОСТ использует два прохода: на первом файл затирается

последовательностью из нулевых бит,

а на втором — случайной последовательностью. }

Randomize;

AssignFile(f, Edit1.Text);

ProgressBar1.min := 0;

Reset(f);

Size := Filesize(f);

ProgressBar1.Max := 2;

ProgressBar1.Position := 0;

c := 0;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := 1;

Application.ProcessMessages;

For i := 0 to Size - 1 do

begin

seek(f, i);

c := Random(255);

write(f, c);

end;

ProgressBar1.Position := 2;

Application.ProcessMessages;

CloseFile(f);

end;

procedure TForm1.Method3;

var

f: file of byte;

Size: Longint;

i: Longint;

c: byte;

begin

{ Первая просто перезаписывает файл последовательностью нулей. }

AssignFile(f, Edit1.Text);

ProgressBar1.min := 0;

Reset(f);

Size := Filesize(f);

ProgressBar1.Max := 1;

ProgressBar1.Position := 0;

c := 0;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := 1;

Application.ProcessMessages;

CloseFile(f);

end;

procedure TForm1.Method4;

var

f: file of byte;

Size: Longint;

i, j: Longint;

c: byte;

begin

{ Семипроходовый алгоритм. Сначала выполняются

два прохода с перезаписыванием файла нулевыми

и единичными последовательностями. Затем эти

два прохода повторяются еще два раза.

И седьмой проход из случайной последовательности. }

Randomize;

AssignFile(f, Edit1.Text);

ProgressBar1.min := 0;

Reset(f);

Size := Filesize(f);

ProgressBar1.Max := 7;

ProgressBar1.Position := 0;

for j := 1 to 3 do

begin

c := 0;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

c := 255;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

end;

For i := 0 to Size - 1 do

begin

seek(f, i);

c := Random(255);

write(f, c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

CloseFile(f);

end;

procedure TForm1.Method5;

var

f: file of byte;

Size: Longint;

i, j: Longint;

c: byte;

begin

{ Снова семипроходовый алгоритм.

? Первый проход — перезапись файла последовательностью нулевых бит.

? Второй проход —перезапись файла последовательностью единичных бит.

? Проходы с 3 по 7 — перезапись случайной последовательностью.

}

Randomize;

AssignFile(f, Edit1.Text);

ProgressBar1.min := 0;

Reset(f);

Size := Filesize(f);

ProgressBar1.Max := 7;

ProgressBar1.Position := 0;

begin

c := 0;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

c := 255;

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

for j := 1 to 4 do

For i := 0 to Size - 1 do

begin

seek(f, i);

c := Random(255);

write(f, c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

end;

CloseFile(f);

end;

procedure TForm1.Method6;

const

a:array[5..31] of byte=($55,$AA,$92,$49,$24,$00,$11,$22, $33,$44,$55,$66,$77,$88,$99,$AA,$BB,$CC,$DD,$EE,$FF,$92,$49,$24, $6D,$B6,$DB);

var

f: file of byte;

Size: Longint;

i, j: Longint;

c: byte;

begin

{

В первые четыре прохода записываются случайно выбранные символы в каждый байт каждого сектора,

с 5 по 31 проход происходит запись определенной последовательности символов из специальной таблицы,

в последние 4 прохода снова записываются случайно выбранные символы.

}

Randomize;

AssignFile(f, Edit1.Text);

ProgressBar1.min := 0;

Reset(f);

Size := Filesize(f);

ProgressBar1.Max := 35;

ProgressBar1.Position := 0;

for j := 1 to 4 do begin

c := 0;

For i := 0 to Size - 1 do

begin

seek(f, i);

c := Random(255);

write(f,c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

end;

for j := 5 to 31 do begin

c := a[j];

For i := 0 to Size - 1 do

begin

seek(f, i);

write(f, c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

end;

for j := 1 to 4 do

begin

For i := 0 to Size - 1 do

begin

seek(f, i);

c := Random(255);

write(f, c);

end;

ProgressBar1.Position := ProgressBar1.Position + 1;

Application.ProcessMessages;

end;

CloseFile(f);

end;

// -------------------------------------------

procedure TForm1.BitBtn1Click(Sender: TObject);

begin

If OpenDialog1.Execute() then

Edit1.Text := OpenDialog1.FileName;

end;

procedure TForm1.BitBtn2Click(Sender: TObject);

begin

If RadioButton1.Checked then

Method1;

If RadioButton2.Checked then

Method2;

If RadioButton3.Checked then

Method3;

If RadioButton4.Checked then

Method4;

If RadioButton5.Checked then

Method5;

If RadioButton6.Checked then

Method6;

ShowMessage('Затирание файла завершено успешно');

end;

end.