Департамент образования города Москвы

Государственное бюджетное общеобразовательное учреждение города Москвы «Гимназия №1505 «Московская городская педагогическая

гимназия-лаборатория»»

**Диплом**

**на тему**

**Алгоритмы хэширования паролей. Моделирование алгоритма хэширования.**

Выполнил:

Шигаров Дмитрий Алексеевич

Руководитель:

Пяткина Галина Александровна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись руководителя)

Рецензент:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись рецензента)

Москва

2017/2018 уч.г.

Оглавление

[I Введение 3](#_Toc514102034)

[II Алгоритмы хэширования и их назначение. 5](#_Toc514102035)

[Свойства алгоритмов хэширования и их составляющие. 7](#_Toc514102036)

[III Современные алгоритмы хэширования. 10](#_Toc514102037)

[I. SHA-1 11](#_Toc514102038)

[II. ГОСТ Р 34.11-94 13](#_Toc514102039)

[III. MD5 14](#_Toc514102040)

[IV Разработка модели алгоритма хэширования 15](#_Toc514102041)

[V Заключение 25](#_Toc514102042)

[VI Список литературы. 27](#_Toc514102043)

[VII Приложение 28](#_Toc514102044)

# Введение

В наши дни IT сфера является ключевым аспектом нашей жизни. Например, купле продажа во многих случаях проводится с использованием интернет ресурсов, где наибольшее внимание необходимо уделять вопросу о защите личных данных. У каждой информации есть своя ценность, значит и её безопасность играет важную роль. Поэтому программисты хешированием стараются наиболее надёжно защитить доступ к ней. Таким образом, в современном мире, выбранная мной тема наиболее актуальна в современном мире.

Мы живём в мире технологий, где самым ценным ресурсом является информация. Человек, который имеет доступ к чужим данным, получает полное преимущество. Поэтому люди придумывают всё более сложные, многоуровневые системы защиты личной информации. Следующим этапом после простого сравнивания пароля идёт хэширование этих паролей. В январе 1953 года сотрудник фирмы IBM Ханс Петер Лун[[1]](#footnote-1) впервые предложил использовать «хэш-кодирование».

Хэширование — в первую очередь это преобразование массива данных какой-либо длины в строку, с использованием определённого алгоритма. Также необходимо решать проблемы коллизии, которые решаются путём использования различных технологий, такие как: Метод цепочек, Открытая адресация, Хэширование с солью.

Цель моего исследования – это изучить способы взлома и защиты данных, а также привести пример собственного алгоритма хэширования.

**Для выполнения этой цели были поставлены конкретные задачи**:

* Изучить информацию по данной теме реферата.
* Объяснить необходимость надёжной защиты информации.
* Изучить виды различных атак и способов взлома.
* Рассмотреть виды различных способов сокрытия и защиты информации.
* Создать собственную программу с определённым алгоритмом хэширования.

# Алгоритмы хэширования и их назначение.

Многие наверняка знают известную крылатую фразу Натана Ротшильда: «Кто владеет информацией, тот владеет миром». Используя своё преимущество перед остальными в скорости получения информации, он манипулировал биржей, и заработал 40 миллионов фунтов на разорившихся конкурентах.

В современном мире скорость передачи информации высока, и основной заботой является её защита. Так, например, самая известная и распространённая защита паролем - первый этап по обеспечению безопасности данных. Использование пароля позволяет защитить информацию на компьютере от многих угроз:

•несанкционированного доступа (как от прямого вторжения, так и по сети);

•от вирусов и шпионских программ.

С нынешним уровнем технологий одного пароля для защиты любой информации мало. Основной проблемой является необходимость защитить местоположение и содержание пароля. Для этого придумывается множество алгоритмов изменения и сокрытия, названные хэш-функциями.

В январе 1953 года сотрудник фирмы IBM Ханс Петер Лун впервые предложил использовать «хэш-кодирование». Принятую во всём мире идею хэширования описал в 1956 году Арнольд Думи в своей работе «Computers and automation». Думи рассматривал «хэширование», как решение проблемы перечня, предложил использовать в качестве «хэш-адреса» остаток от деления на простое число. После Думи термин хэширование употреблялся в таких изданиях, как:

1. Статья Уэсли Питерсона (1957 год).
2. Работа Вернера Бухгольца (1963 год).
3. Книга Херберта Хеллермана «Принципы цифровых вычислительных систем» (1967 год).

Но наибольшее распространение хэширование получило после публикации в 1968 году статьи Робертом Моррисом. Эта работа считается ключевой публикацией, вводящей понятие о «хешировании» в научный оборот, и закрепившей термин хеширование, ранее применявшийся только специалистами.

Хэш-функция это особый алгоритм, уникальным образом переделывающий пароль. Результат этих действий называется хэш-кодом. Важной особенностью хэширования является необратимость. С помощью хэш-функции мы можем получить для каждого пароля хэш-код, но, зная хэш-код, мы не сможем узнать саму функцию без алгоритма. Однако при простой хэш-функции довольно легко понять концепцию, поэтому для улучшения защищённости придумываются всё более изощрённые способы изменения.

Так что может сделать среднестатистический пользователь для защиты личной информации? Практически единственное, что зависит от обычного человека, это использование грамотного пароля. Некоторые интернет ресурсы, требующие пароль, дают различные советы пользователю по этому вопросу. Таким образом, они вынуждают пользователя выбрать правильный по нескольким пунктам пароль, такие как:

* Пароль должен быть достаточно длинным, обычно от 8 до 12 символов.
* Пароль должен содержать заглавные и прописные буквы, символы, знаки препинания и пробелы.
* Пароль не должен ассоциироваться с пользователем, такие пароли как имена, фамилии, памятные даты, номера телефонов и т.д. не подходят.
* Пароль не должен состоять из известных фраз или популярных слов.
* Пароль не должен быть просто запоминающимся и широко используемым.
* Пароль не должен совпадать с логином, именем сайта или любой общедоступным связанным словом.

# Свойства алгоритмов хэширования и их составляющие.

Кроме хэш-кода есть и другие составляющие, например Хэш-таблица. Хэш-таблица — это массивы, хранящие пары информации: ключ, значение, и выполняющие три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу. Выполнение операции в хэш-таблице начинается с вычисления хэш-функции от ключа. Получающийся хэш играет роль индекса в массиве. Затем выполняемая операция перенаправляется объекту, который хранится в соответствующей ячейке массива.

Также, частой проблемой многих алгоритмов это коллизия. Коллизия хэш-функции — ситуация, при которой два разных результата имеют одинаковый хэш-код. Это возможно, если длина последовательности символов будет достаточно большой. Чем больше длина, тем больше вероятность совпадения хэша. Коллизии осложняют использование хэш-таблиц, так как нарушают исключительность соответствия между хэш-кодами и данными. Тем не менее, существуют специальные методики для преодоления возникающих сложностей:

1. Метод цепочек
2. Открытая адресация
3. Хэширование с солью

Метод цепочек: технология сцепления элементов состоит в том, что элементы множества, которым соответствует одно и то же хэш-значение, связываются в цепочку-список. В позиции с определённым номером хранится указатель на список тех элементов, у которых хэш-значение ключа равно тому же номеру.

Открытая адресация: в отличие от хэширования с цепочками, при открытой адресации никаких списков нет, а все записи хранятся в самой хэш-таблице. Каждая ячейка таблицы содержит элемент такого множества.

Конечно, ради чужой информации люди придумали множество способов получения доступа к ней, такие как:

1. Использование "радужных таблиц".
2. Фишинг
3. Перебор по словарям.
4. Вредоносное программное обеспечение

Радужная таблица [[2]](#footnote-2)- это комбинация уже готовых словарей хэш-функций, используемая для замены вычислений на поиск. Таблица включает в себя хэши всех возможных комбинаций паролей для многих видов алгоритма хэширования. Время, необходимое для взлома пароля с помощью радужной таблицы - это время поиска нужного захэшированного пароля в списке. Тем не менее, сама таблица огромна и для просмотра требует серьезных вычислительных мощностей. Такой вариант атак не подходит для поиска хэша с использованием соли, направленной против этого.

Применение радужных таблиц намного проще и быстрее использования метода грубой силы для подбора, но само первостепенное создание требует огромные вычислительные мощности, без которой это невозможно сделать приемлемое время. К примеру, для захешированных паролей длиной не более 8 символов, состоящих из букв, цифр и специальных символов, таблицы могут быть сгенерированы на компьютере за 3 года.

Фишинг – это ещё один часто используемый способ получения доступа к чужой информации. Это процедура получения паролей путём обмана пользователей всемирной сети. Она включает в себя создание ложных сайтов, где пользователи вводят свои персональные данные, которые передаются в руки злоумышленников. Обычно такие сайты рассчитывают на невнимательность людей, так как адрес подставного сайта от оригинала отличается только одним символом.

Перебор это метод поиска, включающий в себя прогон большого количества вариантов, в поисках подходящего. Чаще всего перебираются «простые пароли», реже более сложные, так как количество возможных вариантов сильно влияет на время перебора.

Простые пароли это комбинации символов, для поиска которых затрачивается малое время (не больше дня).

Простыми паролями чаще всего считаются:

1. Небольшие пароли, состоящие из набора цифр
2. Небольшие пароли, состоящие из набора строчных или заглавных букв
3. Односложные пароли любой доступной длины (77777777)
4. Пароль, состоящий из рядом расположенных символов на клавиатуре.
5. Часто используемые пароли, комбинации паролей
6. “слитые” пароли популярных сервисов.

Также, есть способ получения паролей и хэшей, внедряясь в устройство. Программа перехвата фиксирует всю вводимую информацию и переправляет копию этого файла на адреса взломщиков.

После появления способов взлома паролей возникла острая необходимость в более высоком уровне защиты. Для защиты от поиска по радужным таблицам была придумано хэширование с солью. Это комбинация из определённых символов, добавляемая к паролю перед вычислением хэша. Тем самым хэш удлиняется до размера, не обрабатываемого никакой радужной таблицей.

Для защиты от метода грубой силы и перебора по словарю нет особого алгоритма защиты. На различных сайтах присутствуют специализированные требования к паролям и всплывающие окна, задерживающие перебор. Но основной преградой является правильный алгоритм хэширования и трудность пароля.

При правильном использовании интернета, скачивание из надёжных источников, шансы быть атакованным вредоносными программами минимальны. Но специализированную защиту, обеспечивают антивирусы. Они предотвращают скачивание вирусов, а в случае проникновения обнаруживают и удаляют.

Чтобы избежать потери данных из-за фишинга, необходимо всегда внимательно проверять адрес сайта, прежде чем вводить персональные данные, и избегать переходов по подозрительным ссылкам.

# Современные алгоритмы хэширования.

В современном мире существует огромное множество уникальных хэш-функций. Их секретность является залогом безопасности данных компании. Уникальность достигается за счёт различных вариаций одного алгоритма. Наиболее известными и значимыми, пожалуй, являются такие хеш-функции как:

* SHA-1
* ГОСТ Р 34.11-94
* MD5
* CRC

И я расскажу про некоторые из них.

# SHA-1

SHA-1 –очень известная криптографическая функция.

SHA 1 расшифровывается как Secure Hashing Algorithm, что значит Безопасный Алгоритм Хэширования. Он разработан в National Security Agency[[3]](#footnote-3) – Агентством национальной безопасности США. SHA 1 служит для того, чтобы сделать идентификаторы фиксированной длиной из набора данных.

Разработанный в 1993 году совместно с NIST алгоритм безопасного хеширования, названный SHA-0, был опубликован в документе FIPS PUB 180. Однако из-за обнаруженной ошибки первая версия была отозвана через небольшой отрезок времени. Позже, в 1995 году в FIPS PUB 180-1 была опубликована уже исправленная версия, которую и принято сейчас называть SHA-1. Возможно, исправленная ошибка была найдена французскими исследователями и представлена на конференции CRYPTO в 1998 году. Представленная ими атака успешно действовала на алгоритме SHA-0, но на алгоритме SHA-1 не проходила. Но эта информация так и не была подтверждена NSA.

Сама механика алгоритма заключается в обработке кусков информации. Входные данные произвольной длинны алгоритм делит на части, каждая из которых длинной по 512 бит. Также, происходит дополнение, путём дополнения длинны исходного массива и пустых ячеек в конец блока. После деления основной цикл итеративно обрабатывает каждый из поделенных кусков. Сама обработка состоит из четырёх частей, в каждой из которых происходят двадцать операций. В результате, каждый отрезок информации подвергается изменению 80 раз. После выполнения хэш-функции получается искомый хэш, длинной 160 бит.

SHA-1 является лицом всего семейства SHA. Он получил наибольшее распространение среди собратьев и активно применялся криптографических приложениях и алгоритмах. SHA-1 использовался для подписи сертификатов TLS, а также входил в состав многих приложений: Torrent, Git, PGP, SHACAL и др.

Со временем алгоритм, очевидно, стал устаревать, росли вычислительные мощности и компьютерные возможности. И поэтому крупнейшая компания Google объявила своё недоверие к SHA-1, особенно в TLS[[4]](#footnote-4). И в 2014 году разработчики Chrome официально объявили о постепенном отказе от использования SHA-1. Через какое-то время и другие разработчики постепенно переходили на использование других алгоритмов взамен SHA-1.

# ГОСТ Р 34.11-94

После SHA-1 я хочу рассказать про распространённый в России алгоритм ГОСТ Р 34.11-94.

ГОСТ Р 34.11-94 был создан в России как криптографический стандарт вычисления для государственных учреждений, где стал обязательным для использования. Он был назван по характерному принципу для его группы Русских алгоритмов, у которых в конце ставился год создания. Поэтому ГОСТ Р 34.11-94 так и называется, так как был создан в 19994 году группой программистов ГУБС ФАПСИ[[5]](#footnote-5) и Всероссийского научно-исследовательского института стандартизации.

Аналогично принципу SHA-1, ГОСТ Р 34.11-94 разбивает вводимые данные на части, только длинной 256 бит. Вводимые данные не имеют ограничений, а выходное значение фиксировано имеет длину 256 бит. После деления происходит промежуточная обработка. При обработке полученных блоков используются преобразования по предшествующему алгоритму ГОСТ 28147—89. Это устаревший алгоритм, который имеет одну особенность, за которую вероятно и был выбран для использования. В ГОСТ 28147—89 присутствуют узлы замены (S-блоки), они могут использоваться как увеличители длины хэша, что существенно осложняет поиск коллизий в хэш-функции. Помимо этого, в алгоритме происходит расчёт контрольной суммы всего исходного сообщения, которая является частью финального вычисления хэша. Данное действие также является препятствием для коллизионных атак. Однако уже в 2008 году Польские и Австрийские программисты указали на недостаток программы, сокращая время поиска коллизий в 223 раза.

1 января 2013 года стал официально использоваться следующий алгоритм из той же группы ГОСТ Р 34.11-2012. В хэш-функцию добавили безусловно важную функцию - сжатие с тремя различными преобразованиями. Но несмотря на это, криптоанализ хэш-функции показал результат хуже, чем был у предшественника ГОСТ 28147—89.

# MD5

Алгоритм шифрования MD5 это, пожалуй самый известный алгоритм хэширования.

Алгоритм MD5 был разработан профессором Рональдом Л. Ривестом из Массачусетского технологического института в 1991 году. Данный алгоритм стал заменой предыдущей версии алгоритма MD4. Алгоритм быстро обрел большую популярность и стал использоваться повсеместно. Однако уже через 2 года Берт ден Бур и Антон Босселарс нашли первый недочёт в алгоритме. В 1996 году Ганс Доббертин также обнаружил уязвимость, и уже тогда были предложены в качестве замены другие алгоритмы, такие как Whirlpool[[6]](#footnote-6), SHA-1 или RIPEMD-160.

Начиная с 1993 года, регулярно появляются исследования, которые обнаруживают все новые уязвимости в алгоритме MD5. На данный момент алгоритм MD5 считается уязвимым и постепенно заменяется алгоритмом SHA.

Сейчас уже этот метод можно считать устаревшим, но раньше он применялся в FreeBSD[[7]](#footnote-7), а также в некоторых системах Linux. Сейчас MD5 продолжает использоваться для проверки целостности данных при передачи, хэширования паролей в UNIX. Из-за востребованности MD5 Linux имеет собственную команду «md5sum», использующую алгоритм MD5 для проверки файла.

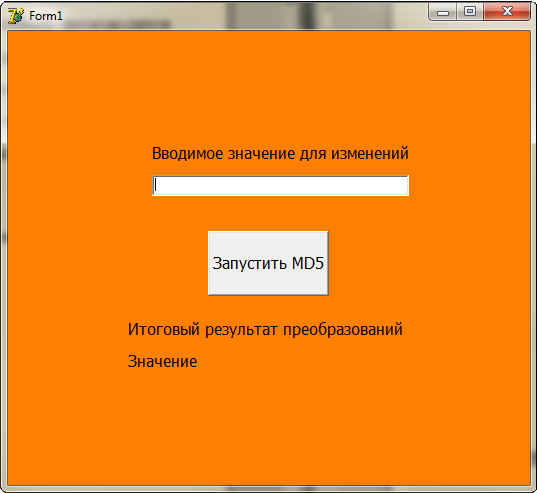
# Разработка модели алгоритма хэширования

Целью практической части моего диплома является создание программы, моделирующей действие какого-либо алгоритма. Я выбрал моделирование алгоритма MD5, так как это наиболее известный, важный открытый алгоритм.

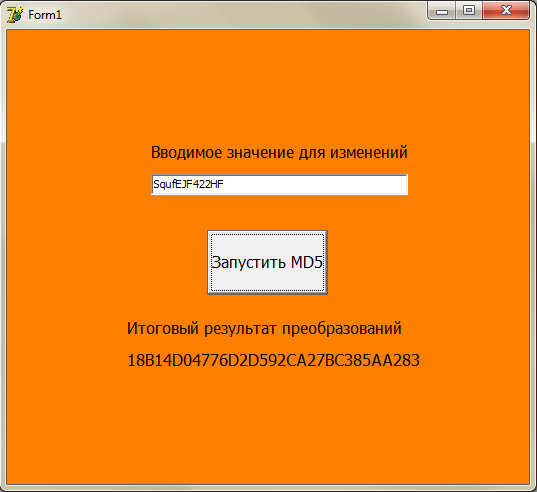
Для моделирования необходимо также выбрать среду разработки. Выбранной мною средой стала Delphi 7[[8]](#footnote-8). Причиной данного выбора стали несколько факторов.

* Во первых, Delphi –многофункциональная и удобная среда разработки, просто-доступная для восприятия.
* Во вторых, я мог использовать базу знаний и опыт работы, полученные в 10 классе.
* В третьих, Delphi имеет собственные встроенные библиотеки, что позволяет воздержаться от подключения сторонних библиотек.

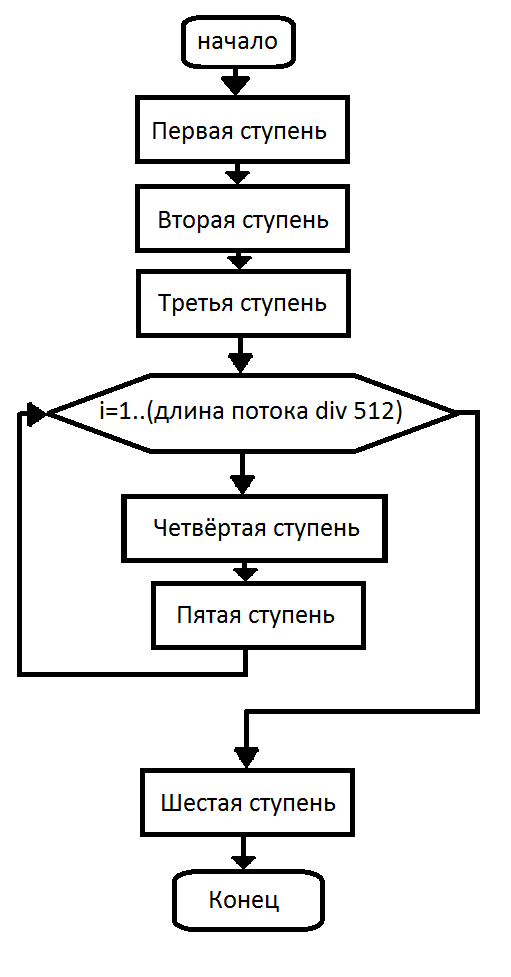
Прежде всего, перед нами находится фрейм с надписями, строкой для ввода и кнопкой. Программа начинает работать по нажатию на кнопку. В строку для ввода записывается набор символов, которые будут обработаны.



В программе присутствует несколько программно-задаваемых переменных, которые влияют на результат работы программы. В своём программном коде я их определил как: «Bit32» - массив, состоящий из 16 строк, и «A», «B», «C», «D», которые также задаются двоичными строками.

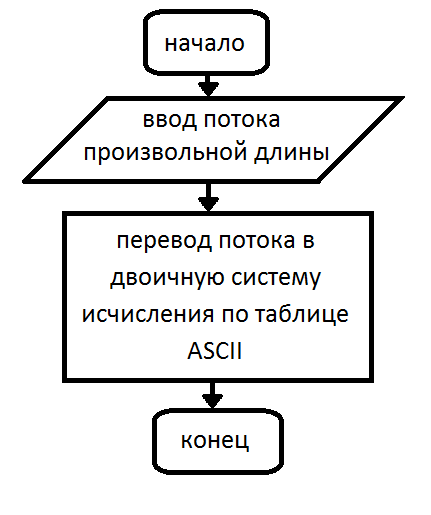
На вход поступает поток произвольной длинны, который в итоге преобразуется в 128 битный хэш, состоящий из 32 шестнадцатеричных символов.

Принцип работы алгоритма MD5 я разделил на 6 частей. В совокупности они создают работающий код, которые переделывает получаемое значение в хэш.



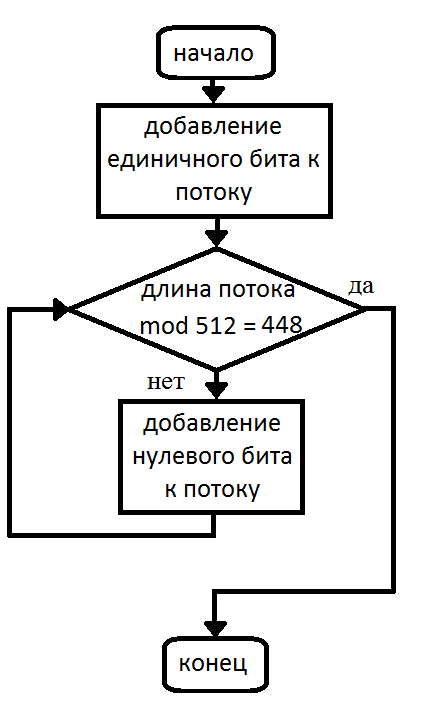
**Первая ступень**

В первой части происходит ввод данных в программу для последующих действий, а так же происходит обработка этих данных для получения их двоичного вида. По итогу, мы получаем двоичную строку произвольного размера, которая являлась вводными данными.



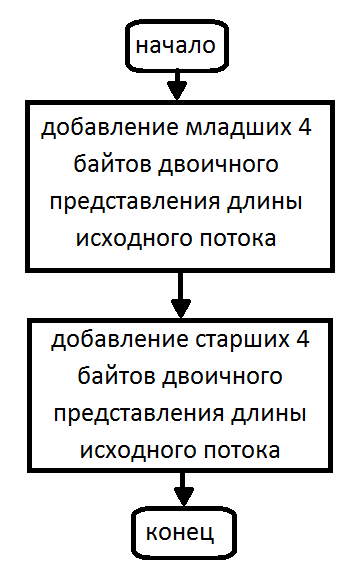
**Вторая ступень**

Во второй части происходит заполнение потока. Сначала добавляется единичный бит, после неопределённое количество нулевых, пока не будет выполняться это условие: (Inputvalue512length mod 512)=448. Эти действия необходимы для того, чтобы привести поток к определённой длине.



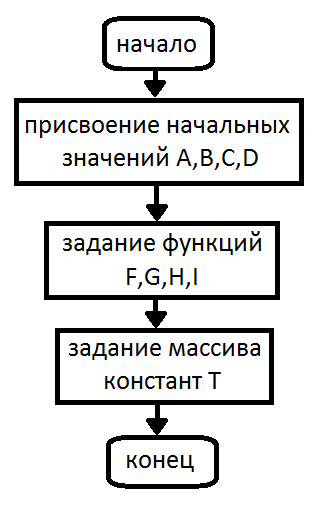
**Третья ступень**

В третьей части также происходит дополнение потока. К потоку прибавляется в конец длина исходных данных, сначала младшие 4 байта длины, затем старшие. На этом этапе длина потока становится кратной 512 и удобной для дальнейшего деления и работы. Все действия происходят в двоичной системе и хранятся в строковых переменных



**Четвёртая ступень**

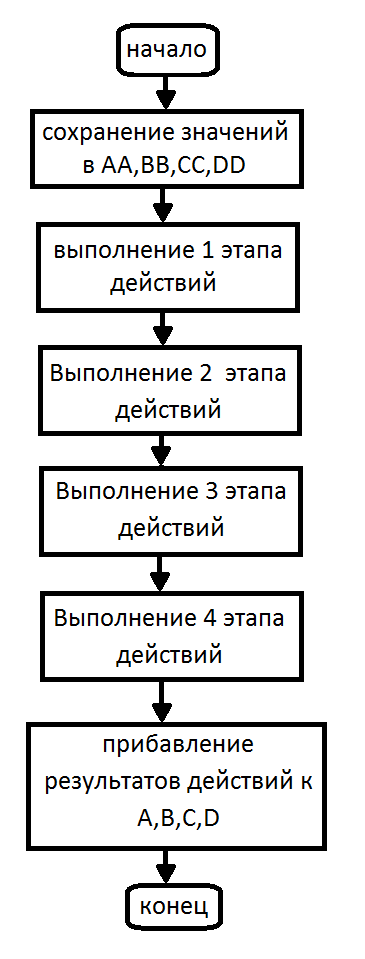
Четвертая часть используется в основном для задания функций и заполнения массивов. На этом этапе я организовал функции для финальных вычислений.

1. FunF(A1, B1, C1)=(A1 and B1)or (not A1 and C1)
2. FunG(A1, B1, C1)= (A1 and C1)or (not C1 and B1)
3. FunH(A1, B1, C1)= A1 xor B1 xor C1
4. FunI(A1, B1, C1) =B1 xor (not B1 or C1)
5. ROL(a, s)= A shl s) or (A shr (32-s)) - побитовый сдвиг

Ещё я задал значения массиву из 64 элементов, который имеет формулу:

T[n]=int(232\*|sin(n)|)

Помимо этого, в четвёртой ступени происходит заполнение массива для каждого куска в 512 бит, состоящий из 16 элементов по 32 бита от куска потока, который также используется в основных вычислениях.

**Пятая ступень**

Пятая часть представляет собой основные вычисления алгоритма. В этой ступени происходит 4 этапа изменений по 16 действий каждый. Этапы различаются функциями, которые в них используются, и номерами ячеек массивов. Эти значения k,I,s задаются программно и влияют на результат. Так например, общая формула первого этапа выглядит:

a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s).

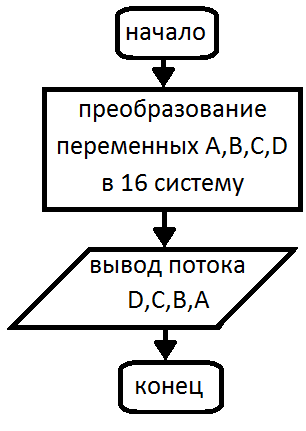
Так выглядит формула второго этапа:

a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s).

Сначала мы сохраняем начальные результаты, а после вычислений суммируем. По итогу, у нас получается 4 строки: A,B,C,D.

**Шестая ступень**

Шестая часть служит для вывода конечного результата. Кроме вывода в этой ступени присутствуют действия по переходу в 16 систему исчисления, так как принято считать результатом алгоритма MD5 16 битную строчку определённой длины. Вычисления проводились в алфавитном порядке, но на выход поступает строчка из соединенных в обратном порядке переменных.



**Проблемы**

Во время написания программы мне пришлось столкнуться с различными трудностями, которые необходимо было по возможности решать.

Главная проблема, это ограниченность ячеек. В Delphi отсутствует тип данных для работы с бинарными числами. Поэтому я пытался использовать различные переменные, типа Longword, Extended, но в результате использую для хранения строки, а все вычисления провожу в десятичной системе типа integer.

Так же в алгоритме присутствовали элементы, не известные мне, которые требовалось воссоздать в программном коде.

# Заключение

Основной темой моей дипломной работы являлись алгоритмы хэширования, играющие важную роль в современной жизни. Так как сейчас информационный век и идёт безостановочное развитие IT сферы, вопрос безопасности и сохранности является одним из ключевых. В общих учебных учреждениях эта тема подробно не описывается, и поэтому большинство людей попросту не знают ничего о средствах защиты своей информации и их различиях.

Для данной дипломной работы была поставлена определённая цель: изучить способы взлома и защиты данных, а также привести пример собственного алгоритма хэширования.

Эта цель достигалась на протяжении работы над дипломом, путём выполнения отдельных задач:

* Изучить информацию по данной теме реферата.
* Объяснить необходимость надёжной защиты информации.
* Изучить виды различных атак и способов взлома.
* Рассмотреть виды различных способов сокрытия и защиты информации.
* Создать собственную программу с определённым алгоритмом хэширования.

# Список литературы.

* Корбит А.Г., Кривоносова Т.М. Методическое пособие. Алгоритмы вычислительной математики. [ Электронный ресурс]

Режим доступа: <http://www.studfiles.ru/preview/5514229/page:7/>

* Разработчики PHP. Технология хэширования паролей[Электронный ресурс]

Режим доступа: <http://phpfaq.ru/tech/hashing>

* Издательский дом "К-Пресс". Бинарные деревья. [Электронный ресурс]

Режим доступа: <http://www.k-press.ru/CS/2000/4/bintree_htm/hash.asp>

* Сергей Соболь. Хэш-таблицы. [Электронный ресурс]

Режим доступа: <https://acm.bsu.by/w/images/c/c0/HashTables.pdf>

* Эрик Грубер. Подбор LM-хеша – радужные таблицы против прямого перебора при помощи GPU. [Электронный ресурс]

Режим доступа: <http://www.securitylab.ru/analytics/462201.php>

* Сергей Суягин. Как создать и запомнить надежный пароль. [Электронный ресурс]

Режим доступа: <https://lifehacker.ru/2014/06/07/nadyozhnyj-parol/>

* Атака на хэши. (практическое руководство). [Электронный ресурс]

Режим доступа: <https://forum.antichat.ru/threads/68675/>

* Список словарей: Электронный ресурс.

Режим доступа: <http://wordbook.xyz/download>

* Форум программистов и сисадминов Киберфорум[Электронный ресурс]

Режим доступа: <http://www.cyberforum.ru/>

* Описание алгоритма MD5[Электронный ресурс]

Режим доступа: <http://re.mipt.ru/infsec/2004/essay/2004_MD5_Message-Digest_Algorithm__Strelnikov.pdf>

* Коды символов ASCII[Электронный ресурс]

Режим доступа: <http://book.itep.ru/10/ascii.htm>

* Хэш-функция MD5[Электронный ресурс]

Режим доступа: <https://habrahabr.ru/sandbox/26876/>

* Учебное пособие: Проектирование, анализ и программная реализация структур данных и алгоритмов[Электронный ресурс]

Режим доступа: <https://studopedia.ru/3_53192_proektirovanie-analiz-i-programmnaya-realizatsiya-struktur-dannih-i-algoritmov.html>

# Приложение

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Math, StdCtrls;

type

TForm1 = class(TForm)

Edit1: TEdit;

Button1: TButton;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

Inputvalue: string; //Вводимый поток

Binaryinputvalues: string; //Поток в десятичном представлении

Inputvaluelength: Integer; //длина вводимого потока

Binaryinputvalueslength:integer; //длина вводимого бинарного потока

N,z,R: integer;

RC:string;

i: integer;

Inputvalue512: string; //поток с нулевыми битами

Inputvalue512length: integer; //длина потока с нулевыми битами

BinaryinputvalueslengthBinary: string; //бинарная длина вводимого бинарного потока

R2: Integer;

RC2:string;

Inputvalue512bit64:string;

//Задание 4 ступени

A,B,C,D,AA,BB,CC,DD: string;

AttachThreadInput: Integer;

Inputvalue512bit64Second,Inputvalue512bit64Second32, second321,second322: string;

second321try4, second322try4, analog321,analog322, Value32S,Sint16mass,AString,BString,CString, DString: string;

j,p1,p2,p3: Integer;

mass16int: Extended;

Bit32: array[1..16] of string;

T: array[1..64] of string =

('11010111011010101010010001111000', '11101000110001111011011101010110', '100100001000000111000011011011', '11000001101111011100111011101110', '11110101011111000000111110101111', '1000111100001111100011000101010',

'10101000001100000100011000010011', '11111101010001101001010100000001', '1101001100000001001100011011000', '10001011010001001111011110101111', '11111111111111110101101110110001', '10001001010111001101011110111110',

'1101011100100000001000100100010', '11111101100110000111000110010011', '10100110011110010100001110001110', '1001001101101000000100000100001', '11110110000111100010010101100010', '11000000010000001011001101000000',

'100110010111100101101001010001', '11101001101101101100011110101010', '11010110001011110001000001011101', '10010001000001010001010011', '11011000101000011110011010000001', '111001111101001111111011110010008',

'100001111000011100110111100110', '11000011001101110000011111010110', '11110100110101010000110110000111', '1000101010110100001010011101101', '10101001111000111110100100000101', '11111100111011111010001111111000',

'1100111011011110000001011011001', '10001101001010100100110010001010', '11111111111110100011100101000010', '10000111011100011111011010000001', '1101101100111010110000100100010', '11111101111001010011100000001100',

'10100100101111101110101001000100', '1001011110111101100111110101001', '11110110101110110100101101100000', '10111110101111111011110001110000', '101000100110110111111011000110', '11101010101000010010011111111010',

'11010100111011110011000010000101', '100100010000001110100000101', '11011001110101001101000000111001', '11100110110110111001100111100101', '11111101000100111110011111000', '11000100101011000101011001100101',

'11110100001010010010001001000100', '1000011001010101111111110010111', '10101011100101000010001110100111', '11111100100100111010000000111001', '1100101010110110101100111000011', '10001111000011001100110010010010',

'11111111111011111111010001111101', '10000101100001000101110111010001', '1101111101010000111111001001111', '11111110001011001110011011100000', '10100011000000010100001100010100', '1001110000010000001000110100001',

'11110111010100110111111010000010', '10111101001110101111001000110101', '101010110101111101001010111011', '11101011100001101101001110010001');

implementation

{$R \*.dfm}

function ROL(a, s: Integer): Integer;

begin

result:= (A shl s) or (A shr (32-s));

end;

function BinToInt(a: string):Integer;

Var

a2:string;

h,l:Integer;

begin

a2:=a;

for l:=1 to Length(a) do

begin

Delete(a2,1,Length(a2)-l);

if l>1 then

Delete(a2,2,Length(a2)-1);

if a2='1' then

begin

h:=h+Round(IntPower(2,l-1));

end;

a2:=a;

end;

result:=h;

end;

function FunF(A1, B1, C1: string) : integer;

begin

result :=(BinToInt(A1) and BinToInt(B1))or (not BinToInt(A1) and BinToInt(C1));

end;

function FunG(A1, B1, C1: string) : integer;

begin

result :=(BinToInt(A1) and BinToInt(C1))or (not BinToInt(C1) and BinToInt(B1));

end;

function FunH(A1, B1, C1: string) : integer;

begin

result :=BinToInt(A1) xor BinToInt(B1) xor BinToInt(C1);

end;

function FunI(A1, B1, C1: string) : integer;

begin

result :=BinToInt(B1) xor (not BinToInt(B1) or BinToInt(C1));

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

//Первая ступень

for i:=1 to 1 do

begin

Binaryinputvalues:='';

RC:='';

Inputvaluelength:=Length(Edit1.Text);

for N:= 1 to Inputvaluelength do

begin

R:=ord(Edit1.Text[N]); //Получение кода буквы по таблице ASCII

for z:=1 to Length(IntToStr(R)) do

begin

while (R div 2)>0 do //Перевод кода буквы в 2 систему исчисления

begin

RC:=IntToStr(R mod 2) + RC;

R:=R div 2;

end

end;

RC:=IntToStr(1)+RC;

Binaryinputvalues:= Binaryinputvalues + RC;

end;

Binaryinputvalueslength:=Length(Binaryinputvalues);

end;

//Конец первой ступени

//Binaryinputvalues(str) - бинарный поток

//Binaryinputvalueslength(int) - длина бинарного потока

//Вторая ступень

for i:=1 to 1 do

begin

Inputvalue512:=Binaryinputvalues+'1'; //Добавление 1 бита

repeat

Inputvalue512:=Inputvalue512+'0';

Inputvalue512length:=Length(Inputvalue512);

until(((Inputvalue512length mod 512)=448)and(((Inputvalue512length div 512)>0)));

end;

//Конец второй ступени

//inputvalues512(str) бинарный поток с нулвыми битами, выравненный под 512\*n +448

//Inputvalue512length(int) длина бинарного потока с нулевыми битами, выравненного под 512\*n +448

//Третья ступень

for i:=1 to 1 do

begin

R2:=Binaryinputvalueslength;

RC2:='';

while (R2 div 2)>0 do

begin

RC2:=IntToStr(R2 mod 2) + RC2;

R2:=R2 div 2;

end;

RC2:=IntToStr(1)+RC2;

BinaryinputvalueslengthBinary:=RC2;

Inputvalue512bit64:=Inputvalue512;

if ((Length(BinaryinputvalueslengthBinary)mod 2)=0) then

begin

for z:=1 to (Length(BinaryinputvalueslengthBinary)div 2) do

begin

Inputvalue512bit64:=Inputvalue512bit64+BinaryinputvalueslengthBinary[(Length(BinaryinputvalueslengthBinary)div 2)+z];

end;

end;

if ((Length(BinaryinputvalueslengthBinary)mod 2)=1) then

begin

for z:=1 to ((Length(BinaryinputvalueslengthBinary)div 2)+1) do

begin

Inputvalue512bit64:=Inputvalue512bit64+BinaryinputvalueslengthBinary[(Length(BinaryinputvalueslengthBinary)div 2)+z];

end;

end;

for z:=1 to (Length(BinaryinputvalueslengthBinary)div 2) do

begin

Inputvalue512bit64:=Inputvalue512bit64+BinaryinputvalueslengthBinary[z];

end;

if (Length(BinaryinputvalueslengthBinary)<65) then

Begin

for z:=1 to (64-Length(BinaryinputvalueslengthBinary)) do

begin

Inputvalue512bit64:=Inputvalue512bit64+IntToStr(0);

end;

end;

end;

//Конец третьей ступени

//Inputvalue512bit64(str) поток с добавлением нулевых битов и изначальной длинны

//Четвёртая ступень

A:='1001000110100010101100111';

B:='10001001101010111100110111101111';

C:='11111110110111001011101010011000';

D:='1110110010101000011001000010000';

for i:=1 to (Length(Inputvalue512bit64) div 512)do

begin

Inputvalue512bit64Second:=Inputvalue512bit64;

if i>1 then

Delete(Inputvalue512bit64Second, 1, 512\*(i-1));

Delete(Inputvalue512bit64Second,513, (Length(Inputvalue512bit64)-i)\*512);

for j:=1 to 16 do

begin

Inputvalue512bit64Second32:=Inputvalue512bit64Second;

Delete(Inputvalue512bit64Second32, (32\*j)+1, Length(Inputvalue512bit64Second)-32\*j);

if j>1 then

Delete(Inputvalue512bit64Second32, 1, 32\*(j-1));

if (j=2)and(i=1) then

Bit32[j]:=Inputvalue512bit64Second32;

//присвоение массива 16 отрезков

end;

//Конец четвёртой ступени

//Пятая ступень

//Этапы вычислений

AA:=IntToStr(abs(BinToInt(A)));

BB:=IntToStr(abs(BinToInt(B)));

CC:=IntToStr(abs(BinToInt(C)));

DD:=IntToStr(abs(BinToInt(D)));

//Этап 1

A:=IntToStr(abs(BinToInt(B)+ROL(BinToInt(A)+FunF(B,C,d)+BinToInt(Bit32[1])+ BinToInt(T[1]), 7)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunF(A,B,C)+BinToInt(Bit32[2])+ BinToInt(T[2]), 12)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunF(D,A,B)+BinToInt(Bit32[3])+ BinToInt(T[3]), 17)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunF(C,D,A)+BinToInt(Bit32[4])+ BinToInt(T[4]), 22)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunF(B,C,d)+BinToInt(Bit32[5])+ BinToInt(T[5]), 7)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunF(A,B,C)+BinToInt(Bit32[6])+ BinToInt(T[6]), 12)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunF(D,A,B)+BinToInt(Bit32[7])+ BinToInt(T[7]), 17)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunF(C,D,A)+BinToInt(Bit32[8])+ BinToInt(T[8]), 22)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunF(B,C,d)+BinToInt(Bit32[9])+ BinToInt(T[9]), 7)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunF(A,B,C)+BinToInt(Bit32[10])+BinToInt(T[10]),12)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunF(D,A,B)+BinToInt(Bit32[11])+BinToInt(T[11]),17)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunF(C,D,A)+BinToInt(Bit32[12])+BinToInt(T[12]),22)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunF(B,C,d)+BinToInt(Bit32[13])+BinToInt(T[13]), 7)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunF(A,B,C)+BinToInt(Bit32[14])+BinToInt(T[14]),12)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunF(D,A,B)+BinToInt(Bit32[15])+BinToInt(T[15]),17)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunF(C,D,A)+BinToInt(Bit32[16])+BinToInt(T[16]),22)));

//Этап 2

A:=IntToStr(abs(BinToInt(B)+ROL(BinToInt(A)+FunG(B,C,d)+BinToInt(Bit32[2])+ BinToInt(T[17]), 5)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunG(A,B,C)+BinToInt(Bit32[7])+ BinToInt(T[18]), 9)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunG(D,A,B)+BinToInt(Bit32[12])+BinToInt(T[19]),14)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunG(C,D,A)+BinToInt(Bit32[1])+ BinToInt(T[20]),20)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunG(B,C,d)+BinToInt(Bit32[6])+ BinToInt(T[21]), 5)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunG(A,B,C)+BinToInt(Bit32[11])+BinToInt(T[22]), 9)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunG(D,A,B)+BinToInt(Bit32[16])+BinToInt(T[23]),14)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunG(C,D,A)+BinToInt(Bit32[5])+ BinToInt(T[24]),20)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunG(B,C,d)+BinToInt(Bit32[10])+BinToInt(T[25]), 5)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunG(A,B,C)+BinToInt(Bit32[15])+BinToInt(T[26]), 9)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunG(D,A,B)+BinToInt(Bit32[4])+ BinToInt(T[27]),14)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunG(C,D,A)+BinToInt(Bit32[9])+ BinToInt(T[28]),20)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunG(B,C,d)+BinToInt(Bit32[14])+BinToInt(T[29]), 5)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunG(A,B,C)+BinToInt(Bit32[3])+ BinToInt(T[30]), 9)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunG(D,A,B)+BinToInt(Bit32[8])+ BinToInt(T[31]),14)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunG(C,D,A)+BinToInt(Bit32[13])+BinToInt(T[32]),20)));

//Этап 3

A:=IntToStr(abs(BinToInt(B)+ROL(BinToInt(A)+FunH(B,C,d)+BinToInt(Bit32[6])+ BinToInt(T[33]), 4)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunH(A,B,C)+BinToInt(Bit32[9])+ BinToInt(T[34]),11)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunH(D,A,B)+BinToInt(Bit32[12])+BinToInt(T[35]),16)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunH(C,D,A)+BinToInt(Bit32[15])+BinToInt(T[36]),23)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunH(B,C,d)+BinToInt(Bit32[2])+ BinToInt(T[37]), 4)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunH(A,B,C)+BinToInt(Bit32[5])+ BinToInt(T[38]),11)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunH(D,A,B)+BinToInt(Bit32[8])+ BinToInt(T[39]),16)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunH(C,D,A)+BinToInt(Bit32[11])+BinToInt(T[40]),23)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunH(B,C,d)+BinToInt(Bit32[14])+BinToInt(T[41]), 4)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunH(A,B,C)+BinToInt(Bit32[1])+ BinToInt(T[42]),11)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunH(D,A,B)+BinToInt(Bit32[4])+ BinToInt(T[43]),16)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunH(C,D,A)+BinToInt(Bit32[7])+ BinToInt(T[44]),23)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunH(B,C,d)+BinToInt(Bit32[10])+BinToInt(T[45]), 4)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunH(A,B,C)+BinToInt(Bit32[13])+BinToInt(T[46]),11)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunH(D,A,B)+BinToInt(Bit32[16])+BinToInt(T[47]),16)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunH(C,D,A)+BinToInt(Bit32[3])+ BinToInt(T[48]),23)));

//Этап 4

A:=IntToStr(abs(BinToInt(B)+ROL(BinToInt(A)+FunI(B,C,d)+BinToInt(Bit32[1])+ BinToInt(T[49]), 6)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunI(A,B,C)+BinToInt(Bit32[8])+ BinToInt(T[50]),10)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunI(D,A,B)+BinToInt(Bit32[15])+BinToInt(T[51]),15)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunI(B,C,d)+BinToInt(Bit32[13])+BinToInt(T[53]), 6)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunI(A,B,C)+BinToInt(Bit32[4])+ BinToInt(T[54]),10)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunI(D,A,B)+BinToInt(Bit32[11])+BinToInt(T[55]),15)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunI(C,D,A)+BinToInt(Bit32[2])+ BinToInt(T[56]),21)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunI(B,C,d)+BinToInt(Bit32[9])+ BinToInt(T[57]), 6)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunI(A,B,C)+BinToInt(Bit32[16])+BinToInt(T[58]),10)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunI(D,A,B)+BinToInt(Bit32[7])+ BinToInt(T[59]),15)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunI(C,D,A)+BinToInt(Bit32[14])+BinToInt(T[60]),21)));

a:=IntToStr(abs(BinToInt(b)+ROL(BinToInt(a)+FunI(B,C,d)+BinToInt(Bit32[5])+ BinToInt(T[61]), 6)));

D:=IntToStr(abs(BinToInt(A)+ROL(BinToInt(D)+FunI(A,B,C)+BinToInt(Bit32[12])+BinToInt(T[62]),10)));

C:=IntToStr(abs(BinToInt(D)+ROL(BinToInt(a)+FunI(D,A,B)+BinToInt(Bit32[3])+ BinToInt(T[63]),15)));

B:=IntToStr(abs(BinToInt(C)+ROL(BinToInt(a)+FunI(C,D,A)+BinToInt(Bit32[10])+BinToInt(T[64]),21)));

A:=IntToStr(StrToInt(AA)+StrToInt(A));

B:=FloatToStr(StrToFloat(BB)+StrToFloat(B));

C:=IntToStr(StrToInt(CC)+StrToInt(C));

D:=IntToStr(StrToInt(DD)+StrToInt(D));

end;

//Конец пятой ступени

//Перевод в 16 систему

//Элемент A

for i:=1 to 1 do

begin

AttachThreadInput:=StrToInt(A);

AString:='';

while AttachThreadInput>=16 do

begin

if (AttachThreadInput mod 16)=0 then

AString:=IntToStr(0)+AString;

if (AttachThreadInput mod 16)=1 then

AString:=IntToStr(1)+AString;

if (AttachThreadInput mod 16)=2 then

AString:=IntToStr(2)+AString;

if (AttachThreadInput mod 16)=3 then

AString:=IntToStr(3)+AString;

if (AttachThreadInput mod 16)=4 then

AString:=IntToStr(4)+AString;

if (AttachThreadInput mod 16)=5 then

AString:=IntToStr(5)+AString;

if (AttachThreadInput mod 16)=6 then

AString:=IntToStr(6)+AString;

if (AttachThreadInput mod 16)=7 then

AString:=IntToStr(7)+AString;

if (AttachThreadInput mod 16)=8 then

AString:=IntToStr(8)+AString;

if (AttachThreadInput mod 16)=9 then

AString:=IntToStr(9)+AString;

if (AttachThreadInput mod 16)=10 then

AString:='A'+AString;

if (AttachThreadInput mod 16)=11 then

AString:='B'+AString;

if (AttachThreadInput mod 16)=12 then

AString:='C'+AString;

if (AttachThreadInput mod 16)=13 then

AString:='D'+AString;

if (AttachThreadInput mod 16)=14 then

AString:='E'+AString;

if (AttachThreadInput mod 16)=15 then

AString:='F'+AString;

AttachThreadInput:=AttachThreadInput div 16;

end;

if (AttachThreadInput mod 16)=0 then

AString:=IntToStr(0)+AString;

if (AttachThreadInput mod 16)=1 then

AString:=IntToStr(1)+AString;

if (AttachThreadInput mod 16)=2 then

AString:=IntToStr(2)+AString;

if (AttachThreadInput mod 16)=3 then

AString:=IntToStr(3)+AString;

if (AttachThreadInput mod 16)=4 then

AString:=IntToStr(4)+AString;

if (AttachThreadInput mod 16)=5 then

AString:=IntToStr(5)+AString;

if (AttachThreadInput mod 16)=6 then

AString:=IntToStr(6)+AString;

if (AttachThreadInput mod 16)=7 then

AString:=IntToStr(7)+AString;

if (AttachThreadInput mod 16)=8 then

AString:=IntToStr(8)+AString;

if (AttachThreadInput mod 16)=9 then

AString:=IntToStr(9)+AString;

if (AttachThreadInput mod 16)=10 then

AString:='A'+AString;

if (AttachThreadInput mod 16)=11 then

AString:='B'+AString;

if (AttachThreadInput mod 16)=12 then

AString:='C'+AString;

if (AttachThreadInput mod 16)=13 then

AString:='D'+AString;

if (AttachThreadInput mod 16)=14 then

AString:='E'+AString;

if (AttachThreadInput mod 16)=15 then

AString:='F'+AString;

end;

//Элемент B

for i:=1 to 1 do

begin

AttachThreadInput:=StrToInt(B);

BString:='';

while AttachThreadInput>=16 do

begin

if (AttachThreadInput mod 16)=0 then

BString:=IntToStr(0)+BString;

if (AttachThreadInput mod 16)=1 then

BString:=IntToStr(1)+BString;

if (AttachThreadInput mod 16)=2 then

BString:=IntToStr(2)+BString;

if (AttachThreadInput mod 16)=3 then

BString:=IntToStr(3)+BString;

if (AttachThreadInput mod 16)=4 then

BString:=IntToStr(4)+BString;

if (AttachThreadInput mod 16)=5 then

BString:=IntToStr(5)+BString;

if (AttachThreadInput mod 16)=6 then

BString:=IntToStr(6)+BString;

if (AttachThreadInput mod 16)=7 then

BString:=IntToStr(7)+BString;

if (AttachThreadInput mod 16)=8 then

BString:=IntToStr(8)+BString;

if (AttachThreadInput mod 16)=9 then

BString:=IntToStr(9)+BString;

if (AttachThreadInput mod 16)=10 then

BString:='A'+BString;

if (AttachThreadInput mod 16)=11 then

BString:='B'+BString;

if (AttachThreadInput mod 16)=12 then

BString:='C'+BString;

if (AttachThreadInput mod 16)=13 then

BString:='D'+BString;

if (AttachThreadInput mod 16)=14 then

BString:='E'+BString;

if (AttachThreadInput mod 16)=15 then

BString:='F'+BString;

AttachThreadInput:=AttachThreadInput div 16;

end;

if (AttachThreadInput mod 16)=0 then

BString:=IntToStr(0)+BString;

if (AttachThreadInput mod 16)=1 then

BString:=IntToStr(1)+BString;

if (AttachThreadInput mod 16)=2 then

BString:=IntToStr(2)+BString;

if (AttachThreadInput mod 16)=3 then

BString:=IntToStr(3)+BString;

if (AttachThreadInput mod 16)=4 then

BString:=IntToStr(4)+BString;

if (AttachThreadInput mod 16)=5 then

BString:=IntToStr(5)+BString;

if (AttachThreadInput mod 16)=6 then

BString:=IntToStr(6)+BString;

if (AttachThreadInput mod 16)=7 then

BString:=IntToStr(7)+BString;

if (AttachThreadInput mod 16)=8 then

BString:=IntToStr(8)+BString;

if (AttachThreadInput mod 16)=9 then

BString:=IntToStr(9)+BString;

if (AttachThreadInput mod 16)=10 then

BString:='A'+BString;

if (AttachThreadInput mod 16)=11 then

BString:='B'+BString;

if (AttachThreadInput mod 16)=12 then

BString:='C'+BString;

if (AttachThreadInput mod 16)=13 then

BString:='D'+BString;

if (AttachThreadInput mod 16)=14 then

BString:='E'+BString;

if (AttachThreadInput mod 16)=15 then

BString:='F'+BString;

end;

//Элемент C

for i:=1 to 1 do

begin

AttachThreadInput:=StrToInt(C);

CString:='';

while AttachThreadInput>=16 do

begin

if (AttachThreadInput mod 16)=0 then

CString:=IntToStr(0)+CString;

if (AttachThreadInput mod 16)=1 then

CString:=IntToStr(1)+CString;

if (AttachThreadInput mod 16)=2 then

CString:=IntToStr(2)+CString;

if (AttachThreadInput mod 16)=3 then

CString:=IntToStr(3)+CString;

if (AttachThreadInput mod 16)=4 then

CString:=IntToStr(4)+CString;

if (AttachThreadInput mod 16)=5 then

CString:=IntToStr(5)+CString;

if (AttachThreadInput mod 16)=6 then

CString:=IntToStr(6)+CString;

if (AttachThreadInput mod 16)=7 then

CString:=IntToStr(7)+CString;

if (AttachThreadInput mod 16)=8 then

CString:=IntToStr(8)+CString;

if (AttachThreadInput mod 16)=9 then

CString:=IntToStr(9)+CString;

if (AttachThreadInput mod 16)=10 then

CString:='A'+CString;

if (AttachThreadInput mod 16)=11 then

CString:='B'+CString;

if (AttachThreadInput mod 16)=12 then

CString:='C'+CString;

if (AttachThreadInput mod 16)=13 then

CString:='D'+CString;

if (AttachThreadInput mod 16)=14 then

CString:='E'+CString;

if (AttachThreadInput mod 16)=15 then

CString:='F'+CString;

AttachThreadInput:=AttachThreadInput div 16;

end;

if (AttachThreadInput mod 16)=0 then

CString:=IntToStr(0)+CString;

if (AttachThreadInput mod 16)=1 then

CString:=IntToStr(1)+CString;

if (AttachThreadInput mod 16)=2 then

CString:=IntToStr(2)+CString;

if (AttachThreadInput mod 16)=3 then

CString:=IntToStr(3)+CString;

if (AttachThreadInput mod 16)=4 then

CString:=IntToStr(4)+CString;

if (AttachThreadInput mod 16)=5 then

CString:=IntToStr(5)+CString;

if (AttachThreadInput mod 16)=6 then

CString:=IntToStr(6)+CString;

if (AttachThreadInput mod 16)=7 then

CString:=IntToStr(7)+CString;

if (AttachThreadInput mod 16)=8 then

CString:=IntToStr(8)+CString;

if (AttachThreadInput mod 16)=9 then

CString:=IntToStr(9)+CString;

if (AttachThreadInput mod 16)=10 then

CString:='A'+CString;

if (AttachThreadInput mod 16)=11 then

CString:='B'+CString;

if (AttachThreadInput mod 16)=12 then

CString:='C'+CString;

if (AttachThreadInput mod 16)=13 then

CString:='D'+CString;

if (AttachThreadInput mod 16)=14 then

CString:='E'+CString;

if (AttachThreadInput mod 16)=15 then

CString:='F'+CString;

end;

//Элемент D

for i:=1 to 1 do

begin

AttachThreadInput:=StrToInt(D);

DString:='';

while AttachThreadInput>=16 do

begin

if (AttachThreadInput mod 16)=0 then

DString:=IntToStr(0)+DString;

if (AttachThreadInput mod 16)=1 then

DString:=IntToStr(1)+DString;

if (AttachThreadInput mod 16)=2 then

DString:=IntToStr(2)+DString;

if (AttachThreadInput mod 16)=3 then

DString:=IntToStr(3)+DString;

if (AttachThreadInput mod 16)=4 then

DString:=IntToStr(4)+DString;

if (AttachThreadInput mod 16)=5 then

DString:=IntToStr(5)+DString;

if (AttachThreadInput mod 16)=6 then

DString:=IntToStr(6)+DString;

if (AttachThreadInput mod 16)=7 then

DString:=IntToStr(7)+DString;

if (AttachThreadInput mod 16)=8 then

DString:=IntToStr(8)+DString;

if (AttachThreadInput mod 16)=9 then

DString:=IntToStr(9)+DString;

if (AttachThreadInput mod 16)=10 then

DString:='A'+DString;

if (AttachThreadInput mod 16)=11 then

DString:='B'+dString;

if (AttachThreadInput mod 16)=12 then

DString:='C'+DString;

if (AttachThreadInput mod 16)=13 then

DString:='D'+DString;

if (AttachThreadInput mod 16)=14 then

DString:='E'+DString;

if (AttachThreadInput mod 16)=15 then

DString:='F'+DString;

AttachThreadInput:=AttachThreadInput div 16;

end;

if (AttachThreadInput mod 16)=0 then

DString:=IntToStr(0)+DString;

if (AttachThreadInput mod 16)=1 then

DString:=IntToStr(1)+DString;

if (AttachThreadInput mod 16)=2 then

DString:=IntToStr(2)+DString;

if (AttachThreadInput mod 16)=3 then

DString:=IntToStr(3)+DString;

if (AttachThreadInput mod 16)=4 then

DString:=IntToStr(4)+DString;

if (AttachThreadInput mod 16)=5 then

DString:=IntToStr(5)+DString;

if (AttachThreadInput mod 16)=6 then

DString:=IntToStr(6)+DString;

if (AttachThreadInput mod 16)=7 then

DString:=IntToStr(7)+DString;

if (AttachThreadInput mod 16)=8 then

DString:=IntToStr(8)+DString;

if (AttachThreadInput mod 16)=9 then

DString:=IntToStr(9)+DString;

if (AttachThreadInput mod 16)=10 then

DString:='A'+DString;

if (AttachThreadInput mod 16)=11 then

DString:='B'+dString;

if (AttachThreadInput mod 16)=12 then

DString:='C'+DString;

if (AttachThreadInput mod 16)=13 then

DString:='D'+DString;

if (AttachThreadInput mod 16)=14 then

DString:='E'+DString;

if (AttachThreadInput mod 16)=15 then

DString:='F'+DString;

end;

//Вывод конечного результата

Label3.Caption:=DString+CString+BString+AString;

end;

end.

1. Статья о Hans Peter Luhn [Электронный ресурс]: <https://en.wikipedia.org/wiki/Hans_Peter_Luhn> [↑](#footnote-ref-1)
2. Список словарей: Электронный ресурс: <http://wordbook.xyz/download> [↑](#footnote-ref-2)
3. Статья о National Security Agency [Электронный ресурс]: <https://en.wikipedia.org/wiki/National_Security_Agency> [↑](#footnote-ref-3)
4. # Статья о TLS [Электронный ресурс]:

   <https://habr.com/post/258285/> [↑](#footnote-ref-4)
5. Сайт ФАПСИ [Электронный ресурс]:

   http://xn--80aa2bkafhg.xn--p1ai/26142/%D4%E5%E4%E5%F0%E0%EB%FC%ED%EE%E5-%E0%E3%E5%ED%F2%F1%F2%E2%EE-%EF%F0%E0%E2%E8%F2%E5%EB%FC%F1%F2%E2%E5%ED%ED%EE%E9-%F1%E2%FF%E7%E8-%E8-%E8%ED%F4%EE%F0%EC%E0%F6%E8%E8-%EF%F0%E8-%CF%F0%E5%E7%E8%E4%E5%ED%F2%E5-%D0%D4-%D4%C0%CF%D1%C8 [↑](#footnote-ref-5)
6. Статья о Whirlpool[Электронный ресурс]:

   <http://mind-control.wikia.com/wiki/Whirlpool_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)> [↑](#footnote-ref-6)
7. Статья о FreeBSD [Электронный ресурс]:

   <https://dic.academic.ru/dic.nsf/ruwiki/2538> [↑](#footnote-ref-7)
8. Статья о Delphi [Электронный ресурс]:

   <https://7lafa.com/pageanswer.php?id=6194> [↑](#footnote-ref-8)