ВВЕДЕНИЕ

Появившиеся в начале 80-ых персональные ЭВМ (ПЭВМ или ПК) прочно вошли во все сферы человеческой деятельности. Вместе с ними у эксплуатирующих ПЭВМ организаций и ведомств возникли и многочисленные проблемы. Одна из них — защита информации. Согласно статистическим данным более 80% компаний и агентств несут финансовые убытки из-за нарушения безопасности данных. Проблема защиты информации представляет собой совокупность тесно связанных проблем в областях права, организации управления, разработки технических средств, программирования и математики. Одна из центральных задач проектирования систем защиты состоит в рациональном распределении имеющихся ресурсов.

Характерная особенность использования ПЭВМ в нашей стране заключатся в том, что доступ к ним имеют многие пользователи. В связи с таким "многопользовательским" режимом работы возникает целый набор взаимосвязанных вопросов по защите информации, хранящейся в ПЭВМ. При создании и использовании ПЭВМ возникает целый ряд взаимосвязанных

теоретических и практических проблем. В коммерческих и военных областях одной из основных является проблема защиты информации. Так можно выделить следующие объективные причины, определяющие важность проблемы защиты информации:

· высокие темпы роста парка ПЭВМ, находящихся в эксплуатации;

· широкое применение ПЭВМ в самых различных сферах человеческой деятельности;

· высокая степень концентрации информации в ПЭВМ;

· совершенствование способов доступа пользователей к ресурсам ПЭВМ;

· усложнение вычислительного процесса в ПЭВМ.

Усложнение методов и средств организации машинной обработки информации приводят к тому, что информация становится все более уязвимой. Этому способствуют такие факторы, как постоянно возрастающие объемы обрабатываемых данных, накопление и хранение данных в ограниченных местах, постоянное расширение круга пользователей, имеющих доступ как к ресурсам ПЭВМ, так и к программам и данным, хранящихся в них, усложнение режимов эксплуатации вычислительных систем и т. п.

Основной целью выполнения курсового проекта является расширение, углубление знаний по дисциплине «Основы алгоритмизации и программирование», формирование навыков научно-исследовательской деятельности.

Задачи выполнения курсового проекта:

* систематизация знаний в области программирования на языке Pascal.
* углубление уровня и расширение объема профессионально значимых знаний, умений и навыков разработки программ в среде Delphi;
* формирование умений и навыков самостоятельной организации научно-исследовательской работы;

- овладение современными методами поиска, обработки и использования информации.

Данная пояснительная записка содержит следующие разделы:

1. Раздел «Постановка задачи» содержит описание всех функций проекта, входных и выходных данных.
2. В разделе «Вычислительная система» приводится описание операционной системы, языка программирования и среды разработки выбранных для реализации проекта.
3. Раздел «Проектирование» содержит описание внешнего пользовательского интерфейса, описание алгоритма программы и структуры справочной системы.
4. В разделе «Описание программы» описываются все библиотечные модули, входящие в состав программы, а также назначение всех функций, входящих в приложение, здесь же приводится интерфейс программы с рисунками пользовательских форм.
5. Раздел «Отладка и испытание программы» иллюстрирует проверку работы программы на различных тестах и содержит анализ полученных результатов.
6. В разделе «Описание применения программы» содержатся сведения о назначении программного средства, классе решаемых задач и ограничениях, накладываемых на область применения.
7. В «Заключении» описываются использованные методы и средства, полнота раскрытия проблемы и возможность дальнейшей модификации проекта.
8. ПОСТАНОВКА ЗАДАЧИ
   1. Формулировка задания

Необходимо разработать аппаратно-программный комплекс «Защита данных от НСД». Реализовать следующие операции:

1. Аутентификация пользователя по ключевому USB-носителю
2. Шифрование выбранных файлов по паролю или по USB-носителю
3. Пример работы шифрования

Предусмотреть возможность выбора USB-носителя для шифрования файлов пользователем.

Реализовать контроль за вводом пароля, чтоб пароль не пыл пустым, при вводе пароля пустым будет выдаваться соответствующее сообщение.

Разработать справочную систему для приложения.

* 1. Определение данных и их представление

Входные данные:

Т. к. программа работает с файлами (шифрование) то входными данными являются файлы различного типа для шифрования.

Выходные данные:

Зашифрованные файлы и защищенные программы.Вычислительная система

**2**

* 1. Обоснование выбора языка программирования

Pascal – один из наиболее известных языков программирования, который очень эффективен для обучения структурному программированию, благодаря небольшому количеству синтаксических конструкций.

Особенностями языка является строгая типизация и наличие средств структурного программирования. Pascal способствует дисциплинированному программированию, так как на ряду со строгой типизацией там сведены к минимуму возможные синтаксические неоднозначности, при этом синтаксис языка интуитивно понятен.

Диалект Object Pascal разработанный в 1985 году поддерживает объектно-ориентированное программирование.

* 1. Обоснование выбора среды разработки

Delphi – интегрированная среда разработки программного обеспечения для операционной системы Windows.

Высоко производительный инструмент визуального построения приложений включает в себя компилятор кода и предоставляет средства визуального программирования. В Delphi так же входят локальный SQL-сервер, генераторы отчетов, библиотеки компонентов.

Преимущества Delphi:

* Быстрая разработка приложений
* Высокая производительность разработанных приложений
* Низкие требование разработанного приложения к ресурсам компьютера
* Наращиваемость за счет встраивания новых компонент в среду Delphi
* Возможность разработки новых компонентов собственными средствами Delphi
* Удачная проработка иерархии объектов
  1. Требования к конфигурации программного и аппаратного обеспечения

Windows 7 – пользовательская операционная система семейства Windows NT. В этой операционной системе реализована поддержка Unicode 5.1, она обладает поддержкой мультитач-управления.

Дополнительным преимуществом Windows 7 является более тесная интеграция с производителями драйверов, 50 новых шрифтов, улучшенная совместимость со старыми приложениями.

Сетевая технология Branch Cache позволяет кэшировать содержимое интернет трафика.

Минимальные требования:

- процессор Intel Pentium III 420 Мгц

- ОЗУ 256 Мбайт

- Место на HDD 1024 Мбайт

1. Проектирование
   1. Проектирование интерфейса

Главная форма приложения содержит 4 компонента типа TButton:

* Шифрование файлов
* Пример работы
* О программе
* Выход

При нажатии на кнопку «Шифрование файлов», открывается новая форма «Шифрование файлов», на которой расположены 1 компонент TMemo для вывода информации о шифровании пользователю. 4 копонента TButton для выбора метода шифрования или расшифровки файлов. 2 компонента TComboBox для выбора USB-носителя для шифрования. 1 компонент TImage для выхода в главное меню.

При нажатии на кнопку «Пример работы» открывается новая форма «Пример работы», на которой расположены 2 компонента TMemo для ввода текста для шифрования и вывода шифрованного текста. Ещё компонент TButton для шифрования введённого текста. 1 компонент TImage для выхода в главное меню.

При нажатии на кнопку «О программе» открывается справочная система программы. Где пользователь может посмотреть как пользоваться программой, различные правила шифрования или расшифровки файлов.

Так же есть форма, которую мы будем видеть самой первой, но она не является главной. На этой форме содержится 2 компонента TButton для входа в приложение или же создания ключевого USB-носителя. 1 компонент TComboBox для выбора USB-носителя­.

* 1. Описание алгоритма

Перед описанием алгоритма следует ввести некоторые термины.

Зашифрованием данных называется процесс преобразования открытых данных в зашифрованные с помощью шифра, а расшифрованием данных — процесс преобразования закрытых данных в открытые с помощью шифра.

Шифрованием называется процесс зашифрования или расшифрования данных.

Дешифрованием будем называть процесс преобразования закрытых данных в открытые при неизвестном ключе и, возможно, неизвестном алгоритме.

Криптографическая защита — это защита данных с помощью криптографического преобразования, под которым понимается преобразование данных шифрованием.

Уравнение зашифрования — соотношение, описывающее процесс образования зашифрованных данных из открытых данных в результате преобразований, заданных алгоритмом криптографического преобразования.

Уравнение расшифрования — соотношение, описывающее процесс образования открытых данных из зашифрованных данных в результате преобразований, заданных алгоритмом криптографического преобразования.

Под шифром понимается совокупность обратимых преобразований множества открытых данных на множество зашифрованных данных, заданных алгоритмом криптографического преобразования.

Криптостойкостью называется характеристика шифра, определяющая его стойкость к дешифрованию. Обычно эта характеристика определяется периодом времени, необходимым для дешифрования.

Принцип зашифрования заключается в генерации гаммы шифра с помощью датчика псевдослучайных чисел (ПСЧ) и наложением полученной гаммы на открытые данные с помощью логической операции “исключающее ИЛИ” (т. е. обратимым образом). Процесс расшифрования данных сводится к повторной генерации гаммы шифра при известном ключе и наложению такой гаммы на зашифрованные данные. Полученный зашифрованный текст является достаточно трудным для раскрытия в том случае, когда гамма шифра не содержит повторяющихся битовых последовательностей. По сути дела гамма шифра должна изменяться случайным образом для каждого шифруемого слова. Фактически если период гаммы превышает длину всего зашифрованного текста и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (подбором ключа). В этом случае криптостойкость определяется размером ключа. Чтобы получить линейные последовательности элементов гаммы, используются датчики ПСЧ. К настоящему времени на основе теории групп разработано несколько типов таких датчиков.

В своей программе я использовал так называемый конгруэнтный генератор ПСЧ — наиболее доступный и эффективный. Для этого класса генераторов ПСЧ можно сделать математически строгое заключение о том, какими свойствами обладают выходные сигналы этих генераторов с точки зрения периодичности и случайности. Данный линейный конгруэнтный датчик ПСЧ вырабатывает последовательности псевдослучайных чисел T(i), описываемые соотношением:

, (1)

T(0) — исходная величина, выбранная в качестве порождающего числа. Этот датчик ПСЧ генерирует псевдослучайные числа с определенным периодом повторения, зависящим от выбранных значений A и C. Значение М обычно устанавливается равным 2b, где b — длина слова ЭВМ в битах. Датчик имеет максимальный период М до того, как генерируемая последовательность чисел начнет повторяться. Линейный конгруэнтный датчик ПСЧ имеет максимальную длину М тогда и только тогда, когда A mod 4 = 1 и С — нечетное. В своей программе я положил А = 5, С = 27, Т(0) — пароль, вводимый пользователем.

С полученной последовательностью Т(i) поступают следующим образом:

, (2)

где D(i) — последовательность открытых данных, F(i) — последовательность зашифрованных данных.

Также при разработке алгоритма шифрования использовался алгоритм американского федерального стандарта на шифрование данных — Data Encryption Standard (DES).

При зашифровании входные данные шифруются по формуле (2), далее обрабатываются блоками по 64 слова (word). Эта обработка заключается в следующем: 4 слова переставляются в соответствии с таблицей, изображенной на рисунке 1:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **40** | **8** | **48** | **16** | **56** | **24** | **64** | **32** | |  |
| **39** | **7** | **47** | **15** | **55** | **23** | **63** | **31** | |  |
| **38** | **6** | **46** | **14** | **54** | **22** | **62** | **30** | |  |
| **37** | **5** | **45** | **13** | **53** | **21** | **61** | **29** | |  |
| **36** | **4** | **44** | **12** | **52** | **20** | **60** | **28** | |  |
| **35** | **3** | **43** | **11** | **51** | **19** | **59** | **27** | |  |
| **34** | **2** | **42** | **10** | **50** | **18** | **58** | **26** | |  |
| **33** | **1** | **41** | **9** | **49** | **17** | **57** | | **25** | |
|  |  |  |  |  |  |  | |  | |

*Рисунок 1. Перестановка после зашифрования.*

Как видно из данной таблицы, слово 40 входной последовательности становится 1-ым, слово 8 — 2-ым и т. д. Процесс расшифрования данных является инверсным относительно процесса зашифрования. Т. е. данные сначала переставляются в соответствии с таблицей, изображенной на рисунке 2, а затем преобразуются по формуле (2). Как легко видеть, данная перестановка является обратной по отношению к начальной.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **58** | **50** | **42** | **34** | **26** | **18** | **10** | **2** | |  |
| **60** | **52** | **44** | **36** | **28** | **20** | **12** | **4** | |  |
| **62** | **54** | **46** | **38** | **30** | **22** | **14** | **6** | |  |
| **64** | **56** | **48** | **40** | **32** | **24** | **16** | **8** | |  |
| **57** | **49** | **41** | **33** | **25** | **17** | **9** | **1** | |  |
| **59** | **51** | **43** | **35** | **27** | **19** | **11** | **3** | |  |
| **61** | **53** | **45** | **37** | **29** | **21** | **13** | **5** | |  |
| **63** | **55** | **47** | **39** | **31** | **23** | **15** | | **7** | |
|  |  |  |  |  |  |  | |  | |
|  |  |  |  |  |  |  |  |  |  |

*Рисунок 2. Перестановка перед расшифрованием.*

**Описание программы**

* 1. Логическая структура

Основная форма (Form1):

- procedure Button1Click(Sender: TObject); - открытие формы «Шифрование файлов».

- procedure Button2Click(Sender: TObject); - открытие формы «Пример работы».

- procedure Button3Click(Sender: TObject); - открывает справку.

- procedure FormClose(Sender: TObject; var Action: TCloseAction); - закрывает форму и выходит из программы.

Форма «Шифрование файлов»(Form2):

- procedure Button1Click(Sender: TObject); - шифрование выбранного файла по паролю.

- procedure Button2Click(Sender: TObject); - расшифрование выбранного файла по паролю.

- procedure FormCreate(Sender: TObject); - записывает определённые инструкции в TMemo1.

- procedure Image1Click(Sender: TObject); - переход на главную форму.

- procedure FormClose(Sender: TObject; var Action: TCloseAction); - закрывает форму и выходит из программы.

- procedure Button3Click(Sender: TObject); - выбор USB-носителя для шифрования.

- procedure Button4Click(Sender: TObject); - выбор USB-носителя для расшифрования.

- procedure ComboBox1Change(Sender: TObject); - шифрование файла по выбранному USB-носителю.

- procedure ComboBox2Change(Sender: TObject); - расшифрование файла по выбранному USB-носителю.

- procedure Image1MouseEnter(Sender: TObject); - изменение курсора при наведении на картинку.

- procedure Image2Click(Sender: TObject); - переход назад к выбору метода шифрования.

- procedure Image2MouseEnter(Sender: TObject); - изменение курсора при наведении на картинку.

- procedure pass; - получение пароля.

Форма «Пример работы» (Form3):

- procedure Button1Click(Sender: TObject); -шифрование введённого текста.

- procedure FormCreate(Sender: TObject); -очистка полей TMemo1 и TMemo2.

- procedure FormClose(Sender: TObject; var Action: TCloseAction); - закрывает форму и выходит из программы.

- procedure Image1Click(Sender: TObject); - переход на главную форму.

- procedure Memo1Change(Sender: TObject); - активирует кнопку если в TMemo1 был введён какой-то текст.

- procedure Image1MouseEnter(Sender: TObject); - изменение курсора при наведении на картинку.

Форма «Вход в приложение» (Form4):

- procedure Button1Click(Sender: TObject); - считывание всех подключённых USB-носителей к ПК.

- procedure ComboBox1Change(Sender: TObject); - создание ключевого USB-носителя.

- procedure FormCreate(Sender: TObject); - проверка на то, был ли создан ключевой USB-носитель ранее.

- procedure Button2Click(Sender: TObject); - вход в приложение по ключевому USB-носителю.

* 1. Физическая структура

Unit1 – основная форма программы, служит для выбора выполняемой операции.

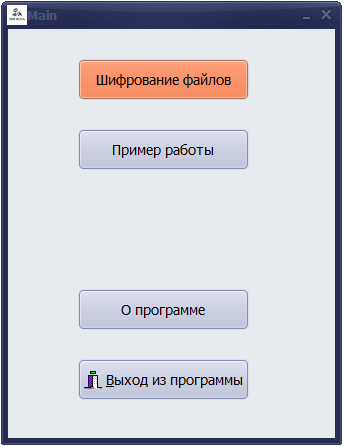
Unit2 – форма «Шифрование файлов», служит для шифрования выбранных файлов.

Unit3 – форма «Пример работы», служит для демонстрации примера работы шифрования.

Unit4 – форма «Вход в приложение», служит для проверки входа по ключевому USB-носителю.

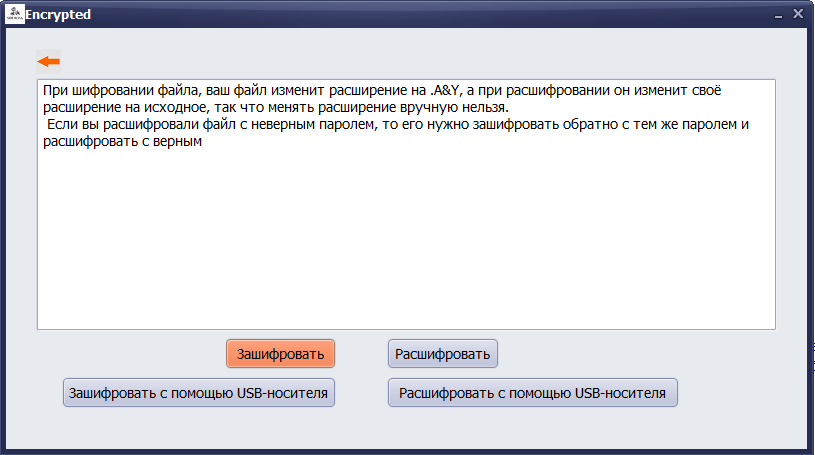
Особенности интерфейса

Интерфейс главной формы приложения показан на рисунке 3.



*Рисунок 3 – Главная форма приложения*

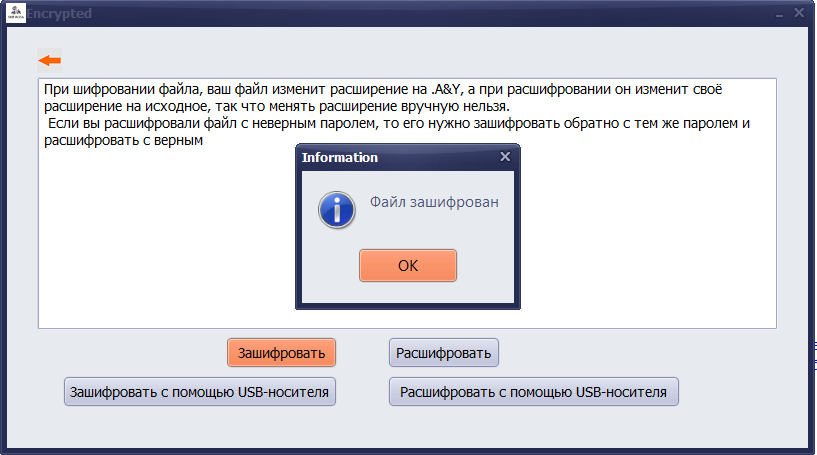
Интерфейс формы «Шифрование файлов» показан на рисунке 4.



*Рисунок 4 – Форма «Шифрование файлов»*

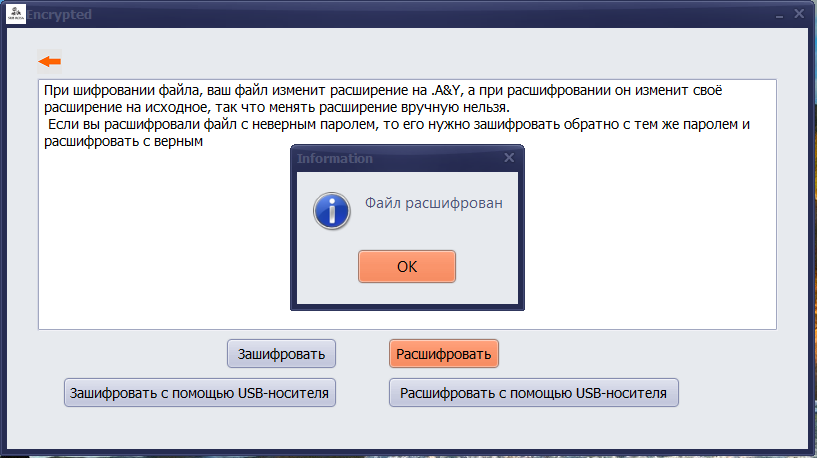
1. ОТЛАДКА И ИСПЫТАНИЕ ПРОГРАММЫ
   1. Тестовые примеры

Тест №1: Шифрование выбранного файла с верно введённым паролем или выбранным USB-носителем. Результат приведён на рисунке 5.



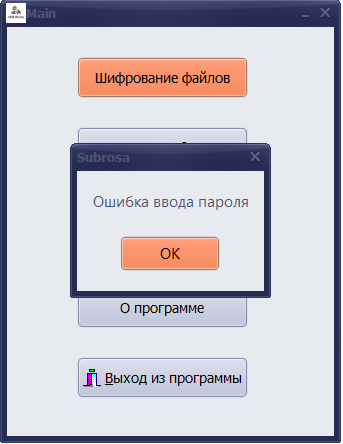
*Рисунок 5 – Тест №1*

Тест №2: Расшифрование выбранного файла с верно введённым паролем или USB-носителем. Результат приведён на рисунке 6.



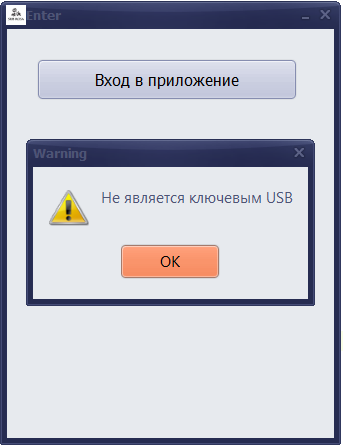
*Рисунок 6 – Тест №2*

Тест №3: Шифрование или расшифрование выбранного файла с неверно введённым паролем. Результат приведён на рисунке 7.



*Рисунок 7 – Тест №3*

Тест №4: Вход в приложение с не ключевым USB-носителем. Результат приведён на рисунке 8.



*Рисунок 8 – Тест №4*

* 1. Анализ полученных результатов

Тест №1 показал, что при корректно введённом пароле файл зашифровывается корректно и без каких-либо ошибок.

Тест №2 показал, что при корректно введённом пароле файл расшифровывается корректно и без каких-либо ошибок.

Тест №3 показал, что при некорректно введённом пароле выдаётся соответствующее сообщение и программа выполнена не будет.

Тест №4 показал, что при не ключевом USB-носителе вход в приложение выполнен не будет.

1. ОПИСАНИЕ ПРИМЕНЕНИЯ ПРОГРАММЫ
   1. Назначение программы и условия применения

Данный аппаратно-программный используется для шифрования файлов или же любых программ, для защиты личной информации и собственной же безопасности.

Минимальные требования для корректной работы приложения:

- процессор Intel Pentium III 420 Мгц

- ОЗУ 256 Мбайт

- Место на HDD 1024 Мбайт

- VGA или монитор с более высоким разрешением;

- мышь, клавиатура.

Для запуска программы нужно зайти в папку Курсовая работа и запустить файл «SubRosa.exe». После запуска откроется окно. Пользователь при первом запуске программы должен создать ключевой USB-носитель, если же запуск программы осуществляется не в первый раз, то надо будет просто выполнить вход по ключевому USB-носителю. Далее представится меню с выбором функций программы.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте была разработана программа «Защита данных от несанкционированного доступа».

Я назвал свою программу SubRosa – втайне (буквально: под розой).

У древних римлян роза была эмблемой тайны. Когда хозяин дома вешал розу над пиршественным столом, гости знали, что все сказанное должно оставаться тайной.

Впоследствии изображение розы на потолке помещений, где происходили совещания, свидетельствовало о том, что все сказанное здесь не должно быть разглашено.

Было произведено тестирование разработанного приложения. Программа работает корректно при любых действиях пользователя.

Проект может быть доработан следующим образом:

* Добавить шифрование текстовых сообщений в мессенджерах на телефоне, а так же и на ПК при его отправлении от одного пользователя к другому и следовательно сделать расшифрование сообщений при их полученииььпользователем.Список использованных источников

1. Архангельский А.Я.Программирование в Delphi для Windows. М.: Бином-Пресс,2007.
2. Галисеев Г.В.Компоненты в Delphi 7. Профессиональная работа. М.: Издательский дом «Вильямс», 2004.
3. Долинский М.С.Алгоритмизация и программирование на TurboPascal: от простых до олимпиадных задач. СПб: Питер, 2004.
4. Хомоненко А.Д. и др. Delphi 7. СПб.: БХВ-Петербург, 2008.
5. Руссинович М. Внутреннее устройство Microsoft Windows, издательство Питер, 2013

Приложение 1

**(обязательное)**

**Листинг программы**

unit Unit1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

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

type

TForm1 = class(TForm)

Button1: TButton;

Button2: TButton;

Button3: TButton;

BitBtn1: TBitBtn;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

uses Unit2, Unit3;

procedure TForm1.Button1Click(Sender: TObject);

begin

Form1.Hide;

Form2.Show;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

Form1.Hide;

Form3.Show;

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Application.Terminate;

end;

end.

unit Unit2;

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.Menus,

Vcl.ExtCtrls;

type

TForm2 = class(TForm)

OpenDialog1: TOpenDialog;

Button1: TButton;

Button2: TButton;

OpenDialog2: TOpenDialog;

Memo1: TMemo;

Image1: TImage;

Button3: TButton;

Memo2: TMemo;

Button4: TButton;

Panel1: TPanel;

ComboBox1: TComboBox;

ComboBox2: TComboBox;

Image2: TImage;

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure Image1Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure ComboBox1Change(Sender: TObject);

procedure ComboBox2Change(Sender: TObject);

procedure Image1MouseEnter(Sender: TObject);

procedure Image2Click(Sender: TObject);

procedure Image2MouseEnter(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

const

A = 5; {Константы для}

C = 7; {генератора}

M =5536; {псевдослучайных чисел, далее - ПСЧ}

var

Form2: TForm2;

var

TempFile : file of byte;

InpF, OutF : file of word; {файлы на входе и выходе}

Password, Password1 : string; {переменные для работы с паролями}

OutputFileName, Exten : string; {переменные имен файлов}

I, J, K, tmp : byte; {переменные кодирования}

Temp, SCode, TByte, Code: word;

Position : LongInt; {переменные данных о процессе}

NowPos : real;

TPassword : array [1..255] of word;

MasByte, Mas, MasEnd, PS: array [1..64] of word; {массивы перестановок}

T : array [0..64] of word;

DirInfo, DirInfo1 : TSearchRec; {данные о файле}

Exten1 : string[3];

kolvo : integer;

implementation

{$R \*.dfm}

uses Unit1;

procedure pass;

var check, check1:boolean;

begin

kolvo := 0;

check := False;

check1 := False;

password := '';

password1 := '';

while check1 = False do

begin

password := InputBox('П А Р О Л Ь', 'ВВЕДИТЕ ПАРОЛЬ:', password);

if kolvo < 3 then

begin

if (password = '') then

begin

kolvo := kolvo + 1;

end;

if password <> '' then

begin

check1 := True;

end;

end;

if kolvo = 3 then

begin

Form2.Hide;

Form1.Show;

ShowMessage('Ошибка ввода пароля');

check := True;

break;

end;

end;

while check = False do

begin

if kolvo < 3 then

begin

password1 := '';

password1 := InputBox('П А Р О Л Ь', 'ВВЕДИТЕ ПАРОЛЬ ЕЩЁ РАЗ:', password1);

if (password1 = '') or (password1 <> password) then

begin

kolvo := kolvo + 1;

end;

if (password = password1) and (length(password) <> 0) then

begin

check := True;

end;

{else

begin

password1 := '';

password1 := InputBox('П А Р О Л Ь', 'ПАРОЛИ НЕ СОВПАДАЮТ, ВВЕДИТЕ ЕЩЁ РАЗ:', password1);

end; }

end;

if kolvo = 3 then

begin

Form2.Hide;

Form1.Show;

ShowMessage('Ошибка ввода пароля');

break;

end;

end;

end;

function GetHardDiskSerial(const DriveLetter: PChar): string;

var

NotUsed: DWORD;

VolumeFlags: DWORD;

VolumeInfo: array[0..MAX\_PATH] of Char;

VolumeSerialNumber: DWORD;

begin

GetVolumeInformation(PChar(DriveLetter),

nil, SizeOf(VolumeInfo), @VolumeSerialNumber, NotUsed,

VolumeFlags, nil, 0);

Result := Format('%8.8X', [VolumeSerialNumber])

end;

procedure TForm2.Button1Click(Sender: TObject);{шифрование файла}

var check, check1:boolean;

begin

if OpenDialog1.Execute then

begin

pass;{Получение пароля}

end;

if (OpenDialog1.FileName <> '') and (kolvo <> 3) then

begin

{Преобразовать файл}

FindFirst(OpenDialog1.FileName, faAnyFile, DirInfo);

if DirInfo.Size mod 2 = 1 then

begin

AssignFile(TempFile, OpenDialog1.FileName);

reset(TempFile);

while not EOF(TempFile) do read(TempFile, tmp);

tmp := 255;

write(TempFile, tmp);

CloseFile(TempFile);

end;

{Преобразование расширения файла}

assignFile(InpF, OpenDialog1.FileName);

reset(InpF);

for i := length(OpenDialog1.FileName) downto 1 do

if OpenDialog1.FileName[i] = '.' then

begin

OutputFileName := copy(OpenDialog1.FileName, 1, i) + 'A&Y';

break;

end;

assignFile(OutF, OutputFileName);

rewrite(OutF);

for i:= 0 to length(OpenDialog1.FileName) do

if OpenDialog1.FileName[length(OpenDialog1.FileName) - i] = '.' then

case i of

0: Exten := chr(0) + chr(0) + chr(0);

1: Exten := copy(OpenDialog1.FileName, length(OpenDialog1.FileName)-2, i) + chr(0) + chr(0);

2: Exten := copy(OpenDialog1.FileName, length(OpenDialog1.FileName)-2, i) + chr(0)

else Exten := copy(OpenDialog1.FileName, length(OpenDialog1.FileName)-2, 3)

end;

for i := 1 to 3 do

begin

Temp := ord(Exten[i]);

Write(OutF, Temp);

end;

{Начать шифрование}

k := 1;

repeat

{Считать из исходного файла блок размером 64\*word}

begin

for i := 1 to 64 do

if EOF(InpF) then MasByte[i] := 0 else read(InpF, MasByte[i]);

mas := MasByte;

T[0] := ord(Password[k]);

if k < length(password) then inc(k) else k := 1;

for i := 1 to 64 do

begin

{Шифровать с помощью ПСЧ}

code := mas[i];

T[i] := (A \* T[i-1] + C) mod M;

Code:=T[i] xor Code;

Mas[i] := Code;

end;

for i := 1 to 8 do

for j := 1 to 8 do

case i of

1: MasEnd[8\*(j-1)+i] := Mas[41-j];

2: MasEnd[8\*(j-1)+i] := Mas[09-j];

3: MasEnd[8\*(j-1)+i] := Mas[49-j];

4: MasEnd[8\*(j-1)+i] := Mas[17-j];

5: MasEnd[8\*(j-1)+i] := Mas[57-j];

6: MasEnd[8\*(j-1)+i] := Mas[25-j];

7: MasEnd[8\*(j-1)+i] := Mas[65-j];

8: MasEnd[8\*(j-1)+i] := Mas[33-j]

end;

for i := 1 to 64 do write(OutF, MasEnd[i]);

end;

until EOF(InpF);

closeFile(InpF);

Erase(InpF);

CloseFile(OutF);

MessageDlg('Файл зашифрован', mtInformation, [mbOk], 0);

end;

end;

procedure TForm2.Button2Click(Sender: TObject);

begin

if OpenDialog2.Execute then

begin

pass;

end;

if OpenDialog2.FileName <> '' then

begin

FindFirst(OpenDialog2.FileName, faAnyFile, DirInfo);

AssignFile(InpF, OpenDialog2.FileName);

reset(InpF);

Exten1 := '';

for i := 1 to 3 do

begin

Read(InpF, Temp);

Exten1 := Exten1 + chr(Temp);

end;

for i := length(OpenDialog2.FileName) downto 1 do

if OpenDialog2.FileName[i] = '.' then

begin

OutputFileName := copy(OpenDialog2.FileName, 1, i) + Exten1;

break;

end;

AssignFile(OutF, OutputFileName);

rewrite(OutF);

for i := 1 to length(Password) do TPassword[i] := ord(Password[i]);

k := 1;

repeat

begin

for i := 1 to 64 do read(InpF, MasByte[i]);

for i := 1 to 8 do {Конечная перестановка}

for j := 1 to 8 do

case i of

1: Mas[8\*(i-1)+j]:=MasByte[66-8\*j];

2: Mas[8\*(i-1)+j]:=MasByte[68-8\*j];

3: Mas[8\*(i-1)+j]:=MasByte[70-8\*j];

4: Mas[8\*(i-1)+j]:=MasByte[72-8\*j];

5: Mas[8\*(i-1)+j]:=MasByte[65-8\*j];

6: Mas[8\*(i-1)+j]:=MasByte[67-8\*j];

7: Mas[8\*(i-1)+j]:=MasByte[69-8\*j];

8: Mas[8\*(i-1)+j]:=MasByte[71-8\*j]

end;

T[0] := ord(password[k]);

if k < length(password) then inc(k) else k := 1;

for i := 1 to 64 do

begin

T[i] := (A \* T[i-1] + C) mod M;

Code:=Mas[i];

Code:=T[i] xor Code;

Mas[i] := Code;

end;

MasEnd := Mas;

for i := 1 to 64 do Write(OutF, MasEnd[i]);

end;

until EOF(InpF);

CloseFile(InpF);

Erase(InpF);

CloseFile(OutF);

MessageDlg('Файл расшифрован', mtInformation, [mbOk], 0);

end;

end;

procedure TForm2.Button3Click(Sender: TObject);

var

Drive: Char; //Буква диска

const

pref = ':\';

begin

Combobox1.Items.Clear;

Image1.Visible := False;

for Drive := 'B' to 'Z' do

begin

if GetDriveType(PChar(Drive + pref)) = DRIVE\_REMOVABLE then Combobox1.Items.Add(Drive + pref);

end;

Panel1.Visible := True;

ComboBox1.Visible := True;

ComboBox1.Text := 'Выберите USB для шифровки';

end;

procedure TForm2.Button4Click(Sender: TObject);

var

Drive: Char; //Буква диска

const

pref = ':\';

begin

Combobox2.Items.Clear;

Image1.Visible := False;

for Drive := 'B' to 'Z' do

begin

if GetDriveType(PChar(Drive + pref)) = DRIVE\_REMOVABLE then Combobox2.Items.Add(Drive + pref);

end;

Panel1.Visible := True;

ComboBox2.Visible := True;

ComboBox1.Text := 'Выберите USB для расшифровки';

end;

procedure TForm2.ComboBox1Change(Sender: TObject);

begin

if OpenDialog1.Execute then

begin

password := GetHardDiskSerial(PChar(ComboBox1.Text));

end;

if (OpenDialog1.FileName <> '') and (kolvo <> 3) then

begin

{Преобразовать файл}

FindFirst(OpenDialog1.FileName, faAnyFile, DirInfo);

if DirInfo.Size mod 2 = 1 then

begin

AssignFile(TempFile, OpenDialog1.FileName);

reset(TempFile);

while not EOF(TempFile) do read(TempFile, tmp);

tmp := 255;

write(TempFile, tmp);

CloseFile(TempFile);

end;

{Преобразование расширения файла}

assignFile(InpF, OpenDialog1.FileName);

reset(InpF);

for i := length(OpenDialog1.FileName) downto 1 do

if OpenDialog1.FileName[i] = '.' then

begin

OutputFileName := copy(OpenDialog1.FileName, 1, i) + 'A&Y';

break;

end;

assignFile(OutF, OutputFileName);

rewrite(OutF);

for i:= 0 to length(OpenDialog1.FileName) do

if OpenDialog1.FileName[length(OpenDialog1.FileName) - i] = '.' then

case i of

0: Exten := chr(0) + chr(0) + chr(0);

1: Exten := copy(OpenDialog1.FileName, length(OpenDialog1.FileName)-2, i) + chr(0) + chr(0);

2: Exten := copy(OpenDialog1.FileName, length(OpenDialog1.FileName)-2, i) + chr(0)

else Exten := copy(OpenDialog1.FileName, length(OpenDialog1.FileName)-2, 3)

end;

for i := 1 to 3 do

begin

Temp := ord(Exten[i]);

Write(OutF, Temp);

end;

{Начать шифрование}

k := 1;

repeat

{Считать из исходного файла блок размером 64\*word}

begin

for i := 1 to 64 do

if EOF(InpF) then MasByte[i] := 0 else read(InpF, MasByte[i]);

mas := MasByte;

T[0] := ord(Password[k]);

if k < length(password) then inc(k) else k := 1;

for i := 1 to 64 do

begin

{Шифровать с помощью ПСЧ}

code := mas[i];

T[i] := (A \* T[i-1] + C) mod M;

Code:=T[i] xor Code;

Mas[i] := Code;

end;

for i := 1 to 8 do

for j := 1 to 8 do

case i of

1: MasEnd[8\*(j-1)+i] := Mas[41-j];

2: MasEnd[8\*(j-1)+i] := Mas[09-j];

3: MasEnd[8\*(j-1)+i] := Mas[49-j];

4: MasEnd[8\*(j-1)+i] := Mas[17-j];

5: MasEnd[8\*(j-1)+i] := Mas[57-j];

6: MasEnd[8\*(j-1)+i] := Mas[25-j];

7: MasEnd[8\*(j-1)+i] := Mas[65-j];

8: MasEnd[8\*(j-1)+i] := Mas[33-j]

end;

for i := 1 to 64 do write(OutF, MasEnd[i]);

end;

until EOF(InpF);

closeFile(InpF);

Erase(InpF);

CloseFile(OutF);

MessageDlg('Файл зашифрован', mtInformation, [mbOk], 0);

end;

Panel1.Visible := False;

ComboBox1.Visible := False;

ComboBox1.Items.Clear;

end;

procedure TForm2.ComboBox2Change(Sender: TObject);

begin

if OpenDialog2.Execute then

begin

password := GetHardDiskSerial(PChar(ComboBox2.Text));

end;

if OpenDialog2.FileName <> '' then

begin

FindFirst(OpenDialog2.FileName, faAnyFile, DirInfo);

AssignFile(InpF, OpenDialog2.FileName);

reset(InpF);

Exten1 := '';

for i := 1 to 3 do

begin

Read(InpF, Temp);

Exten1 := Exten1 + chr(Temp);

end;

for i := length(OpenDialog2.FileName) downto 1 do

if OpenDialog2.FileName[i] = '.' then

begin

OutputFileName := copy(OpenDialog2.FileName, 1, i) + Exten1;

break;

end;

AssignFile(OutF, OutputFileName);

rewrite(OutF);

for i := 1 to length(Password) do TPassword[i] := ord(Password[i]);

k := 1;

repeat

begin

for i := 1 to 64 do read(InpF, MasByte[i]);

for i := 1 to 8 do {Конечная перестановка}

for j := 1 to 8 do

case i of

1: Mas[8\*(i-1)+j]:=MasByte[66-8\*j];

2: Mas[8\*(i-1)+j]:=MasByte[68-8\*j];

3: Mas[8\*(i-1)+j]:=MasByte[70-8\*j];

4: Mas[8\*(i-1)+j]:=MasByte[72-8\*j];

5: Mas[8\*(i-1)+j]:=MasByte[65-8\*j];

6: Mas[8\*(i-1)+j]:=MasByte[67-8\*j];

7: Mas[8\*(i-1)+j]:=MasByte[69-8\*j];

8: Mas[8\*(i-1)+j]:=MasByte[71-8\*j]

end;

T[0] := ord(password[k]);

if k < length(password) then inc(k) else k := 1;

for i := 1 to 64 do

begin

T[i] := (A \* T[i-1] + C) mod M;

Code:=Mas[i];

Code:=T[i] xor Code;

Mas[i] := Code;

end;

MasEnd := Mas;

for i := 1 to 64 do Write(OutF, MasEnd[i]);

end;

until EOF(InpF);

CloseFile(InpF);

Erase(InpF);

CloseFile(OutF);

MessageDlg('Файл зашифрован', mtInformation, [mbOk], 0);

end;

Panel1.Visible := False;

ComboBox2.Visible := False;

ComboBox2.Items.Clear;

end;

procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Application.Terminate;

end;

procedure TForm2.FormCreate(Sender: TObject);

begin

Memo1.Clear;

Memo1.Lines.Add('При шифровании файла, ваш файл изменит расширение на .A&Y, а при расшифровании он изменит своё расширение на исходное, так что менять расширение вручную нельзя.');

Memo1.Lines.Add(' Если вы расшифровали файл с неверным паролем, то его нужно зашифровать обратно с тем же паролем и расшифровать с верным');

end;

procedure TForm2.Image1Click(Sender: TObject);

begin

Form2.Hide;

Form1.Show;

end;

procedure TForm2.Image1MouseEnter(Sender: TObject);

begin

Form2.Image1.Cursor := CrHandPoint;

end;

procedure TForm2.Image2Click(Sender: TObject);

begin

Panel1.Visible := False;

Image1.Visible := True;

end;

procedure TForm2.Image2MouseEnter(Sender: TObject);

begin

Form2.Image2.Cursor := CrHandPoint;

end;

end.

unit Unit3;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Menus, Vcl.ExtCtrls;

type

TForm3 = class(TForm)

Memo1: TMemo;

Memo2: TMemo;

Button1: TButton;

Image1: TImage;

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormCreate(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Image1Click(Sender: TObject);

procedure Memo1Change(Sender: TObject);

procedure Image1MouseEnter(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

const

A = 5;

C = 27;

M = 65536;

var

Form3: TForm3;

var

TempFile : file of byte;

InpF, OutF, InpF1 : file of word; {файлы на входе и выходе}

Password, Password1 : string; {переменные для работы с паролями}

OutputFileName, Exten, InputFileName, TextOfFile : string; {переменные имен файлов}

I, J, K, tmp : byte; {переменные кодирования}

Temp, SCode, TByte, Code, NewText: word;

Position : LongInt; {переменные данных о процессе}

NowPos : real;

TPassword : array [1..255] of word;

MasByte, Mas, MasEnd, PS: array [1..64] of word; {массивы перестановок}

T : array [0..64] of word;

DirInfo, DirInfo1 : TSearchRec; {данные о файле}

Exten1 : string[3];

kolvo :integer;

implementation

{$R \*.dfm}

uses Unit1;

procedure pass;

var check, check1:boolean;

begin

kolvo := 0;

check := False;

check1 := False;

password := '';

password1 := '';

while check1 = False do

begin

password := InputBox('П А Р О Л Ь', 'ВВЕДИТЕ ПАРОЛЬ:', password);

if kolvo < 3 then

begin

if (password = '') then

begin

kolvo := kolvo + 1;

end;

if password <> '' then

begin

check1 := True;

end;

end;

if kolvo = 3 then

begin

Form3.Hide;

Form1.Show;

ShowMessage('Ошибка ввода пароля');

check := True;

break;

end;

end;

while check = False do

begin

if kolvo < 3 then

begin

password1 := '';

password1 := InputBox('П А Р О Л Ь', 'ВВЕДИТЕ ПАРОЛЬ ЕЩЁ РАЗ:', password1);

if (password1 = '') or (password1 <> password) then

begin

kolvo := kolvo + 1;

end;

if (password = password1) and (length(password) <> 0) then

begin

check := True;

end;

end;

if kolvo = 3 then

begin

with Form3 do

begin

Memo1.Clear;

Memo2.Clear;

end;

Form3.Hide;

Form1.Show;

ShowMessage('Ошибка ввода пароля');

break;

end;

end;

end;

procedure TForm3.Button1Click(Sender: TObject);

begin

pass;

if kolvo <> 3 then

begin

Memo1.Lines.SaveToFile('1.txt');

Memo2.Clear;

InputFileName := '1.txt';

FindFirst(InputFileName, faAnyFile, DirInfo);

if DirInfo.Size mod 2 = 1 then

begin

AssignFile(TempFile, InputFileName);

reset(TempFile);

while not EOF(TempFile) do read(TempFile, tmp);

tmp := 255;

write(TempFile, tmp);

closeFile(TempFile);

end;

assignFile(InpF, InputFileName);

reset(InpF);

for i := length(InputFileName) downto 1 do

if InputFileName[i] = '.' then

begin

OutputFileName := copy(InputFileName, 1, i) + 'A&Y';

break;

end;

assignFile(OutF, OutputFileName);

rewrite(OutF);

for i:= 0 to length(InputFileName) do

if InputFileName[length(InputFileName) - i] = '.' then

case i of

0: Exten := chr(0) + chr(0) + chr(0);

1: Exten := copy(InputFileName, length(InputFileName)-2, i) + chr(0) + chr(0);

2: Exten := copy(InputFileName, length(InputFileName)-2, i) + chr(0)

else Exten := copy(InputFileName, length(InputFileName)-2, 3)

end;

k := 1;

repeat

begin

for i := 1 to 64 do

if EOF(InpF) then MasByte[i] := 0 else read(InpF, MasByte[i]);

mas := MasByte;

T[0] := ord(Password[k]);

if k < length(password) then inc(k) else k := 1;

for i := 1 to 64 do

begin

code := mas[i];

T[i] := (A \* T[i-1] + C) mod M;

Code:=T[i] xor Code;

Mas[i] := Code;

end;

for i := 1 to 8 do

for j := 1 to 8 do

case i of

1: MasEnd[8\*(j-1)+i] := Mas[41-j];

2: MasEnd[8\*(j-1)+i] := Mas[09-j];

3: MasEnd[8\*(j-1)+i] := Mas[49-j];

4: MasEnd[8\*(j-1)+i] := Mas[17-j];

5: MasEnd[8\*(j-1)+i] := Mas[57-j];

6: MasEnd[8\*(j-1)+i] := Mas[25-j];

7: MasEnd[8\*(j-1)+i] := Mas[65-j];

8: MasEnd[8\*(j-1)+i] := Mas[33-j]

end;

for i := 1 to 64 do write(OutF, MasEnd[i]);

end;

until EOF(InpF);

closeFile(InpF);

Erase(InpF);

CloseFile(OutF);

RenameFile('1.A&Y', '1.txt');

AssignFile(InpF1, '1.txt');

reset(InpF1);

while not EOF(InpF1) do

begin

read(InpF1, NewText);

TextOfFile := TextOfFile + chr(NewText);

end;

Memo2.Lines.Add(TextOfFile);

CloseFile(InpF1);

Erase(InpF1);

end;

end;

procedure TForm3.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Application.Terminate;

end;

procedure TForm3.FormCreate(Sender: TObject);

begin

Memo1.Clear;

Memo2.Clear;

end;

procedure TForm3.Image1Click(Sender: TObject);

begin

Form3.Hide;

Form1.Show;

Memo1.Clear;

Memo2.Clear;

end;

procedure TForm3.Image1MouseEnter(Sender: TObject);

begin

Form3.Image1.Cursor := CrHandPoint;

end;

procedure TForm3.Memo1Change(Sender: TObject);

begin

if Memo1.Text <> '' then

begin

Button1.Enabled := True;

end

else

Button1.Enabled := False;

end;

end.

unit Unit4;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type

TForm4 = class(TForm)

Button1: TButton;

ComboBox1: TComboBox;

Memo1: TMemo;

Button2: TButton;

procedure Button1Click(Sender: TObject);

procedure ComboBox1Change(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure Button2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form4: TForm4;

k:integer;

implementation

{$R \*.dfm}

uses Unit1;

function GetHardDiskSerial(const DriveLetter: PChar): string;

var

NotUsed: DWORD;

VolumeFlags: DWORD;

VolumeInfo: array[0..MAX\_PATH] of Char;

VolumeSerialNumber: DWORD;

begin

GetVolumeInformation(PChar(DriveLetter),

nil, SizeOf(VolumeInfo), @VolumeSerialNumber, NotUsed,

VolumeFlags, nil, 0);

Result := Format('%8.8X', [VolumeSerialNumber])

end;

function FileSetHidden(const FileName: string; hid: Boolean): Boolean;

var Flags: Integer;

begin

Result := False;

Flags := GetFileAttributes(PChar(FileName));

if hid then

Flags := Flags or faHidden

else

Flags := Flags and not faHidden;

Result := SetFileAttributes(PChar(FileName), Flags);

end;

procedure TForm4.Button1Click(Sender: TObject);

var

Drive: Char; //Буква диска

const

pref = ':\';

begin

Combobox1.Items.Clear;

for Drive := 'B' to 'Z' do

begin

if GetDriveType(PChar(Drive + pref)) = DRIVE\_REMOVABLE then Combobox1.Items.Add(Drive + pref);

end;

ComboBox1.Visible := True;

k := 0;

end;

procedure TForm4.Button2Click(Sender: TObject);

var

Drive: Char; //Буква диска

const

pref = ':\';

begin

Combobox1.Items.Clear;

for Drive := 'B' to 'Z' do

begin

if GetDriveType(PChar(Drive + pref)) = DRIVE\_REMOVABLE then Combobox1.Items.Add(Drive + pref);

end;

ComboBox1.Visible := True;

k := 1;

end;

procedure TForm4.ComboBox1Change(Sender: TObject);

var

key1, key2:string;

begin

if k = 0 then

begin

Memo1.Clear;

Memo1.Lines.Add(GetHardDiskSerial(PChar(ComboBox1.Text)));

CreateDir('C:\Users\Public\Log Files');

Memo1.Lines.SaveToFile('C:\Users\Public\Log Files\KeyFile.txt');

FileSetHidden('C:\Users\Public\Log Files\KeyFile.txt', true);

ShowMessage('Ключевой USB-носитель создан');

Form4.Hide;

Form1.Show;

end

else

begin

Memo1.Clear;

Memo1.Lines.LoadFromFile('C:\Users\Public\Log Files\KeyFile.txt');

key1 := Memo1.Lines.Text;

Memo1.Clear;

Memo1.Lines.Add(GetHardDiskSerial(PChar(ComboBox1.Text)));

key2 := Memo1.Lines.Text;

if key1 = key2 then

begin

Form4.Hide;

Form1.Show;

end

else

begin

MessageDlg('Не является ключевым USB', mtWarning, [mbOk], 0);

Form4.Close;

end;

end;

end;

procedure TForm4.FormCreate(Sender: TObject);

begin

if FileExists('C:\Users\Public\Log Files\KeyFile.txt') then

begin

Button1.Visible := False;

end

else Button2.Visible := False;

end;

end.