Введение

На курсовое проектирование была поставлена задача разработать программный продукт на тему «Шифр «Вернама».

Цель курсового проекта заключается в реализации шифрования файлов и их сокрытия под видом изображений в формате png.

Создаваемая программа будет рассчитана на любого рода пользователей. Она будет особенно полезна людям, увлекающимся шифрованием или желающим передать своё сообщение в «нечитаемом» виде.

Далее приведем краткое описание разделов пояснительной записки.

Первый раздел носит название “Анализ задачи”. В нем вы сможете ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, определение ее организационно-экономической сущности. Также в этом разделе вы сможете узнать о том, как данная задача решается в настоящее время. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе “Инструменты разработки” будет рассмотрена среда, в которой создается данный курсовой проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачи.

В разделе «Проектирование задачи» будут рассмотрены основные аспекты разработки программного продукта. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет четко описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации, описана разработка системы справочной информации.

«Реализация задачи» – это третий раздел пояснительной записки, в котором описываются все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе будут четко описаны функции пользователя и их структура. Здесь можно будет найти таблицу, в которой будет представлена полная аннотация файлов используемых в данном проекте.

Четвертый раздел – «Тестирование». В нем будет описано полное и функциональное тестирование данной программы, т.е. будет оттестирован каждый пункт меню, каждая операция, которая выполняется приложением. Будут смоделированы все возможные действия пользователя при работе с программой, начиная от запуска до выхода.

В разделе «Применение» будет описано назначение, область применения, среда функционирования курсовой программы. Также в нем будет описано использование справочной системы.

«Заключение» будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В приложении к пояснительной записке будет приведен листинг программы с необходимыми комментариями.

Схема работы системы будет представлена в графической части.

1 Анализ задачи

1.1 Постановка задачи

Темой данного курсового проекта является «Разработка программного продукта «Шифр Вернама».

Шифрование играет большую роль в нашем современном мире. Ранее оно использовалось для обменом сообщениями на войне, между лидерами мнений и правительством, а так же преступностью и крайне редко простыми людьми. Сейчас же данные почти каждого человека в той или иной мере зашифрованы и нуждаются в защите. В эпоху Интернета и глобальной информатизации и цифровизации свои данные нужно хранить в секрете и защищать, чтобы они не попали к злоумышленникам.

С древности шифрование шло в ногу с дешифрованием. С времён Цезаря и названного в его честь шифра прошли сотни лет, были разработаны и взломаны десятки шифров, даже такие сложные, как «Энигма» и «Виженера», которые в своё время считались не взламываемыми.

В моём проекте вы увидите по-настоящему несокрушимый шифр. И название ему шифр «Вернама». Он заключается в том, что для каждой единицы информации используется свой «ключ». Из-за такой особенности шифра, где для каждой единицы информации(бит) нужна единица ключа (бит для ключа) данный вид шифрования не имеет широкого распространения, так как возникает проблема с передачей большого ключа, а также его хранением. Если ключ будет повторяться или окажется не криптостойким, то шифр больше не является абсолютно не взламываемым.

Подвидом шифра «Вернама» является шифр одноразового шифроблокнота. Это тот же шифр, однако ключ в нём не дублируется и является абсолютно случайно сгенерированным. После одного шифрования при помощи ключа он уничтожается. Данная особенность реализована.

В моём проекте данные проблемы решаются благодаря кодированию ключа внутрь картинки в формате png, что является так же шифрованием, называемым «стеногрaфия».

Аналогов такого приложения не существует. Существуют различные онлайн реализации шифра «Вернама» на сайтах, офлайн приложений с такой возможностью, а также много статей на эту тему. Однако приложений для кодирования файлов внутрь картинок мне найти не удалось, а совместное использование обоих этих алгоритмов тем более.

Периодичность использования данного программного продукта не ограничена. Пользователь может в любое время установить, а также удалить приложение.

Цель данного курсового проекта – разработать программный продукт, который покажет возможности шифра «Вернама», а также позволит зашифровать и дешифровать информацию пользователя.

В приложении должен быть простой и понятный интерфейс, который будет интуитивно понятен любому пользователю, который ранее не имел большого опыта с шифрованием файлов.

Разрабатываемый программный продукт должен позволять выполнять следующие действия:

- шифрование информации;
- дешифрование информации;
- создание ключа, криптостойкого ключа;
- просмотр справки;
- просмотр пункта «о разработчике»;
- выбор файлов дли шифрования и дешифрования;
- выбор ключа.

В разрабатываемой программе будут использоваться три вида данных.

К входной информации можно отнести вводимые пользователем информацию, например кастомный ключ, файл для кодирования и декодирования информации.

К выходной – результат кодирования файла в картинку, а именно обновлённая картинка, введённая ранее для этой нужды, а также результат декодирования в виде файла, который был ранее зашифрован.

Постоянной информацией в проекте будут являться текстовые файлы, картинки, медиафайлы и др.

Программный продукт предоставляет функционал для следующего ряда пользователей: создатель ключа, простой пользователь.

1.2 Инструменты разработки

Для разработки данного проекта будет выбрана среда Delphi 12, так как это самое удобная и доступная среда разработки на данный момент. Delphi 12 - среда разработки, относящийся к классу RAD- (Rapid Application Development – «Средство быстрой разработки приложений») средств CASE – технологии. Delphi 12 сделал разработку приложений для Windows быстрым и приятным процессом.

Теперь разрабатывать сложные и интересные проекты можно только одним человеком, использующим Delphi 12.

Интерфейс Windows обеспечивает полное перенесение CASE-технологии в интегральную систему поддержки работ по созданию прикладной системы на всех фазах жизненного цикла работы и проектирования системы.

Delphi 12 обладает широким набором возможностей, начиная от проектировщика форм и заканчивая поддержкой всех форматов популярных баз данных. Среда устраняет необходимость программировать такие компоненты Windows общего назначения, как метки, программы и даже диалоговые панели. Работая в Windows, можно видеть одинаковые «объекты» во многих разнообразных приложениях. Диалоговые панели (например, Choose File и Save File) являются примерами многократно-используемых компонентов, встроенных непосредственно в Delphi 12, который позволяет приспособить эти компоненты к имеющийся задаче, чтобы они работали именно так, как требуется создаваемому приложению. Также здесь имеются предварительно- определенные визуальные и не визуальные объекты.

Три основные части разработки интерфейса следующие: проектирование панели, проектирование диалога и представление окон. Для общего пользовательского доступа также должны учитываться условия применения архитектуры прикладных систем.

Сегодня появилась реальная возможность с помощью моделирования на современных многофункциональных средствах обработки и отображения информации таких как Delphi 12 конкретизировать тип и характеристики используемых информационных моделей, выявить основные особенности будущей деятельности операторов, сформулировать требования к параметрам аппаратно-программных средств интерфейса взаимодействия и т.д.

Delphi 12 позволяет создать различные виды программ: консольные приложения, оконные приложения, приложения для работы с Интернетом и базами данных. То есть, Delphi 12 является не только средствами для работы с языком программирования Паскаль, но дополнительные инструменты, призванные для максимального упрощения и ускорения создание приложений.

К дополнительным инструментам можно отнести визуальный редактор форм, благодаря которому можно с легкостью создать полноценную программу, и другие визуальные составляющие разработки программного обеспечения. С Delphi вам не нужно вручную просчитывать расположение каждого элемента интерфейса пользователя, поэтому при разработке программы значительно экономится время.

Выгоды от проектирования в среде Windows с помощью Delphi 12:

- устраняется необходимость в повторном вводе данных;

- обеспечивается согласованность проекта и его реализации;

- увеличивается производительность разработки и переносимость программ.

Ни одно серьезное программное обеспечение не обходится без модуля справочной информации и руководства пользователя. Это придает программе законченный вид и показывает заботу о пользователе.

Help + Manual 9– легкий в использовании и функциональный инструмент, упрощающий создание справочных файлов Windows, печать справочных руководств и документации в целом. Программа имеет интуитивно понятный интерфейс. Все созданные проекты можно сохранить в различных форматах:

Для создания инсталлятора будет использоваться мощное и удобное средство - Smart Install Maker.

Программа обладает удобным и интуитивно понятным интерфейсом, а также полным набором необходимых функций для создания профессиональных инсталляторов с минимальным размером, высокой степенью сжатия файлов и приятным интерфейсом.

Помимо стандартного минимума, Smart Install Maker позволяет редактировать системный реестр и INI-файлы, создавать программные ярлыки, запускать ассоциируемые и исполняемые файлы, регистрировать новые шрифты и ActiveX компоненты, отображать тексты информации и лицензионного соглашения. Также, с помощью этой утилиты, можно создать мультиязыковые инсталляторы с поддержкой более 20-ти популярных языков мира.

Microsoft Word 2022 – редактор текста для написания документации.

Разработка ведется на ноутбуке Lenovo Legion. У данного ноутбука следующие параметры:

- процессор Intel Core i5;
- объем ОЗУ 16 гб;
- объем места на HDD – 1 тб ;
- видеоподсистема 1920x1080 точек с глубиной цвета 32 Bit;
- ОС – Windows 10.

Как видно разрабатываемое приложение не очень требовательно к аппаратным ресурсам, что, является большим плюсом.

1.3 Требования к приложению

На этапе исследования предметной области был установлен целый ряд требований, которые предъявляются к разрабатываемой программе.

При моделировании форм следует учесть такой момент: приложение нацелено на кодирование и декодирование информации, из чего следует, что оно не нуждается в экстраординарном интерфейсе.

Требования к интерфейсу: в связи с частым использование программы она должна быть с приятной цветовой гаммой и понятной для пользователя. Следовательно, каждое окно должно иметь ясную визуальную иерархию своих элементов. Фрагменты текста должны располагаться на экране так, чтобы пользователя было просто и понятно принимать информацию.

Пользователь не должен испытывать какого-либо дискомфорта в плане восприятия информация, отображённой на экране. Объекты (рисунки и символы) не должны быть слишком мелкие. Все окна приложения по возможности должны помещаться на экран полностью, так как использование в процессе работы полос прокруток достаточно неудобно.

На одной форме нельзя допускать избытка и нагромождения данных.

Формы должны быть эффектно оформлены согласно тематике разрабатываемого проекта.

Требования к надежности: специальных требований к надежности не предъявляется.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбое (не крахом) операционной системы, не должно превышать 30-ти минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы программы вследствие некорректных действий пользователя при взаимодействии с программой через графический интерфейс не должны влиять на конечный результат. ПП должен иметь средства фиксации всех действий в процессе изготовления программного продукта. Это необходимо для восстановления состояния процесса разработки на любом его этапе (при комплексной отладке).

2 Проектирование задачи

2.1 Организация данных

Проектирование задачи – это очень важный и ответственный этап в разработке любого приложения.

Важным является он вследствие того, что методы, по средствам которых пользователь управляет формами, построены на высокой степени специализации каждого из компонентов.

Необходимым условием при разработке данного приложения является описание организации данных, т.е. логическая и физическая структура данных в контексте среды разработки. В разрабатываемой программе будут использоваться три вида данных.

Первым видом являются данные, которые будут введены разработчиком на этапе реализации задачи. Сюда можно отнести изображения (иконки, кнопки), описание.

Вторым видом данных, используемых в программе, является вводимая пользователем информация. К входной информации можно отнести вводимые пользователем значения, например пути к файлам.

Третьим видом данных является результат программы – например, сгенерированный ключ.

Таким образом, организация данных является важной задачей при разработке данной и любой программы.

2.2 Процессы

Согласно всем перечисленным требованиям и указаниям, которые были рассмотрены в разделе «Анализ задачи», было определено, чем конкретно должна заниматься разрабатываемая курсовая программа. Главной ее задачей будет являться шифрование и дешифрование файлов с использованием шифра «Вернама».

Программа будет создавать ключ, потом на основе этого ключа кодировать файл, после чего записывать закодированный файл в заранее выбранное изображение. Так же будет реализовано декодирование. Имея нужный ключ, пользователь сможет декодировать файл из изображения.

2.3 Описание внешнего пользовательского интерфейса

Важным при выполнении курсового проекта является организация диалога между пользователем и самой программой. Во многом это зависит от того, как программист разработает данную программу, какие компоненты будут использованы и какие методы будут автоматизированы.

Особое внимание следует уделить интерфейсу. Разработчик должен так организовать внешний вид своей программы, что бы пользователь понял, что от него требуется.

Для организации эффектной работы пользователя нужно создать целостное приложение данной предметной области, в которой все компоненты приложения будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс пользователя.

Таким образом, для успешной работы всего проекта в целом следует обеспечить интуитивно понятный интерфейс с приятной гаммой цветов и шрифтами.

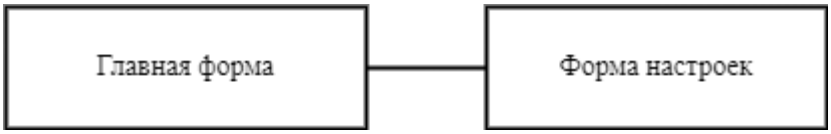

Рисунок 1 – Структура навигации по проекту

3 Реализация

3.1 Реализация проекта

3.1.1 Структура программы

Данный курсовой проект содержит 7 модулей. Далее рассмотрим назначение каждого модуля:

1 Модуль Conf - отвечает за форму настроек и сохранение результатов настроек;

2 Модуль Cypher – отвечает за шифрование и дешифрование, а так же вспомогательные функции для шифрования;

3 Модуль Loading – отвечает за заставку формы, после которой запускается главная форма (Main);

4 Модуль Main – отвечает за главную форму программы, где производится шифрование;

5 Модуль Picture – модуль для описания класса;

6 Модуль Rand – модуль, отвечающий за генерацию чисел;

7 Модуль Saves – модуль, отвыкающий за сохранение и загрузку настроек.

3.1.2 Структура и описание процедур и функций пользователя

Таблица 1 – Процедуры и функции

Имя процедуры (функции)	В каком модуле находится	За каким компонентом закреплена	Назначение
1	2	3	4
1 procedure TForm2.Button1Click(Sender: TObject);	Conf.pas	Button1	сохранение настроек и закрытие формы по нажатию кнопки
2 procedure TForm2.Save();	Conf.pas	TForm2	сохранение настроек в файл
3 procedure TForm2.SpeedButton1Click(Sender: TObject);	Conf.pas	SpeedButton1	выбор картинки для ключа по нажатию кнопки
4 procedure TForm2.SpeedButton2Click(Sender: TObject);	Conf.pas	SpeedButton2	выбор картинки для хранения закодированного файла по кнопке
5 procedure TForm2.SpeedButton3Click(Sender: TObject);	Conf.pas	SpeedButton3	выбор файла для кодирования по кнопке
6 procedure TForm2.FormActivate(Sender: TObject);	Conf.pas	TForm2	загрузка всех значений настроек из файла на форму

Продолжение таблицы 1

1	2	3	4
7 procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);	Conf.pas	TForm2	сохранение настроек при закрытии
8 procedure LoadKeyPng();	Cypher.pas	-	загрузка картинки-ключа в переменную
9 procedure LoadFilePng();	Cypher.pas	-	загрузка картинки-источника в переменную
10 procedure SaveNameToPic(const name:string);	Cypher.pas	-	сохранение названия файла в картинку-источник
11 function GetNameFromPic():string;	Cypher.pas	-	получение названия файла из картинки-источника
12 function GetSizeFromPic():int64;	Cypher.pas	-	получение размера файла из картинки-источника
13 procedure SaveSizeToPic(len:Int64);	Cypher.pas	-	сохранение размера файла в картинку-источник
14 procedure SavePosXToPic(num:Int64);	Cypher.pas	-	сохранение x координаты в картинку-ключ
15 function GetPosXFromPic():int64;	Cypher.pas	-	чтение x координаты из картинки-ключа
16 procedure SavePosYToPic(num:Int64);	Cypher.pas	-	сохранение y координаты в картинку-ключ
17 function GetPosYFromPic():int64;	Cypher.pas	-	чтение y координаты из картинки-ключа
18 procedure SetPosXYToDefault();	Cypher.pas	-	установка значений по умолчанию для позиций x и y
19 function EncryptDecryptByte(source:TColor; key: byte):TColor;	Cypher.pas	-	закодировать байт в цвет
20 function HigherShapeFilter(c:byte):byte;	Cypher.pas	-	функция для корректной записи в цвет
21 function GetEncryptedShape(c:byte; en:char):byte;	Cypher.pas	-	кодирование одной 1/3 через 1/3 байта ключа
22 function DecryptDecryptByte(source:TColor):byte;	Cypher.pas	-	декодирование зашифрованного байта из цвета
23 procedure EncryptPicture();	Cypher.pas	-	кодирование картинки-ключа

Продолжение таблицы 1

1	2	3	4
24 procedure writebytestopic(pic:PNG; bytes:TArray<Byte>);	Cypher.pas	-	запись байтов в картинку
25 procedure EncryptFile();	Cypher.pas	-	шифрование файла
26 procedure DecryptFile();	Cypher.pas	-	декодирование файла
27 function GetFileBytes():TArray<Byte>;	Cypher.pas	-	получение байтов файла
28 function GetDecryptedBytes(count:integer):TArray<Byte>;	Cypher.pas	-	получение байтов ключа
29 procedure TForm3.Timer1Timer(Sender: TObject);	Loading.pas	Timer1	активация формы при конце загрузки
30 procedure TForm3.FormCreate(Sender: TObject);	Loading.pas	TForm3	установка значений по умолчанию для загрузки и запуск таймера
31 procedure Tform1.UpdateStateString;	Main.pas	-	обновить надпись с описанием текущего состояния приложения(настроек)
32 procedure TForm1.Close1Click(Sender: TObject);	Main.pas	TMainMenu	закрытие по нажатию пункта меню
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);	Main.pas	TForm1	полное закрытие приложения по нажатию кнопки
33 procedure TForm1.Localize();	Main.pas	-	отвечает за локализацию приложения
34 procedure Tform1.LoadPic();	Main.pas	-	загрузка картинки, которая показывает какой процесс настроен
35 procedure TForm1.FormCreate(Sender: TObject);	Main.pas	TForm1	первоначальная настройка формы
36 procedure TForm1.OpenForm1Click(Sender: TObject);	Main.pas	TMenuItem	открытие формы настроек
37 procedure TForm1.Projectreference1Click(Sender: TObject);	Main.pas	TMenuItem	открытие справки
38 procedure TForm1.Selectfilepath1Click(Sender: TObject);	Main.pas	TMenuItem	выбор пути к любому файлу

Продолжение таблицы 1

1	2	3	4
39 procedure TForm1.Selectkeypath1Click(Sender: TObject);	Main.pas	TMenuItem	выбор пути к картинке-ключу
40 procedure TForm1.Selectsourcepath1Click(Sender: TObject);	Main.pas	TMenuItem	выбор пути к картинке-источнику
41 procedure TForm1.SpeedButton1Click(Sender: TObject);	Main.pas	TForm1	реализация выбранного процесса по нажатию кнопки
42 procedure TForm1.SwitchCC1Click(Sender: TObject);	Main.pas	TMenuItem	смена пользователя
43 procedure TForm1.SwitchCG1Click(Sender: TObject);	Main.pas	TMenuItem	смена вида генерации байтов ключа
44 procedure TForm1.Switchsourcemode1Click(Sender: TObject);	Main.pas	TMenuItem	смена шифрования/дешифрования
45 procedure TForm1.Swithlanquage1Click(Sender: TObject);	Main.pas	TMenuItem	смена языка
46 constructor Create(path:string);	Picture.pas	Png	контруктор класса. Загружает картинку и устанавливает поле size
47 procedure LoadPicture(path:string);	Picture.pas	Png	загружает картинку в переменную
48 procedure SaveResult(filename:string);	Picture.pas	Png	сохраняет результат
49 procedure SetPixelColor(x, y:integer; color:TColor);	Picture.pas	Png	устанавливает цвет пикселю
50 function GetPixelColor(x, y:integer):TColor;	Picture.pas	Png	возвращает цвет пикселя
51 function GenerateRandomBytes(len: ulong):TArray<Byte>;	Rand.pas	-	генерация случайных байтов ключа
52 function ReadData():TArray<Byte>;	Rand.pas	-	чтение байтов, сгенерированных отдельным приложением
53 function ListToArray(list:Tlist):Tarray<Byte>;	Rand.pas	-	перевод списка в массив
54 procedure GenSettingsFileIfNotExists();	Saves.pas	-	создание настроек по умолчанию в файл настроек

Продолжение таблицы 1

1	2	3	4
55 procedure SaveLanguage(lang:string);	Saves.pas	-	сохранение языка в настройки, обновление локализации
56 procedure SavePictureKeyPath(path:string);	Saves.pas	-	сохранение картинки-ключа в настройки
57 procedure SavePictureSourcePath(path:string);	Saves.pas	-	сохранение картинки-источника в настройки
58 procedure SaveFilePath(path:string);	Saves.pas	-	сохранение пути для шифрования в настройки
59 procedure SaveUserMode(mode:string);	Saves.pas	-	сохранение текущего режима пользователя в настройки
60 procedure SaveGenMode(mode:string);	Saves.pas	-	сохранение текущего режима генерации в настройки
61 procedure SaveCryptMode(mode:string);	Saves.pas	-	сохранение текущего режима шифрования в настройки
62 procedure UpdateSettings(lang, um, gm, pm:string);	Saves.pas	-	обновление настроек
63 function GetLang():string;	Saves.pas	-	получение текущего сохранённого языка
64 function GetPictureKeyPath():string;	Saves.pas	-	получение текущего сохранённого пути к картинке-ключу
65 function GetPictureSourcePath():string;	Saves.pas	-	получение текущего сохранённого пути к картинке-источнику
66 function GetUserMode():string;	Saves.pas	-	получение текущего режима пользователя
67 function GetGenMode():string;	Saves.pas	-	получение текущего режима шифрования ключа
68 function GetCryptMode():string;	Saves.pas	-	получение текущего режима генерации байтов ключа
69 function GetFilePath():string;	Saves.pas	-	получение текущего сохранённого пути к файлу
70 function ReadSettingsFile():TStringlist;	Saves.pas	-	чтение всех настроек

Продолжение таблицы 1

1	2	3	4
71 procedure WriteSettingsInFile(list:TStringlist);	Saves.pas	-	запись всех настроек в файл
72 function GetPath():string;	Saves.pas	-	функция для получения пути
73 procedure Selectkeypath();	Saves.pas	-	сохранение пути к файлу
74 procedure Selectsourcepath();	Saves.pas	-	procedure Selectkeypath();
75 procedure SelectFilePath();	Saves.pas	-	Selectsourcepath();

3.1.3 Описание использованных компонентов

Таблица 2 – Использованные компоненты

Компонент	На какой форме расположен	Назначение
1 TMainMenu	Form1	Используется для создания главного меню проекта
2 TLabel	Form2, Form1	Используется для вывода надписей
3 TToggleSwitch	Form2	Используется для создания переключателей
4 TButton	Form2	Используется для создания кнопок
5 TSpeedButton	Form2, Form1	Используется для создания кнопок-изображений
6 TImage	Form3, Form1	Используется как фон
7 TTimer	Form3	Используется для определения длительности загрузочного экрана
8 TMenuItem	Form1	Используется для создания подпунктов меню

3.2 Спецификация программы

Таблица 3 – Спецификация программы

Имя файла	Назначение
1	2
1 HID.exe	Исполняемый файл проекта
2 Conf.dcu	Скомпилированный код модуля Conf
3 Cypher.dcu	Скомпилированный код модуля Cypher
4 Loading.dcu	Скомпилированный код модуля Loading
5 Main.dcu	Скомпилированный код модуля Main
6 Picture.dcu	Скомпилированный код модуля Picture
7 Rand.dcu	Скомпилированный код модуля Rand
8 Saves.dcu	Скомпилированный код модуля Saves

Продолжение таблицы 3

1	2
9 Rand.exe	Исполняемый файл вспомогательной программы
10 data.dat	Файл для сохранения сгенерированных байтов сгенерированных Rand.exe
11 settings.txt	Файл с настройками программы
12 key.png	Картинка, хранящая ключ
13 source.png	Картинка, хранящая зашифрованное сообщение
14 test.txt	Текстовый файл для тестирования работы программы
15 HID.rsm	Хранит настройки проекта
16 c_key.png	Картинки для отображения режима создания ключа
17 decode.png	Картинки для отображения режима дешифрования
18 encode.png	Картинки для отображения режима шифрования
19 Conf.dfm	Файл
20 Conf.pas	Файл программного модуля для формы Conf
21 Picture.pas	Файл программного модуля
22 Cypher.pas	Файл программного модуля
23 Main.pas	Файл программного модуля для формы Main
24 Main.dfm	Главная форма
25 Loading.dfm	Форма с заставкой
26 Loading.pas	Файл программного модуля для формы Loading
27 Rand.pas	Файл программного модуля
28 Saves.pas	Файл программного модуля
29 HID.res	Файл для хранения ресурсов проекта
30 HID.dpr	Файл проекта, связывает все файлы, из которых состоит приложение
31 HID.dproj	Файл, который служит для связи всего проекта
32 HID.identcache	Кэш файл для хранения информации об идентификационных данных файлов
33 HID.local	Файл с локальными данными проекта
34 HID.chm	Файл справки, содержит помощь по работе с программой
35 Setup.exe	Файл для установки приложения

4 Тестирование

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения испытания реализации программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Отчёт о результатах тестирования предоставлен в таблице 4.

Таблица 4 – Отчёт результатах тестирования

№	Тест	Ожидаемый результат	Физический результат	Результат тестирования
1	Проверка создания ключа	Корректно созданный случайный ключ	Корректно созданный случайный ключ	Выполнено
2	Проверка шифрования	Зашифрованный файл	Зашифрованный файл	Выполнено
3	Проверка дешифрования	Дешифрованный файл	Дешифрованный файл	Выполнено
4	Проверка сохранений настроек	Файл с сохранёнными настройками	Файл с сохранёнными настройками	Выполнено
5	Проверка корректного открытия, обновления и закрытия всех форм	Корректные работа приложения и отображаемая информация	Корректные работа приложения и отображаемая информация	Выполнено
6	Проверка изменений настроек и корректной работы приложения	Приложение работает корректно вне зависимости от настроек	Приложение работает корректно вне зависимости от настроек	Выполнено

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в процедурах.

5 Применение

5.1 Общие сведения о программном продукте

Цель данного программного продукта в обеспечении любительского шифрования файлов при помощи шифра Вернама.

Приложение рассчитано на различных типов пользователей, которые имеют разные интересы и цели. Данный шифр считается и является не взламываемым, если соблюдать определённые правила.

Быстродействие программы зависит от характеристик персонального компьютера, а именно, рабочей частоты процессора, объема оперативной памяти и т.д. Но стоит отметить, что данный программный продукт легко запускается и функционирует на относительно современных машинах.

5.2 Инсталляция

Для того, чтобы установить программу необходимо запустить файл Setup.exe. Появится окно установки приложения «HID».

Затем следует выполнять указанные инструкции установки приложения.

5.3 Выполнение программы

5.3.1 Запуск программы

Для запуска программы требуется дважды щелкнуть левой кнопки мыши на ярлыке или исполнительном файле с названием «HID.exe».

По подготовленным тестам будет осуществляться функциональное и полное тестирование программного продукта. Отчет о результатах тестирования будет представлен в 4 разделе пояснительной записки.

5.3.2 Инструкции по работе с программой

Первое, что видит пользователь при запуске программы – это главная форма. В ней находится панель инструментов, кнопка запуска работы программы, строка состояния и изображение, иллюстрирующее текущий режим работы приложения(рисунок 2)**.**

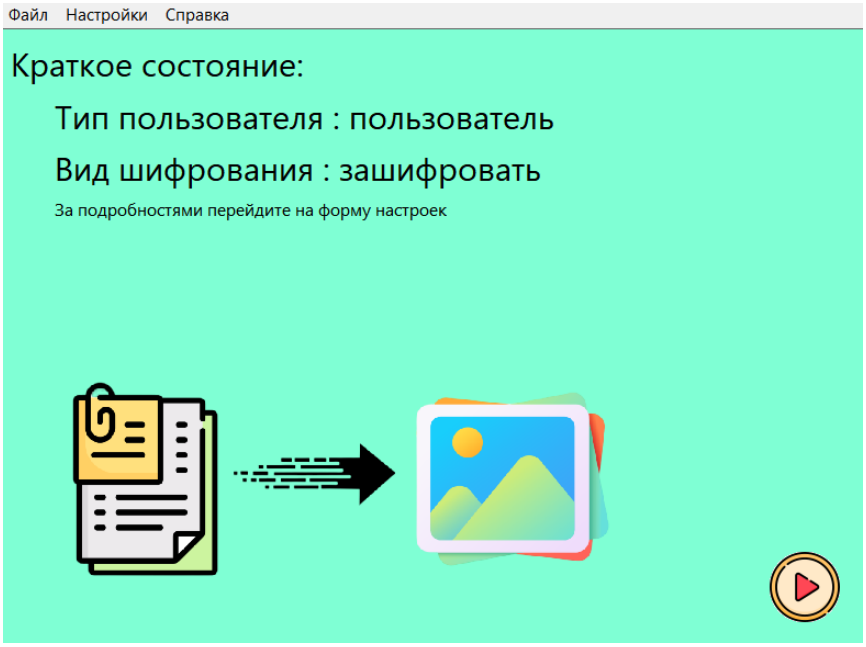

Рисунок 2 – Главная форма

Для того, чтобы изменить режим шифрования, пользователя, язык и прочие настройки, пользователь должен использовать панель инструментов, а именно вкладку «Настройки». Для более подробного просмотра и изменения настроек можно перейти на «Форму настроек». Для этого нужно нажать на «Настройки», а затем выбрать «Открыть форму настроек»(рисунок 3).

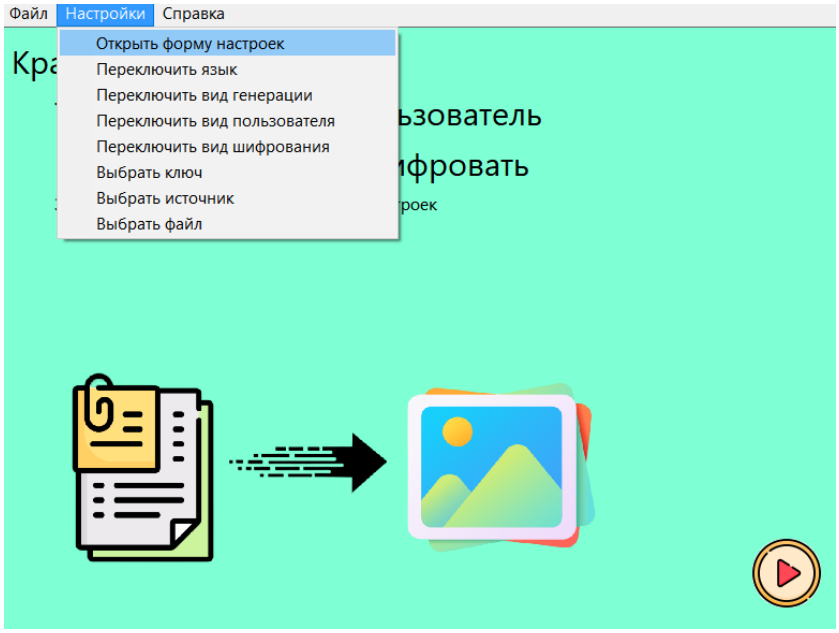

Рисунок 3 – Открытие формы настроек

На форме настроек пользователь сможет просмотреть, а так же обновить все существующие настройки программы. Для этого требуется нажимать на

переключатели, где бегунок справа обозначает включено/да/правый вариант, а бегунок слева выключено/нет/левый вариант(рисунок 4).

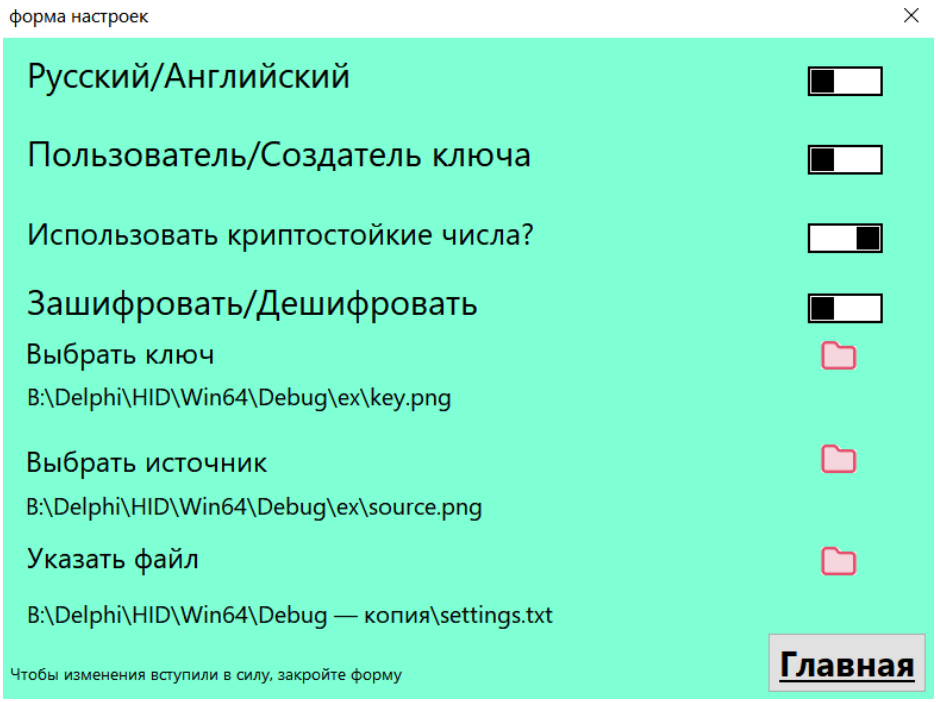

Рисунок 4 - Форма настроек

При нажатии на папку будет показан диалог с выбором файла, а после выбора данный путь будет сохранён в настройки как файл ключа/источника/исходного файла для шифрования.

Создания ключа, шифрование и дешифрование происходит путём изменения настроек пользователя и шифрования. Если выбрать пользователь, то будет реализовано шифрование или дешифрование в зависимости от соответствующей настройки. Если же выбран создатель ключа, то по нажатию по кнопке запуск будет реализовано создание ключа, вне зависимости от других настроек.

Важный момент работы программы. Чтобы другой человек мог декодировать ваше сообщение, вы должны передать ему ключ до шифрования сообщения, так как во время этого процесса ключ стирается, что обеспечивает дополнительную безопасность приложения, но делает декодирование невозможным при несоблюдении этого правила.

Так же стоит помнить, что ключ ограничен своим размером и он постепенно уменьшается, вы будете извещены сообщениями об ошибках в случае недостатка размера ключа.

5.3.3 Завершение работы с программой

Завершить программу можно несколькими способами. Первый способ заключается в нажатии крестика в правом верхнем углу главной формы. Второй способ – это нажатие кнопки «Закрыть» в главном меню во вкладке «Файл».

5.4 Использование системы справочной информации

Справочную систему можно запустить с помощью пункта меню Справка, которая находится в панели инструментов главной формы по пути «Справка» - «Справка проекта».

Заключение

В эпоху Интернета и глобальной информатизации и цифровизации остро встаёт проблема сохранения своих персональных данных. Это касается как каждого человека, так и крупных компании.

Разработанное познавательное приложение «Шифр Вернама» предоставит пользователям возможность скрыть, зашифровать свои данные от злоумышленников.

Задача была выполнена с помощью среды разработки Delphi 12. Для разработки программы использовались различные инструменты и средства, такие как Help + Manual 9 для создания справок, Word 2022 для написания документации, PowerPoint 2023 для создания отчётной презентации, Smart Install Maker для создания инсталлятора. Это позволило создать полноценное десктопное приложение.

Степень соответствия проектных решений заданию является высокой, так как были реализованы все основные функции приложения, такие как шифрование, дешифрование информации в виде файлов и создание ключа.

В ходе разработки программы были найдены следующие нетрадиционные способы решения задачи: использование картинки как средство хранения информации в цвете.

Программа может быть модифицирована для добавления новых функций и материалов, а также для адаптации к другим платформам и устройствам. Она может быть расширена для включения дополнительных функций, так как писалась с ориентиром на гибкость и комфортную разработку.

Список использованных источников

1. Шифр Вернама [Электронный ресурс] – Режим доступа: https://www.youtube.com/results?search_query=шифр+вернама – Дата доступа: 15.06.2024.

2. Конвертация типов данных в байты и обратно Delhi [Электронный ресурс] – Режим доступа https://www.youtube.com /results?search_query=Конвертация+типов+данных+в+байты+и+обратно+Delphi – Дата доступа: 18.06.2024.

3. Базовые компоненты Delphi [Электронный ресурс] – Режим доступа https://www.youtube.com/results?search_query=Базовые+компоненты+Delphi+ – Дата доступа: 18.06.2024.

Приложение А

Листинг

```
unit Main;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Cypher, Picture, Rand, Vcl.Imaging.pngimage, jpeg,
  Vcl.Buttons, Vcl.Menus, Vcl.WinXCtrls, Saves, Conf,
Loading, math, ShellAPI,
  Vcl.ExtCtrls;
type
  TForm1 = class(TForm)
    mm: TMainMenu;
    Settings1: TMenuItem;
    OpenForm1: TMenuItem;
    Swithlanquage1: TMenuItem;
    SwitchCG1: TMenuItem;
    SwitchCC1: TMenuItem;
    Selectkeypath1: TMenuItem;
    Selectsourcepath1: TMenuItem;
    Fi1: TMenuItem;
    Close1: TMenuItem;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Switchsourcemode1: TMenuItem;
    Reference1: TMenuItem;
    Projectreference1: TMenuItem;
    Label4: TLabel;
    Label5: TLabel;
    Image1: TImage;
    SpeedButton1: TSpeedButton;
    Selectfilepath1: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure OpenForm1Click(Sender: TObject);
    procedure SwitchCG1Click(Sender: TObject);
    procedure SwitchCC1Click(Sender: TObject);
    procedure Swithlanquage1Click(Sender: TObject);
    procedure Close1Click(Sender: TObject);
    procedure Selectkeypath1Click(Sender: TObject);
    procedure Selectsourcepath1Click(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var
CanClose: Boolean);
    procedure SpeedButton1Click(Sender: TObject);
    procedure UpdateStateString;
    procedure Localize();
    procedure Switchsourcemode1Click(Sender: TObject);
    procedure LoadPic();
    procedure Selectfilepath1Click(Sender: TObject);
    procedure Projectreference1Click(Sender: TObject);
    private
    public
    var key_small_for_encode:string;
        key_small_for_decode:string;
    { Public declarations }
  end;
var
  MF: TForm1;
implementation
{$R *.dfm}
// обновить надпись с описанием текущего состояния
приложения(настроек)
procedure Tform1.UpdateStateString;
var userModeRenamed, pictureModeRenamed:string;
begin
  if getlang = 'en' then begin
    if getusermode = 'creator' then
      userModeRenamed := 'key creator'
    else
      userModeRenamed := 'user';
    if GetCryptMode = 'pic' then
      pictureModeRenamed := 'decrypt'
    else
      pictureModeRenamed := 'encrypt';
    label1.Caption := 'Type of user: ' + userModeRenamed;
    label4.Caption := 'Type of encryption: ' +
pictureModeRenamed;
  end
  else begin
    if getusermode = 'creator' then
      userModeRenamed := 'создатель ключа'
    else
      userModeRenamed := 'пользователь';
    if GetCryptMode = 'pic' then
      pictureModeRenamed := 'дешифровать'
    else
      pictureModeRenamed := 'зашифровать';
    label1.Caption := 'Тип пользователя : ' +
userModeRenamed;
    label4.Caption := 'Вид шифрования : ' +
pictureModeRenamed;
  end;
end;
// закрытие по нажатию пункта меню
procedure TForm1.Close1Click(Sender: TObject);
begin
  mf.Close;
  Loading.load.Close;
end;
// полное закрытие приложения по кнопке
procedure TForm1.FormCloseQuery(Sender: TObject;
var CanClose: Boolean);
begin
  Loading.load.Close;
end;
// функция, оотвечающая за локализацию приложения
procedure TForm1.Localize();
begin
  UpdateStateString;
  if GetLang = 'en' then begin
    label2.Caption := 'For details, go to the settings form';
    conff.Caption := 'settings form';
    load.Caption := 'loading';
    label5.Caption := 'Short state:';
```

```
    mm.Items[0].Caption := 'File';
    mm.Items[0].Items[0].Caption := 'Close';
    mm.Items[1].Caption := 'Settings';
    mm.Items[1].Items[0].Caption := 'Open settings form';
    mm.Items[1].Items[1].Caption := 'Switch language';
    mm.Items[1].Items[2].Caption := 'Switch generate mode';
    mm.Items[1].Items[3].Caption := 'Switch user mode';
    mm.Items[1].Items[4].Caption := 'Switch encryption mode';
    mm.Items[1].Items[5].Caption := 'Select key';
    mm.Items[1].Items[6].Caption := 'Select source';
    mm.Items[1].Items[7].Caption := 'Select file';
    mm.Items[2].Caption := 'Reference';
    mm.Items[2].Items[0].Caption := 'Project reference';
    conff.Label2.Caption := 'Russian/English';
    conff.Label3.Caption := 'User/Key creator';
    conff.Label4.Caption := 'Use cryptoproof numbers?';
    conff.Label8.Caption := 'Encrypt/Decrypt';
    conff.Label1.Caption := 'Select key';
    conff.Label5.Caption := 'Select source';
    conff.Label11.Caption := 'Select file';
    conff.Label9.Caption := 'For changes to take effect, close the form';
    conff.Button1.Caption := 'Main';
    mf.key_small_for_encode := 'File too large, cannot be encoded';
    mf.key_small_for_decode := 'The key bytes are not enough to decode the data';
  end
  else begin
    label2.Caption := 'За подробностями перейдите на форму настроек';
    conff.Caption := 'форма настроек';
    load.Caption := 'загрузка';
    label5.Caption := 'Краткое состояние:';
    mm.Items[0].Caption := 'Файл';
    mm.Items[0].Items[0].Caption := 'Закрыть';
    mm.Items[1].Caption := 'Настройки';
    mm.Items[1].Items[0].Caption := 'Открыть форму настроек';
    mm.Items[1].Items[1].Caption := 'Переключить язык';
    mm.Items[1].Items[2].Caption := 'Переключить вид генерации';
    mm.Items[1].Items[3].Caption := 'Переключить вид пользователя';
    mm.Items[1].Items[4].Caption := 'Переключить вид шифрования';
    mm.Items[1].Items[5].Caption := 'Выбрать ключ';
    mm.Items[1].Items[6].Caption := 'Выбрать источник';
    mm.Items[1].Items[7].Caption := 'Выбрать файл';
    mm.Items[2].Caption := 'Справка';
    mm.Items[2].Items[0].Caption := 'Справка проекта';
    conff.Label2.Caption := 'Русский/Английский';
    conff.Label3.Caption := 'Пользователь/Создатель ключа';
    conff.Label4.Caption := 'Использовать криптостойкие числа?';
    conff.Label8.Caption := 'Зашифровать/Дешифровать';
    conff.Label1.Caption := 'Выбрать ключ';
    conff.Label5.Caption := 'Выбрать источник';
    conff.Label11.Caption := 'Указать файл';
    conff.Label9.Caption := 'Чтобы изменения вступили в силу, закройте форму';
    conff.Button1.Caption := 'Главная';
    mf.key_small_for_encode := 'Слишком большой файл, невозможно закодировать';
    mf.key_small_for_decode := 'Байт ключа недостаточно, чтобы декодировать данные';
  end;
end;
// загрузка картинки, которая показывает какой процесс настроен
procedure Tform1.LoadPic();
var filename: string;
begin
  filename := extractFilePath(paramstr(0)) + PathDelim + 'imgs' +  PathDelim;
  var pic := TPicture.Create;
  if getUsermode = 'creator' then
    filename := filename + 'c_key.png'
  else if getcryptmode = 'pic' then
    filename := filename + 'decode.png'
  else filename := filename + 'encode.png';
  pic.LoadFromFile(filename);
  Image1.Picture := pic;
  image1.update;
end;
// первоначальная настройка формы
procedure TForm1.FormCreate(Sender: TObject);
begin
  GenSettingsFileIfNotExists;
  LoadKeyPng;
  LoadFilePng;
  LoadPic;
  Localize;
end;
// открытие формы настроек
procedure TForm1.OpenForm1Click(Sender: TObject);
begin
  ConfF.ShowModal;
end;
// открытие справки
procedure TForm1.Projectreference1Click(Sender: TObject);
begin
  ShellExecute(0, PChar('Open'),PChar(extractFilePath(paramstr(0)) + 'HID.chm'),nil,nil,SW_SHOW);
end;
// выбор пути к файлу
procedure TForm1.Selectfilepath1Click(Sender: TObject);
begin
  SelectFilePath;
end;
// выбор пути к картинке-ключу
```

```
procedure TForm1.Selectkeypath1Click(Sender:
TObject);
begin
  Selectkeypath;
end;
// выбор пути к картинке-источнику
procedure TForm1.Selectsourcepath1Click(Sender:
TObject);
begin
  Selectsourcepath;
end;
// реализация выбранного процесса по нажатию
кнопки
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  if getusermode = 'creator' then begin
    EncryptPicture();
    pngForWork.SaveResult(GetPictureKeyPath);
  end
  else begin
    if Saves.GetCryptMode = 'npic' then begin
      EncryptFile();
    end
    else begin
      DecryptFile();
    end;
  end;
end;
// смена пользователя
procedure TForm1.SwitchCC1Click(Sender: TObject);
begin
  if GetUserMode = 'consumer' then
    SaveUserMode('creator')
  else
    SaveUserMode('consumer');
  LoadPic;
end;
// смена вида генерации байтов ключа
procedure TForm1.SwitchCG1Click(Sender: TObject);
begin
  if GetGenMode = 'nc' then
    SaveGenMode('cn')
  else
    SaveGenMode('nc')
end;
// смена шифрования/дешифрования
procedure TForm1.Switchsourcemode1Click(Sender:
TObject);
begin
  if GetCryptMode = 'pic' then
    SaveCryptMode('npic')
  else
    SaveCryptMode('pic');
  LoadPic;
end;
// смена языка
procedure TForm1.Swithlanquage1Click(Sender:
TObject);
begin
  if Getlang = 'en' then
    SaveLanguage('ru')
  else
    SaveLanguage('en')
end;
end.
unit Cypher;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Rand, Vcl.Imaging.pngimage, Picture;
var
  pngForWork,pngForFile:Png;
function EncryptDecryptByte(source:TColor; key:
byte):Tcolor;
function DecryptDecryptByte(source:TColor):byte;
function HigherShapeFilter(c:byte):byte;
function GetEncryptedShape(c:byte; en:char):byte;
function
GetDecryptedBytes(count:integer):TArray<Byte>;
procedure EncryptPicture();
procedure EncryptFile();
procedure DecryptFile();
procedure LoadKeyPng();
procedure LoadFilePng();
procedure writebytestopic(pic:PNG;
bytes:TArray<Byte>);
function GetFileBytes():TArray<Byte>;
procedure SaveSizeToPic(len:Int64);
function GetSizeFromPic():int64;
function GetPosXFromPic():int64;
procedure SavePosXToPic(num:int64);
function GetPosYFromPic():int64;
procedure SavePosYToPic(num:int64);
procedure SetPosXYToDefault();
procedure SaveNameToPic(const name:string);
function GetNameFromPic():string;
implementation
uses Saves, Main;
// загрузка картинки-ключа в переменную
procedure LoadKeyPng();
begin
  pngForWork := Png.Create(GetPictureKeyPath);
  pngForWork.picture.RemoveTransparency();
end;
// загрузка картинки-источника в переменную
procedure LoadFilePng();
begin
  pngForFile := Png.Create(GetPictureSourcePath);
  pngForFile.picture.RemoveTransparency();
end;
// сохранение названия файла в картинку-источник
procedure SaveNameToPic(const name:string);
var Bytes: TArray<byte>;
    color: Tcolor;
begin
  bytes := TEncoding.UTF8.GetBytes(name);
  SetLength(Bytes, 64);
  for var i:= 8 to 71 do begin
```

```
    color                                                    :=
EncryptDecryptByte(pngForFile.GetPixelColor(i,          0),
bytes[i-8]);
    pngForFile.SetPixelColor(i, 0, color);
  end;
  pngForFile.SaveResult(GetPictureSourcePath);
end;
// получение названия файла из картинки-источника
function GetNameFromPic():string;
  var Bytes: TArray<byte>;
      color: Tcolor;
begin
  SetLength(Bytes, 64);
  for var i:= 8 to 71 do begin
    color := pngforfile.GetPixelColor(i, 0);
    bytes[i-8] := DecryptDecryptByte(color);
  end;
  Result := TEncoding.UTF8.GetString(bytes);
end;
// получение размера файла из картинки-источника
function GetSizeFromPic():int64;
  var Bytes: TArray<byte>;
      color: Tcolor;
begin
  SetLength(Bytes, 8);
  for var i:=0 to 7 do begin
    color := pngforfile.GetPixelColor(i, 0);
    bytes[i] := DecryptDecryptByte(color);
  end;
  Result := PInteger(@Bytes[0])^;
end;
// сохранение размера файла в картинку-источник
procedure SaveSizeToPic(len:Int64);
  var Bytes: TArray<byte>;
      color: Tcolor;
begin
  SetLength(Bytes, 8);
  Move(len, Bytes[0], SizeOf(Int64));
  for var i:= 0 to 7 do begin
    color                                                    :=
EncryptDecryptByte(pngForFile.GetPixelColor(i,          0),
bytes[i]);
    pngForFile.SetPixelColor(i, 0, color);
  end;
  pngForFile.SaveResult(GetPictureSourcePath);
end;
// сохранение x координаты в картинку-ключ
procedure SavePosXToPic(num:Int64);
  var Bytes: TArray<byte>;
      color: Tcolor;
begin
  SetLength(Bytes, 8);
  Move(num, Bytes[0], SizeOf(Int64));
  for var i:= 0 to 7 do begin
    color                                                    :=
EncryptDecryptByte(pngForwork.GetPixelColor(i,          0),
bytes[i]);
    pngForwork.SetPixelColor(i, 0, color);
  end;
  pngForwork.SaveResult(GetPictureKeyPath);
end;
// чтение x координаты из картинки-ключа
function GetPosXFromPic():int64;
  var Bytes: TArray<byte>;
      color: Tcolor;
begin
  SetLength(Bytes, 8);
  for var i:= 0 to 7 do begin
    color := pngforwork.GetPixelColor(i, 0);
    bytes[i] := DecryptDecryptByte(color);
  end;
  Result := PInteger(@Bytes[0])^;
end;
// сохранение y координаты в картинку-ключ
procedure SavePosYToPic(num:Int64);
  var Bytes: TArray<byte>;
      color: Tcolor;
begin
  SetLength(Bytes, 8);
  Move(num, Bytes[0], SizeOf(Int64));
  for var i:= 8 to 15 do begin
    color                                                    :=
EncryptDecryptByte(pngForWork.GetPixelColor(i,          0),
bytes[i-8]);
    pngForWork.SetPixelColor(i, 0, color);
  end;
  pngForwork.SaveResult(GetPictureKeyPath);
end;
// чтение y координаты из картинки-ключа
function GetPosYFromPic():int64;
  var Bytes: TArray<byte>;
      color: Tcolor;
begin
  SetLength(Bytes, 8);
  for var i:= 8 to 15 do begin
    color := pngforwork.GetPixelColor(i, 0);
    bytes[i-8] := DecryptDecryptByte(color);
  end;
  Result := PInteger(@Bytes[0])^;
end;
// установка значений по умолчанию для позиций x и
y
procedure SetPosXYToDefault();
var color: Tcolor;
begin
  SavePosXToPic(0);
  SavePosYToPic(1);
end;
// закодировать байт в цвет
function EncryptDecryptByte(source:TColor; key:
byte):TColor;
var r, g, b:byte;
begin
    var keyLikeStr := inttostr(key);
    if keyLikeStr.Length = 1 then keyLikeStr := '00' +
keyLikeStr;
    if keyLikeStr.Length = 2 then keyLikeStr := '0' +
keyLikeStr;
```

```
  r := GetEncryptedShape(HigherShapeFilter(GetRValue(source)), keyLikeStr[1]);
  g := GetEncryptedShape(HigherShapeFilter(GetGValue(source)), keyLikeStr[2]);
  b := GetEncryptedShape(HigherShapeFilter(GetBValue(source)), keyLikeStr[3]);
  var color := RGB(r, g, b);
  Result:= color;
end;
// функция для корректной записи в цвет
function HigherShapeFilter(c:byte):byte;
begin
  if c >= 250 then c := c - 10;
  Result := c;
end;
// кодирование одной 1/3 через 1/3 байта ключа
function GetEncryptedShape(c:byte; en:char):byte;
begin
  Result := c - c mod 10 + strtoint(en);
end;
// декодирование зашифрованного байта из цвета
function DecryptDecryptByte(source:TColor):byte;
var r, g, b:byte;
begin
  r := (GetRValue(source) mod 10) * 100;
  g := (GetGValue(source) mod 10) * 10;
  b := GetBValue(source) mod 10;
  Result:= r + g + b;
end;
// кодирование картинки-ключа
procedure EncryptPicture();
var
 bytes: Tarray<Byte>;
begin
 bytes := GenerateRandomBytes(pngForWork.size);
 writebytestopic(pngforwork,bytes);
 SetPosXYToDefault;
end;
// запись байтов в картинку
procedure writebytestopic(pic:PNG; bytes:TArray<Byte>);
var color, encryptedColor:TColor;
begin
 var counter := 0;
  for var y := 1 to pic.picture.Width - 1 do begin
  for var x := 0 to pic.picture.Height - 1 do begin
    color := pic.GetPixelColor(x, y);
    encryptedColor := EncryptDecryptByte(color, bytes[counter]);
    pic.SetPixelColor(x, y, encryptedColor);
    counter := counter + 1;
    if counter >= length(bytes) then
      break
  end;
  if counter >= length(bytes) then
    break
 end;
end;
// шифрование файла
procedure EncryptFile();
var
 FileStream: TFileStream;
 bytes: TArray<Byte>;
begin
 LoadKeyPng;
 LoadFilePng;
 FileStream := TFileStream.Create(GetFilePath, fmOpenRead);
 try
  SetLength(bytes, FileStream.Size);
  FileStream.ReadBuffer(bytes[0], FileStream.Size);
 finally
  FileStream.Free;
 end;
 if length(bytes) > pngForWork.size then begin
   messageDlg(mf.key_small_for_encode, mtError, [mbOK],0);
   exit
 end;
 var encryptedBytes := GetDecryptedBytes(Length(bytes));
 for var i := 0 to Length(bytes) - 1 do begin
//   showmessage(IntToStr(encryptedBytes[i]));
  bytes[i] := bytes[i] xor encryptedBytes[i];
 end;
 SaveSizeToPic(length(bytes));
 SaveNameToPic(ExtractFileName(GetFilePath));
 writebytestopic(pngForFile, bytes);
 pngforfile.SaveResult(GetPictureSourcePath);
end;
// декодирование файла
procedure DecryptFile();
var
 color:TColor;
 FileStream: TFileStream;
begin
 LoadKeyPng;
 LoadFilePng;
 var fileBytes := GetFileBytes;
  if length(filebytes) > (pngForWork.size - GetPosXFromPic- GetPosYFromPic * pngforwork.picture.Width - 1) then begin
   messageDlg(mf.key_small_for_decode, mtError, [mbOK],0);
   exit
 end;
 var decryptedBytes := GetDecryptedBytes(GetSizeFromPic);
 for var i := 0 to Length(fileBytes) - 1 do begin
//   showmessage(IntToStr(decryptedBytes[i]));
  fileBytes[i] := fileBytes[i] xor decryptedBytes[i];
 end;
 FileStream := TFileStream.Create(extractFilePath(paramstr(0)) + 'res' + PathDelim + GetNameFromPic, fmCreate);
 try
  FileStream.Write(fileBytes[0], Length(fileBytes));
```

```
  finally
    FileStream.Free;
  end;
end;
// получение байтов файла
function GetFileBytes():TArray<Byte>;
var
  color:TColor;
  bytes:TArray<byte>;
begin
  var counter:int64:=0;
  var size := GetSizeFromPic;
  SetLength(bytes, size);
    for var y := 1 to pngForFile.picture.Width - 1 do begin
    for var x := 0 to pngForFile.picture.Height - 1 do begin
      color := pngForFile.GetPixelColor(x, y);
      bytes[counter] := DecryptDecryptByte(color);
      counter := counter + 1;
      if counter >= size then begin
        break
      end;
    end;
    if counter >= size then
       break;
  end;
  Result := bytes;
end;
// получение байтов ключа
function
GetDecryptedBytes(count:integer):TArray<Byte>;
  var
  color:TColor;
  bytes:TArray<byte>;
  encryptedColor: TColor;
begin
  var counter:int64:=0;
  SetLength(bytes, count);
  var posX :int64 := GetPosXFromPic;
  var posY :int64 := GetPosYFromPic;
//  showmessage('poss');
    for var y := posY to pngForWork.picture.Width - 1 do
begin
    for var x := posX to pngForWork.picture.Height - 1 do
begin
//      showmessage(IntToStr(x));
      PosX := x;
      color := pngForWork.GetPixelColor(x, y);
      bytes[counter] := DecryptDecryptByte(color);
      if (counter + 1) <> count then begin
        color          :=          pngForWork.GetPixelColor(x,
y);//затирание значения
        encryptedColor     :=     EncryptDecryptByte(color,
Random(256));
        pngForWork.SetPixelColor(x, y, encryptedColor);
      end;
      counter := counter + 1;
      if counter >= count then begin
        break
      end;
    end;
    if counter >= count then
       break;
       posY := y;
  end;
  SavePosXToPic(posX);
  SavePosYToPic(posY);
  pngForWork.SaveResult(GetPictureKeyPath);
  Result := bytes;
end;
end.
unit Rand;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, ShellApi, Saves;
function                GenerateRandomBytes(len:
ulong):TArray<Byte>;
function ReadData():TArray<Byte>;
function ListToArray(list:Tlist):Tarray<Byte>;
implementation
uses Main;
// генерация случайных байтов ключа
function                GenerateRandomBytes(len:
ulong):TArray<Byte>;
begin
  if Saves.GetGenMode = 'cn' then begin
    ShellExecute(0,          'open',          'Rand.exe',
PChar(PWideString(UIntToStr(len))),          nil,
SW_SHOWNORMAL);
    sleep(500);
    Result:= ReadData();
  end
  else begin
    var arr: TArray<Byte>;
    setLength(arr, len);
    for var i := 0 to len-1 do begin
      arr[i] := Random(256);
    Result:= arr;
    end;
  end;
end;
// чтение байтов, сгенерированных отдельным
приложением
function ReadData():TArray<Byte>;
var
  f: file of byte;
  list: Tlist;
  b:byte;
begin
  Reset(f, 'data.dat');
  list := Tlist.Create();
  while not eof(f) do
  begin
    read(f, b);
    list.Add(Pointer(b))
  end;
  Close(F);
  Result:= ListToArray(list);
end;
```

```
// перевод списка в массив
function ListToArray(list:Tlist):Tarray<Byte>;
var arr: TArray<Byte>;
begin
  SetLength(arr, list.Count);
  for var i:=0 to list.Count - 1 do
    arr[i] := Integer(list[i]);
  Result:= arr;
end;
end.
unit Picture;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Rand, Vcl.Imaging.pngimage, jpeg;
type
  Png = class
    picture: TPngImage;
    size: integer;
    constructor Create(path:string);
    procedure LoadPicture(path:string);
    procedure SaveResult(filename:string);
    procedure SetPixelColor(x, y:integer; color:TColor);
    function GetPixelColor(x, y:integer):TColor;
  end;
implementation
// контруктор класса. Загружает картинку и
устанавливает поле size
constructor Png.Create(path:string);
begin
  LoadPicture(path);
  size:= picture.Width * picture.Height;
end;
// загружает картинку в переменную
procedure Png.LoadPicture(path:string);
begin
  picture := TPngImage.Create;
  picture.loadfromfile(path);
end;
// сохраняет результат
procedure Png.SaveResult(filename:string);
begin
  picture.SaveToFile(filename)
end;
// устанавливает цвет пикселю
procedure Png.SetPixelColor(x, y:integer; color:TColor);
begin
  picture.Canvas.Pixels[y, x] := color;
end;
// возвращает цвет пикселя
function Png.GetPixelColor(x, y:integer):TColor;
begin
  Result := picture.Canvas.Pixels[y, x];
end;
end.
unit Saves;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, ShellApi;
procedure GenSettingsFileIfNotExists();
procedure SaveLanguage(lang:string);
procedure SavePictureKeyPath(path:string);
procedure SavePictureSourcePath(path:string);
procedure SaveFilePath(path:string);
procedure SaveUserMode(mode:string);
procedure SaveGenMode(mode:string);
procedure SaveCryptMode(mode:string);
procedure UpdateSettings(lang, um, gm, pm:string);
function GetLang():string;
function GetPictureKeyPath():string;
function GetPictureSourcePath():string;
function GetUserMode():string;
function GetGenMode():string;
function GetCryptMode():string;
function GetFilePath():string;
function ReadSettingsFile():TStringlist;
procedure WriteSettingsInFile(list:TStringlist);
function GetPath():string;
procedure Selectkeypath();
procedure Selectsourcepath();
procedure SelectFilePath();
implementation
uses Main;
// создание настроек по умолчанию
procedure GenSettingsFileIfNotExists();
var F:TextFile;
begin
  if not FileExists(extractFilePath(paramstr(0)) +
'settings.txt') then begin
    AssignFile(F, extractFilePath(paramstr(0)) +
'settings.txt');
    Rewrite(F);
    writeln(F, 're');
    writeln(F, extractFilePath(paramstr(0)) + 'ex' +
PathDelim + 'key.png');
    writeln(F, 'creator');
    writeln(F, 'cn');
    writeln(F, extractFilePath(paramstr(0)) + 'ex' +
PathDelim + 'source.png');
    writeln(F, 'pic');
    writeln(F, extractFilePath(paramstr(0)) + 'ex' +
PathDelim + 'test.txt');
    close(F);
  end;
end;
// сохранение языка в настройки, обновление
локализации
procedure SaveLanguage(lang:string);
var
  list: TStringlist;
begin
  list := ReadSettingsFile();
  list[0] := lang;
  WriteSettingsInFile(list);
  mf.Localize();
end;
```

```
// сохранение картинки-ключа в настройки
procedure SavePictureKeyPath(path:string);
var
  list: TStringlist;
begin
  list := ReadSettingsFile();
  list[1] := path;
  WriteSettingsInFile(list);
end;

// сохранение картинки-источника в настройки
procedure SavePictureSourcePath(path:string);
var
  list: TStringlist;
begin
  list := ReadSettingsFile();
  list[4] := path;
  WriteSettingsInFile(list);
end;

// сохранение текущего режима пользователя в настройки
procedure SaveUserMode(mode:string);
var
  list: TStringlist;
begin
  list := ReadSettingsFile();
  list[2] := mode;
  WriteSettingsInFile(list);

  mf.UpdateStateString;
end;

// сохранение текущего режима генерации в настройки
procedure SaveGenMode(mode:string);
var
  list: TStringlist;
begin
  list := ReadSettingsFile();
  list[3] := mode;
  WriteSettingsInFile(list);
end;

// сохранение текущего режима шифрования в настройки
procedure SaveCryptMode(mode:string);
var
  list: TStringlist;
begin
  list := ReadSettingsFile();
  list[5] := mode;
  WriteSettingsInFile(list);
  mf.UpdateStateString;
end;
// сохранение пути к файлу для шифрования в настройки
procedure SaveFilePath(path:string);
var
  list: TStringlist;
begin
  list := ReadSettingsFile();
  list[6] := path;
  WriteSettingsInFile(list);
end;
// обновление настроек
procedure UpdateSettings(lang, um, gm, pm:string);
var list: TStringlist;
begin
  list := ReadSettingsFile;
  list[0] := lang;
  list[2] := um;
  list[3] := gm;
  list[5] := pm;
  WriteSettingsInFile(list);
  mf.Localize;
  mf.LoadPic;
end;
// получение текущего сохранённого языка
function GetLang():string;
begin
   Result := ReadSettingsFile[0];
end;
// получение текущего сохранённого пути к картинке-ключу
function GetPictureKeyPath():string;
begin
   Result := ReadSettingsFile[1];
end;
// получение текущего сохранённого пути к файлу
function GetFilePath():string;
begin
   Result := ReadSettingsFile[6];
end;
// получение текущего сохранённого пути к картинке-источнику
function GetPictureSourcePath():string;
begin
   Result := ReadSettingsFile[4];
end;
// получение текущего режима пользователя
function GetUserMode():string;
begin
   Result := ReadSettingsFile[2];
end
// получение текущего режима генерации байтов ключа
function GetGenMode():string;
begin
   Result := ReadSettingsFile[3];
end;
// получение текущего режима шифрования ключа
function GetCryptMode():string;
begin
   Result := ReadSettingsFile[5];
end;
// чтение всех настроек
function ReadSettingsFile():TStringlist;
var
  F:textFile;
```

```
  list: TStringlist;
  s:string;
begin
  list := TStringList.Create();
  AssignFile(F,          extractFilePath(paramstr(0))          +
'settings.txt');
  Reset(F);
  while not eof(F) do begin
    readln(F, s);
    list.Add(s);
  end;
  close(F);
  Result := list;
end;
// запись всех настроек в файл
procedure WriteSettingsInFile(list:TStringlist);
var
  F:textFile;
  s:string;
begin
  AssignFile(F,          extractFilePath(paramstr(0))          +
'settings.txt');
  Rewrite(F);
  for var i := 0 to list.Count-1 do begin
    s := list[i];
    writeln(F, s);
  end;
  close(F);
end;
// функция для получения пути
function GetPath():string;
begin
  var path:='';
  var OpenDialog := TOpenDialog.Create(nil);
  if OpenDialog.Execute then
    path := OpenDialog.FileName;
  Result:= path
end;
// сохранение пути к файлу
procedure SelectFilePath();
begin
  var path := GetPath;
  if path <> '' then
    SaveFilePath(path);
end;
// сохранение пути к картинке-ключу
procedure Selectkeypath();
begin
  var path := GetPath;
  if path <> '' then
    SavePictureKeyPath(path);
end;
// сохранение пути к картинке-источнику
procedure Selectsourcepath();
begin
  var path := GetPath;
  if path <> '' then
    SavePictureSourcePath(path);
end;
end.
```

```
unit Conf;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.WinXCtrls, Vcl.Buttons, saves;
type
  TForm2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    ToggleSwitch1: TToggleSwitch;
    ToggleSwitch2: TToggleSwitch;
    ToggleSwitch3: TToggleSwitch;
    Button1: TButton;
    Label5: TLabel;
    SpeedButton2: TSpeedButton;
    Label6: TLabel;
    Label7: TLabel;
    SpeedButton1: TSpeedButton;
    Label8: TLabel;
    ToggleSwitch4: TToggleSwitch;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    SpeedButton3: TSpeedButton;
    procedure Button1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure Save();
    procedure FormClose(Sender: TObject; var Action:
TCloseAction);
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  ConfF: TForm2;
implementation
{$R *.dfm}
uses cypher;
//сохранение настроек и закрытие формы по кнопке
procedure TForm2.Button1Click(Sender: TObject);
begin
  Save;
  Close;
end;
//сохранение настроек
procedure TForm2.Save();
var lang, um, gm, pm:string;
begin
  if ToggleSwitch1.State = tssOn then
  lang := 'en'
else
  lang := 'ru';
```

```
if ToggleSwitch2.State = tssOn then
  um := 'creator'
else
  um := 'consumer';
if ToggleSwitch3.State = tssOn then
  gm := 'cn'
else
  gm := 'nc';
if ToggleSwitch4.State = tssOn then
  pm := 'pic'
else
  pm := 'npic';
UpdateSettings(lang, um, gm, pm);
Loadkeypng;
end;
// выбор картинки для ключа по кнопке
procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Selectkeypath;
  label6.Caption := GetPictureKeyPath;
end;
// выбор картинки для хранения закодированного файла по кнопке
procedure TForm2.SpeedButton2Click(Sender: TObject);
begin
  Selectsourcepath;
  label7.Caption := GetPictureSourcePath;
end;
// выбор файла для кодирования по кнопке
procedure TForm2.SpeedButton3Click(Sender: TObject);
begin
  Selectfilepath;
  label10.Caption := GetFilePath;
end;
// загрузка всех значений настроек
procedure TForm2.FormActivate(Sender: TObject);
begin
  if GetLang = 'en' then
    ToggleSwitch1.State := tssOn
  else
    ToggleSwitch1.State := tssOff;
  if GetUserMode = 'creator' then
    ToggleSwitch2.State := tssOn
  else
    ToggleSwitch2.State := tssOff;
  if GetGenMode = 'cn' then
    ToggleSwitch3.State := tssOn
  else
    ToggleSwitch3.State := tssOff;
  if GetCryptMode = 'pic' then
    ToggleSwitch4.State := tssOn
  else
    ToggleSwitch4.State := tssOff;
  label6.Caption := GetPictureKeyPath;
  label7.Caption := GetPictureSourcePath;
  label10.Caption := GetFilePath;
end;
//сохранение настроек при закрытии
procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   Save;
end;
end.
unit Loading;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls, Vcl.ExtCtrls;
type
  TForm3 = class(TForm)
    ProgressBar1: TProgressBar;
    Timer1: TTimer;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    await:integer;
  end;
var
  load: TForm3;
implementation
uses Main;
// активация формы при конце загрузки
procedure TForm3.Timer1Timer(Sender: TObject);
begin
  ProgressBar1.StepBy(1);
  if ProgressBar1.Position = ProgressBar1.Max then begin
                await := await - 1;
                if await = 0 then begin
                  Timer1.Enabled := False;
                  hide;
                  mf.Show();
                end;
              end;
            end;
            {$R *.dfm}
            // установка значений по умолчанию для загрузки и запуск таймера
            procedure TForm3.FormCreate(Sender: TObject);
            begin
              await := 40;
              ProgressBar1.Max := 100;
              ProgressBar1.Position := 0;
              Timer1.Enabled := True;
            end;
            end.
```