Содержание

Обозначения и сокращения

ПО – программное обеспечение
ЭВМ – электронно-вычислительная машина
ОС – операционная система
ЯП – язык программирования
ООП – объектно-ориентированное программирование

Введение

Практика является одним из важнейших этапов подготовки высшими образовательными заведениями высококвалифицированных специалистов. Практика студентов Южного федерального университета является составной частью образовательной программы высшего профессионального образования и представляет собой одну из форм организации учебного процесса, заключающуюся в профессионально-практической подготовке студентов на базах практик.

В ходе практики студент решает поставленную задачу, изучая дополнительную литературу, вникая в тонкости той или иной сферы профессиональной деятельности.

Основными целями практики, которые стоят перед студентами при ее прохождении согласно «Положению о практиках студентов Южного федерального университета» являются:

1) Закрепление, углубление и расширение теоретических знаний, умения и навыков, полученных студентами в процессе теоретического обучения;
2) Овладение профессионально-практическими умениями, производственными навыками и передовыми методами труда;
3) Овладение основами профессии в операционной сфере: ознакомление и усвоение методологии и технологии решения профессиональных задач.

Студентам всегда не хватает практических знаний, опыта работы с определенным ПО, отсюда и возникает потребность в практике, дающей необходимые практические знания будущим специалистам в своей сфере.

1 Постановка задачи

Цель практики – изучение необходимого теоретического материала и последующая разработка проекта по заданному плану:

1. Изучение требований к стилю кодирования на высокоуровневых языках.
2. Изучение особенностей объектного дизайна несложных приложений.
3. Изучение инженерных методик кодирования и отладки приложений на высокоуровневых языках.
4. Разработка проекта, его тестирование и верификация.

2 Ход работы

Первым шагом в разработке проекта однозначно является анализ поставленной задачи.

2.1 Анализ задачи

В качестве темы я выбрал разработку программы для социальной сети Вконтакте.

Суть работы программы:

Программа предназначена для мониторинга состояния пользователей социальной сети Вконтакте. Она проверяет каждые 1-3 сек. в сети ли пользователь, если в сети то подаёт соответствующий сигнал, если же нет, то дальше проверяет его состояние. Также имеются некоторые дополнительные функции, такие как, авторизация и «горячая» отправка сообщений, открытие профиля пользователя при его появлении в сети, а также отключение подающего сигнала. Программа имеет GUI.

После того, как анализ задачи завершен, необходимо составить требования к проекту.

2.2 Требования к проекту

1.1. Требования к системе в целом

- Требования к структуре и функционированию системы

Перечень подсистем их назначение и основные характеристики

В состав должны входить следующие модули:

Модуль аутентификации, позволяющий авторизоваться пользователю

Модуль отправки мгновенного сообщения активному пользователю

Модуль парсинга данных с ответа VK API

Также криптографический пакет OpenSSL, через который осуществляется запросы к API

Требование к обмену данными с внешними системами

Для обеспечения обмена данными с внешними системами, как основной, должен использоваться формат JSON.

Разработка описания структуры и формата обменных данных для обеспечения обмена с конкретной внешней системой не является предметом настоящего Технического задания.

Требование к обеспечению взаимодействия с предыдущими версиями системы

Требуется разработать инструментарий для передачи в систему основных данных из базы данных предыдущей версии системы.

- Требования к защите информации от несанкционированного доступа

Компоненты подсистемы защиты от НСД должны обеспечивать:

Идентификацию пользователя;

Аутентификация пользователя;

Разграничение доступа пользователей на уровне задач и информационных массивов.

- Требование к хранению данных учета

Система должна обеспечивать следующие свойства в отношении хранения данных учета:

данные учёта не могут быть исключены без следа из системы учёта (*неисключаемость*);

данные учёта не могут быть изменены или дополнены в системе учёта (в том числе и для целей исправления ошибок) без сохранения в системе учёта исходного состояния данных или возможности восстановления этого состояния (*сохранение*);

должна быть возможность восстановления истории внесения учётных данных, изменений и дополнений в них, и выполнения иных действий в отношении определённых объектов учёта (*восстановимость*).

Для обеспечения перечисленных выше свойств неисключаемости, сохранения и восстановимости состояний должно использоваться *журналирование* учётных событий, предусматривающее наличие специальных форм ведения учёта – журналов учётных событий, накапливающих учётные данные в хронологической последовательности их внесения или обновления в системе учёта.

1.2. Требования к видам обеспечения

- *Требования к информационному обеспечению*

Состав, структура и способы организации данных в системе

Требования к входной информации

Система должна быть способна принимать входную информацию оформленную только в виде JSON ответов.

Основными источниками входной информации для системы являются:

Ответ сервера на Get запрос определённого метода VK API

- Требования к лингвистическому обеспечению

Требования к языку взаимодействия с пользователями системы

Все прикладное программное обеспечение системы для организации взаимодействия с пользователем должно использовать русский язык.

Требования к применению языков программирования и форматов

В разработке системы могут быть использованы следующие форматы и языки:

JSON

Delphi

Требования к средствам описания предметной области

В качестве языка описания предметной области, был использован язык UML.

- *Требования к программному обеспечению*

Требования к общесистемным программным средствам

Рабочие станции должны функционировать под управлением следующих ОС: Windows XP/Vista/ 7 / 8 / 8.1

На рабочих станциях, работающих под управлением указанных выше ОС, обязательно наличие Internet Explorer версии не ниже 8.0 или любого другого браузера.

Требования к наличию дополнительных программных средств

Для корректной работы программы необходимы библиотеки ssleay32.dll и libeay32.dll криптографического пакета OpenSSL.

- *Требования к техническому обеспечению*

Требования к вычислительным комплексам

Требования к техническим характеристикам клиентских рабочих станций:

Процессор – Intel i3 1,2 ГГц;

Объем оперативной памяти – 1 Гб;

Дисковая подсистема – 16 Гб;

Сетевой адаптер – 100 Мбит.

Требования к вычислительным сетям:

Для функционирования и взаимодействия серверов и приложения необходимо наличие сети Ethernet с протоколом TCP/IP и средой передачи данных со скоростью от 256 Кбит/сек.

2.3 Критерии качества программы

1. ***Правильность*** – программа должна работать корректно, согласно зафиксированным требованиям.
2. ***Надежность*** – в работе программы не должно быть сбоев, отказов или зацикливаний вне зависимости от исходных данных. К примеру, число ошибок, не найденных программистом или тестировщиком, должно составлять 10-12 штук на 1000 строк кода.
3. ***Эффективность*** – рациональное использование программой ресурсов ЭВМ.
4. ***Защита*** – помимо секретности данных программы это еще ее устойчивость к исключительным ситуациям, которые могут возникнуть в ходе ее работы, несмотря на все попытки разработчика избежать ошибок, а также действия пользователя. Можно составить следующую логическую цепочку, приводящую к необходимости принятия мер защиты от возникновения исключительных ситуаций: Опасность → Последствия → Меры защиты.
5. ***Портируемость*** – программа должна работать на других ЭВМ и семействе ОС Windows.
6. ***Краткость*** – отсутствие в коде повторяющихся элементов или частей процедур.
7. ***Понятность*** – смысл работы программы должен быть понятен из самой программы.

2.4 Изучение требований к стилю кодирования на высокоуровневых языках

В качестве языка разработки я выбрал Delphi. Проект разработан в среде RAD Studio XE6, тип проекта: GUI VCL Application.

При написании кода были применены стандарты “Coding Requirements”, которые я получил в ООО “Программные технологии”.

Стандарт кодирования – это комплекс правил, применяемых при написании исходного кода на определенном языке программирования.

2.4.1 Элементы стиля кодирования

1. Наименования переменных, классов, структур, методов, исходных файлов, пространств имен и т.д.
2. Логическая структура кода.
   Как правило, имеются в виду отступы, значительно упрощающие читабельность кода.
3. Хороший объектный дизайн.
   Примером такого дизайна могут служить адекватные абстракции, отношения.
4. Надежность, устойчивость кода.
   Любые модификации в коде должны быть максимально легкими, не приводящими к изменению других его частей. Также программный код должен быть устойчив к различным исключениям, которые могут быть выброшены во время работы программы, и корректно их обрабатывать.
5. Комментарии.
   Пишутся разработчиком, чтобы в дальнейшем упростить понимание кода для него самого и других программистов. Комментариев не должно быть много – лучше всего комментировать неоднозначные, неочевидные или непонятные ситуации.
6. Размер модуля.
   Желательно, чтобы реализация каждого метода умещалась в пределах одного экрана или хотя бы страницы формата А4.

2.5 Изучение особенностей объектного дизайна несложных приложений

Проект разработан в стиле объектно-ориентированного программирования, была реализована иерархия классов.

3.1 Принципы объектного дизайна

1. При наличии множества классов, взаимодействующих между собой, целесообразно выстроить из них определенную иерархию.
2. В самом начале разработки следует реализовать программу, которая будет работоспособной. Только после этого можно вдаваться в детали и дорабатывать отдельные фрагменты кода.
3. Код нужно писать так, чтобы система была для клиента черным ящиком с необходимыми функциями, к которым он будет обращаться и которые будут ему понятны.
4. При создании класса или метода лучше выбирать такие имена, по которым сразу можно будет понять, для чего этот класс или метод предназначен.
5. Отсутствие обобщения в коде так же плохо, как и его избыток. Поэтому следует применять абстракции, инкапсуляцию и т.п. только там, где это имеет смысл.
6. Атомарность классов – признак хорошего кода. Классы должны быть максимально простыми. Следует избегать сложностей с классами (оператор switch, большое число полей или методов класса).
7. Количество параметров в любом методе не должно быть длинным, вследствие чего необходимо следить за их числом и при надобности разбивать сложный метод на несколько простых.
8. Базовые функции программы должны находиться в основном классе.

9. Интерфейс каждого класса необходимо разрабатывать методом “от простого к сложному” – чем проще интерфейс, тем очевиднее его полезность и заметнее серьезные ошибки в его структуре. Следует соблюдать особую осторожность при расширении интерфейсов.
10. Нельзя создавать слишком большие объекты, реализовывать весь функционал в пределах 2-3 объектов.
11. Пожалуй, самым важным критерием дизайна является сквозной контроль. Все ошибки, недочеты и нюансы проще увидеть человеку, который не писал проверяемый код.

2.6 Изучение инженерных методик кодирования и отладки на высокоуровневых языках

2.6.1 Этапы разработки ПО

Разработка ПО – процесс трудоемкий, включающий в себя много этапов. Среди них можно выделить самые главные:

1. ***Проектирование программной системы.***
   Основную часть этапа занимает работа с заказчиком, составление и формализация требований.
2. ***Разработка программной системы.***
   Этап предполагает создание программного продукта, удовлетворяющего всем предъявленным к нему требованиям.
3. ***Тестирование программной системы.***
   На данном этапе производится проверка, как отдельных модулей, так и всей системы в целом на наличие ошибок с последующим их исправлением до тех пор, пока не будут соблюдены некоторые условия (обычно они оговариваются командой тестировщиков или разработчиков).
4. ***Внедрение и сопровождение программной системы.***
   Этот этап включает в себя выпуск разработанного ПО, его установка на ЭВМ, обучение пользованию программой, а также гарантийное обслуживание ПО.

2.6.2 Отладка. Методы отладки

Отладка – это процесс поиска, выявления и исправления дефектов (как правило, до тех пор, пока стоимость исправления ошибки не сделает ее исправление невыгодным).

Методы отладки:

1. Внедрение в проект специальных логгеров. Логгирование – отличный способ отслеживать ошибки в программе, и самое главное, что этот метод является потоконезависимым.
2. Use-case diagrams – отладка программы по специальным сценариям.
3. Запуск программы в режиме отладки. В этом случае используется отладчик, интегрированный в среду разработки.

Разумеется, методов отладки превеликое множество. Вышеперечисленные методы показались мне наиболее интересными и эффективными.

2.7 Разработка проекта, его тестирование и верификация

2.7.1 Разработка

После изучения предметной области и закрепления теоретического материала, пришла очередь разработки проекта. Вначале были разработаны требования к программе.

В программе реализована многопоточность и критические секции.

В данной системе я использовал VK API для отправки Get запросов и парсил с помощью собственного метода.

Листинг программного кода приведен в приложении.

2.7.2 Тестирование

Для тестирования своей программы я использовал инструмент DUnit, который позволял производить автоматическое тестирование модулей. Также мною была применена техника программного обеспечения TDD (test-driven development) - которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам. Для отладки проекта использовался стандартный встроенный отладчик RAD Studio XE6, а также отладочный вывод.

2.7.3 Верификация

Верификацией называется проверка соответствия результатов отдельных этапов разработки программной системы требованиям и ограничениям, сформулированным для них на предыдущих этапах.

1. При авторизации с помощью официального приложения для Android пользователь должен ввести свои данные, если же он не вводит их и оставляет поля пустые, то выдать ошибку и поместить курсор снова на поле ввода. При данной проверки этот тест на всех системах отработал корректно.
2. При отправке запросов на сервера Вконтакте, но при медленном интернете было обнаружено некоторые задержки, но в целом так как программа многопоточна она справилась. Что и наглядно демонстрирует требование к наличию широкополосного доступа в интернет.

2.7.4 Тесты

Для проверки портируемости программы следует перенести проект на другую ЭВМ с любой версией ОС Windows. Успешные запуск и работа программы подтвердят, что она обладает хорошей переносимостью.

Конкретные входные-выходные данные для тестирования правильности работы программы приведены в таблице 1.

Входные данные	Выходные данные
Ответ с Get запроса в VK API методе user.get	Парсинг, а далее вывод в переменную, если же не удалось вывести, то ошибка в парсинге, если же удалось, то проверить что вывелось в переменную, если же это информация о конкретном пользователе, то отобразить на экране, если же нет, то ошибка.
Содержимое переменной	Проверить в какой кодировке находятся символы в переменной и конвертировать их в Unicode, а затем вывести на экран.
Post запрос в VK API при аутентификации	Проверить на наличие валидного access token и если он валиден то парсить, если же нет, то выдать ошибку авторизации

Листинг

```
unit About;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, IdTCPConnection,
  IdTCPClient, IdHTTP, IdBaseComponent, IdComponent, IdIOHandler,
  IdIOHandlerSocket, IdIOHandlerStack, IdSSL, IdSSLOpenSSL, Vcl.ExtCtrls,
  IdAntiFreezeBase, Vcl.IdAntiFreeze, UserGet, ActiveX,
  System.win.ScktComp, System.StrUtils, Winapi.shellapi, Vcl.Clipbrd,
  Winapi.WinInet;

type
  TAboutBox = class(TForm)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    OKButton: TButton;
    Label_source: TLabel;
    procedure OKButtonClick(Sender: TObject);
    procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Panel1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure CommentsMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure CopyrightMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Label_sourceMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ProductNameMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ProgramIconMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure VersionMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormKeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure Label_sourceMouseEnter(Sender: TObject);
```

```
    procedure Label_sourceMouseLeave(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

implementation

{$R *.dfm}

procedure TAboutBox.CommentsMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  AboutBox.Perform(WM_SysCommand, $F012, 0);
end;

procedure TAboutBox.CopyrightMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  AboutBox.Perform(WM_SysCommand, $F012, 0);
end;

procedure TAboutBox.FormKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_ESCAPE then
  begin
    AboutBox.Close;
  end;
end;

procedure TAboutBox.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  AboutBox.Perform(WM_SysCommand, $F012, 0);
end;

procedure TAboutBox.Label_sourceMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
```

```
begin
  ShellExecute(Handle, nil, PChar('https://github.com/akimdi/Litvinova'), nil,
    nil, SW_SHOW);
end;

procedure TAboutBox.Label_sourceMouseEnter(Sender: TObject);
begin
  AboutBox.Label_source.Font.Color := clred;
  AboutBox.Label_source.Cursor := crHandPoint;
end;

procedure TAboutBox.Label_sourceMouseLeave(Sender: TObject);
begin
  AboutBox.Label_source.Font.Color := clblue;
end;

procedure TAboutBox.OKButtonClick(Sender: TObject);
begin
  AboutBox.Close;
end;

procedure TAboutBox.Panel1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  AboutBox.Perform(WM_SysCommand, $F012, 0);
end;

procedure TAboutBox.ProductNameMouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  AboutBox.Perform(WM_SysCommand, $F012, 0);
end;

procedure TAboutBox.ProgramIconMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  AboutBox.Perform(WM_SysCommand, $F012, 0);
end;

procedure TAboutBox.VersionMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
```

```
  ReleaseCapture;
  AboutBox.Perform(WM_SysCommand, $F012, 0);
end;

end.
```

```
unit Auth;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, System.StrUtils,
  Authorization, SendMessage;

type
  TForm_Auth = class(TForm)
    Edit_login: TEdit;
    Edit_password: TEdit;
    Button_enter_auth: TButton;
    Label_1: TLabel;
    Button_no_enter_auth: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button_enter_authClick(Sender: TObject);
    procedure Button_no_enter_authClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Auth: TForm_Auth;

implementation

uses Main;

{$R *.dfm}

procedure TForm_Auth.Button_enter_authClick(Sender: TObject);
```

```
begin
  My_Auth_VK := TAuthorizationThread.Create(False);
  My_Auth_VK.Priority := tpLower;
  My_Auth_VK.FreeOnTerminate := True;
  Form_send_message.Show();
end;

procedure TForm_Auth.Button_no_enter_authClick(Sender: TObject);
begin
  Form_Auth.Close();
  Main_Form.Timer_2.Enabled := True;
  Main_Form.Timer_1.Enabled := False; // Beep()
end;

procedure TForm_Auth.FormCreate(Sender: TObject);
begin
  // Form_Auth.Edit_login.Clear();
  // Form_Auth.Edit_password.Clear();
end;

end.



unit Authorization;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, IdTCPConnection,
  IdTCPClient, IdHTTP, IdBaseComponent, IdComponent, IdIOHandler,
  IdIOHandlerSocket, IdIOHandlerStack, IdSSL, IdSSLOpenSSL, Vcl.ExtCtrls,
  ActiveX, System.win.ScktComp, System.StrUtils, System.DateUtils,
  UTC;

type
  TAuthorizationThread = class(TThread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
    procedure AuthorizationVK();
  end;
```

```
implementation

uses Main, Auth;

{ TAuthorizationThread }

procedure TAuthorizationThread.AuthorizationVK;
begin
  GetAuth := Main_Form.IdHTTP_1.Get

('https://oauth.vk.com/token?grant_type=password&scope=notify,friends,photos,audio,video,docs,notes,pages,status,wall,groups,messages,notifications,stats,ads,offline,email&client_id=2274003&client_secret=hHbZxrka2uZ6jB1inYsH&username='
    + Form_Auth.Edit_login.Text + '&password=' + Form_Auth.Edit_password.Text +
    '&v=5.21');
  GetParsAuth := Copy(GetAuth, 1, MaxInt);
  GetParsAuth := AnsiReplaceStr(GetParsAuth, '}', '');
  GetParsAuth := AnsiReplaceStr(GetParsAuth, '"', '');
  GetParsAuth := AnsiReplaceStr(GetParsAuth, '_', '');
  // ïàðñèíã AccessToken
  GetAccessToken := Copy(GetParsAuth, 1, MaxInt);
  Delete(GetAccessToken, 1, 12);
  Delete(GetAccessToken, AnsiPos(',', GetAccessToken), MaxInt);
  GetAccessToken := AnsiReplaceStr(GetAccessToken, ':', '');
  // ïàðñèíã UserId êîòîðîìó ïðèíàäëåæèò âûøåèçëîæåííûé AccessToken
  GetUserId := Copy(GetParsAuth, AnsiPos('userid:', GetParsAuth), MaxInt);
  Delete(GetUserId, 1, 7);
end;

procedure TAuthorizationThread.Execute;
begin
  { Place thread code here }
  Synchronize(AuthorizationVK);
end;

end.



unit About;
interface
uses
Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, IdTCPConnection,
IdTCPClient, IdHTTP, IdBaseComponent, IdComponent, IdIOHandler,
```

```
IdIOHandlerSocket, IdIOHandlerStack, IdSSL, IdSSLOpenSSL, Vcl.ExtCtrls,
IdAntiFreezeBase, Vcl.IdAntiFreeze, UserGet, ActiveX,
System.win.ScktComp, System.StrUtils, Winapi.shellapi, Vcl.Clipbrd,
Winapi.WinInet;
type
TAboutBox = class(TForm)
Panel1: TPanel;
ProgramIcon: TImage;
ProductName: TLabel;
Version: TLabel;
Copyright: TLabel;
Comments: TLabel;
OKButton: TButton;
Label_source: TLabel;
procedure OKButtonClick(Sender: TObject);
procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure Panel1MouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure CommentsMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure CopyrightMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure Label_sourceMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure ProductNameMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure ProgramIconMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure VersionMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure FormKeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
procedure Label_sourceMouseEnter(Sender: TObject);
procedure Label_sourceMouseLeave(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
AboutBox: TAboutBox;
implementation
{$R *.dfm}
procedure TAboutBox.CommentsMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
```

```
ReleaseCapture;
AboutBox.Perform(WM_SysCommand, $F012, 0);
end;
procedure TAboutBox.CopyrightMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
ReleaseCapture;
AboutBox.Perform(WM_SysCommand, $F012, 0);
end;
procedure TAboutBox.FormKeyUp(Sender: TObject; var Key: Word;
Shift: TShiftState);
begin
if Key = VK_ESCAPE then
begin
AboutBox.Close;
end;
end;
procedure TAboutBox.FormMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
ReleaseCapture;
AboutBox.Perform(WM_SysCommand, $F012, 0);
end;
procedure TAboutBox.Label_sourceMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
ShellExecute(Handle, nil, PChar('https://github.com/akimdi/Litvinova'), nil,
nil, SW_SHOW);
end;
procedure TAboutBox.Label_sourceMouseEnter(Sender: TObject);
begin
AboutBox.Label_source.Font.Color := clred;
AboutBox.Label_source.Cursor := crHandPoint;
end;
procedure TAboutBox.Label_sourceMouseLeave(Sender: TObject);
begin
AboutBox.Label_source.Font.Color := clblue;
end;
procedure TAboutBox.OKButtonClick(Sender: TObject);
begin
AboutBox.Close;
end;
procedure TAboutBox.Panel1MouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
ReleaseCapture;
```

```
AboutBox.Perform(WM_SysCommand, $F012, 0);
end;
procedure TAboutBox.ProductNameMouseDown(Sender: TObject; Button:
TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
ReleaseCapture;
AboutBox.Perform(WM_SysCommand, $F012, 0);
end;
procedure TAboutBox.ProgramIconMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
ReleaseCapture;
AboutBox.Perform(WM_SysCommand, $F012, 0);
end;
procedure TAboutBox.VersionMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
ReleaseCapture;
AboutBox.Perform(WM_SysCommand, $F012, 0);
end;
end.
```

```
unit Main;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, IdTCPConnection,
  IdTCPClient, IdHTTP, IdBaseComponent, IdComponent, IdIOHandler,
  IdIOHandlerSocket, IdIOHandlerStack, IdSSL, IdSSLOpenSSL, Vcl.ExtCtrls,
  IdAntiFreezeBase, Vcl.IdAntiFreeze, UserGet, ActiveX,
  System.win.ScktComp, System.StrUtils, Winapi.shellapi, Vcl.Clipbrd,
  Winapi.WinInet, System.DateUtils, UTC, UserTimer, ABOUT, Vcl.AppEvnts,
  System.SyncObjs, Auth, Authorization, SendMessage, MessageVK;

type
  TMain_Form = class(TForm)
    Edit_enter: TEdit;
    Button_start: TButton;
    Button_open_URL: TButton;
```

```
Button_paste: TButton;
Button_hideTray: TButton;
Timer_1: TTimer;
Timer_2: TTimer;
IdSSLIOHandlerSocketOpenSSL_1: TIdSSLIOHandlerSocketOpenSSL;
IdHTTP_1: TIdHTTP;
Button_help: TButton;
Label_1: TLabel;
Label_2: TLabel;
Label_3: TLabel;
Label_name: TLabel;
Label_surname: TLabel;
Label_on_off: TLabel;
Label_4: TLabel;
Button_off_sound: TButton;
Edit_temp: TEdit;
IdAntiFreeze1: TIdAntiFreeze;
TrayIcon1: TTrayIcon;
Memo_1: TMemo;
Label_5: TLabel;
Label_ID: TLabel;
Label_7: TLabel;
Label_sex: TLabel;
Label_6: TLabel;
Label_status: TLabel;
Label_8: TLabel;
Label_bdate: TLabel;
Label_9: TLabel;
Label_last_seen: TLabel;
Button_copy: TButton;
Button_about: TButton;
ApplicationEvents1: TApplicationEvents;
Button_clear: TButton;
Button_further: TButton;
procedure Button_startClick(Sender: TObject);
procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_4MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_nameMouseDown(Sender: TObject; Button: TMouseButton;
```

```
  Shift: TShiftState; X, Y: Integer);
procedure Label_surnameMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_on_offMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Timer_1Timer(Sender: TObject);
procedure Button_hideTrayClick(Sender: TObject);
procedure Button_open_URLClick(Sender: TObject);
procedure Button_pasteClick(Sender: TObject);
procedure GetMethodVK;
function ConnectedToInet: Boolean; // ïðîâåðêà ïîäêëþ÷åíèÿ ê èíòåðíåòó
procedure Timer_2Timer(Sender: TObject);
procedure TrayIcon1MouseMove(Sender: TObject; Shift: TShiftState;
  X, Y: Integer);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Button_off_soundClick(Sender: TObject);
procedure Label_5MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_IDMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_sexMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure FormKeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
procedure Button_startMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_statusMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_6MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_8MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_bdateMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_9MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Label_last_seenMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
function UnixTimeToDateTime(UnixTime: LongInt): String;
function DateTimeToUnixTime(DelphiDate: TDate): LongInt;
procedure FormCreate(Sender: TObject);
procedure Button_copyClick(Sender: TObject);
procedure Button_helpClick(Sender: TObject);
procedure TrayIcon1Click(Sender: TObject);
procedure Button_aboutClick(Sender: TObject);
```

```
    procedure Edit_enterEnter(Sender: TObject);
    procedure Edit_enterKeyPress(Sender: TObject; var Key: Char);
    procedure Button_clearClick(Sender: TObject);
    procedure FormMouseLeave(Sender: TObject);
    procedure ApplicationEvents1Activate(Sender: TObject);
    procedure Button_furtherClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Main_Form: TMain_Form;
  GetVKobj, GetParsObj: string;

  My_UserGet_VK: TUserGetThread;
  My_UserTimer_VK: TUserTimerThread;
  My_Auth_VK: TAuthorizationThread;
  My_Message_VK: TSendMessageThread;

  dwConnectionTypes: DWORD;
  Res_Time: TDateTime;

  GetParsObj_for_name, GetParsObj_for_surname, GetParsObj_for_sex,
    GetParsObj_for_online_offline, GetParsObj_for_ID, GetParsObj_for_status,
    GetParsObj_for_bdate, GetParsObj_for_last_seen: string;

  temp_name: string;
  temp_last_seen, temp_bdate, temp_status2: Integer;

  Position_deleted, Position_banned, Position_error: Integer;

  Check_ID_Pos_int: Integer;
  Check_ID_Pos_str: string;

  GetAuth, GetParsAuth, GetAccessToken, GetUserId: string;

const
  My_SecPerDay = 86400; // 86400 ñåêóíä â ñóòêàõ (24 ÷àñà)
  My_Offset1970 = 25569;
  My_four_hours = 14400; // 14400 ñåêóíä â 4 ÷àñàõ

implementation

{$R *.dfm}
```

```
procedure TMain_Form.ApplicationEvents1Activate(Sender: TObject);
begin
  Main_Form.Edit_enter.Clear();
  if Clipboard.HasFormat(CF_TEXT) then // åñëè ýòî òåêñò, òî...
  begin
    Edit_enter.Text := Clipboard.AsText;
  end;
  Main_Form.Edit_enter.SetFocus();
end;

procedure TMain_Form.Button_aboutClick(Sender: TObject);
begin
  AboutBox.ShowModal();
end;

procedure TMain_Form.Button_clearClick(Sender: TObject);
begin
  Main_Form.Label_name.Caption := '';
  Main_Form.Label_surname.Caption := '';
  Main_Form.Label_on_off.Caption := '';
  Main_Form.Label_ID.Caption := '';
  Main_Form.Label_sex.Caption := '';
  Main_Form.Label_status.Caption := '';
  Main_Form.Label_bdate.Caption := '';
  Main_Form.Label_last_seen.Caption := '';
  Edit_enter.Clear();
  Clipboard.Clear();
  Main_Form.Timer_1.Enabled := False;
  Main_Form.Timer_2.Enabled := False;
  // EmptyClipboard();
end;

procedure TMain_Form.Button_copyClick(Sender: TObject);
begin
  Edit_enter.SelectAll; // âûäåëèëè âñ¸
  Edit_enter.CopyToClipboard; // ñêîïèðîâàëè :)
end;

procedure TMain_Form.Button_furtherClick(Sender: TObject);
begin
  Form_Auth.Show();
end;

procedure TMain_Form.Button_helpClick(Sender: TObject);
begin
```

```
  // ïðàâèëà ââîäà !!!!!!!!!!
end;

procedure TMain_Form.Button_hideTrayClick(Sender: TObject);
begin
  TrayIcon1.Visible := True;
  Application.MainForm.Hide;
end;

procedure TMain_Form.Button_off_soundClick(Sender: TObject);
begin
  Timer_1.Enabled := False;
end;

procedure TMain_Form.Button_open_URLClick(Sender: TObject);
begin
  // ShellExecute(0, 'open', 'www.google.ru', nil, nil, SW_SHOW); ýòîò âàðèàíò ïîäõîäèò äëÿ ñòðîêîâûõ êîíñòàíò
  // ShellExecute(Handle, 'Open', PChar(VarToStr(Edit1.Text)), nil, nil,SW_SHOWNORMAL); åñòü åù¸ âîò òàê âîò
  ShellExecute(Handle, nil, PChar('https://vk.com/id' + GetParsObj_for_ID), nil,
    nil, SW_SHOW);
  // åñëè æå URL àäðåñ îòñóòñòâóåò â Edit2 òî ïðîèñõîäèò îòêðûòèå ñòðàíèöû https://vk.com/id ÷òî ïðèâîäèò ÂÊ ê îøèáêè
end;

procedure TMain_Form.Button_pasteClick(Sender: TObject);
begin
  // Main_Form.Edit_enter.Clear();
  // if Clipboard.HasFormat(CF_TEXT) then // åñëè ýòî òåêñò, òî...
  // begin
  // Edit_enter.Text := Clipboard.AsText;
  // end;
  // Main_Form.Edit_enter.SetFocus();
end;

procedure TMain_Form.Button_startClick(Sender: TObject);
var
  ask_int: Integer;
begin
  if Edit_enter.Text = '' then
  begin
    MessageBox(Handle, PChar('Âû íè÷åãî íå ââåëè :( '), PChar('Îøèáêà'),
      (MB_OK + MB_ICONERROR));
    Main_Form.Label_name.Caption := '';
    Main_Form.Label_surname.Caption := '';
```

```
  Main_Form.Label_on_off.Caption := '';
  Main_Form.Label_ID.Caption := '';
  Main_Form.Label_sex.Caption := '';
  Main_Form.Label_status.Caption := '';
  Main_Form.Label_bdate.Caption := '';
  Main_Form.Label_last_seen.Caption := '';
 end
 else
 begin
  Check_ID_Pos_int := AnsiPos('vk.com/', Main_Form.Edit_enter.Text);
  if not(Check_ID_Pos_int = 0) then // åñëè îí íàøåë ñîâïàäåíèå
  begin
   Check_ID_Pos_str := AnsiReplaceStr(Main_Form.Edit_enter.Text, ' ', '');
   Delete(Check_ID_Pos_str, 1, AnsiPos('vk.com/', Check_ID_Pos_str));
   Delete(Check_ID_Pos_str, 1, 6);
   ask_int := AnsiPos('?', Check_ID_Pos_str);
   if not(ask_int = 0) then // åñëè íàø¸ë çíàê "?"
   begin
    Delete(Check_ID_Pos_str, AnsiPos('?', Check_ID_Pos_str), MaxInt);
    Main_Form.Edit_temp.Text := Check_ID_Pos_str;
   end;
   Main_Form.Edit_temp.Text := Check_ID_Pos_str;
   if ConnectedToInet() then
   begin
    GetMethodVK();
   end
   else
   begin
    MessageBox(Handle, PChar('Ïðîâåðüòå ïîäêëþ÷åíèå ê èíòåðíåòó !'),
     PChar('Îòñóòñòâóåò ïîäêëþ÷åíèå'), (MB_OK + MB_ICONERROR));
   end;
  end
  else
  begin
   MessageBox(Handle, PChar('Íåâåðíûå äàííûå :('), PChar('Îøèáêà'),
    (MB_OK + MB_ICONERROR));
   Main_Form.Label_name.Caption := '';
   Main_Form.Label_surname.Caption := '';
   Main_Form.Label_on_off.Caption := '';
   Main_Form.Label_ID.Caption := '';
   Main_Form.Label_sex.Caption := '';
   Main_Form.Label_status.Caption := '';
   Main_Form.Label_bdate.Caption := '';
   Main_Form.Label_last_seen.Caption := '';
  end;
 end;
```

```
end;

procedure TMain_Form.Button_startMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  Main_Form.Button_start.Font.Style := [fsbold];
  Main_Form.Button_start.Font.Size := 15;
end;

function TMain_Form.ConnectedToInet: Boolean;
begin
  dwConnectionTypes := INTERNET_CONNECTION_MODEM +
INTERNET_CONNECTION_LAN +
    INTERNET_CONNECTION_PROXY;
  Result := InternetGetConnectedState(@dwConnectionTypes, 0);
end;

function TMain_Form.DateTimeToUnixTime(DelphiDate: TDate): LongInt;
begin
  Result := Trunc((DelphiDate - My_Offset1970) * My_SecPerDay);
end;

procedure TMain_Form.Edit_enterEnter(Sender: TObject);
{ ïðè ïîëó÷åíèè ôîêóñà }
begin
  // çàãëóøêà
end;

procedure TMain_Form.Edit_enterKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then
  begin
    Main_Form.Button_start.Font.Style := [fsbold];
    Main_Form.Button_start.Font.Size := 15;
    Main_Form.Button_startClick(Sender);
  end;
end;

procedure TMain_Form.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree; // îñâîáîæäåíèå ïàìÿòè ïðè çàêðûòèè ôîðìû
  Clipboard.Clear();
end;

procedure TMain_Form.FormCreate(Sender: TObject);
begin
```

```
  Clipboard.Clear();
  Main_Form.Label_name.Caption := '';
  Main_Form.Label_surname.Caption := '';
  Main_Form.Label_on_off.Caption := '';
  Main_Form.Label_ID.Caption := '';
  Main_Form.Label_sex.Caption := '';
  Main_Form.Label_status.Caption := '';
  Main_Form.Label_bdate.Caption := '';
  Main_Form.Label_last_seen.Caption := '';
end;

procedure TMain_Form.FormKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key = VK_ESCAPE then
  begin
    Main_Form.Close;
  end;
end;

procedure TMain_Form.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.FormMouseLeave(Sender: TObject);
begin
  if Main_Form.Focused = False then
  begin
    Main_Form.SetFocus();
  end;
end;

procedure TMain_Form.GetMethodVK;
begin
  My_UserGet_VK := TUserGetThread.Create(False);
  My_UserGet_VK.Priority := tpLowest;
  My_UserGet_VK.FreeOnTerminate := True;
end;

procedure TMain_Form.Label_1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
```

```
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_4MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_5MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_6MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_8MouseDown(Sender: TObject; Button: TMouseButton;
```

```
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_9MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_bdateMouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_IDMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_last_seenMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_nameMouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_on_offMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
```

```
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_sexMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_statusMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Label_surnameMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Main_Form.Perform(WM_SysCommand, $F012, 0);
end;

procedure TMain_Form.Timer_1Timer(Sender: TObject);
begin
  Beep();
  // MessageBeep(MM_MIDI_MAPPER);
end;

procedure TMain_Form.Timer_2Timer(Sender: TObject);
begin
  My_UserTimer_VK := TUserTimerThread.Create(False);
  My_UserTimer_VK.Priority := tpLower;
  My_UserTimer_VK.FreeOnTerminate := True;
end;

procedure TMain_Form.TrayIcon1Click(Sender: TObject);
begin
  Application.MainForm.Show;
end;

procedure TMain_Form.TrayIcon1MouseMove(Sender: TObject; Shift: TShiftState;
  X, Y: Integer);
begin
```

```
  // posit2 := AnsiPos('online":0', str_temp2);
  // if posit2 = 0 then
  // // åñëè îí íå íàøåë ñîâïàäåíèå òî...
  // begin
  // TrayIcon_1.Hint := str_name + ' ' + str_surname + ' â ñåòè  :)';
  // end
  // else
  // begin
  // TrayIcon_1.Hint := str_name + ' ' + str_surname + ' íå â ñåòè :(';
  // end;
end;

function TMain_Form.UnixTimeToDateTime(UnixTime: Integer): String;
begin
  Result := DateTimeToStr(UnixTime / My_SecPerDay + My_Offset1970);
end;

end.
```

```
unit MessageVK;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, IdTCPConnection,
  IdTCPClient, IdHTTP, IdBaseComponent, IdComponent, IdIOHandler,
  IdIOHandlerSocket, IdIOHandlerStack, IdSSL, IdSSLOpenSSL, SendMessage;

type
  TSendMessageThread = class(TThread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
    procedure SendMessageVK();
  end;

implementation

uses Main, UserGet, Authorization;
```

```
{ TSendMessageThread }

procedure TSendMessageThread.Execute;
begin
  { Place thread code here }
  Synchronize(SendMessageVK);
end;

procedure TSendMessageThread.SendMessageVK;
var
  temp: string;
  temp2: AnsiString;
begin
  temp2 := Form_send_message.Memo_1.Lines.Text;
  temp := Main_Form.IdHTTP_1.Get
    ('https://api.vk.com/method/messages.send?user_id=' + GetParsObj_for_ID +
    '&message=' + temp2 + '&v=5.21&access_token=' + GetAccessToken);
  MessageBox(Form_send_message.Handle, PChar('Ñîîáùåíèå îòïðàâëåíî'),
    PChar('Èíôîðìàöèÿ'), (MB_OK + MB_ICONINFORMATION));
end;

end.




unit SendMessage;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type
  TForm_send_message = class(TForm)
    Button_send_message: TButton;
    Memo_1: TMemo;
    procedure Button_send_messageClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form_send_message: TForm_send_message;

implementation

uses Main, MessageVK;

{$R *.dfm}

procedure TForm_send_message.Button_send_messageClick(Sender: TObject);
begin
  // ñîçäà¸ì ïîòîê
  My_Message_VK := TSendMessageThread.Create(False);
  My_Message_VK.Priority := tpLower;
  My_Message_VK.FreeOnTerminate := True;
end;

end.


unit UTC;

interface

// Ïðåîáðàçîâàíèå âðåìåíè UTC ê ëîêàëüíîìó âðåìåíè è îáðàòíî
// ñ ó÷¸òîì Windows-íàñòðîåê ëîêàëüíîãî GMT-ñìåùåíèÿ è ïðàâèëàìè ïåðåõîäà íà ëåòíåå âðåìÿ
function UTCToLocalTime(AValue: TDateTime): TDateTime;
function LocalTimeToUTC(AValue: TDateTime): TDateTime;

implementation

uses
  SysUtils, Windows;

function UTCToLocalTime(AValue: TDateTime): TDateTime;
// AValue - âðåìÿ UTC
// Result - âðåìÿ ñ ó÷¸òîì ëîêàëüíîãî GMT-ñìåùåíèÿ è ïðàâèëàìè ïåðåõîäà íà ëåòíåå âðåìÿ
var
  ST1, ST2: TSystemTime;
  TZ: TTimeZoneInformation;
begin
  // TZ - ëîêàëüíûå íàñòðîéêè Windows
  GetTimeZoneInformation(TZ);

  // Ïðåîáðàçîâàíèå TDateTime ê WindowsSystemTime
  DateTimeToSystemTime(AValue, ST1);
```

```
  // Ïðèìåíåíèå ëîêàëüíûõ íàñòðîåê êî âðåìåíè
  SystemTimeToTzSpecificLocalTime(@TZ, ST1, ST2);

  // Ïðåîáðàçóåì SystemTime ê TDateTime
  Result := SystemTimeToDateTime(ST2);
end;

function LocalTimeToUTC(AValue: TDateTime): TDateTime;
// AValue - ëîêàëüíîå âðåìÿ
// Result - âðåìÿ UTC
var
  ST1, ST2: TSystemTime;
  TZ: TTimeZoneInformation;
begin
  // TZ - ëîêàëüíûå (Windows) íàñòðîéêè
  GetTimeZoneInformation(TZ);
  // ò.ê. íàäî áóäåò äåëàòü îáðàòíûå ïðåîáðàçîâàíèå - èíâåðòèðóåì bias
  TZ.Bias := -TZ.Bias;
  TZ.StandardBias := -TZ.StandardBias;
  TZ.DaylightBias := -TZ.DaylightBias;

  DateTimeToSystemTime(AValue, ST1);

  // Ïðèìåíåíèå ëîêàëüíûõ íàñòðîåê êî âðåìåíè
  SystemTimeToTzSpecificLocalTime(@TZ, ST1, ST2);

  // Ïðåîáðàçóåì WindowsSystemTime ê TDateTime
  Result := SystemTimeToDateTime(ST2);
end;

end.
```

```
unit UserGet;

interface

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

```
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, IdTCPConnection,
  IdTCPClient, IdHTTP, IdBaseComponent, IdComponent, IdIOHandler,
  IdIOHandlerSocket, IdIOHandlerStack, IdSSL, IdSSLOpenSSL, Vcl.ExtCtrls,
  ActiveX, System.win.ScktComp, System.StrUtils, System.DateUtils,
  UTC, Auth;

type
  TUserGetThread = class(TThread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
    procedure GetVK;
  end;

implementation

uses Main;

{ TAuthThread }

procedure TUserGetThread.Execute;
begin
  Synchronize(GetVK);
end;

procedure TUserGetThread.GetVK;
var
  i: Integer; // счётчик
  temp_data_time: TDateTime;
begin
  {
https://api.vk.com/method/users.get?user_ids=id1&fields=sex,id,first_name,last_name,bdate,city
,photo_max_orig,deactivated,lists,blacklisted,verified,site,home_town,domain,followers_count,n
ickname,occupation,personal,relatives,relation,home_phone,counters,mobile_phone,universities,
university_name,faculty,graduation,faculty_name,schools,can_post,connections,exports,status,co
untry,online,online_mobile,online_app,has_mobile,can_see_all_posts,can_see_audio,movies,ga
mes,about,can_post,quotes,books,tv,timezone,can_write_private_message,activities,interests,mu
sic,wall_comments,contacts,education,last_seen&name_case=nom&v=5.21&access_token
    в методе users.get поле user_ids это список строк, разделенных через запятую,
    количество элементов которых должно составлять не более 1000 }
  GetVKobj := Main_Form.IdHTTP_1.Get
    ('https://api.vk.com/method/users.get?user_ids=' + Check_ID_Pos_str +
    '&fields=sex,id,first_name,last_name,bdate,' +
    'deactivated,lists,blacklisted,verified,home_town' +
    ',domain,last_seen,nickname,' + 'counters,' +
```

```
  'status,online,online_mobile,online_app,' + 'has_mobile,' +
  'timezone,activities,' + '&name_case=nom&v=5.21&access_token');
// копирование результата
GetParsObj := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_name := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_status := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_last_seen := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_bdate := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_surname := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_sex := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_online_offline := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_ID := Copy(GetVKobj, 1, MaxInt);
// вывод JSON ответа в Memo что бы было визуально и наглядно
// (в релиз версии убрать)
Main_Form.Memo_1.Lines.Text := GetParsObj;
Position_error := AnsiPos('error', GetParsObj);
Position_deleted := AnsiPos('"deactivated":"deleted"', GetParsObj);
Position_banned := AnsiPos('"deactivated":"banned"', GetParsObj);
if (Position_error <> 0) then // если нашел ошибку (error), то...
begin
  MessageBox(Main_Form.Handle, PChar('Неверные данные !'), PChar('Ошибка'),
    (MB_OK + MB_ICONERROR));
  Main_Form.Label_name.Caption := '';
  Main_Form.Label_surname.Caption := '';
  Main_Form.Label_on_off.Caption := '';
  Main_Form.Label_ID.Caption := '';
  Main_Form.Label_sex.Caption := '';
  Main_Form.Label_status.Caption := '';
  Main_Form.Label_bdate.Caption := '';
  Main_Form.Label_last_seen.Caption := '';
end
else
begin
  if ((Position_banned <> 0) or (Position_deleted <> 0)) then
  // если нашёл или deleted или banned, то...
  begin
    { начало парсинга }
    // парсим имя
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '{', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '}', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, ']', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '"', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, ':', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '_', '');
    temp_name := Copy(GetParsObj_for_name,
      AnsiPos('firstname', GetParsObj_for_name),
```

```
  AnsiPos(',last', GetParsObj_for_name));
Delete(temp_name, 1, 9);
Delete(temp_name, AnsiPos(',', temp_name), MaxInt);
// выводим имя
Main_Form.Label_name.Caption := temp_name;
// парсим фамилию
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '{', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '}', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, ']', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '"', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, ':', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '_', '');
Delete(GetParsObj_for_surname, 1, 8);
GetParsObj_for_surname := Copy(GetParsObj_for_surname,
  AnsiPos('lastname', GetParsObj_for_surname), MaxInt);
Delete(GetParsObj_for_surname, 1, 8);
Delete(GetParsObj_for_surname,
  AnsiPos(',', GetParsObj_for_surname), MaxInt);
// выводим фамилию
Main_Form.Label_surname.Caption := GetParsObj_for_surname;
// парсим ID
Delete(GetParsObj_for_ID, 1, 19);
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '{', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '}', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, ']', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '"', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, ':', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '_', '');
for i := length(GetParsObj_for_ID) downto 1 do
  if not(CharInSet(GetParsObj_for_ID[i], ['0' .. '9'])) then
  begin
    Delete(GetParsObj_for_ID, i, MaxInt);
  end;
// выводим ID
Main_Form.Label_ID.Caption := GetParsObj_for_ID;
// парсим пол
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '{', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '}', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, ']', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '"', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, ':', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '_', '');
GetParsObj_for_sex := Copy(GetParsObj_for_sex,
  AnsiPos('sex', GetParsObj_for_sex), MaxInt);
Delete(GetParsObj_for_sex, 6, MaxInt);
GetParsObj_for_sex := Copy(GetParsObj_for_sex, 4, 1);
```

```
// выводим пол
if (GetParsObj_for_sex = '0') then
begin
  Main_Form.Label_sex.Caption := 'не указан';
  Main_Form.Label_sex.Font.Color := clRed;
  Main_Form.Label_sex.Font.Size := 12;
end;
if (GetParsObj_for_sex = '1') then
begin
  Main_Form.Label_sex.Caption := 'женский';
end;
if (GetParsObj_for_sex = '2') then
begin
  Main_Form.Label_sex.Caption := 'мужской';
end;
// парсим онлайн
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '{', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '}', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, ']', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '"', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, ':', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '_', '');
Delete(GetParsObj_for_online_offline, 1, 8);
GetParsObj_for_online_offline := Copy(GetParsObj_for_online_offline,
  AnsiPos('online', GetParsObj_for_online_offline), MaxInt);
GetParsObj_for_online_offline :=
  Copy(GetParsObj_for_online_offline, 7, 1);
// выводим онлайн
if GetParsObj_for_online_offline = '0' then
begin
  Main_Form.Label_on_off.Font.Color := clRed;
  Main_Form.Label_on_off.Font.Size := 12;
  Main_Form.Label_on_off.Caption := 'не в сети';
  Main_Form.Timer_2.Enabled := True;
  Main_Form.Timer_1.Enabled := False; // Beep()
  Main_Form.Label_status.Caption := '';
  Main_Form.Label_bdate.Caption := '';
  Main_Form.Label_last_seen.Caption := '';
  MessageBox(Main_Form.Handle,
    PChar(temp_name + ' забанен или его страница удалена !' + #13 +
```

```
        'Но как только пользователь восстановит свою страницу или его разбанят, сразу
будет подана серия звуковых сигналов.'),
          PChar('Предупреждение'), (MB_OK + MB_ICONWARNING));
        end
        else
        begin
          Main_Form.Label_on_off.Font.Color := clGreen;
          Main_Form.Label_on_off.Font.Size := 12;
          Main_Form.Label_on_off.Caption := 'в сети';
          MessageBox(Main_Form.Handle, PChar(temp_name + ' сейчас Вконтакте :)' +
            #13 + 'Можете что-нибудь написать...'), PChar('Информация'),
            (MB_OK + MB_ICONINFORMATION));
          Main_Form.Timer_2.Enabled := False;
          Main_Form.Timer_1.Enabled := True; // Beep()
        end;
        { конец парсинга }
      end
      else
      begin
        { НАЧАЛО ОСНОВНОГО ТЕЛА }
        // парсим имя
        GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '{', '');
        GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '}', '');
        GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, ']', '');
        GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '"', '');
        GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, ':', '');
        GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '_', '');
        temp_name := Copy(GetParsObj_for_name,
          AnsiPos('firstname', GetParsObj_for_name),
          AnsiPos(',last', GetParsObj_for_name));
        Delete(temp_name, 1, 9);
        Delete(temp_name, AnsiPos(',', temp_name), MaxInt);
        // выводим имя
        Main_Form.Label_name.Caption := temp_name;
        // парсим ID
        Delete(GetParsObj_for_ID, 1, 19);
        GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '{', '');
        GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '}', '');
        GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, ']', '');
        GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '"', '');
        GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, ':', '');
        GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '_', '');
        for i := length(GetParsObj_for_ID) downto 1 do
          if not(CharInSet(GetParsObj_for_ID[i], ['0' .. '9'])) then
          begin
            Delete(GetParsObj_for_ID, i, MaxInt);
```

```
  end;
// выводим ID
Main_Form.Label_ID.Caption := GetParsObj_for_ID;
// парсим фамилию
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '{', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '}', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, ']', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '"', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, ':', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '_', '');
Delete(GetParsObj_for_surname, 1, 8);
GetParsObj_for_surname := Copy(GetParsObj_for_surname,
  AnsiPos('lastname', GetParsObj_for_surname), MaxInt);
Delete(GetParsObj_for_surname, 1, 8);
Delete(GetParsObj_for_surname,
  AnsiPos(',', GetParsObj_for_surname), MaxInt);
// выводим фамилию
Main_Form.Label_surname.Caption := GetParsObj_for_surname;
// парсим пол
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '{', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '}', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, ']', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '"', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, ':', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '_', '');
GetParsObj_for_sex := Copy(GetParsObj_for_sex,
  AnsiPos('sex', GetParsObj_for_sex), MaxInt);
Delete(GetParsObj_for_sex, 6, MaxInt);
GetParsObj_for_sex := Copy(GetParsObj_for_sex, 4, 1);
// выводим пол
if (GetParsObj_for_sex = '0') then
begin
  Main_Form.Label_sex.Caption := 'не указан';
  Main_Form.Label_sex.Font.Color := clRed;
  Main_Form.Label_sex.Font.Size := 12;
end;
if (GetParsObj_for_sex = '1') then
begin
  Main_Form.Label_sex.Caption := 'женский';
end;
if (GetParsObj_for_sex = '2') then
begin
  Main_Form.Label_sex.Caption := 'мужской';
end;
// парсим статус пользователя
GetParsObj_for_status := Copy(GetParsObj_for_status,
```

```
  AnsiPos('status', GetParsObj_for_status), MaxInt);
Delete(GetParsObj_for_status, 1, 6);
Delete(GetParsObj_for_status, 1, 3);
GetParsObj_for_status := Copy(GetParsObj_for_status, 1,
  AnsiPos('","last_seen":{"time":', GetParsObj_for_status));
temp_status2 := length(GetParsObj_for_status);
temp_status2 := temp_status2 - 1;
GetParsObj_for_status := Copy(GetParsObj_for_status, 1, temp_status2);
GetParsObj_for_status := AnsiReplaceStr(GetParsObj_for_status, '\/', '/');
GetParsObj_for_status := AnsiReplaceStr(GetParsObj_for_status, '\', '');
// выводим статус пользователя
Main_Form.Label_status.Caption := GetParsObj_for_status;
// парсим день рождения пользователя
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, '{', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, '}', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, ']', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, '"', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, ':', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, '_', '');
temp_bdate := AnsiPos('bdate', GetParsObj_for_bdate);
if temp_bdate = 0 then // если он не нашел, то...
begin
  Main_Form.Label_bdate.Caption := 'не указан';
  // Main_Form.Label_bdate.Font.Color := clRed;
  Main_Form.Label_bdate.Font.Size := 12;
end
else
begin
  Delete(GetParsObj_for_bdate, 1, AnsiPos('bdate', GetParsObj_for_bdate));
  Delete(GetParsObj_for_bdate, 1, 4);
  GetParsObj_for_bdate := Copy(GetParsObj_for_bdate, 1,
    AnsiPos(',', GetParsObj_for_bdate));
  GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, ',', '');
  // выводим день рождения пользователя
  Main_Form.Label_bdate.Caption := GetParsObj_for_bdate;
end;
// парсим время когда последний раз заходил пользователь
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, '{', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, '}', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, ']', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, '"', '');
GetParsObj_for_last_seen :=
```

```
  AnsiReplaceStr(GetParsObj_for_last_seen, ':', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, '_', '');
Delete(GetParsObj_for_last_seen, 1, AnsiPos('lastseentime',
  GetParsObj_for_last_seen));
Delete(GetParsObj_for_last_seen, 1, 11);
GetParsObj_for_last_seen := Copy(GetParsObj_for_last_seen, 1,
  AnsiPos(',', GetParsObj_for_last_seen));
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, ',', '');
// конвертируем значение из String в Integer
temp_last_seen := StrToInt(GetParsObj_for_last_seen);
// конвертируем из Unix Time в Windows Time функцией UnixTimeToDateTime
GetParsObj_for_last_seen := Main_Form.UnixTimeToDateTime(temp_last_seen);
// конвертируем полученное значение в формате Windows Time типа String в тип DateTime, функцией StrToDateTime
temp_data_time := StrToDateTime(GetParsObj_for_last_seen);
// переводим часовой пояс из среднего времени по Гринвичу (GMT или UTC±0 => в данном случае UTC±0 совпадает с GMT)
// в локальный часовой пояс (в данном случае это Московское время => MSK (UTC+4))
GetParsObj_for_last_seen :=
  DateTimeToStr(UTC.UTCToLocalTime(temp_data_time));
// выводим время когда последний раз заходил пользователь
Main_Form.Label_last_seen.Caption := GetParsObj_for_last_seen;
// парсим онлайн
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '{', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '}', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, ']', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '"', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, ':', '');
GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '_', '');
Delete(GetParsObj_for_online_offline, 1, 8);
GetParsObj_for_online_offline := Copy(GetParsObj_for_online_offline,
  AnsiPos('online', GetParsObj_for_online_offline), MaxInt);
GetParsObj_for_online_offline :=
  Copy(GetParsObj_for_online_offline, 7, 1);
// выводим онлайн
if GetParsObj_for_online_offline = '0' then
begin
```

```
        Main_Form.Label_on_off.Font.Color := clRed;
        Main_Form.Label_on_off.Font.Size := 12;
        Main_Form.Label_on_off.Caption := 'не в сети';
        Main_Form.Timer_2.Enabled := True;
        Main_Form.Timer_1.Enabled := False;
        MessageBox(Main_Form.Handle,
          PChar(temp_name + ' сейчас не в сети.' + #13 +
          'Как только пользователь появится в сети' + #13 +
          'будет подана серия звуковых сигналов.'), PChar('Информация'),
          (MB_OK + MB_ICONINFORMATION));
      end
      else
      begin
        Main_Form.Label_on_off.Font.Color := clGreen;
        Main_Form.Label_on_off.Font.Size := 12;
        Main_Form.Label_on_off.Caption := 'в сети';
        MessageBox(Main_Form.Handle, PChar(temp_name + ' сейчас Вконтакте :)' +
          #13 + 'Можете что-нибудь написать...'), PChar('Информация'),
          (MB_OK + MB_ICONINFORMATION));
        Main_Form.Timer_2.Enabled := False;
        Main_Form.Timer_1.Enabled := True; // Beep()
        Main_Form.Button_further.Visible := True;
      end;
      { КОНЕЦ ОСНОВНОГО ТЕЛА }
    end;
  end;
end;

end.
```

```
unit UserTimer;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, IdTCPConnection,
  IdTCPClient, IdHTTP, IdBaseComponent, IdComponent, IdIOHandler,
  IdIOHandlerSocket, IdIOHandlerStack, IdSSL, IdSSLOpenSSL, Vcl.ExtCtrls,
  ActiveX, System.win.ScktComp, System.StrUtils, System.DateUtils,
  UTC, Auth;
```

```
type
  TUserTimerThread = class(TThread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
    procedure TimerVK;
  end;

implementation

uses Main;

{ TUserTimer }

procedure TUserTimerThread.Execute;
begin
  Synchronize(TimerVK);
end;

procedure TUserTimerThread.TimerVK;
var
  i: Integer; // ñ÷¸ò÷èê
  temp_data_time: TDateTime;
begin
  {
https://api.vk.com/method/users.get?user_ids=id1&fields=sex,id,first_name,last_name,bdate,city
,photo_max_orig,deactivated,lists,blacklisted,verified,site,home_town,domain,followers_count,n
ickname,occupation,personal,relatives,relation,home_phone,counters,mobile_phone,universities,
university_name,faculty,graduation,faculty_name,schools,can_post,connections,exports,status,co
untry,online,online_mobile,online_app,has_mobile,can_see_all_posts,can_see_audio,movies,ga
mes,about,can_post,quotes,books,tv,timezone,can_write_private_message,activities,interests,mu
sic,wall_comments,contacts,education,last_seen&name_case=nom&v=5.21&access_token
    â ìåòîää users.get ïîëå user_ids ýòî ñïèñîê ñòðîê, ðàçäåëåííûõ ÷åðåç çàïÿòóþ,
    êîëè÷åñòâî ýëåìåíòîâ êîòîðûõ äîëæíî ñîîòâåòñòâîâàòü íå áîëåå 1000 }
  GetVKobj := Main_Form.IdHTTP_1.Get
    ('https://api.vk.com/method/users.get?user_ids=' + Check_ID_Pos_str +
    '&fields=sex,id,first_name,last_name,bdate,' +
    'deactivated,lists,blacklisted,verified,home_town' +
    ',domain,last_seen,nickname,' + 'counters,' +
    'status,online,online_mobile,online_app,' + 'has_mobile,' +
    'timezone,activities,' + '&name_case=nom&v=5.21&access_token');
  // êîïèðîâàíèå ðåçóëüòàòà
  GetParsObj := Copy(GetVKobj, 1, MaxInt);
  GetParsObj_for_name := Copy(GetVKobj, 1, MaxInt);
```

```
GetParsObj_for_status := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_last_seen := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_bdate := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_surname := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_sex := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_online_offline := Copy(GetVKobj, 1, MaxInt);
GetParsObj_for_ID := Copy(GetVKobj, 1, MaxInt);
// âûâîä JSON îòâåòà â Memo ÷òî áû áûëî âèçóàëüíî è íàãëÿäíî
// (â ðåëèç âåðñèè óáðàòü)
Main_Form.Memo_1.Lines.Text := GetParsObj;
Position_error := AnsiPos('error', GetParsObj);
Position_deleted := AnsiPos('"deactivated":"deleted"', GetParsObj);
Position_banned := AnsiPos('"deactivated":"banned"', GetParsObj);
if (Position_error <> 0) then // åñëè íàøåë îøèáêó (error), òî...
begin
  // MessageBox(Main_Form.Handle, PChar('Íåâåðíûå äàííûå !'), PChar('Îøèáêà'),
  // (MB_OK + MB_ICONERROR));
  Main_Form.Label_name.Caption := 'Íåâåðíûå äàííûå !';
  Main_Form.Label_name.Font.Color := clRed;
  Main_Form.Label_surname.Caption := '';
  Main_Form.Label_on_off.Caption := '';
  Main_Form.Label_ID.Caption := '';
  Main_Form.Label_sex.Caption := '';
  Main_Form.Label_status.Caption := '';
  Main_Form.Label_bdate.Caption := '';
  Main_Form.Label_last_seen.Caption := '';
end
else
begin
  if ((Position_banned <> 0) or (Position_deleted <> 0)) then
  // åñëè íàø¸ë èëè deleted èëè banned, òî...
  begin
    { íà÷àëî ïàðñèíãà }
    // ïàðñèì èìÿ
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '{', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '}', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, ']', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '"', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, ':', '');
    GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '_', '');
    temp_name := Copy(GetParsObj_for_name,
      AnsiPos('firstname', GetParsObj_for_name),
      AnsiPos(',last', GetParsObj_for_name));
    Delete(temp_name, 1, 9);
    Delete(temp_name, AnsiPos(',', temp_name), MaxInt);
    // âûâîäèì èìÿ
```

```
Main_Form.Label_name.Caption := temp_name;
// ïàðñèì ôàìèëèþ
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '{', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '}', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, ']', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '"', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, ':', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '_', '');
Delete(GetParsObj_for_surname, 1, 8);
GetParsObj_for_surname := Copy(GetParsObj_for_surname,
  AnsiPos('lastname', GetParsObj_for_surname), MaxInt);
Delete(GetParsObj_for_surname, 1, 8);
Delete(GetParsObj_for_surname,
  AnsiPos(',', GetParsObj_for_surname), MaxInt);
// âûâîäèì ôàìèëèþ
Main_Form.Label_surname.Caption := GetParsObj_for_surname;
// ïàðñèì ID
Delete(GetParsObj_for_ID, 1, 19);
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '{', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '}', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, ']', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '"', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, ':', '');
GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '_', '');
for i := length(GetParsObj_for_ID) downto 1 do
  if not(CharInSet(GetParsObj_for_ID[i], ['0' .. '9'])) then
  begin
    Delete(GetParsObj_for_ID, i, MaxInt);
  end;
// âûâîäèì ID
Main_Form.Label_ID.Caption := GetParsObj_for_ID;
// ïàðñèì ïîë
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '{', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '}', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, ']', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '"', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, ':', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '_', '');
GetParsObj_for_sex := Copy(GetParsObj_for_sex,
  AnsiPos('sex', GetParsObj_for_sex), MaxInt);
Delete(GetParsObj_for_sex, 6, MaxInt);
GetParsObj_for_sex := Copy(GetParsObj_for_sex, 4, 1);
// âûâîäèì ïîë
if (GetParsObj_for_sex = '0') then
begin
  Main_Form.Label_sex.Caption := 'íå óêàçàí';
```

```
  Main_Form.Label_sex.Font.Color := clRed;
  Main_Form.Label_sex.Font.Size := 12;
 end;
 if (GetParsObj_for_sex = '1') then
 begin
  Main_Form.Label_sex.Caption := 'æåíñêèé';
 end;
 if (GetParsObj_for_sex = '2') then
 begin
  Main_Form.Label_sex.Caption := 'ìóæñêîé';
 end;
 // ïàðñèì îíëàéí
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '{', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '}', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, ']', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '"', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, ':', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '_', '');
 Delete(GetParsObj_for_online_offline, 1, 8);
 GetParsObj_for_online_offline := Copy(GetParsObj_for_online_offline,
  AnsiPos('online', GetParsObj_for_online_offline), MaxInt);
 GetParsObj_for_online_offline :=
  Copy(GetParsObj_for_online_offline, 7, 1);
 // âûâîäèì îíëàéí
 if GetParsObj_for_online_offline = '0' then
 begin
  Main_Form.Label_on_off.Font.Color := clRed;
  Main_Form.Label_on_off.Font.Size := 12;
  Main_Form.Label_on_off.Caption := 'çàáàíåí èëè åãî ñòðàíèöà óäàëåíà !';
  // Main_Form.Timer_2.Enabled := True;
  // Main_Form.Timer_1.Enabled := False;
  Main_Form.Label_status.Caption := '';
  Main_Form.Label_bdate.Caption := '';
  Main_Form.Label_last_seen.Caption := '';
  // MessageBox(Main_Form.Handle,
  // PChar(temp_name + ' çàáàíåí èëè åãî ñòðàíèöà óäàëåíà !' + #13 +
  // 'Íî êàê òîëüêî ïîëüçîâàòåëü âîññòàíîâèò ñâîþ ñòðàíèöó èëè åãî ðàçáàíèò, ñðàçó áóäåò
ïîäàíà ñåðèÿ çâóêîâûõ ñèãíàëîâ.'),
  // PChar('Ïðåäóïðåæäåíèå'), (MB_OK + MB_ICONWARNING));
 end
```

```
  else
  begin
    // Main_Form.Timer_2.Enabled := False;
    // Main_Form.Timer_1.Enabled := False;
    Main_Form.Label_on_off.Font.Color := clGreen;
    Main_Form.Label_on_off.Font.Size := 12;
    Main_Form.Label_on_off.Caption := 'â ñåòè';
    // MessageBox(Main_Form.Handle, PChar(temp_name + ' ñåé÷àñ Âêîíòàêòå :)' +
    // #13 + 'Ìîæåòå ÷òî-íèáóäü íàïèñàòü...'), PChar('Èíôîðìàöèÿ'),
    // (MB_OK + MB_ICONINFORMATION));
  end;
  { êîíåö ïàðñèíãà }
end
else
begin
  { ÍÀ×ÀËÎ ÎÑÍÎÂÍÎÃÎ ÒÅËÀ }
  // ïàðñèì èìÿ
  GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '{', '');
  GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '}', '');
  GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, ']', '');
  GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '"', '');
  GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, ':', '');
  GetParsObj_for_name := AnsiReplaceStr(GetParsObj_for_name, '_', '');
  temp_name := Copy(GetParsObj_for_name,
    AnsiPos('firstname', GetParsObj_for_name),
    AnsiPos(',last', GetParsObj_for_name));
  Delete(temp_name, 1, 9);
  Delete(temp_name, AnsiPos(',', temp_name), MaxInt);
  // âûâîäèì èìÿ
  Main_Form.Label_name.Caption := temp_name;
  // ïàðñèì ID
  Delete(GetParsObj_for_ID, 1, 19);
  GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '{', '');
  GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '}', '');
  GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, ']', '');
  GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '"', '');
  GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, ':', '');
  GetParsObj_for_ID := AnsiReplaceStr(GetParsObj_for_ID, '_', '');
  for i := length(GetParsObj_for_ID) downto 1 do
    if not(CharInSet(GetParsObj_for_ID[i], ['0' .. '9'])) then
    begin
      Delete(GetParsObj_for_ID, i, MaxInt);
    end;
  // âûâîäèì ID
  Main_Form.Label_ID.Caption := GetParsObj_for_ID;
  // ïàðñèì ôàìèëèþ
```

```
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '{', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '}', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, ']', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '"', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, ':', '');
GetParsObj_for_surname := AnsiReplaceStr(GetParsObj_for_surname, '_', '');
Delete(GetParsObj_for_surname, 1, 8);
GetParsObj_for_surname := Copy(GetParsObj_for_surname,
  AnsiPos('lastname', GetParsObj_for_surname), MaxInt);
Delete(GetParsObj_for_surname, 1, 8);
Delete(GetParsObj_for_surname,
  AnsiPos(',', GetParsObj_for_surname), MaxInt);
// âûâîäèì ôàìèëèþ
Main_Form.Label_surname.Caption := GetParsObj_for_surname;
// ïàðñèì ïîë
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '{', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '}', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, ']', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '"', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, ':', '');
GetParsObj_for_sex := AnsiReplaceStr(GetParsObj_for_sex, '_', '');
GetParsObj_for_sex := Copy(GetParsObj_for_sex,
  AnsiPos('sex', GetParsObj_for_sex), MaxInt);
Delete(GetParsObj_for_sex, 6, MaxInt);
GetParsObj_for_sex := Copy(GetParsObj_for_sex, 4, 1);
// âûâîäèì ïîë
if (GetParsObj_for_sex = '0') then
begin
  Main_Form.Label_sex.Caption := 'íå óêàçàí';
  Main_Form.Label_sex.Font.Color := clRed;
  Main_Form.Label_sex.Font.Size := 12;
end;
if (GetParsObj_for_sex = '1') then
begin
  Main_Form.Label_sex.Caption := 'æåíñêèé';
end;
if (GetParsObj_for_sex = '2') then
begin
  Main_Form.Label_sex.Caption := 'ìóæñêîé';
end;
// ïàðñèì ñòàòóñ ïîëüçîâàòåëÿ
GetParsObj_for_status := Copy(GetParsObj_for_status,
  AnsiPos('status', GetParsObj_for_status), MaxInt);
Delete(GetParsObj_for_status, 1, 6);
Delete(GetParsObj_for_status, 1, 3);
GetParsObj_for_status := Copy(GetParsObj_for_status, 1,
```

```
  AnsiPos('","last_seen":{"time":', GetParsObj_for_status));
temp_status2 := length(GetParsObj_for_status);
temp_status2 := temp_status2 - 1;
GetParsObj_for_status := Copy(GetParsObj_for_status, 1, temp_status2);
GetParsObj_for_status := AnsiReplaceStr(GetParsObj_for_status, '\/', '/');
GetParsObj_for_status := AnsiReplaceStr(GetParsObj_for_status, '\', '');
// âûâîäèì ñòàòóñ ïîëüçîâàòåëÿ
Main_Form.Label_status.Caption := GetParsObj_for_status;
// ïàðñèì äåíü ðîæäåíèÿ ïîëüçîâàòåëÿ
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, '{', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, '}', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, ']', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, '"', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, ':', '');
GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, '_', '');
temp_bdate := AnsiPos('bdate', GetParsObj_for_bdate);
if temp_bdate = 0 then // åñëè îí íå íàøåë, òî...
begin
  Main_Form.Label_bdate.Caption := 'íå óêàçàí';
  // Main_Form.Label_bdate.Font.Color := clRed;
  Main_Form.Label_bdate.Font.Size := 12;
end
else
begin
  Delete(GetParsObj_for_bdate, 1, AnsiPos('bdate', GetParsObj_for_bdate));
  Delete(GetParsObj_for_bdate, 1, 4);
  GetParsObj_for_bdate := Copy(GetParsObj_for_bdate, 1,
    AnsiPos(',', GetParsObj_for_bdate));
  GetParsObj_for_bdate := AnsiReplaceStr(GetParsObj_for_bdate, ',', '');
  // âûâîäèì äåíü ðîæäåíèÿ ïîëüçîâàòåëÿ
  Main_Form.Label_bdate.Caption := GetParsObj_for_bdate;
end;
// ïàðñèì âðåìÿ êîãäà ïîñëåäíèé ðàç çàõîäèë ïîëüçîâàòåëü
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, '{', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, '}', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, ']', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, '"', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, ':', '');
GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, '_', '');
Delete(GetParsObj_for_last_seen, 1, AnsiPos('lastseentime',
```

```
  GetParsObj_for_last_seen));
 Delete(GetParsObj_for_last_seen, 1, 11);
 GetParsObj_for_last_seen := Copy(GetParsObj_for_last_seen, 1,
  AnsiPos(',', GetParsObj_for_last_seen));
 GetParsObj_for_last_seen :=
  AnsiReplaceStr(GetParsObj_for_last_seen, ',', '');
 // êîíâåðòèðóåì çíà÷åíèå èç String â Integer
 temp_last_seen := StrToInt(GetParsObj_for_last_seen);
 // êîíâåðòèðóåì èç Unix Time â Windows Time ôóíêöèåé UnixTimeToDateTime
 GetParsObj_for_last_seen := Main_Form.UnixTimeToDateTime(temp_last_seen);
 // êîíâåðòèðóåì ïîëó÷åííîå çíà÷åíèå â ôîðìàòå Windows Time òèïà String â òèï DateTime,
ôóíêöèåé StrToDateTime
 temp_data_time := StrToDateTime(GetParsObj_for_last_seen);
 // ïåðåâîäèì ÷àñîâîé ïîÿñ èç ñðåäíåãî âðåìåíè ïî Ãðèíâè÷ó (GMT èëè UTC±0 => â äàííîì
ñëó÷àå UTC±0 ñîâïàäàåò ñ GMT)
 // â ëîêàëüíûé ÷àñîâîé ïîÿñ (â äàííîì ñëó÷àå ýòî Ìîñêîâñêîå âðåìÿ => MSK (UTC+4))
 GetParsObj_for_last_seen :=
  DateTimeToStr(UTC.UTCToLocalTime(temp_data_time));
 // âûâîäèì âðåìÿ êîãäà ïîñëåäíèé ðàç çàõîäèë ïîëüçîâàòåëü
 Main_Form.Label_last_seen.Caption := GetParsObj_for_last_seen;
 // ïàðñèì îíëàéí
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '{', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '}', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, ']', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '"', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, ':', '');
 GetParsObj_for_online_offline :=
  AnsiReplaceStr(GetParsObj_for_online_offline, '_', '');
 Delete(GetParsObj_for_online_offline, 1, 8);
 GetParsObj_for_online_offline := Copy(GetParsObj_for_online_offline,
  AnsiPos('online', GetParsObj_for_online_offline), MaxInt);
 GetParsObj_for_online_offline :=
  Copy(GetParsObj_for_online_offline, 7, 1);
 // âûâîäèì îíëàéí
 if GetParsObj_for_online_offline = '0' then
 begin
  Main_Form.Label_on_off.Font.Color := clRed;
  Main_Form.Label_on_off.Font.Size := 12;
  Main_Form.Label_on_off.Caption := 'íå â ñåòè';
 end
 else
```

```
      begin
        Main_Form.Timer_2.Enabled := False;
        Main_Form.Timer_1.Enabled := True; // Beep()
        Main_Form.Label_on_off.Font.Color := clGreen;
        Main_Form.Label_on_off.Font.Size := 12;
        Main_Form.Label_on_off.Caption := 'â ñåòè';
        Main_Form.Button_further.Visible := True;
      end;
      { ÊÎÍÅÖ ÎÑÍÎÂÍÎÃÎ ÒÅËÀ }
    end;
  end;
end;

end.
```

Заключение

В ходе практики мною были изучены и закреплены особенности стиля кодирования, принципы объектного дизайна, инженерные методики кодирования и отладки. При создании проекта я расширил свои знания о ЯП Delphi, а также получил опыт в разработке и тестировании программного обеспечения. Полученные знания я буду применять в своей будущем.

Список использованных источников

1 Положение о практиках студентов Южного федерального университета: принято на Ученом совете Южного федерально университета 27 апреля 2013 года.
2 Coding Requirements, Confidential. © LLC “Software Technologies”, 2009.
3 https://vk.com/dev/main - официальная документация Вконтакте.