**СОДЕРЖАНИЕ**

Введение 5

1 Аналитический обзор литературы и существующих аналогов 6

1.1 Аналитический обзор литературы 6

1.2 Обзор аналогов 6

1.3 Постановка задачи 8

2 Разработка алгоритма 9

2.1 Добавление ветки диалоги или действия 9

2.2 Возможность открывать приложения . 10

3 Разработка структуры данных 12

3.1 Интерфейс программного средства 12

3.2 Структура программы 13

4 Обоснование технических приемов программировании 15

4.1 Обоснование технических приемов в коде 15

4.2 Обоснование технических компонентов на форме 15

5 Тестирование, анализ полученных данных 16

5.1 Общие ошибки 16

5.2 Работоспособность компонентов 16

6 Руководство пользователя программы 18

6.1 Инструкция по использованию 18

Заключение 21

Список использованных источников 22

Приложение А. Исходный код программы 23

**Введение**

Виртуальный ассистент – приложение, которое может выполнять задачи для пользователя на основе введенной пользователем информации, его местоположения, информации, полученной из интернет-ресурсов, а также его интересов.

На данный момент большинство ассистентов ориентированы преимущественно на мобильные устройства, а существующие ассистенты для ПК или ноутбуков не удовлетворяют потребностям русскоязычных пользователей. К примеру, у ассистента Cortana от Microsoft отсутствует русская локализация, что очень сильно затрудняет процесс взаимодействия с ней, а это, можно считать, единственный ассистент для ПК и ноутбуков под управлением Windows. С мобильными ассистентами дела обстоят немного лучше, там уже есть из чего выбрать пользователю, но, к сожалению, они не адаптированы для пользования вне мобильных устройств.

Задача ассистента достаточно понятна для любого современного человека – это помощь в некоторых повседневных задачах, небольшое общение с ним и получение необходимой повседневной информации (новости, погода и т.д.).

Главной проблемой является то, что большие компании в нынешний момент в первую очередь ориентированы на телефоны, ведь они есть у всех и их продажи выше, чем у компьютеров и ноутбуков. А если и появляется ассистент для компьютеров, то на нём в большинстве случаев нет специфического функционала, что требует обычный пользователь из стран СНГ.

Целью данной курсовой работы является создание виртуального ассистента для ПК и ноутбуков с русской локализацией и возможностью добавлять личные команды либо ответы на вопросы, тем самым улучшая своего личного виртуального ассистента.

В первом разделе будет рассмотрена требуемая литература для написания курсовой работы. Также будут рассмотрены аналоги программы и поставлена задача по написанию курсовой работы с описанием её преимуществ.

1. **Анализ прототипов, литературных источников и формирование требований к проектиромуему программному средству**

**1.1 Аналитический обзор литературы**

При поиске информации по работе с динамическими структурами данных широко использовалось методическое пособие Серебряной Л.В «Структуры и алгоритмы обработки данных».

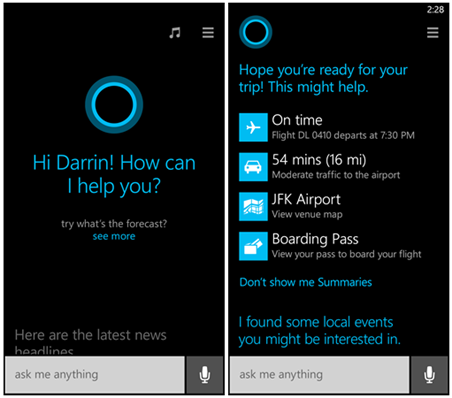
В этом пособии содержится вся необходимая информация о формировании одно- и двусвязных списков, вставке и удалении из них элементов. Материал сопровождается примерами кода и картинками, которые описывают принцип взаимодействия.

Так же широко использовались такие интернет-ресурсы как: Cyberforum.ru и delphi-manual.ru. На этих сайтах была найдена вся необходимая информация по взаимодействию с Windows-формами их настройке и взаимодействию с ними.

Материал на этих ресурсах излагается другими программистами, поэтому он, в большинстве своем, понятен.

**1.2 Обзор аналогов**

Самыми популярными аналогами моего программного средства (далее ПС) являются: Сortana, Google Assistant и Алиса от Яндекс. За основу была взята Cortana, так как она разработана для платформы Windows и, соответственно, первой была проанализирована Cortana.

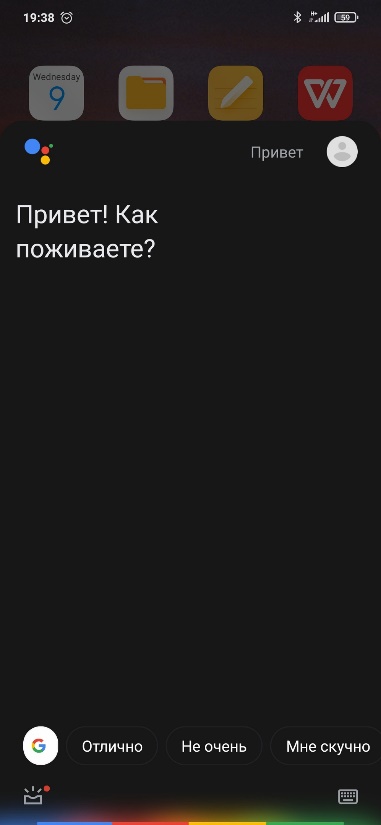


**Рисунок 1.1 – Интерфейс программы “Cortana” от компании Microsoft**

Достоинства: хороший ассистент для Windows

Недостатки: присутствует только на ОС Windows, отсутствие русской локализации, не доступен в странах СНГ.

Из конкурентов есть ещё Google Assistant. Приложение, созданное преимущественно для карманных устройств (смартфонов). Дизайн немного отличается от Cortana, но функционал примерно одинаковый.

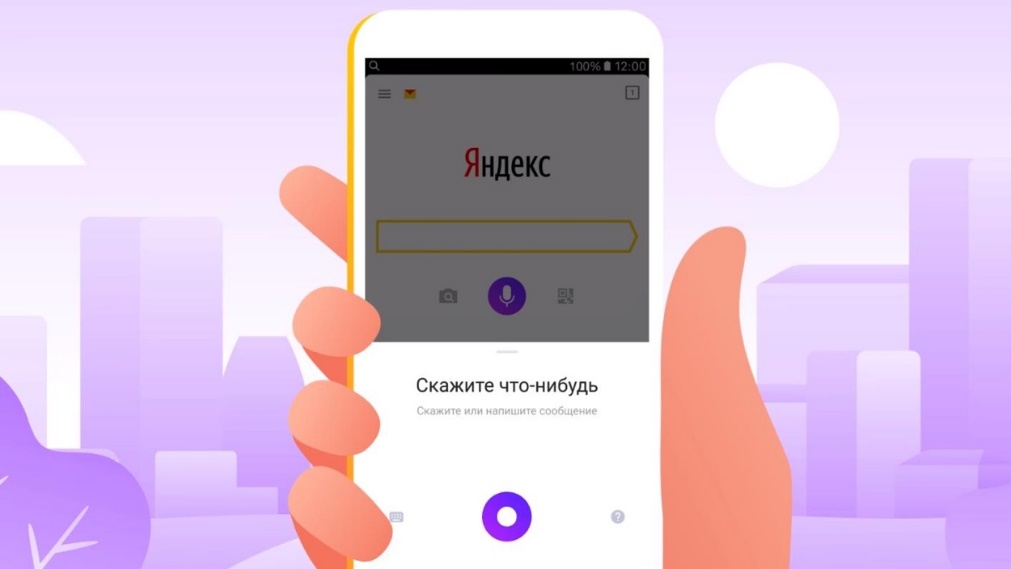


**Рисунок 1.2 – Интерфейс программы Google Assistant от Google**

Достоинства: присутствует голосовой ввод, поиск по интернету, озвучка ответа ассистента.

Недостатки: преимущественно для смартфонов

Также есть голосовой ассистент Алиса от компании Яндекс. Функционал и дизайн примерно схож с Google Assistant. Как и ассистент от Google, Алиса разрабатывалась в первую очередь для смартфонов. Также компания Яндекс производит колонки для дома с уже встроенным ассистентом.



**Рисунок 1.3 – Интерфейс программы Алиса от Яндекс**

Преимущества: присутствует голосовой ввод, поиск по интернету, озвучка ответов ассистента, присутствует продукция в виде колонок с встроенным ассистентом

Недостатки: преимущественно для смартфонов

**1.3 Постановка задачи**

В рамках данной курсовой работы было принято решение разработать программное средство “Виртуальный ассистент”, имя которому – Eve.

После рассмотрения аналогов, было принято решение сделать ассистента, который сможет взаимодействовать с пользователем, отвечать ему, помогать работать с текстовыми редакторами, запускать игры и искать в интернете запросы пользователя.

В программном средстве планируется реализовать следующие возможности:

— Возможность ведение диалога с ассистентом;

— Создание новых диалогов, которые захочет пользователь;

— Возможность запускать текстовые редакторы и игры;

— Озвучка виртуального ассистента;

Для разработки программного средства будет использоваться язык программирования Delphi и среда разработки RAD Studio 10.3.

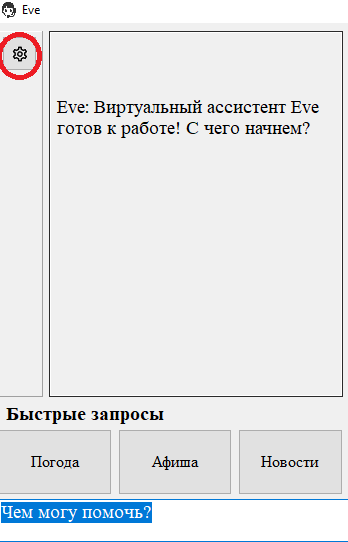
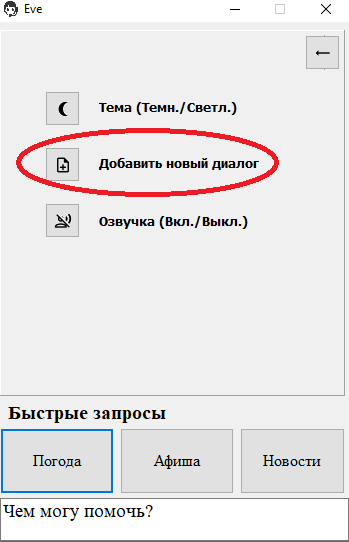
**2. Разработка алгоритма**

При разработке было принято решение использовать в качестве динамической структуры данных списки.

**2.1.Добавление ветки диалога или действия.**

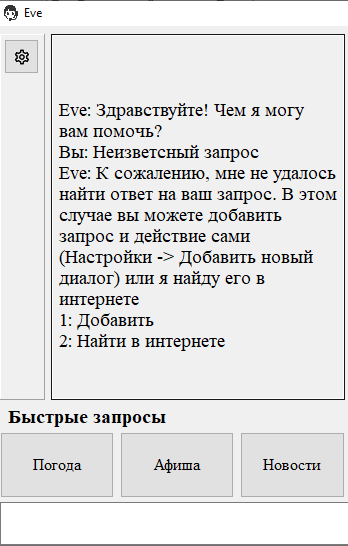
Добавление новой диалоговой ветки или действия производится при помощи приложения, которое пользователь может вызвать следующими путями:

1. Открыть меню и вызвать соответствующий пункт

** **

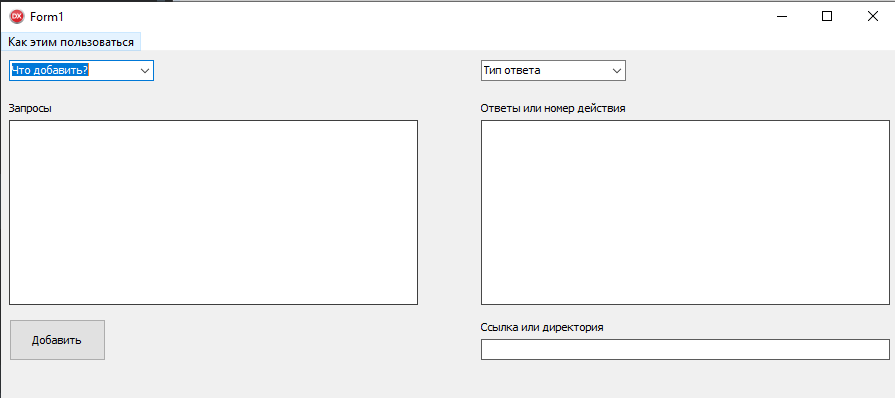
**Рис. 2.1 – первый способ добавления новой ветки диалога**

2. Написать запрос или действие, которого нет в списке, после чего Eve предложит следующие действия:

****

**Рис 2.2 – второй способ добавления новой ветки диалога**

При вводе “1”, “Добавить” или “Добавь” будет открыто следующее приложение:

****

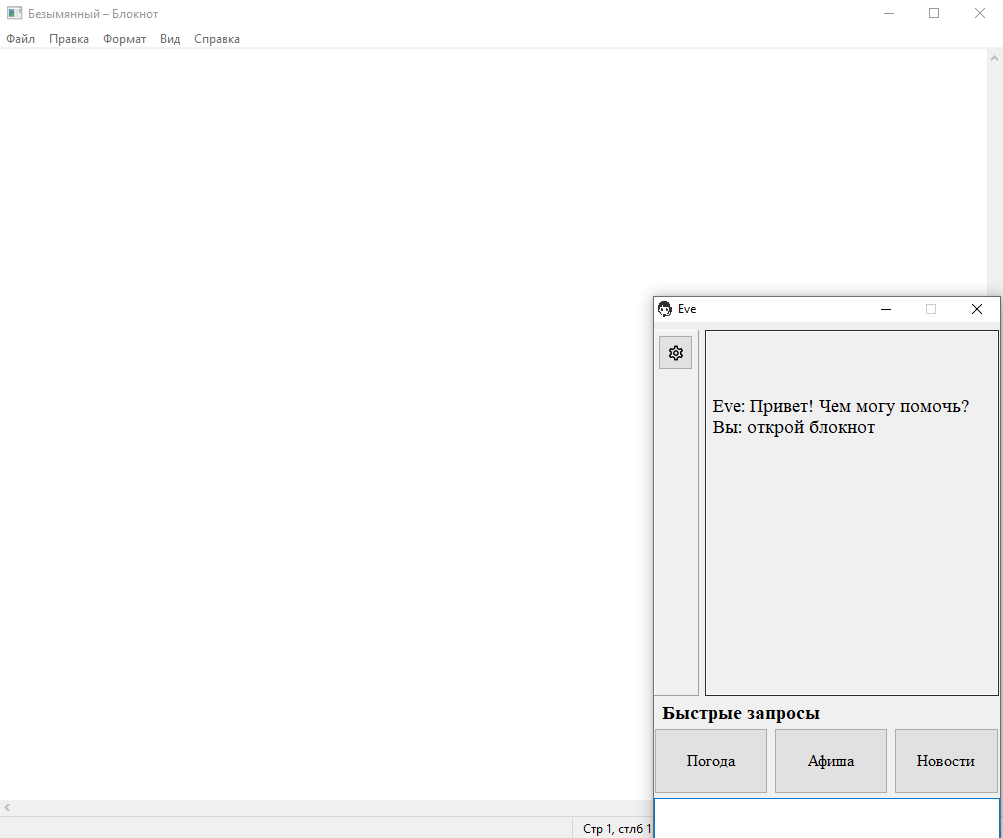
**Рис. 2.3 – Приложение для добавления новой диалоговой ветки**

При вводе “2”, “Найди” или “Поиск” ваш запрос будет найден в интернете.

**2.2 Возможность открывать приложения**

Для реализации этого пункта была использована стандартная библиотека ShellAPI.

Если попросить ассистента открыть приложение Блокнот, то ассистент успешно выполнит эту задачу:

****

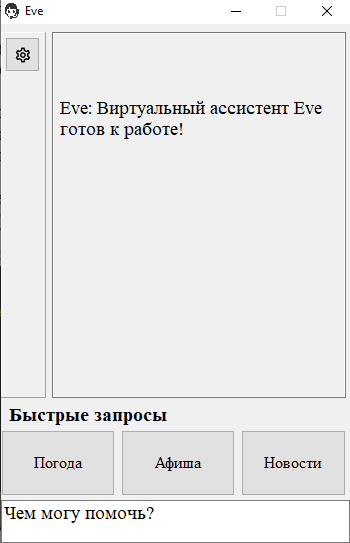
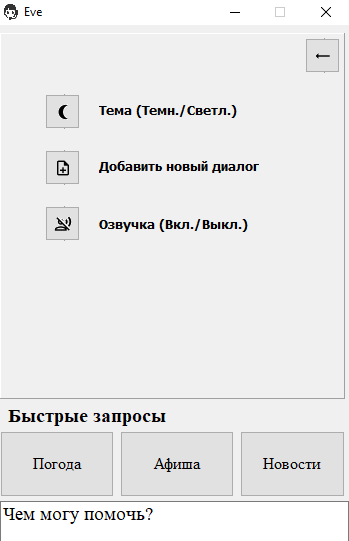
**Рис. 2.4 – пример выполнения действия ассистентом**

**3. Разработка программного средства**

**3.1 Интерфейс программного средства**

В ходе разработки данного ПС были использованы следующие формы:

-MainForm – главная форма, на которой располагается диалоговое окно и панель с меню.

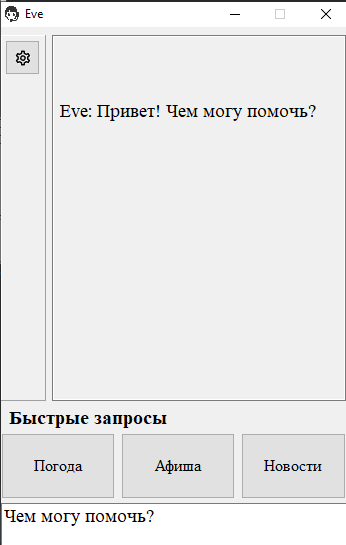
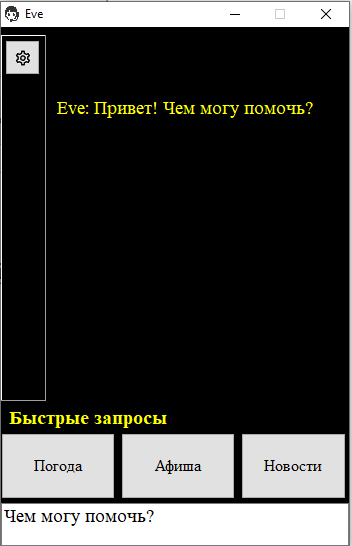
**Рис. 3.1 – Главная форма**

На форме располагаются следующие компоненты: TEdit – поле, в которую пользователь вводит свой запрос. TButton – кнопки быстрых запросов, TMemo – компонент, в котором отображается диалог.

SettingsPanel (TPanel), на которой располагаются кнопки TBitBtn взаимодействия с меню и Tlabel, описывающие пункты меню.

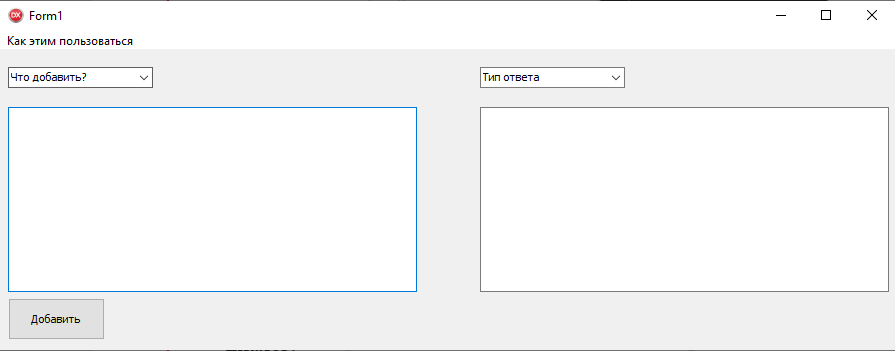
Для создания анимации появления меню настроек был использован TTimer. Для изменения темы приложения используется TColorDialogue.

Для озвучки ассистента используется TSpVoice.

** **

**Рис. 3.2 – Светлая и темная тема интерфейса**

-AddDialogue – форма, которая позволяет пользователю добавить свой диалог.

****

**Рис. 3.3 – форма для добавления диалога**

На форме используются 2 TMemo, в которые пользователь должен ввести запрос и ответ через специальный символ-разделитель “ ┐”, 2 TComboBox, в которых нужно выбрать нужные типы добавляемой информации, TEdit, в который пользователь должен ввести ссылку или директорию на ресурс и кнопка добавления TButton.

**3.2 Структура программы**

Для хранения данных используется типизированный файл типа TUsefulInfo (Brains.txt), который является записью и имеет следующий вид:

TUsefulData = record

Req: String[255];

case TypeOfReq: boolean of

true: (Ans: String[255]);

false: (

NumberOfAction: integer;

Link: ShortString

);

end;

Поле Req – поле хранящее в себе запросы одного типа (прим.: привет┐здравствуй┐здарова┐ и т.п.);

Поле TypeOfReq – поле, определяющее действие, которое должен выполнить ассистент. Если оно установлено в состояние True, то Eve ответит на ваш запрос фразой, если в False, то выполнит действие в зависимости от значения поля NumberOfAction. Поле Link хранит в себе ссылку на интернет-ресурс или директорию файла на компьютере.

При запуске программы, в памяти компьютера создается динамический список, который хранит в себе информацию, загруженную из файла Brains.txt.

Ptr = ^TElement;

TElement = record

Next, prev: Ptr;

Data: TUsefulData;

end;

1. Next – указатель на следующий элемент списка;
2. Prev - указатель на предыдущий элемент списка;
3. Data – поле, хранящее в себе всю информацию о запросе и ответе;

TList = record

head, tail: Ptr;

end;

1. Head – указатель на первый элемент списка;
2. Tail – указатель на последний элемент списка;

При запуске программы, в диалоговом окне пользователь видит приветственную фразу, которая загружается из типизированного файла Greetings.txt типа TGreeting, который является записью.

TGreeting = record

Greeting: ShortString;

end;

Greeting – поле, содержащее в себе все приветственные фразы.

Если пользователь попросит ассистента рассказать шутку, то она загрузится из типизированного файла Jokes.txt типа TJoke, который является записью.

TJoke = record

Joke: ShortString;

end;

Joke – поле, в котором записана шутка.

**4. Обоснования технических приемов программирования**

**4.1 Обоснование технических приемов в коде**

В данной программе были использованы такой прием, как использование процедур и функций. Процедура — это независимая именованная часть программы, которую после однократного описания можно многократно вызвать по имени из последующих частей программы для выполнения определенных действий. Функция – это фрагмент программного кода, к которому можно обратиться из другого места программы. В большинстве случаев с функцией связывается идентификатор, но многие языки допускают и безымянные функции. Использование функций и процедур облегчает код, упрощает его понимание. Это связано с тем, что процедура и функция пишется один раз, а далее идет вызывание этих функций в других подпрограммах, что облегчает код и уменьшает его.

Динамический список — это интерфейсный объект встроенного языка, который используется для отображения различных списков объектов базы данных, в конкретном случае записей в типизированном файле. Преимуществом динамического списка является то, что не нужно каждый раз обращаться к файлу и создавать большое множество переменных, так как динамический список позволяет хранить всю нужную информацию в динамической памяти компьютера. Так же динамические списки легко изменять, то есть удалять не нужные элементы и добавлять новые.

**4.2 Обоснование технических компонентов на форме**

Компоненты – это более совершенные объекты. С ними можно работать визуально, 1 для этого у них есть необходимые свойства и методы.

При разработке данного ПО было решено осуществить вывод диалога с помощью компонента TMemo, так как он поддерживает многострочность, что увеличивает удобство ведения диалога. На формах находятся такие компоненты, как: TBitBtn, TButton, TEdit, TLabel, TPanel, TComboBox, TColorDialog, TMenu, TTimer. Данные компоненты помогли сделать удобный и красивый интерфейс, который прост и лаконичен в использовании и понимании.

Использование двух видов кнопок обосновывается тем, что в TBitBtn можно вставить пиктограммы, которые делают интерфейс максимально простым.

**5. Тестирование**

**5.1 Общие ошибки**

В ходе тестирования приложения было выявлено, что при поиске нужного запроса в списке, происходило создание нового списка, в следствие чего в поле данных элемента списка хранился мусор. Было принято решение сделать список глобальным и изменять его только при запуске приложения или добавлении новой записи через специальное приложение.

**5.2 Работоспособность компонентов**

У пользователя есть возможность создавать свои действия и диалоги.

Изначально пользователь должен был ввести запрос на действие и номер действия, а затем добавить в коде процедуру ShellExecute. В этот момент тип TUsefulData приложение добавления имели следующий вид:

TUsefulData = record

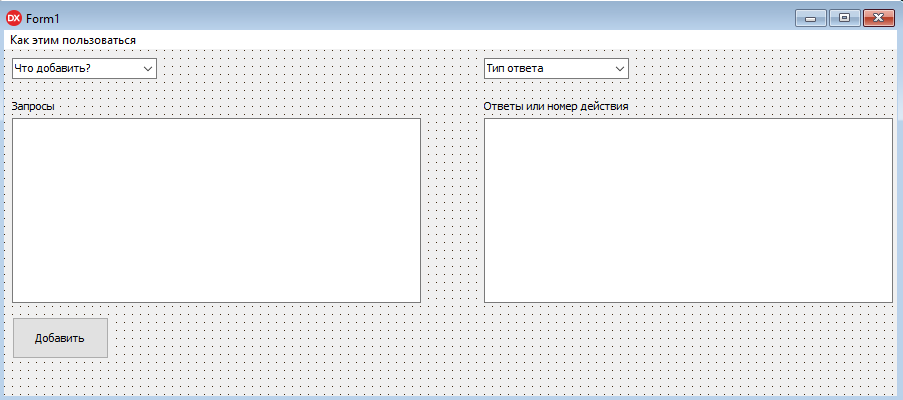
Req: String[255];

case TypeOfReq: boolean of

true: (Ans: String[255]);

false: ( NumberOfAction: integer );

end;



**Рис. 5.1 – изначальный вид приложения**

Затем было принято решение о переработки данного приложения с целью облегчения добавления записей.

В тип TUsefulData было добавлено новое поле Link, которое хранит в себе ссылку или директорию. Новый вид TUsefulData и нтерфейса прилжения:

TUsefulData = record

Req: String[255];

Case TypeOfReq: boolean of

true: (Ans: String[255]);

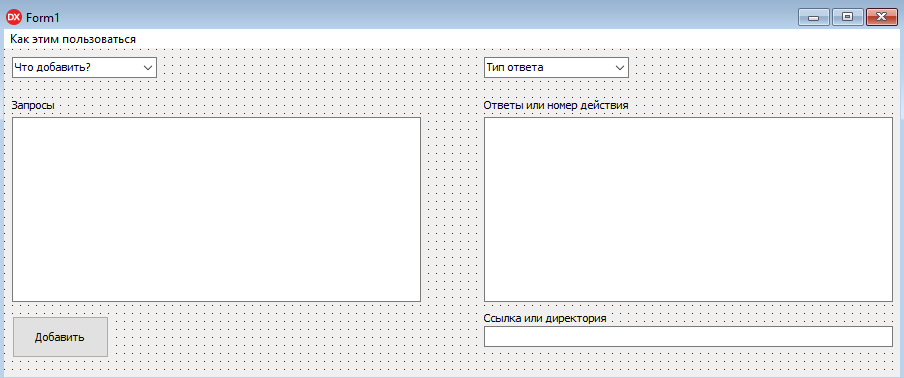
false: (

NumberOfAction: integer;

Link: ShortString

);

end;



**Рис. 5.2 – финальный вид приложения для добавления записей**

Теперь, для того чтобы пользователь мог добавить действие, ему достаточно знать ссылку на интернет-ресурс или директорию файла.

Также, в ходе тестирования было выявлено, что после добавления записи в файл, необходимо перезагрузить приложение виртуального ассистента, чтобы он смог работать с этим запросом.

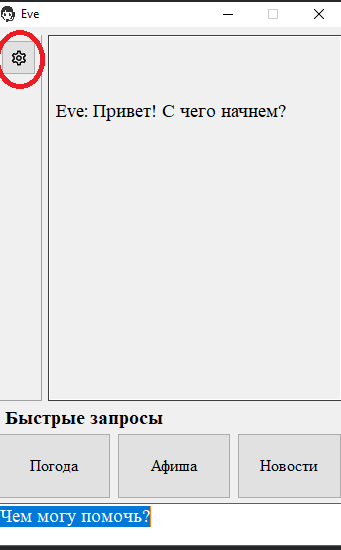
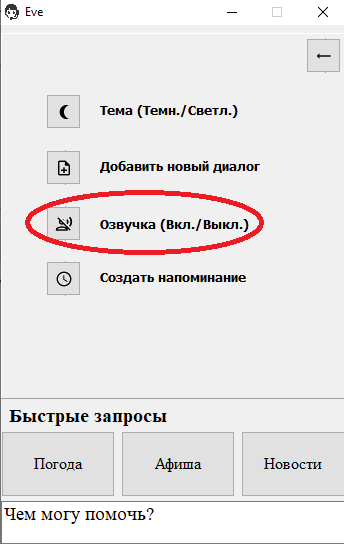
Это было исправлено путем создания флага OpenFlag ( переменной типа Boolean), который устанавливается в True при вызове приложения для добавления записи и при следующем вводе запроса проверяется, если он установлен в True, то динамический список пересоздастся и будет включать в себя новую запись из файла.

**6. Руководство пользователя**

Для успешного выполнения задачи пользователь должен выполнять следующие шаги. Далее последует подробное руководство к использованию данного программного средства.

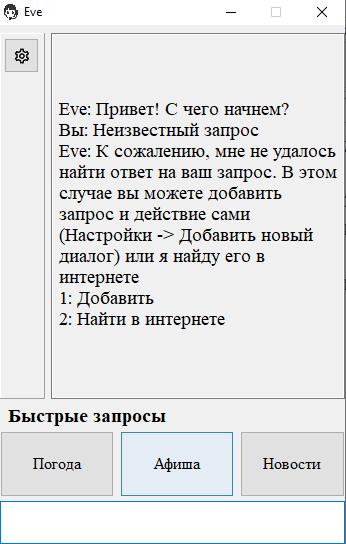
**6.1 Инструкция по использованию**

Для включения и выключения озвучки реплик ассистента пользователь должен выбрать соответствующий пункт меню.

** **

**Рис. 6.1 – Включение озвучки ассистента**

Для ввода запроса пользователь должен ввести свой запрос в специальное поле внизу формы. После чего ассистент будет искать ответ на ваш запрос в списке. Если запрос не был найден, до пользователю будут предложены следующие действия: найти запрос в интернете или добавить его.

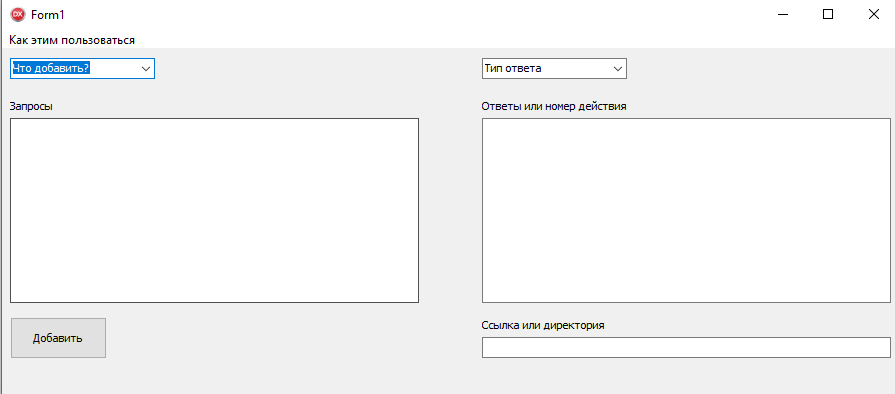


**Рис. 6.2 – Ассистент не нашел запрос в списке**

В этом случае пользователь может ввести “1”, после чего будет открыто приложение для добавления записи, ввести “2”, после чего будет открыт интернет-браузер, в котором будет найден запрос пользователя или ввести другой запрос.

Для добавления напоминания пользователь должен выбрать соответствующий пункт меню, после чего станет активна панель для установки напоминаний.

Приложение добавления записей имеет следующий вид:



**Рисунок 6.3 – Приложение для добавления записей**

В выпадающем меню с текстом “Что добавить” пользователь должен выбрать тип информации, которую он хочет добавить (Greetings, UsefulInfo, Joke).

Для добавления приветствия достаточно ввести какой-либо текст (если вариантов запроса несколько, то ввести их с символом-разделителем '┐' на конце) и нажать кнопку добавить.

Для добавления нового диалога, в выпадающем меню с текстом “Что добавить” пользователь должен выбрать пункт UsefulInfo, вписать в поле “Запросы” фразу, которая вызовет ответ, который добавляется следующим образом: в выпадающем меню “Тип ответа” пользователь должен выбрать пункт “Answer” и ввести в поле “Ответы или номер действия“ все возможные варианты ответов через специальный символ-разделитель ('┐'), после чего нажать кнопку добавить.

Для добавления нового действия, в выпадающем меню с текстом “Что добавить” пользователь должен выбрать пункт UsefulInfo, вписать в поле “Запросы” фразу, которая вызовет действие, которое добавляется следующим опбразом: в выпадающем меню “Тип ответа” пользователь должен выбрать пункт “Action” и ввести в поле “Ответы или номер действия“ цифру 1, после чего в поле “Ссылка или директория” указать ссылку на интернет-ресурс или директорию файла. После чего нажать кнопку добавить.

**Заключение**

Результатом данной курсовой работы является программное средство, созданное в интегрированной среде разработки Delphi 10.3.

С помощью данного программного средства пользователь сможет выполнять различные операции, такие как:

- Добавление своей диалоговой ветки в список;

- Добавление своего действия в список;

- Ведение диалога с компьютером;

- Прослушивание реплик ассистента;

Одна из ключевых задач, поставленная перед написанием программного средства, – сделать удобный и понятный интерфейс. В результате, программное средство может использовать как профессионал, так и пользователь, впервые запустивший данное приложение. Преимуществом перед аналогами получилось то, что пользователь, фактически, может программировать ассистента под свои нужды, добавляя необходимые события в специальном приложении.

В ходе разработки было изучено большое количество литературы из различных источников. Были полученные более углубленные знания о формах и модулях, компонентах и графическом интерфейсе.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Серебряная, Л. В. Структуры и алгоритмы обработки данных : учебно-метод. пособие / Л. В. Серебряная, И. М. Марина. – Минск : БГУИР, 2013. – 51 с. : ил.
2. Процедуры и функции Delphi 10.4.1 [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/122929/> - Дата доступа: 01.06.2021
3. Компоненты Delphi 10.4.1 [Электронный доступ]. – Режим доступа: 10.05.2021
4. Создание компонентов Delphi 10.4.1 [Электронный доступ]. – Режим доступа: <https://thedelphi.ru/les/les90.php> - Дата доступа: 10.05.2021
5. Компоненты форм Delphi 10.4.1 [Электронный доступ]. – Режим досту-  
   па: <http://delphi-faq.ru/palitra-komponentov-delphi/vizyalnie-biblioteki-vcl/svojstva-i-xarakteristiki-formy-form-v-delfi-delphi.html> - Дата доступа: 20.05.2021
6. Осипов, Д. Delphi. Профессиональное программирование / Д. Осипов. - СПб.: Символ-плюс, 2015. - 1056 c.
7. Белов, В.В. Программирование в Delphi: процедурное, объектно-ориентированное, визуальное: Учебное пособие / В.В. Белов. - М.: ГЛТ, 2009. - 240 c.
8. Формы и её свойства [Электронный ресурс]. – Режим доступа:   
   https://thedelphi.ru/les/les8.php - Дата доступа: 15.04.2021

**ПРИЛОЖЕНИЕ А**

**Текст программного модуля управления таблицами**

**MainForm:**

unit MainForm;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls, Vcl.Menus, APIUnit, Math,

Vcl.ToolWin, Vcl.ComCtrls, Vcl.Buttons, Vcl.OleServer, SpeechLib\_TLB,

Vcl.WinXPickers, Vcl.WinXCalendars, Vcl.Grids, Vcl.Samples.Calendar,

System.Notification;

type

TGreeting = record

Greeting: ShortString;

end;

TForm1 = class(TForm)

BWeather: TButton;

BAfisha: TButton;

BNews: TButton;

Label1: TLabel;

ColorDialog: TColorDialog;

ERequest: TEdit;

SettingsPanel: TPanel;

OpenSettingsButton: TBitBtn;

CloseSettingsButton: TBitBtn;

MDialogue: TMemo;

Timer1: TTimer;

OffLight: TBitBtn;

ThemeLabel: TLabel;

AddDialogue: TBitBtn;

Label2: TLabel;

SpeakerButton: TBitBtn;

Label3: TLabel;

SpVoice1: TSpVoice;

procedure BWeatherClick(Sender: TObject);

procedure BAfishaClick(Sender: TObject);

procedure BNewsClick(Sender: TObject);

procedure ERequestKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure FormShow(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure OpenSettingsButtonClick(Sender: TObject);

procedure CloseSettingsButtonClick(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

procedure OffLightClick(Sender: TObject);

procedure AddDialogueClick(Sender: TObject);

procedure SpeakerButtonClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

uses

ShellApi;

var

UserRequest: string;

List: APIUnit.TList;

Flag, LightFlag, Speaker, AnsNotFound, OpenFlag: boolean;

NotesArrCount: integer;

{$R \*.dfm}

procedure TForm1.FormCreate(Sender: TObject);

var

greetings: file of TGreeting;

GreetArr: array of string;

buf: TGreeting;

phrase: String;

i, j: byte;

list: APIUnit.TList;

begin

NotesArrCount := 1;

OpenFlag := False;

SettingsPanel.Width := 45;

LightFlag := false;

Speaker := false;

AnsNotFound := false;

SystemParametersInfo(SPI\_SETBEEP, 0, nil, SPIF\_SENDWININICHANGE);

randomize;

assignFile(greetings, 'Greetings.txt');

reset(greetings);

if FileSize(greetings) = 0 then exit;

seek (greetings, 0 );

read(Greetings, buf);

phrase := '';

i := 1;

j := 0;

setLength(GreetArr, 1);

while True do

begin

if buf.Greeting[i] = '┐' then

begin

delete(buf.Greeting, 1, length(phrase)+1 );

GreetArr[j] := phrase;

phrase := '';

i := 1;

inc(j);

if buf.Greeting = '' then break;

SetLength(GreetArr,j+1);

end

else

begin

phrase := phrase + buf.Greeting[i];

inc(i);

end;

end;

MDialogue.Visible := true;

Phrase := GreetArr[ random ( Length(GreetArr) ) ];

MDialogue.Lines.Add('Eve: ' + phrase );

CreateList;

end;

procedure TForm1.ERequestKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

const

enter = 13;

var

k, ErrPos: integer;

begin

if key = enter then

begin

if OpenFlag then

begin

APIUnit.CreateList;

OpenFlag := False;

end;

if (ERequest.Text = '') then exit;

if AnsNotFound then

begin

AnsNotFound := false;

val(ERequest.Text, k, ErrPos);

case k of

1:

begin

ShellExecute(0, 'open', 'G:\CourseProject\AddDialogue\Final\Win32\Debug\AddRecord.exe',

nil, nil, SW\_SHOWNORMAL);

OpenFlag := true;

ERequest.Clear;

exit;

end;

2:

begin

MDialogue.Lines.Add('Eve: Хорошо');

Sleep(500);

ShellExecute(0, 'open', PWideChar('https://www.google.com/search?q='+UserRequest),

nil, nil, SW\_SHOWNORMAL);

ERequest.Clear;

exit;

end;

end;

if (ERequest.Text = 'добавить') or (ERequest.Text = 'добавь') or (ERequest.Text = 'Добавить') or (ERequest.Text = 'Добавь') then

begin

ShellExecute(0, 'open', 'G:\CourseProject\AddDialogue\Final\Win32\Debug\AddRecord.exe',

nil, nil, SW\_SHOWNORMAL);

OpenFlag := true;

ERequest.Clear;

exit;

end

else if (ERequest.Text = 'найди') or (ERequest.Text = 'поиск') or (ERequest.Text = 'Найди') or (ERequest.Text = 'Поиск')then

begin

MDialogue.Lines.Add('Eve: Хорошо');

Sleep(500);

ShellExecute(0, 'open', PWideChar('https://www.google.com/search?q='+UserRequest),

nil, nil, SW\_SHOWNORMAL);

ERequest.Clear;

exit;

end;

end;

UserRequest := trim(ERequest.Text);

MDialogue.lines.Add('Вы: '+ UserRequest);

FormatInput(UserRequest);

ERequest.Clear;

FindAns(UserRequest, Speaker, AnsNotFound);

if AnsNotFound then

begin

MainForm.Form1.MDialogue.Lines.Add('Eve: К сожалению, мне не удалось найти ответ на ваш запрос. В этом случае вы можете добавить запрос и действие сами (Настройки -> Добавить новый диалог) или я найду его в интернете');

MainForm.Form1.MDialogue.Lines.Add('1: Добавить');

MainForm.Form1.MDialogue.Lines.Add('2: Найти в интернете');

end;

end;

end;

//Apps

procedure TForm1.AddDialogueClick(Sender: TObject);

begin

ShellExecute(Handle, 'open', 'G:\CourseProject\AddDialogue\Final\Win32\Debug\AddRecord.exe',

nil, nil, SW\_SHOWNORMAL) ;

OpenFlag := true;

end;

procedure TForm1.BAfishaClick(Sender: TObject);

begin

ShellExecute(Handle, 'open', 'https://afisha.relax.by/',

nil, nil, SW\_SHOWNORMAL) ;

end;

procedure TForm1.BNewsClick(Sender: TObject);

begin

ShellExecute(Handle, 'open', 'https://www.onliner.by/',

nil, nil, SW\_SHOWNORMAL) ;

end;

procedure TForm1.BWeatherClick(Sender: TObject);

begin

ShellExecute(Handle, 'open', 'https://www.accuweather.com/ru/by/minsk/28580/weather-forecast/28580',

nil, nil, SW\_SHOWNORMAL) ;

end;

//Interface

procedure TForm1.FormShow(Sender: TObject);

begin

ERequest.SetFocus;

end;

procedure TForm1.SpeakerButtonClick(Sender: TObject);

begin

if not Speaker then

begin

speaker := true;

SpeakerButton.Glyph.LoadFromFile('G:\CourseProject\Images\Sound\BMP\outline\_record\_voice\_over\_black\_18dp.bmp');

end

else

begin

speaker := false;

SpeakerButton.Glyph.LoadFromFile('G:\CourseProject\Images\Sound\BMP\outline\_voice\_over\_off\_black\_18dp.bmp');

end;

end;

procedure TForm1.OffLightClick(Sender: TObject);

var

flag: Boolean;

begin

case LightFlag of

false:

begin

form1.color := clBlack;

mDialogue.Color := clBlack;

ERequest.color := clBlack;

ERequest.Font.Color := clYellow;

label1.font.Color := clYellow;

label2.font.Color := clYellow;

Label3.Font.Color := clYellow;

ThemeLabel.font.Color := clYellow;

MDialogue.Font.Color := clYellow;

MDialogue.BorderStyle := bsNone;

LightFlag := true;

exit;

end;

true:

begin

form1.color := clWhite;

mDialogue.Color := clWhite;

ERequest.color := clWhite;

ERequest.Font.Color := clBlack;

mDialogue.Font.Color := clBlack;

label1.font.Color := clBlack;

label2.font.Color := clBlack;

Label3.Font.Color := clBlack;

ThemeLabel.font.Color := clBlack;

MDialogue.BorderStyle := bsSingle;

LightFlag := false;

exit;

end;

end;

end;

procedure TForm1.OpenSettingsButtonClick(Sender: TObject);

begin

flag := true;

Timer1.Interval := 10;

OpenSettingsButton.Visible := False;

MDialogue.Visible := False;

timer1.Enabled := true;

end;

procedure TForm1.CloseSettingsButtonClick(Sender: TObject);

begin

flag := false;

Timer1.Interval := 10;

OpenSettingsButton.Visible := true;

MDialogue.Visible := true;

timer1.Enabled := true;

end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin

case flag of

True:

begin

if SettingsPanel.Width <> 345 then

begin

SettingsPanel.Width := SettingsPanel.Width + 20;

end

else

Timer1.Enabled := false;

end;

False:

begin

if SettingsPanel.Width <> 45 then

begin

SettingsPanel.Width := SettingsPanel.Width - 20;

end

else

begin

Timer1.Enabled := false;

MDialogue.Visible := true;

end;

end;

end;

end;

end.

**APIUnit:**

unit APIUnit;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, ShellApi,SpeechLib\_TLB, math;

type

TForm2 = class(TForm)

private

{ Private declarations }

public

{ Public declarations }

end;

type

TUsefulData = record

Req: String[255];

case TypeOfReq: boolean of

true: (Ans: String[255]);

false: (

NumberOfAction: integer;

Link: ShortString

);

end;

Ptr = ^TElement;

TElement = record

next, prev: Ptr;

data: TUsefulData;

end;

TList = record

head, tail: Ptr;

end;

var

Form2: TForm2;

Brains: file of TUsefulData;

Jokes: file of ShortString;

List: TList;

procedure CreateList ();

procedure FormatInput(var s: string);

procedure AddNewEl (var curr: Ptr);

procedure FindAns (const UserRequest: String; Voice: Boolean; var AnsNotFound: Boolean);

procedure Action (const NumOfAction: integer; const Link: String; Voice: Boolean);

implementation

uses

MainForm;

{$R \*.dfm}

procedure FormatInput(var s: string);

var

i: byte;

begin

for i := (length(s) - 1) downto 1 do

begin

if (s[i] = ' ') and (s[i + 1] = ' ') then

begin

delete(s, i + 1, 1)

end;

end;

for i := 1 to length(s) do

begin

if (s[i] in ['A'..'Z', 'А'..'Я']) and (s[i] <> ' ') then

s[i] := chr(ord(s[i]) + 32);

if s[i] in ['`','~','#','^','&','{','}','\'] then

delete(s,pos(s[i],s),1);

end;

end;

procedure CreateList;

var

curr, tmp: Ptr;

begin

AssignFile(Brains,'Brains.txt');

List.head := nil;

list.tail := nil;

new(curr);

reset(Brains);

read(Brains, curr.data);

AddNewEl ( curr);

while not Eof(Brains) do

begin

tmp := curr;

new(curr);

read(Brains,curr.data);

AddNewEl(curr);

tmp.next := curr;

curr.prev := tmp;

end;

closeFile(Brains);

dispose(curr);

end;

procedure AddNewEl (var curr: Ptr);

begin

if list.head = nil then

begin

list.head := curr;

list.tail := curr;

curr.next := nil;

curr.prev := nil;

exit;

end;

curr.next := nil;

curr.prev := list.tail;

list.tail.next := curr;

list.tail := curr;

end;

procedure FindAns (const UserRequest: String; Voice: Boolean; var AnsNotFound: Boolean);

var

curr: Ptr;

AnsArr: Array of string;

phrase: string;

i, j: byte;

flag: boolean;

begin

Randomize;

new(curr);

curr := list.head;

flag := false;

while curr <> nil do

begin

if pos(UserRequest, curr.data.Req) <> 0 then

begin

case curr.data.TypeOfReq of

true:

begin

SetLength(AnsArr,0);

i := 0;

j := 1;

SetLength(AnsArr,i+1);

phrase := '';

while True do

begin

if curr.data.Ans[j] = '┐' then

begin

AnsArr[i] := phrase;

phrase := '';

inc(i);

SetLength(AnsArr,i+1);

inc(j)

end

else

begin

phrase := phrase + curr.data.Ans[j];

inc(j)

end;

if j = length(curr.data.Ans) then

begin

phrase := '';

phrase := AnsArr[ randomRange(0, Length(AnsArr)-1 ) ];

if voice then

begin

MainForm.Form1.MDialogue.Lines.Add('Eve: ' + phrase );

MainForm.Form1.SpVoice1.Speak(phrase,SVSFlagsAsync);

exit;

end

else

begin

MainForm.Form1.MDialogue.Lines.Add('Eve: ' + phrase );

exit;

end;

end;

end;

flag := true;

end;

False:

begin

Action(curr.data.NumberOfAction, curr.data.Link, Voice);

flag := true;

end;

end;

end;

curr := curr.next;

end;

if not (flag) then AnsNotFound := true;

end;

procedure Action (const NumOfAction: integer; const Link: String; Voice: Boolean);

var

jokesArr: array of ShortString;

i: integer;

Joke: ShortString;

begin

case NumOfAction of

0:

begin

MainForm.Form1.MDialogue.Lines.Add('Eve: Сегодня '+ FormatDateTime('dddddd', now) );

if voice then

MainForm.Form1.SpVoice1.Speak('Сегодня' + FormatDateTime('dddddd', now), SVSFlagsAsync);

end;

1:

ShellExecute(0, 'open', PWideChar(Link), nil, nil, SW\_SHOWNORMAL);

2:

begin

Randomize;

AssignFile(Jokes,'Jokes.txt');

reset(Jokes);

i := 0;

while not Eof (Jokes) do

begin

SetLength(JokesArr,i+1);

read(Jokes,JokesArr[i]);

inc(i);

end;

Joke := JokesArr[ randomRange(1, Length(JokesArr)-1 ) ];

if voice then

MainForm.Form1.SpVoice1.Speak(joke ,SVSFlagsAsync);

MainForm.Form1.MDialogue.Lines.Add('Eve: ' + joke );

CloseFile(Jokes);

end;

end;

end;

end.

**AddRecord:**

unit Unit1;

interface

uses

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

System.Classes, Vcl.Graphics,

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

type

TUsefulData = record

Req: String[255];

case TypeOfReq: boolean of

true: (Ans: String[255]);

false: (

NumberOfAction: integer;

Link: ShortString

);

end;

TGreeting = record

Greeting: ShortString;

end;

TJoke = record

Joke: ShortString;

end;

TForm1 = class(TForm)

AddCB: TComboBox;

AddButton: TButton;

AddMemo: TMemo;

ActionNumber: TMemo;

ActionCB: TComboBox;

MainMenu1: TMainMenu;

N1: TMenuItem;

Label1: TLabel;

Label2: TLabel;

LinkEdit: TEdit;

Label3: TLabel;

procedure AddButtonClick(Sender: TObject);

procedure N1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);

var

greeting: TGreeting;

Greetings: File of TGreeting;

Joke: TJoke;

Jokes: file of TJoke;

Brains: file of TUsefulData;

data: TUsefulData;

i: byte;

buf: string;

ErrorPos: integer;

begin

case AddCB.ItemIndex of

0:

begin

greeting.Greeting := '';

for I := 0 to AddMemo.Lines.Count-1 do

greeting.Greeting := greeting.Greeting + AddMemo.Lines[i];

assignFile(greetings,'G:\CourseProject\26.05\Win32\Debug\Greetings.txt');

reset(greetings);

seek(Greetings,fileSize(Greetings));

write(greetings, greeting);

CloseFile(greetings);

AddMemo.Lines.Clear;

end;

1:

begin

data.Req := '';

for I := 0 to AddMemo.Lines.Count-1 do

begin

data.Req := data.Req + AddMemo.Lines[i];

end;

case ActionCB.ItemIndex of

0: Data.TypeOfReq := True;

1: Data.TypeOfReq := False;

end;

data.Ans := '';

data.NumberOfAction := 0;

case data.TypeOfReq of

True:

begin

for I := 0 to ActionNumber.Lines.Count-1 do

begin

data.Ans := data.Ans + ActionNumber.Lines[i];

end;

end;

False:

begin

val (ActionNumber.Lines[0], Data.NumberOfAction, ErrorPos);

if ErrorPos <> 0 then

begin

ShowMessage('В поле добавления номера действия необходимо ввести целое число!');

ActionNumber.Lines.Clear;

exit;

end;

data.Link := LinkEdit.Text;

LinkEdit.Clear;

end;

end;

AssignFile(Brains, 'G:\CourseProject\26.05\Win32\Debug\Brains.txt');

reset(Brains);

seek(Brains, FileSize(Brains));

write(Brains, Data);

closeFile(Brains);

AddMemo.Lines.Clear;

ActionNumber.Lines.Clear;

end;

2:

begin

Joke.Joke := '';

for I := 0 to AddMemo.Lines.Count-1 do

begin

joke.Joke := joke.Joke + AddMemo.Lines[i];

end;

AssignFile(Jokes,'G:\CourseProject\26.05\Win32\Debug\Jokes.txt');

reset(Jokes);

Seek(Jokes, FileSize(Jokes));

Write(Jokes, Joke);

CloseFile(Jokes);

AddMemo.Lines.Clear;

end;

end;

end;

procedure TForm1.N1Click(Sender: TObject);

begin

ShellExecute(Handle, 'open', 'README.txt', nil, nil, SW\_SHOWNORMAL);

end;

end.