Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №8**

**«Основы событийно-ориентированного программирования»**

**МДК 05.02 «РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Чарушин Егор Вадимович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

1. **Цель работы:**

Получение базовых навыков реализации приложений с графическим интерфейсом пользователя на основе событийно-ориентированной.

1. **Формулировка задания:**
2. Реализовать приложение "Таймер".
3. Предусмотреть возможность подачи звукового сигнала по истечении заданного промежутка времени.
4. Предусмотреть возможность досрочной остановки таймера.
5. **Код программы:**

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, Buttons,

ExtCtrls, Process; // добавили Process

type

{ TTTTTimer }

TTTTTimer = class(TForm)

Pause: TButton;

Exitt: TBitBtn;

Clear: TButton;

Start: TButton;

Hour: TEdit;

Minutes: TEdit;

Seconds: TEdit;

Clock: TLabel;

HourL: TLabel;

MinutsL: TLabel;

SecondsL: TLabel;

Timer1: TTimer;

procedure ExittClick(Sender: TObject);

procedure ClearClick(Sender: TObject);

procedure HourChange(Sender: TObject);

procedure MinutesChange(Sender: TObject);

procedure PauseClick(Sender: TObject);

procedure SecondsChange(Sender: TObject);

procedure StartClick(Sender: TObject);

procedure StopClick(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

private

TimeCounter: Integer;

PausedTimeCounter: Integer;

Paused: Boolean;

public

// existing procedures

end;

var

TTTTimer: TTTTTimer;

implementation

{$R \*.lfm}

{ TTTTTimer }

procedure TTTTTimer.StartClick(Sender: TObject);

var

Hours, Mins, Secs: Integer;

begin

// проверяем, что введенные значения являются целыми числами

if (TryStrToInt(Hour.Text, Hours) and

TryStrToInt(Minutes.Text, Mins) and

TryStrToInt(Seconds.Text, Secs)) then

begin

TimeCounter := Hours \* 3600 + Mins \* 60 + Secs;

PausedTimeCounter := TimeCounter; // сохраняем начальное значение времени

Timer1.Enabled := True;

end

else

ShowMessage('Пожалуйста, введите корректные значения времени');

end;

procedure TTTTTimer.ClearClick(Sender: TObject);

begin

Timer1.Enabled := False;

TimeCounter := 0;

PausedTimeCounter := 0;

Clock.Caption := '00:00:00';

Hour.Text := '0';

Minutes.Text := '0';

Seconds.Text := '0';

end;

procedure TTTTTimer.HourChange(Sender: TObject);

var Hours: Integer;

begin

if not TryStrToInt(Hour.Text, Hours) then

begin

Hour.Text := '0';

Hours := 0;

end

else

begin

if Hours > 61 then

begin

Hour.Text := '61';

Hours := 61;

end;

end;

end;

procedure TTTTTimer.MinutesChange(Sender: TObject);

var Mins: Integer;

begin

if not TryStrToInt(Minutes.Text, Mins) then

begin

Minutes.Text := '0';

Mins := 0;

end

else

begin

if Mins > 61 then

begin

Minutes.Text := '61';

Mins := 61;

end;

end;

end;

procedure TTTTTimer.PauseClick(Sender: TObject);

begin

Paused := not Paused;

if Paused then

begin

Timer1.Enabled := False;

PausedTimeCounter := TimeCounter;

end

else

begin

Timer1.Enabled := True;

TimeCounter := PausedTimeCounter;

end;

end;

procedure TTTTTimer.SecondsChange(Sender: TObject);

var Secs: Integer;

begin

if not TryStrToInt(Seconds.Text, Secs) then

begin

Seconds.Text := '0';

Secs := 0;

end

else

begin

if Secs > 61 then

begin

Seconds.Text := '61';

Secs := 61;

end;

end;

end;

procedure TTTTTimer.ExittClick(Sender: TObject);

begin

Close; // Закрытие формы

end;

procedure TTTTTimer.StopClick(Sender: TObject);

begin

Timer1.Enabled := False;

PausedTimeCounter := TimeCounter;

end;

procedure TTTTTimer.Timer1Timer(Sender: TObject);

var

Hours, Mins, Secs: Integer;

AProcess: TProcess;

begin

if not Paused then

begin

if TimeCounter > 0 then

begin

TimeCounter := TimeCounter - Timer1.Interval div 1000;

Hours := TimeCounter div 3600;

Mins := (TimeCounter - Hours \* 3600) div 60;

Secs := TimeCounter - Hours \* 3600 - Mins \* 60;

Clock.Caption := Format('%2.2d:%2.2d:%2.2d', [Hours, Mins, Secs]);

end

else

begin

Timer1.Enabled := False;

Clock.Caption := '00:00:00';

AProcess := TProcess.Create(nil);

try

AProcess.Executable := '/usr/bin/afplay';

AProcess.Parameters.Add('/Users/mvideomvideo/Desktop/Python/song.mp3');

AProcess.Execute;

finally

AProcess.Free;

end;

end;

end

else

begin

Hours := PausedTimeCounter div 3600;

Mins := (PausedTimeCounter - Hours \* 3600) div 60;

Secs := PausedTimeCounter - Hours \* 3600 - Mins \* 60;

Clock.Caption := Format('%2.2d:%2.2d:%2.2d', [Hours, Mins, Secs]);

end;

end;

end.

1. **Результат выполнения программы.**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание**

Рисунок 1 – Интерфейс программы.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

Рисунок 2 – Выполнение программы.

1. **Вывод:**

В ходе выполнения работы были получены знание в работе с Lazarus, а именно с компонентом TTimer. Были освоены операции по работе с временем, а также взаимодействий кнопок с программным интерфейсом. Была проведена работа по работе со звуковыми эффектами, по окончании истечении времени. А также была продумана система остановки таймера, сброса таймера, а также выхода из программы, путем нажатия на кнопку.

Таким образом, получилась полностью рабочая программа, с интерфейсом включающим в себя ввод часов, минут и секунд, запуска, остановки и остановки на паузу таймер, а также выхода из программы. Полученные цели и задания была достигнуты и выполнены в полном объеме, соответствующе знание были успешно получены и в ходе выполнения работы были закреплены.