Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ №8**

**«ОСНОВЫ СОБЫТИЙНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Яровиков Илья Евгеньевич

Преподаватель:

Сергеева Е.Г.

Киров

2023

1. Цель домашней контрольной работы: Получение базовых навыков реализации приложений с графическим интерфейсом пользователя на основе событийно-ориентированной программы
2. Формулировка задания (25 вариант)  
   1) Реализовать приложение «Таймер».

2) Предусмотреть возможность остановки таймера

1. Код программы

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

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

ExtCtrls;

type

{ TForm1 }

TForm1 = 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; // new variable

public

// existing procedures

end;

var

Form1: TForm1;

implementation

{$R \*.lfm}

{ TForm1 }

procedure TForm1.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 TForm1.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 TForm1.HourChange(Sender: TObject);

var Hours: Integer;

begin

// check if the entered value is a number

if not TryStrToInt(Hour.Text, Hours) then

begin

// if not a number, set the value to 0

Hour.Text := '0';

Hours := 0;

end

else

begin

// check if the value is over 61

if Hours > 61 then

begin

// if over 61, set the value to 61

Hour.Text := '61';

Hours := 61;

end;

end;

end;

procedure TForm1.MinutesChange(Sender: TObject);

var Mins: Integer;

begin

// check if the entered value is a number

if not TryStrToInt(Minutes.Text, Mins) then

begin

// if not a number, set the value to 0

Minutes.Text := '0';

Mins := 0;

end

else

begin

// check if the value is over 61

if Mins > 61 then

begin

// if over 61, set the value to 61

Minutes.Text := '61';

Mins := 61;

end;

end;

end;

procedure TForm1.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 TForm1.SecondsChange(Sender: TObject);

var Secs: Integer;

begin

// check if the entered value is a number

if not TryStrToInt(Seconds.Text, Secs) then

begin

// if not a number, set the value to 0

Seconds.Text := '0';

Secs := 0;

end

else

begin

// check if the value is over 61

if Secs > 61 then

begin

// if over 61, set the value to 61

Seconds.Text := '61';

Secs := 61;

end;

end;

end;

procedure TForm1.ExittClick(Sender: TObject);

begin

end;

procedure TForm1.StopClick(Sender: TObject);

begin

Timer1.Enabled := False;

PausedTimeCounter := TimeCounter;

end;

procedure TForm1.Timer1Timer(Sender: TObject);

var

Hours, Mins, Secs: Integer;

begin

if not Paused then // only update if not paused

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';

end;

end

else // display remaining time if paused

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.

4.Результат выполнения программы

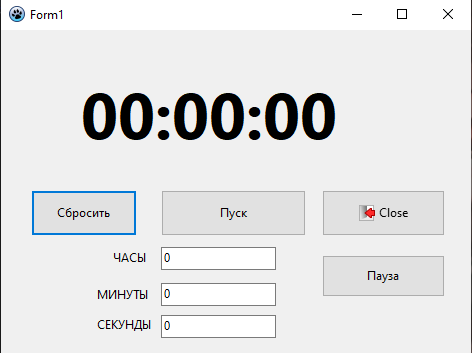


Рисунок1 - Таймер

5. Вывод

Исходя из полученных опытным путем знаний и умений мы можем сформулировать емкий, а главное правильный вывод.

Во время выполнений домашняя контрольная номер №8 мы на практике применили знания, полученные на лекционных занятиях по МДК

В ходе работы мы узнали, как работает Lazarus и как создавать на нем приложения. Узнали как работать с компонентами такими как: Timer, Edit, Button, Label.

Таким образом Домашняя контрольная 8 была выполнена