УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №6.1

по предмету

Основы алгоритмизации и программирования

Вариант 1

Выполнил:

Бетеня К.С.

Проверила:

Данилова Г.В.

Группа 351005

Минск 2024

Задание:

Анимация. Составить программу вывода на экран изображения циферблата механических часов с секундной, минутной и часовой стрелками. Запуск часов осуществляется клавишей "Ввод", при этом перемещение секундной стрелки сопровождается "характерным" для часов звуком.

Код программы на **Delphi**:

Unit MainFormUnit;

Interface

Uses

Winapi.Windows,

System.SysUtils,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.ExtCtrls,

Vcl.Menus,

Vcl.Buttons,

Vcl.MPlayer,

Vcl.StdCtrls;

Type

TClockForm = Class(TForm)

MainMenu: TMainMenu;

SecondTimer: TTimer;

ClockStartSpeedButton: TSpeedButton;

FileList: TMenuItem;

Exit: TMenuItem;

Instraction: TMenuItem;

AboutEditor: TMenuItem;

PickPlayer: TMediaPlayer;

Procedure ClockStartSpeedButtonClick(Sender: TObject);

Procedure SecondTimerTimer(Sender: TObject);

Procedure InstractionClick(Sender: TObject);

Procedure AboutEditorClick(Sender: TObject);

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Procedure ExitClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Const

StartAngle: Double = -Pi / 2;

ThrowOffAngle: Double = 3 \* Pi / 2;

LenSec: Integer = 180;

LenMinute: Integer = 150;

LenHour: Integer = 120;

Var

ClockForm: TClockForm;

SecAngle, MinAngle, HourAngle: Double;

MinSleep: Integer = 0;

X, Y: Integer;

Implementation

Procedure TClockForm.ClockStartSpeedButtonClick(Sender: TObject);

Begin

SecondTimer.Enabled := True;

ClockStartSpeedButton.Enabled := False;

SecAngle := StartAngle;

MinAngle := StartAngle;

HourAngle := StartAngle;

PickPlayer.Open;

X := ClockForm.ClientWidth Div 2;

Y := ClockForm.ClientHeight Div 2;

End;

Function CompareFirstFourDigits(Num1, Num2: Extended): Boolean;

Var

Str1, Str2: String;

Begin

Str1 := FloatToStr(Num1);

Str2 := FloatToStr(Num2);

If Pos('.', Str1) > 0 Then

Str1 := Copy(Str1, 1, Pos('.', Str1) - 1);

If Pos('.', Str2) > 0 Then

Str2 := Copy(Str2, 1, Pos('.', Str2) - 1);

Result := Copy(Str1, 1, 4) = Copy(Str2, 1, 4);

End;

Procedure DrawClock(Canvas: TCanvas);

Const

DotsRadius: Integer = 198;

OuterRadius: Integer = 215;

OuterWidth: Integer = 6;

InnerRadius: Integer = 205;

InnerWidth: Integer = 4;

HourCounter: Integer = 12;

IncAngle: Double = Pi / 6;

AngleWidth: Integer = 5;

Var

Angle: Real;

I: Integer;

Begin

Canvas.Pen.Color := ClBlue;

Canvas.Pen.Width := OuterWidth;

Canvas.MoveTo(X, Y);

Canvas.Ellipse(X - OuterRadius, Y - OuterRadius, X + OuterRadius, Y +

OuterRadius);

Canvas.Pen.Color := ClBlack;

Canvas.Pen.Width := InnerWidth;

Canvas.MoveTo(X, Y);

Canvas.Ellipse(X - InnerRadius, Y - InnerRadius, X + InnerRadius, Y +

InnerRadius);

Canvas.Pen.Color := ClBlue;

Angle := StartAngle;

For I := 1 To HourCounter Do

Begin

Canvas.Pen.Width := AngleWidth;

Angle := Angle + IncAngle;

Canvas.MoveTo(X + Trunc(DotsRadius \* Cos(Angle)), Y + Trunc(DotsRadius \*

Sin(Angle)));

Canvas.LineTo(X + Trunc(DotsRadius \* Cos(Angle)), Y + Trunc(DotsRadius \*

Sin(Angle)));

End;

End;

Procedure DrawArrows(Canvas: TCanvas);

Const

HourWidth: Integer = 10;

MinWidth: Integer = 5;

SecWidth: Integer = 2;

RivetRadius: Integer = 5;

Begin

Canvas.Pen.Color := ClBlack;

Canvas.Pen.Width := HourWidth;

Canvas.MoveTo(X, Y);

Canvas.LineTo(X + Trunc(LenHour \* Cos(HourAngle)), Y + Trunc(LenHour \*

Sin(HourAngle)));

Canvas.Pen.Color := ClBlack;

Canvas.Pen.Width := MinWidth;

Canvas.MoveTo(X, Y);

Canvas.LineTo(X + Trunc(LenMinute \* Cos(MinAngle)), Y + Trunc(LenMinute \*

Sin(MinAngle)));

Canvas.Pen.Color := ClRed;

Canvas.Pen.Width := SecWidth;

Canvas.MoveTo(X, Y);

Canvas.LineTo(X + Trunc(LenSec \* Cos(SecAngle)), Y + Trunc(LenSec \*

Sin(SecAngle)));

Canvas.Pen.Color := ClBlue;

Canvas.MoveTo(X, Y);

Canvas.Ellipse(X - RivetRadius, Y - RivetRadius, X + RivetRadius, Y +

RivetRadius);

End;

Procedure ChangeTheTurnsOfTheArrow();

Const

IncSecMinAngle: Double = Pi / 180 \* 6;

HourSecAngle: Double = Pi / 180 \* 360 / 12 / 60;

Var

RealComparisonStatus: Boolean;

Begin

SecAngle := SecAngle + IncSecMinAngle;

ClockForm.PickPlayer.Play;

RealComparisonStatus := CompareFirstFourDigits(SecAngle, ThrowOffAngle);

If RealComparisonStatus Then

SecAngle := StartAngle;

RealComparisonStatus := CompareFirstFourDigits(SecAngle, MinAngle);

If RealComparisonStatus And (MinSleep <> 1) Then

Begin

MinAngle := MinAngle + IncSecMinAngle;

Inc(MinSleep);

End

Else

MinSleep := 0;

RealComparisonStatus := CompareFirstFourDigits(MinAngle, ThrowOffAngle);

If RealComparisonStatus Then

MinAngle := StartAngle;

RealComparisonStatus := CompareFirstFourDigits(SecAngle, HourAngle);

If RealComparisonStatus Then

HourAngle := HourAngle + HourSecAngle;

RealComparisonStatus := CompareFirstFourDigits(HourAngle, ThrowOffAngle);

If RealComparisonStatus Then

MinAngle := StartAngle;

End;

Procedure TClockForm.SecondTimerTimer(Sender: TObject);

Var

BitMap: TBitmap;

Begin

BitMap := TBitmap.Create();

BitMap.Height := ClockForm.ClientHeight;

BitMap.Width := ClockForm.ClientWidth;

DrawClock(BitMap.Canvas);

DrawArrows(BitMap.Canvas);

ChangeTheTurnsOfTheArrow();

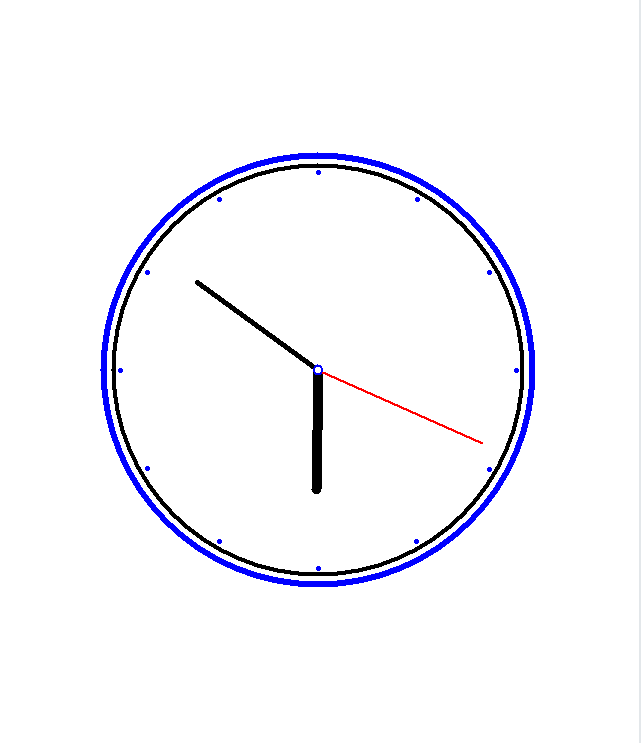
ClockForm.Canvas.Draw(0, 0, BitMap);

BitMap.Free();

End;

End.

Результат в **Delphi**:



Блок-схема:

