Ministerul Educaţiei al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra: Tehnologii Informaționale

**RAPORT**

Lucrare de laborator Nr.1

*la Medii Interactive de Dezvoltare a Produselor Soft*

A efectuat: Pascari Ion

st.gr.TI-144

A verificat: Cojocaru Svetlana

dr.conf.univ.,

Chişinău 2016

**Lucrarea de laborator nr.1**

**Tema:**  **MEDIUL INTEGRAT C++ BUILDER**

**Scopul lucrării:**

**a)** Însuşirea modului de utilizare a celor mai importante componente ale mediului integrat C++ BUILDER . Realizarea unui program simplu care utilizează componente de tip *TButton, TEdit, Tlabel, RadioButton*  etc.

**b)** Însuşirea modului de utilizare a componentei VCL **TTimer.** Însuşirea modului de utilizare a funcţiilor de lucru cu timpul sistem. Realizarea unor aplicaţii de gestionare a resursei timp.

**c)** Însuşirea modului de utilizare a componentelor VCL **TPaintBox** şi **TPanel.** Însuşirea modului de utilizare a principalelor funcţii grafice ale mediului C++BUILDER . Realizarea unor elemente pentru afişarea grafică a informaţiei (diagramă şi bargraf).

**Formularea condiţiei problemei (sarcina de lucru):**

1. Se elaborează un program pentru realizarea unui contor cu funcțiile incrementare/decrementare.
2. Se elaborează un program pentru realizarea unui cronometru.
3. Se elaborează un program pentru realizarea a două elemente de afişare (bargraf şi diagramă cu avans continuu)

**Noţiuni principale din teorie şi metode folosite:**

***Object Selector***

În capătul de sus al lui se află Object Selector care conţine toate componentele de pe formă împreună cu tipul lor.

Object Inspector are două pagini – figurile 1.7, 1.8.:

1. Properties page (figura 1.7) permite stabilirea (setarea) propietăţilor unei componente, şi anume: dimensiunile ei, poziţia în cadrul formei, fonturile folosite, numele etc. Alte propietăţi pot fi setate la momentul execuţiei programului prin scrierea de cod sursă în cadrul handler-lor de evenimente.
2. Event page (figura 1.8) permite legarea unei componente la evenimentele programului. Prin executarea unui dublu click pe un eveniment, de exemplu pe *OnClik*, C++Builder creează un handler de evenimente, care este de fapt o metodă ataşată unei clase şi care se va executa când apare un eveniment particular (în cazul nostru executarea unui click pe buton).

Un handler de evenimente pentru componenta **TButton** din cadrul paginii Standard va arăta în felul următor:

**void\_fastcall Tform1::Button1Click(TObject \*Sender)**

**{**

**//Aici putem scrie cod sursă, care se va executa**

**//în momentul apăsării butonului cu ajutorul mouse-ului**

**}**

În cazul în care alegem evenimentul **OnDblClick** (OnDoubleClick) handler-ul de evenimente va arăta în felul următor (am ales în cazul acesta componenta **RadioButton**):

**void\_fastcall Tform1::RadioButton1DblClick(TObject \*Sender)**

**{**

**//Aici putem scrie cod sursă, care se va executa**

**//în momentul apăsării butonului cu ajutorul mouse-ului**

**}**

Componenta TTimer se găseşte în **Component Palette** (*pagina System*) .

Obiectul de acest tip permite execuţia in cadrul aplicaţiei a unor funcţii la intervale specificate. În context Windows obiectul TTimer lansează către aplicaţie mesaje la intervale prestabilite.

* Proprietatea *Enabled* stabileşte dacă obiectul TTimer răspunde la evenimentul *OnTimer*  (dacă Enabled este *true* atunci se răspunde la eveniment – Timer-ul este activat) ;
* Proprietatea *Interval* specifica numărul de milisecunde dintre două mesaje tip TTimer consecutive ( TTimer este apelat dupa fiecare trecere a intervalului specificat ) ;
* Proprietatea *Name* specifică numele Timer-ului;
* Proprietatea *Tag* este utilizata pentru transferul de informatii suplimentare (variabile de tip int) ;
* Evenimentul *OnTimer*  apare de fiecare dată când trece intervalul (*Interval)* specificat.

Componenta TPaintBox se găseşte în **Component Palette** (*pagina System*) . Obiectul de acest tip furnizează o componentă *TCanvas* care permite desenarea în interiorul unui dreptunghi, prevenind depăşirea marginilor acestuia.

**Funcţii de desenare ale obiectului TCanvas**

|  |  |
| --- | --- |
| **Prototip** | **Funcţie** |
| MoveTo(int x,int y) | fixeaza pozitia curenta in (x,y) |
| LineTo (int x,int y); | linie din poz. curenta pina la (x,y) |
| Ellipse(int stinga,int sus, int dreapta,  int jos) | desenează o elipsă tangentă la  laturile dreptunghiului specificat |
| Rectangle(int stinga,int sus, int dreapta,int jos) | desenează un dreptunghi cu coordonatele colţurilor specificate |
| TextOut (int x, int y, AnsiString Text) | afişează textul *Text* începând cu punctul de coordonate x, z |
| FloodFill (int x, int y, TColor Culoare , TfillStyle Stil)  Stil=fsSurface sau fsBorder | umple o suprafaţă,mărginită de un contur cu culoare *Culoare,* care conţine punctul de coordonate (x,y) cu stilul *Stil* (implicit *fsBorder)* |
| CopyRect (TRect *dest* TCanvas *Canvas,* TRect *sursa*) | copiază zona definită de dreptunghiul *sursă* în Canvas-ul  *Canvas* în zona definită de dreptunghiul *dest* |

1. **Codul programului în limbajul C++ utilizând C++Builder**

*Project.cpp*

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

//---------------------------------------------------------------------------

USEFORM("Unit1.cpp", Form1);

//---------------------------------------------------------------------------

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

{

try

{

Application->Initialize();

Application->CreateForm(\_\_classid(TForm1), &Form1);

Application->Run();

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

catch (...)

{

try

{

throw Exception("");

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

}

return 0;

}

//---------------------------------------------------------------------------

*Counter.cpp*

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

int n;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::UPClick(TObject \*Sender)

{

n += 1;

text->Text = IntToStr(n);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::DOWNClick(TObject \*Sender)

{

n -= 1;

text->Text = IntToStr(n);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::EXITClick(TObject \*Sender)

{

Form1->Close();

}

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::RESETClick(TObject \*Sender)

{

n = 0;

text->Text = IntToStr(n);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::textKeyDown(TObject \*Sender, WORD &Key,

TShiftState Shift)

{

switch( Key )

{

case VK\_DOWN:

n = n - 1;

text->Text = IntToStr(n);

break;

case VK\_DELETE:

ShowMessage("Can't Delete This");

break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::textKeyUp(TObject \*Sender, WORD &Key,

TShiftState Shift)

{

switch( Key )

{

case VK\_UP:

n = n + 1;

text->Text = IntToStr(n);

break;

case VK\_DELETE:

ShowMessage("Can't Delete This");

break;

}

}

//---------------------------------------------------------------------------

*Counter.h*

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TButton \*DOWN;

TButton \*UP;

TEdit \*text;

TButton \*EXIT;

TLabel \*Label1;

TButton \*RESET;

void \_\_fastcall UPClick(TObject \*Sender);

void \_\_fastcall DOWNClick(TObject \*Sender);

void \_\_fastcall EXITClick(TObject \*Sender);

void \_\_fastcall RESETClick(TObject \*Sender);

void \_\_fastcall textKeyDown(TObject \*Sender, WORD &Key,

TShiftState Shift);

void \_\_fastcall textKeyUp(TObject \*Sender, WORD &Key,

TShiftState Shift);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

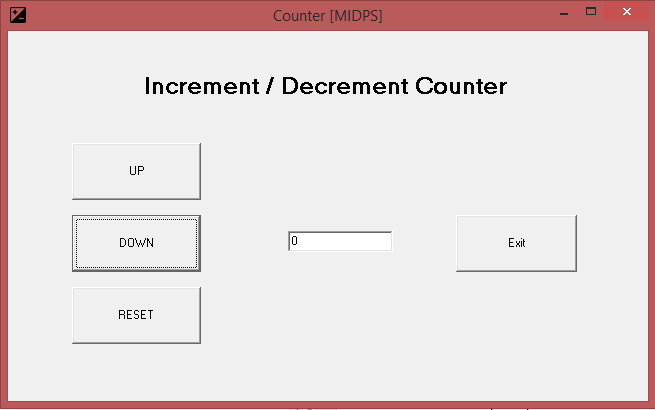
//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

*Screenshot:*



1. **Codul programului în limbajul C++ utilizând C++Builder**

*Project.cpp*

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

//---------------------------------------------------------------------------

USEFORM("Unit1.cpp", Form1);

//---------------------------------------------------------------------------

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

{

try

{

Application->Initialize();

Application->CreateForm(\_\_classid(TForm1), &Form1);

Application->Run();

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

catch (...)

{

try

{

throw Exception("");

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

}

return 0;

}

//---------------------------------------------------------------------------

*Stopwatch.cpp*

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma link "CSPIN"

#pragma resource "\*.dfm"

#include "dos.h"

#include <stdio.h>

struct date d;

struct time t;

unsigned int seconds = 0;

unsigned int minutes = 0;

unsigned int hours = 0;

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormActivate(TObject \*Sender)

{

char buf[20];

char buf2[30];

sprintf(buf," 00 hr 00 min 00 sec");

sprintf(buf2," 00-00-0000 00:00:00");

Timer->Text=(AnsiString)buf;

CurrentTime->Text=(AnsiString)buf2;

Timer1->Interval=1000;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer1Timer(TObject \*Sender)

{

char buf[20];

getdate(&d);

gettime(&t);

sprintf(buf," %02d-%02d-%4d %02d:%02d:%02d",d.da\_day,d.da\_mon,d.da\_year,

t.ti\_hour,t.ti\_min,t.ti\_sec);

CurrentTime->Text=(AnsiString)buf;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer2Timer(TObject \*Sender)

{

char buf[20];

seconds++;

if(seconds == 60){

minutes++;

seconds = 0;

}

if(minutes == 60){

hours++;

minutes = 0;

}

if(hours == 24){

hours = 0;

seconds = 0;

minutes = 0;

}

sprintf(buf," %02d hr %02d min %02d sec", hours, minutes, seconds);

Timer->Text=(AnsiString)buf;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::STARTClick(TObject \*Sender)

{

Timer2->Enabled = true;

START->Enabled = false;

RESET->Enabled = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::STOPClick(TObject \*Sender)

{

Timer2->Enabled = False;

START->Enabled = true;

RESET->Enabled = true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::RESETClick(TObject \*Sender)

{

char buf[20];

hours = 0;

minutes = 0;

seconds = 0;

sprintf(buf," %02d hr %02d min %02d sec", hours, minutes, seconds);

Timer->Text=(AnsiString)buf;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::EXITClick(TObject \*Sender)

{

Form1->Close();

}

//---------------------------------------------------------------------------

*Stopwatch.h*

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ExtCtrls.hpp>

#include "CSPIN.h"

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TEdit \*CurrentTime;

TLabel \*Label1;

TEdit \*Timer;

TButton \*START;

TButton \*STOP;

TButton \*RESET;

TTimer \*Timer1;

TTimer \*Timer2;

TButton \*EXIT;

TGroupBox \*GroupBox2;

void \_\_fastcall FormActivate(TObject \*Sender);

void \_\_fastcall Timer1Timer(TObject \*Sender);

void \_\_fastcall Timer2Timer(TObject \*Sender);

void \_\_fastcall STARTClick(TObject \*Sender);

void \_\_fastcall STOPClick(TObject \*Sender);

void \_\_fastcall RESETClick(TObject \*Sender);

void \_\_fastcall EXITClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

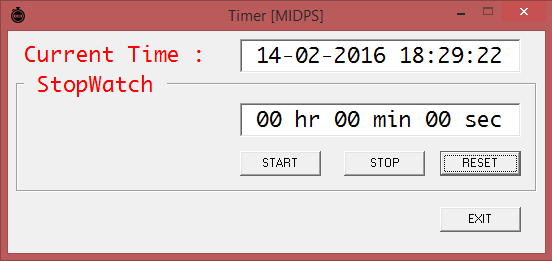
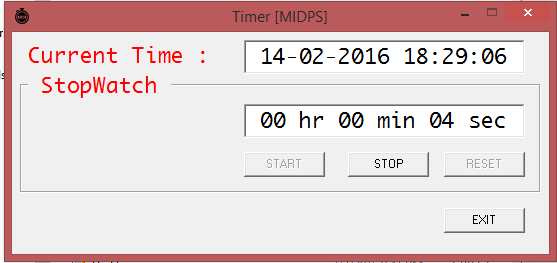
//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

*Screenshot:*



1. **Codul programului în limbajul C++ utilizând C++Builder**

*Project.cpp*

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

//---------------------------------------------------------------------------

USEFORM("Unit2.cpp", Form2);

//---------------------------------------------------------------------------

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

{

try

{

Application->Initialize();

Application->CreateForm(\_\_classid(TForm2), &Form2);

Application->Run();

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

catch (...)

{

try

{

throw Exception("");

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

}

return 0;

}

//---------------------------------------------------------------------------

*Histogram.cpp*

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include <dos.h>

#include "Unit2.h"

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm2 \*Form2;

//---------------------------------------------------------------------------

\_\_fastcall TForm2::TForm2(TComponent\* Owner)

: TForm(Owner)

{

srand(time(NULL));

Timer2->Enabled = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::Timer1Timer(TObject \*Sender)

{

struct time t;

struct date d;

gettime(&t);

getdate(&d);

char buf[30];

sprintf(buf, "%02d-%02d-%02d %02d:%02d:%02d", d.da\_day, d.da\_mon, d.da\_year,t.ti\_hour,t.ti\_min,t.ti\_sec);

Edit1->Text = buf;

}

//---------------------------------------------------------------------------//---------------------------------------------------------------------------}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::PaintBox1Paint(TObject \*Sender)

{

PaintBox1->Canvas->Pen->Color = clBlack;

PaintBox1->Canvas->Brush->Style = bsHorizontal;

for(int i = 0; i <= 240; i += 10){

PaintBox1->Canvas->MoveTo(0,i);

PaintBox1->Canvas->LineTo(240,i);

PaintBox1->Canvas->MoveTo(i,0);

PaintBox1->Canvas->LineTo(i,240);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::Timer2Timer(TObject \*Sender)

{

static int x = 0;

int sign = rand() % 2 == 0 ? 1 : -1;

static int y = ((rand() % 2) \* sign + 120);

PaintBox1->Canvas->Pen->Color = clRed;

PaintBox1->Canvas->Brush->Style = bsHorizontal;

Panel1->Height = y;

PaintBox1->Canvas->MoveTo(x , y);

x =((rand() % 3) + x);

y = ((rand() % 40) \* sign + 120);

PaintBox1->Canvas->LineTo(x,y);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::Button1Click(TObject \*Sender)

{

Timer2->Enabled = true;

Button1->Enabled = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::Button2Click(TObject \*Sender)

{

Button1->Enabled = true;

Timer2->Enabled = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::Button3Click(TObject \*Sender)

{

Form2->Close();

}

//---------------------------------------------------------------------------

*Histogram.h*

//---------------------------------------------------------------------------

#ifndef Unit2H

#define Unit2H

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ExtCtrls.hpp>

//---------------------------------------------------------------------------

class TForm2 : public TForm

{

\_\_published: // IDE-managed Components

TPaintBox \*PaintBox1;

TPanel \*Panel1;

TPanel \*Panel2;

TButton \*Button1;

TButton \*Button2;

TButton \*Button3;

TTimer \*Timer1;

TTimer \*Timer2;

TEdit \*Edit1;

TLabel \*Label1;

TLabel \*Label2;

TLabel \*Label3;

TLabel \*Label4;

void \_\_fastcall Timer1Timer(TObject \*Sender);

void \_\_fastcall PaintBox1Paint(TObject \*Sender);

void \_\_fastcall Timer2Timer(TObject \*Sender);

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall Button2Click(TObject \*Sender);

void \_\_fastcall Button3Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm2(TComponent\* Owner);

};

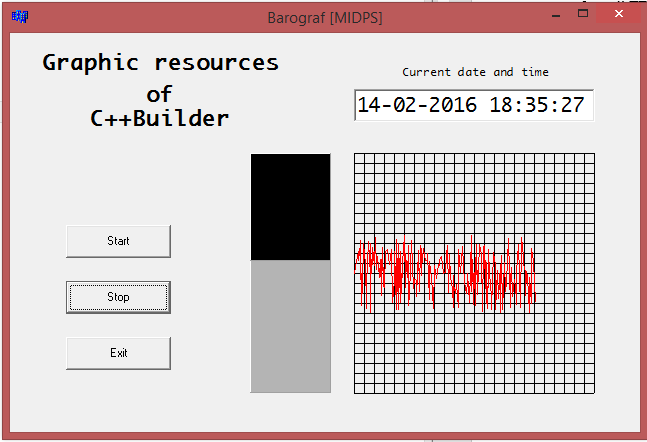
//---------------------------------------------------------------------------

extern PACKAGE TForm2 \*Form2;

//---------------------------------------------------------------------------

#endif

*Screenshot:*



**Concluzii:**

1. În această lucrare am pus în practică IDE-ul C++Builder pentru crearea ferestrelor în Windows cu diverse funcții, ca incrementarea decrementarea, cronometru și barograf.
2. Am învățat cum se creează metode care pot prelucra diferite evenimente

**void\_fastcall Tform1::Button1Click(TObject \*Sender)**

**{**

**//Aici putem scrie cod sursă, care se va executa**

**//în momentul apăsării butonului cu ajutorul mouse-ului**

**}**

1. Am lucrat cu **TCanvas, TPaintBox, TTimer**  și alte componente a C++Builder-ului care mi-au ușurat munca.

|  |
| --- |
| MoveTo(int x,int y) |
| LineTo (int x,int y); |

1. Și m-am învățat să fac un barograf cu ajutorul funcțiilor

**Bibliografie:**

* C++ Primer Plus, Stephen Prata
* Îndrumar metodic pentru lucrările de laborator la MIDPS