Ministerul Educaţiei Republicii Moldova
Universitatea Tehnică a Moldovei

RAPORT

La disciplina: Metode Integrate de Dezvoltare
a Produselor Soft

A efectuat: Petrenco Mihai
st.gr.TI-144

A verificat: Cojocaru Svetlana
lect.univ.

Chişinău, 2016

Lucrarea de laborator Nr.1

Obiectivele 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).

Sarcina lucrarii

1. Vor fi examinate toate componentele prezentate în indicatii teoretice;

2. Se modifică programul din *Project1.cpp* astfel încât să se obţină forma cu obiecte din figura 4.1 ;

3. Se elaborează un program pentru realizarea unui cronometru.

4. Se elaborează un program pentru realizarea a două elemente de afişare (bargraf şi diagramă cu avans continuu).

Noţiuni teoretice:

C++ Builder reprezintă un mediu RAD (rapid application developement), creat de Borland, destinat pentru crearea aplicaţiilor în limbajul C++ , pentru Windows, iOS, Android, etc. El combină librăria vizuală VCL (Visual Component Library) şi IDE-ul scris în Delphi cu un compilator pentru C++.

C++ Builder conţine un set de instrumente destinate pentru tragerea şi plasarea componentelor visuale, făcînd astfel programarea mai uşoară.

Borland C++ Builder este constituit din următoarele instrumente, care permit crearea aplicaţiilor într-un mod rapid şi eficient. Ele sunt:

- Visual Form Designer;
- Object Inspector;
- Component Palette;
- Project Manager;
- Code Editor;
- Debugger;

Efectuarea lucrării:

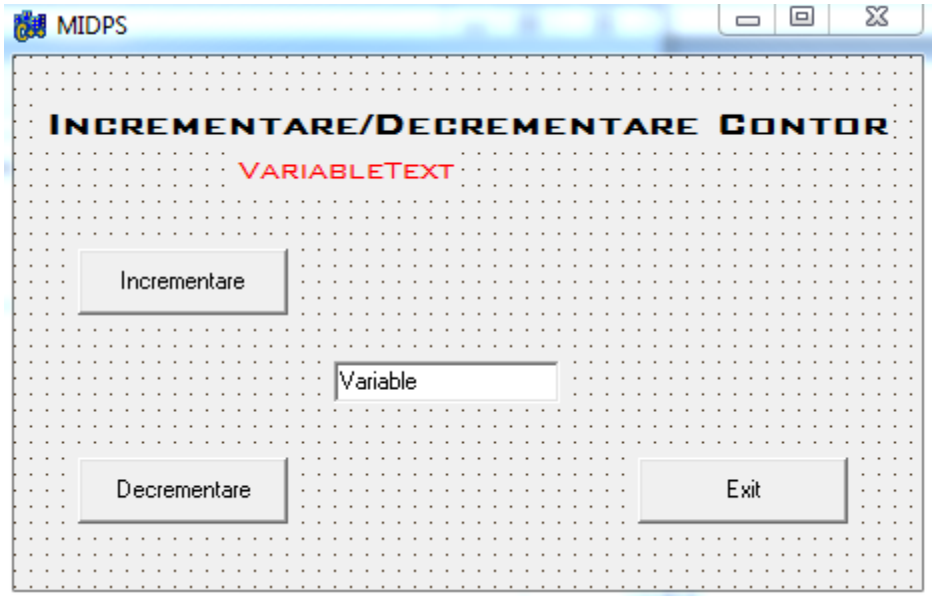
1. ***Crearea unui incrementor / decrementor:***

Programul dat crează o variabilă globală de tip integer care înregistrează valoarea incrementată / decrementată. Odată cu apăsarea butonului de incrementare este apelată funcţia IncrementClick sau DecrementClick, care incrementează/decrementează variabila şi afişează valoarea nouă în spaţiul pentru editare Variable. Cu ajutorul unui label, numit VariableText, este afisat mesajul respectiv de incrementare sau decrementare.

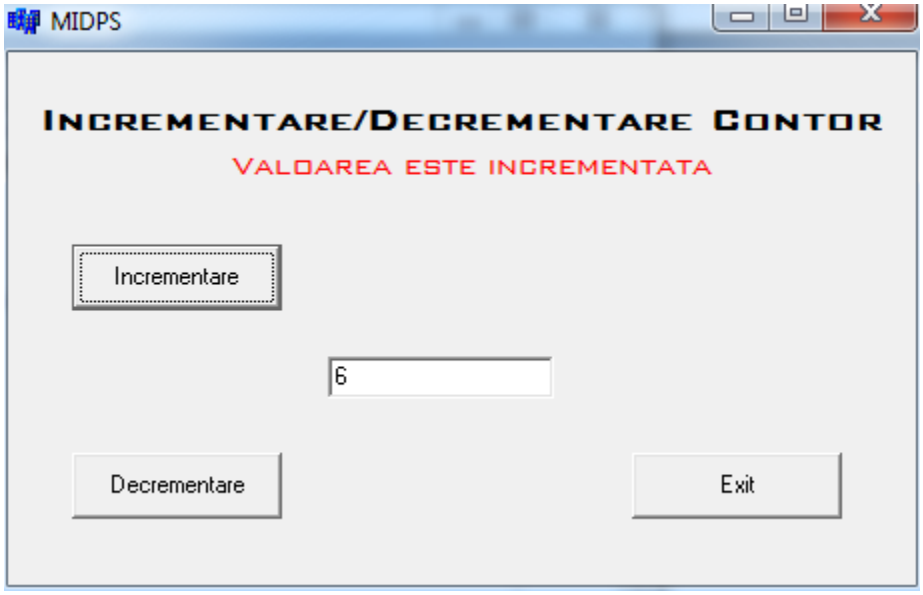
Codul:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int i = 0;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
        Variable-> Text = i;
        VariableText->Caption = "";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::IncrementClick(TObject *Sender)
{
        Variable-> Text = ++i;
        VariableText->Caption = "Valoarea este incrementata";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::DecrementClick(TObject *Sender)
{
        Variable-> Text = --i;
        VariableText->Caption = "Valoarea este decrementata";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ExitClick(TObject *Sender)
{
        Close();
}
//---------------------------------------------------------------------------
```

Forma:



Rezultat:



2. *Crearea unui cronometru:*

Programul introduce libraria dos.h pentru a utiliza structurile de date pentru timpul şi data curentă. Sunt utilizaţi doi timeri, unul pentru afisarea timpului, altul pentru afisarea cronometrului. Afisarea timpului are loc prin intermediul funcţiei getdate() şi gettime(), care mai apoi sunt înregistrate în structura de date şi afişate prin intermediul funcţiei sprintf();

Cronometrul lucrează prin intermediul a 3 variabile, una pentru calcularea milisecundelor, una pentru secunde şi ultima pentru minute. Cu ajutorul timer-ului, variabila pentru milisecunde este incrementată la un interval de 100ms. Astfel, cînd numărul de milisecunde

vor fi egale cu o secundă, numărul de secunde se va incrementa, iar variabila pentru milisecunde va fi resetată la 0. În mod analog lucrează şi variabila minutelor. Butonul RESET resetează valorile variabililor înapoi la 0. Controlul butoanelor se face prin cîmpul Enable, care poate avea valoarea booleană – true sau false.

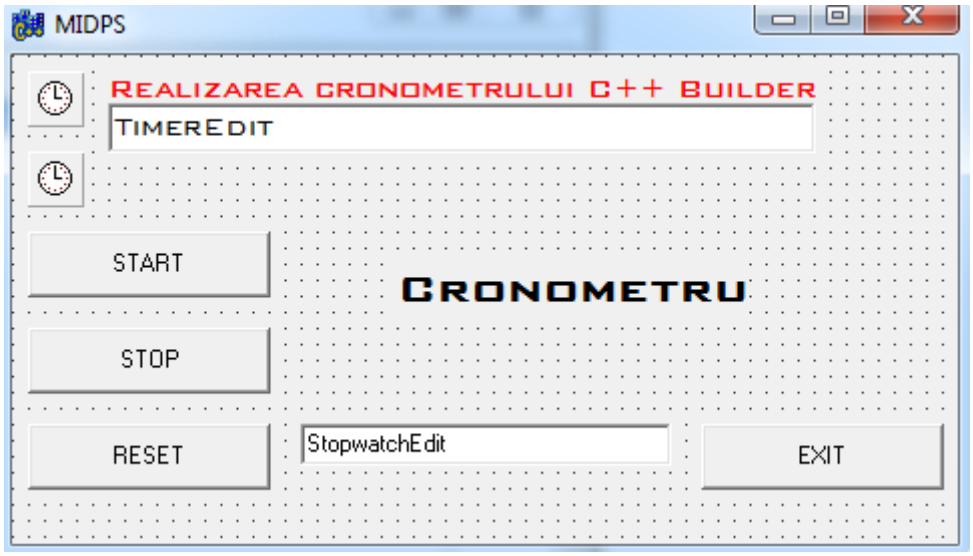
Codul:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
#include "dos.h"
#include <stdio.h>

struct date d;
struct time t;

int ms=0;
int sec=0;
int min=0;
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
   TimerEdit->Clear();
   StopwatchEdit->Clear();
   STOP->Enabled = false;
   Stopwatch->Enabled = false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::STARTClick(TObject *Sender)
{
        START->Enabled = false;
        STOP->Enabled = true;
        Stopwatch->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::STOPClick(TObject *Sender)
{
        START->Enabled = true;
        STOP->Enabled = false;
```

```
        Stopwatch->Enabled = false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::TimeTimer(TObject *Sender)
{
     char buff[20];
     getdate(&d);
     gettime(&t);
     sprintf(buff,"%02d-%02d-%4d\t\t%02d:%02d:%02d",d.da_day,d.da_mon,d.da_year,
     t.ti_hour,t.ti_min,t.ti_sec);

     TimerEdit->Text=(AnsiString)buff;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::StopwatchTimer(TObject *Sender)
{
        ms+=10;
        if(ms == 100)
        {
          sec++;
          ms = 0;
        }
        if(sec == 60)
        {
          min++;
          sec=0;
        }

        char buff[20];
        sprintf(buff,"%02d:%02d:%02d",min,sec,ms);
        StopwatchEdit->Text = (AnsiString)buff;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::EXITClick(TObject *Sender)
{
        Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::RESETClick(TObject *Sender)
{
        ms = 0;
        sec = 0;
        min = 0;

        char buff[20];
        sprintf(buff,"%02d:%02d:%02d",min,sec,ms);
        StopwatchEdit->Text = (AnsiString)buff;
```

}
//---

Forma:



Rezultatul:

3. *Crearea elementelor de afişare:*

Programul dat utilizează funcţiile grafice pentru afişarea unor primitive grafice, cum ar fi linia sau dreptunghiul. Cele două elemente grafice ale programului sunt bargraful şi graficul. Bargraful a fost realizat prin intermediul a două panele, una neagră iar alta roşie. Panelul roşu îşi modifica înălţimea în dependenţă de poziţia Y specificată. Graficul a fost realizat prin intermediul a mai multor funcţii. Iniţial, a fost creată area de desen prin intermediul PaintBox. A fost desenat dreptunghiul cu funcţia Rectangle() şi a fost specificat stilul. Mai apoi, cu ajutorul funcţiei MoveTo(x,y) şi LineTo(x,y) a fost desenată linia grafului cu dimensiuni aleatoare (random).

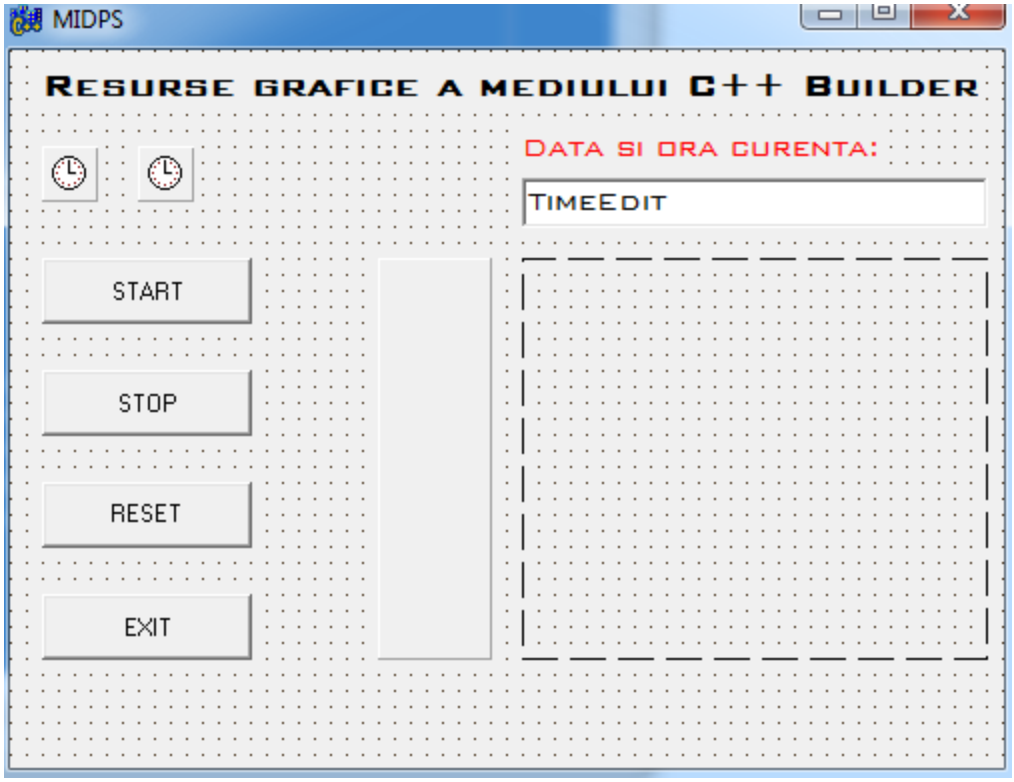
Codul:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit3.h"
#include "dos.h"
#include <stdio.h>
#include <math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

struct date d;
struct time t;

int x = 0;
int y = 100;
int step = 0;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
   TimeEdit->Clear();
   PanelRed->Height = 0;
   PanelRed->Color = RGB(255,0,0);
   PanelBlack->Color = RGB(0,0,0);
   DrawTimer->Enabled = false;
   DrawTimer->Interval = 100;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::TimeTimer(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);
        TimeEdit->Text=(AnsiString)buf;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::STARTClick(TObject *Sender)
{
```

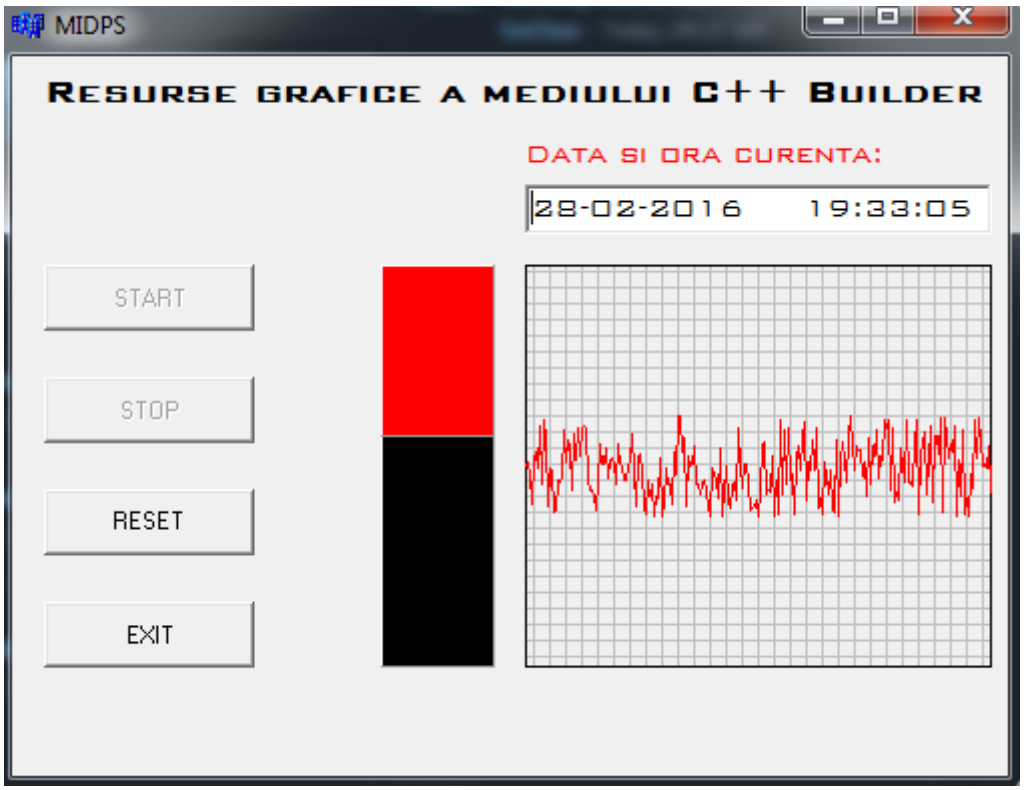
```
        START->Enabled = false;
        STOP->Enabled = true;
        DrawTimer->Enabled = true;
        DrawingArea -> Canvas -> Pen -> Color = clBlack;
        DrawingArea -> Canvas -> Brush -> Color = clSilver;
        DrawingArea -> Canvas -> Brush -> Style = bsCross;
        DrawingArea -> Canvas -> Rectangle(0,0,233,201);
        DrawingArea -> Canvas -> Pen -> Color = clRed;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::STOPClick(TObject *Sender)
{
        START->Enabled = true;
        STOP->Enabled = false;
        DrawTimer->Enabled = false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::EXITClick(TObject *Sender)
{
        Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::DrawTimerTimer(TObject *Sender)
{
        DrawingArea -> Canvas -> MoveTo(x,y);

        do
        {
          if (step % 2 == 0)
          {
            y = y + rand() % 50;
            step++;
          }

          else
          {
            y = y - rand() % 50;
            step++;
          }

        }while (y < 75 || y > 125);

        x +=1 ;
        PanelRed -> Height = y;
```

```
        DrawingArea -> Canvas -> LineTo(x,y);
        if (x >= 233)
        {
          DrawTimer -> Enabled = false;
          //PanelRed -> Height = 0;
          STOP -> Enabled = false;
          START -> Enabled = false;
        }
}
void __fastcall TForm1::RESETClick(TObject *Sender)
{
        if (x >= 233)
        START->Enabled = true;

        DrawingArea->Repaint();
        DrawingArea -> Canvas -> Pen -> Color = clBlack;
        DrawingArea -> Canvas -> Brush -> Color = clSilver;
        DrawingArea -> Canvas -> Brush -> Style = bsCross;
        DrawingArea -> Canvas -> Rectangle(0,0,233,201);
        DrawingArea -> Canvas -> Pen -> Color = clRed;
        y = 100;
        x = 0;
}
//---------------------------------------------------------------------------
```

Forma:



Rezultatul:



Concluzie:

În urma efectuării lucrării de laborator, am obţinut cunoştinţe practice de lucru cu mediul de programare Borland C++ Builder. Am testat unele instrumente ce ne permit să creăm aplicaţii rapid şi eficient prin intermediul drag-and-drop şi ne-am familiarizat cu crearea programelor cu interfaţă grafică. Aceste instrumente ne-au permis să creăm aplicaţii scriind un număr minim de cod.

Am analizat cele mai utilizate funcţii ale mediului de programare, şi am obţinut cunoştinţe despre cum lucrează evenimentele unui obiect.