Ministerul Educaţiei al Republicii Moldova

Universitatea Tehnică a Moldovei
Catedra Tehnologii Informaţionale

RAPORT

Lucrarea de laborator#1

la Medii Interactive de Dezvoltare a Produselor Soft

A efectuat: Profir Andrei
st.gr. TI – 143

A verificat: Cojanu Irina
lect.asist.

Chişinău 2016

Tema: Mediul intergrat C++ Builder

Scopul lucrării:

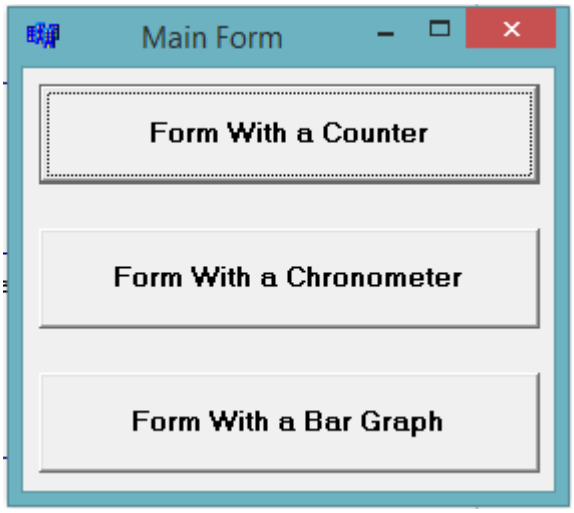
- Însuşirea modului de utilizare a celor mai importante componente ale mediului integrat C++ BUILDER . Realizarea unui program simplu care utilizează componente de tip *TButton, TEdit, Tlabel, RadioButton* etc.
- Însuşirea modului de utilizare a componentei VCL **TTimer.** Însuşirea modului de utilizare a funcţiilor de lucru cu timpul sistem. Realizarea unor aplicaţii de gestionare a resursei timp.
- 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):

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

Implimentare task-uri:

De la inceput a fost creat o forma care este principala, prin intermediul acestei forme putem accesa celelelte task-uri.



Primul butonului ne permite accesarea task-ului care realizeaza un contor.

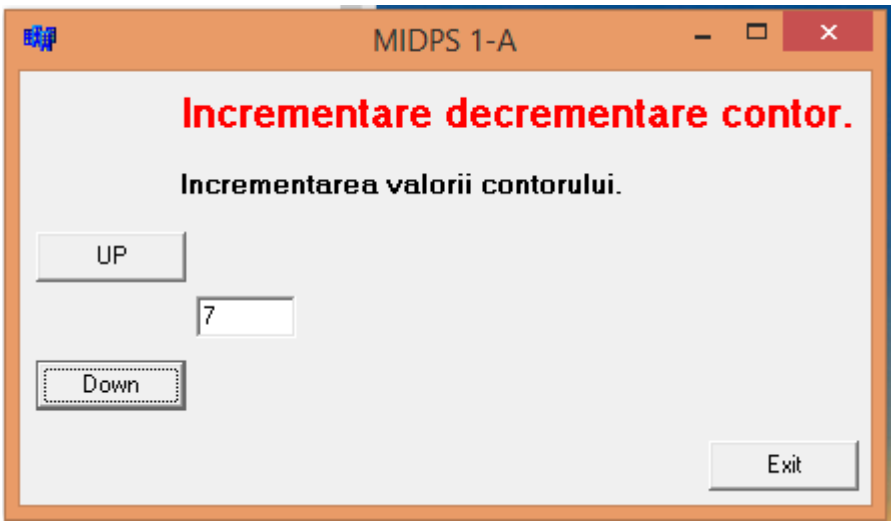
Prin intermediul celui dea-al doilea buton putem accesa o forma care realizeaza un cronometru.

In final cu ajutorul butonului trei putem vedea task-ul care realizeaza doua elemente: un bar graph si p diagram cu avans continuu.

```
//---------------------------------------------------------
void __fastcall TFMain::Button1Click(TObject *Sender)
{
FMain->Hide();
FCounter->ShowModal();
}
//---------------------------------------------------------

void __fastcall TFMain::Button2Click(TObject *Sender)
{
FMain->Hide();
FChronometer->ShowModal();
}
//---------------------------------------------------------

void __fastcall TFMain::Button3Click(TObject *Sender)
{
FMain->Hide();
FBarGraph->ShowModal();
}
```

Aici putem vedea mai detaliat cum a fost realizata functionalitatea acestei forme.

- Se elaborează un program pentru realizarea unui contor cu funcțiile incrementare/decrementare.

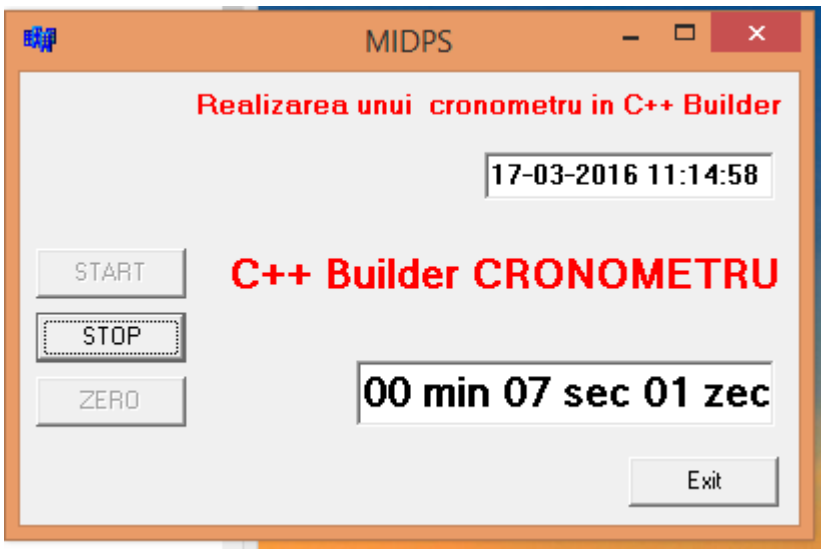


Realizarea acestui task se incepe de la declararea unei variabile de tip int care va reprezenta valoarea curenta a contorului. Dupa care pe forma se adauga doua butoane care permita marirea sau micsoararea valorii contorului. Se mai adauga si un textfield prin intermediul caruia putem vedea valoarea contorului la fiecare modificare a lui. In final a fost adaugat un buton „Exit” care permite inchiderea acestei forme. La fel pe forma au mai fost adaugate doua label caption-ul unui din aceste se modifica la fiecare schimbare a valorii contorului.

```
//---------------------------------------------------------------------------
__fastcall TFCounter::TFCounter(TComponent* Owner)
        : TForm(Owner)
{
counterValue = 0;
}

void TFCounter::increment() {
counterValue++;
}
void TFCounter::decrement() {
counterValue--;
}
int TFCounter::getValueCounter() {
return counterValue;
}

//---------------------------------------------------------------------------
void __fastcall TFCounter::btnExitClick(TObject *Sender)
{
FCounter->Close();
}
//---------------------------------------------------------------------------
void __fastcall TFCounter::FormClose(TObject *Sender, TCloseAction &Action)
{
FMain->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFCounter::btnUpClick(TObject *Sender)
{
FCounter->decrement();
Label2->Caption = "Decrementarea valorii contorului." ;
Edit1->Text = FCounter->getValueCounter();
}
//---------------------------------------------------------------------------

void __fastcall TFCounter::btnDownClick(TObject *Sender)
{
FCounter->increment();
Label2->Caption = "Incrementarea valorii contorului.";
Edit1->Text = FCounter->getValueCounter();
}
```

Codul sursa care arata mai explicit in ce mod a fost realizata functionalitatea formei.

- Se elaborează un program pentru realizarea unui cronometru.



La realizarea acestui task principalul element este timer-ul. Aici au fost folosite 2 componente timer unul cu intervalul 1s. si al doilea cu intervalul 100ms. Primul timer cu intrevalul de 1sec este folosit pentru a afisa data si ora curenta, si faptul ca el la fiecare secunda genereaza un event ne permite sa actualizam ora si data. Ora si data se citesc din sistem cu ajutorul functiilor getdate si gettime. Al doilea timer se foloseste pentru a crea un cronometru, valorile cronometrului sunt zeci de secunda, secunde si minute, aceste 3 valori sunt pastrate in 3 variabile integer si in dependenta de evenimentul timer-ului se incrementeaza.
Mai jos se poate observa mai detaliat in ce mod a fost realizata pedeplin functionalitatea cronometrului.

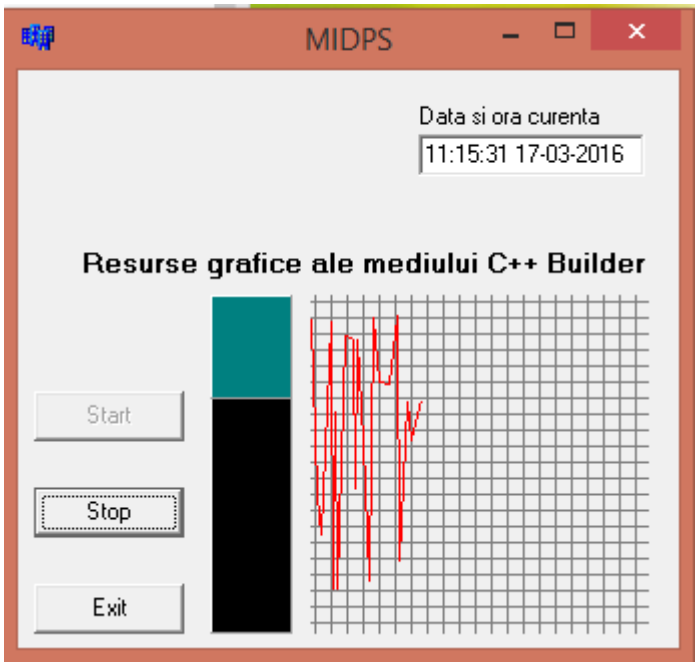
```
struct date d;
struct time t;

struct Chrono {
 int min;
 int sec;
 int zec;
} cr;

TFChronometer *FChronometer;
//---------------------------------------------------------------------
__fastcall TFChronometer::TFChronometer(TComponent* Owner)
        : TForm(Owner)
{
cr.min = 0;
cr.sec = 0;
cr.zec = 0;
char buf[20];
sprintf(buf,"%02d min %02d sec %02d zec",cr.min,cr.sec, cr.zec);
Edit2->Text=(AnsiString)buf;
}
//---------------------------------------------------------------------
void __fastcall TFChronometer::btnExitClick(TObject *Sender)
{
FChronometer->Close();
}
```

```
//---------------------------------------------------------------------
void __fastcall TFChronometer::FormClose(TObject *Sender,
      TCloseAction &Action)
{
FMain->Show();
}
//---------------------------------------------------------------------
void __fastcall TFChronometer::btnStartClick(TObject *Sender)
{
Timer2->Enabled = true;
btnStart->Enabled = false;
btnStop->Enabled = true;
btnZero->Enabled = false;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TFChronometer::Timer2Timer(TObject *Sender)
{
cr.zec++;
if (cr.zec == 10) {
cr.zec = 0;
cr.sec++;
}
if (cr.sec == 60) {
cr.sec = 0;
cr.min++;
}
char buf[20];
sprintf(buf,"%02d min %02d sec %02d zec",cr.min,cr.sec, cr.zec);
Edit2->Text=(AnsiString)buf;
}
//---------------------------------------------------------------------------
void __fastcall TFChronometer::btnStopClick(TObject *Sender)
{
btnStart->Enabled = true;
btnStop->Enabled = false;
btnZero->Enabled = true;
Timer2->Enabled = false;
}
//---------------------------------------------------------------------------
void __fastcall TFChronometer::btnZeroClick(TObject *Sender)
{
btnStart->Enabled = true;
btnStop->Enabled = false;
btnZero->Enabled = false;
cr.min = 0;
cr.sec = 0;
cr.zec = 0;
char buf[20];
sprintf(buf,"%02d min %02d sec %02d zec",cr.min,cr.sec, cr.zec);
Edit2->Text=(AnsiString)buf;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TFChronometer::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);
Edit1->Text=(AnsiString)buf;
}
```

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



Pentru realizarea bargraph-ului si diagramei a fost creat un timer cu uninterval de 500ms de fiecare dat generind un numar random. Numarul acesta este salvat intr-o variabila de tip int si dupa care el este reprezentat pe bargraph si pe diagrama. Bargraphul de fapt sunt doua componente panel care se suprapun si cind este generat o valoare random panel-ul care este deasupra isi schimba dimensiunile. Pe diagrama valoare reprezentata pe bargraph este valoare pe axa „y”, iar axa „x” tot variaza in dependenta de un numar generat care poate fi 1, 2, 3 aceasta ne permite la fiecare generare a unui numar random diagrama sa se deplaseze si treptat sa fie afisata diagrama pe toata dimensiunea PaintBox-ului. Cind toata suprafata PaintBox-ului a fost umpluta se foloseste functia CopyRect pentru a deplasa diagrama in stinga.

```
struct date d;
struct time t;

int valueY = 0;
int valueX = 0;

TFBarGraph *FBarGraph;
```

```
//---------------------------------------------------------------------
__fastcall TFBarGraph::TFBarGraph(TComponent* Owner)
        : TForm(Owner)
{
std::srand(std::time(NULL));
valueY = rand() % 151;
Panel2->Height = valueY;
PaintBox1->Canvas->MoveTo(0, valueY);
}
//---------------------------------------------------------------------
void __fastcall TFBarGraph::btnExitClick(TObject *Sender)
{
FBarGraph->Close();
}
//---------------------------------------------------------------------
void __fastcall TFBarGraph::FormClose(TObject *Sender,
      TCloseAction &Action)
{
FMain->Show();
}
//---------------------------------------------------------------------
void __fastcall TFBarGraph::Timer1Timer(TObject *Sender)
{
char buf[20];
getdate(&d);
gettime(&t);
sprintf(buf,"%02d:%02d:%02d %02d-%02d-%4d",
t.ti_hour,t.ti_min,t.ti_sec,d.da_day,d.da_mon,d.da_year);
Edit1->Text=(AnsiString)buf;
}
//---------------------------------------------------------------------

void __fastcall TFBarGraph::btnStartClick(TObject *Sender)
{
Timer2->Enabled = true;
btnStart->Enabled = false;
btnStop->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TFBarGraph::Timer2Timer(TObject *Sender)
{
valueY = rand() % 151;
valueX += (rand() % 5) + 1;
Panel2->Height = valueY;
PaintBox1->Canvas->LineTo(valueX, valueY);
}if (valueX > 139) {
TRect sursa,destinatie;
sursa=Rect(0,0,169,169);
destinatie=Rect(-(168 - valueX),0,170 - (169 - valueX),169);
PaintBox1->Canvas->CopyRect(destinatie,PaintBox1->Canvas,sursa);
valueX -= (169 - valueX);
}
PaintBox1->Canvas->MoveTo(valueX, valueY);
}
```

```
//---------------------------------------------------------------------
void __fastcall TFBarGraph::btnStopClick(TObject *Sender)
{
Timer2->Enabled = false;
btnStart->Enabled = true;
btnStop->Enabled = false;
}
//---------------------------------------------------------------------

void __fastcall TFBarGraph::PaintBox1Paint(TObject *Sender)
{
  TRect dreptunghi;
  PaintBox1->Canvas->Pen->Color = clRed;
  PaintBox1->Canvas->Pen->Width = 1;
  PaintBox1->Canvas->Brush->Style = bsCross;
  PaintBox1->Canvas->Brush->Color = clGray;
  PaintBox1->Canvas->FloodFill(10, 10, clGray, fsBorder);
}
//---------------------------------------------------------------------
```

Link-ul catre repozitoriu online:

https://github.com/ProfirAndrei/MIDPS/Lab-1

Concluzie:

In urma efectuarii acestei lucrari de laborator au fost capatate in lucrul cu IDE C++ Builder. Au fost folosite mai multe componente de control care permit crearea interfetelor grafice rapid si simplu.

Am capatat depreinderi in lucrul cu butoane, casete de editare, “label” – uri etc. Am folosit componenta Timer pentru a afisa data si ora curenta si pentru a efectua niste operatii la un interval anumit de timp. Timer-ul ne-a ajutat in crearea cronometrului si a unui bargraph pe care erau reprezentate niste numere generate random.

La general putem spune ca C++ Builder este un mediu care ne face unele deprinderi dar totusi sper niciodata sa nu ma mai intorc la el fiindca lucrul in el este greu macar de aceea ca codul nu se aranjeaza asa cum trebuie si cind avem un cod destul de mic inca ne putem descurca dar volum codului sursa creste acest IDE lasa de dorit.