STUDIA UNIVERSITATIS
Revistă științifică a Universității de Stat din Moldova, 2012, nr.9(59)

METODOLOGIA ELABORĂRII APLICAȚIEI EDUCAȚIONALE „ORDINEA EFECTUĂRII OPERAȚIILOR” ÎN MEDIUL DE PROGRAMARE DELPHI

Tatiana CROITOR-CHIRIAC, Marina BOSTAN
Universitatea Pedagogică de Stat „Ion Creangă”

Elaborarea unui software educațional în Borland Delphi (un mediu de dezvoltare integrat pentru a construi ferestre Windows, web şi aplicații mobile) reprezintă astăzi una dintre cele mai potrivite soluții pentru a crea aplicații independent, eficient şi într-un timp foarte scurt. Mediul Delphi oferă un număr mare de aşa-numite componente reutilizabile, care pot fi interconectate pentru a elabora programe didactice valoroase.

În articol se prezintă o metodologie de proiectare a aplicațiilor educaționale pe baza programului numit „Ordinea efectuării operațiilor”, unul dintre secvențele curriculare ale matematicii.

Cuvinte-cheie: *Borland Delphi, programul „Ordinea efectuării operațiilor”, aplicații educaționale.*

METHODOLOGY FOR DESIGNING EDUCATIONAL APPLICATIONS ON THE BASE OF THE PROGRAM NAMED “ORDER OF OPERATION” IN DELPHI SOFTWARE

Designing educational software in Borland Delphi, an integrated development environment for building Windows, web and mobile applications, represents nowadays one of the most suitable solutions to create self-contained, efficient and reliable applications in a short time. Delphi offers a large number of so-called reusable components, which can be interconnected to develop valuable didactic programs.

The present paper has developed a methodology for the design of educational applications on the base of the program named “Order of operations”, one of the curricular sequence of mathematics.

Keywords: *Borland Delphi, “Order of operations” program, educational applications.*

Introducere

Elaborarea aplicațiilor educaționale cu ajutorul tehnicilor de programare a unui limbaj orientat-obiect reprezintă actualmente o direcție nouă vizată de prevederile curriculare la disciplina *Informatica*.

Chiar dacă limbajele de programare orientate pe obiect sunt dificil de învățat şi utilizat, aplicarea acestora prin intermediul unui mediu de programare vizuală în elaborarea programelor didactice evocă una din tehnologiile informaționale de vârf cu referire la metodologia de concepere a produsului educațional performant şi complex.

Există mai multe medii de programare orientate pe obiect, cum ar fi: Borland Delphi, C++, PHP, C-Builder, Visual Basic etc., în care se pot realiza programe executabile cu interfețe de tip Windows, eficiente în soluționarea / optimizarea mai multor probleme didactice.

În acest articol, implementarea informatică în elaborarea aplicației educaționale „Ordinea efectuării operațiilor” s-a bazat pe instrumentarul mediului de programare Borland Delphi. La baza aplicațiilor Delphi se află limbajul Object Pascal, acesta fiind un limbaj orientat-obiect.

Mediul Delphi permite crearea aplicațiilor în mod vizual, adică proiectând un program prin operarea directă asupra unui set de elemente grafice vizuale. Elementele cu ajutorul cărora este realizată interfața unei aplicații formează ***paleta de componente***, ce conține peste o sută de componente organizate în mai multe clase. O componentă reprezintă un icon-obiect care îndeplineşte funcții specifice predefinite, de exemplu: label, buton, richedit, combobox, memo etc. Totalitatea componentelor formează biblioteca de componente vizuale (Visual Components Library (VCL)). Componentele au ***proprietăți*** şi ***evenimente***. Proprietatea controlează modul de operare a unei componente, iar evenimentul este acțiunea în rezultatul condiționării reciproce a unei componente cu utilizatorul.

Proiectarea şi realizarea unei aplicații educaționale ține cont de mai multe aspecte generale psihopedagogice şi metodologice, precum şi de direcția curriculară a disciplinei pentru care se concepe aceasta.

În linii generale, în prezentul studiu vom analiza etapele de realizare informatică a unei aplicații educaționale pe exemplul unei secvențe curriculare la disciplina matematică „Operații aritmetice în concentrul 0-100”, competența „Efectuarea operațiilor aritmetice în exerciții, cu şi fără paranteze” [1, clasa a II-a, p.73-74].

În clasele I–II, elevul rezolvă exerciţii simple în care apar operaţii de acelaşi ordin: adunări, scăderi, înmulţiri sau împărţiri. Aceste exerciţii formează la micul şcolar deprinderea de a efectua operaţiile succesiv, fără să-şi pună problema existenţei unor reguli referitoare la ordinea efectuării acestora.

La finele clasei a II-a, după ce elevii au învăţat cele 4 operaţii cu numere naturale, ei sunt puşi în faţa efectuării unor exerciţii de tipul $a + b \cdot c$. Schimbarea ordinii efectuării operaţiilor duce la rezultate diferite, ceea ce impune stabilirea unor reguli stricte în efectuarea operaţiilor într-un astfel de exerciţiu. În acest caz, pentru a descoperi această ordine, profesorul poate utiliza diferite metode de predare tradiţionale (metoda descoperirii, metoda problematizării şi altele). Iar pentru a forma la elevi competenţe de exersare privind ordinea efectuării operaţiilor aritmetice, profesorul poate apela la un software educaţional elaborat într-un mediu de programare de nivel înalt.

În continuare prezentăm metodologia abordată în dezvoltarea aplicaţiilor educaţionale în mediul de programare Borland Delphi 7.0, pe exemplul sus-numit.

Metodologia elaborării aplicaţiei educaţionale „Ordinea efectuării operaţiilor"

Scopul general: la finele lucrării studenţii trebuie să obţină aplicaţia ilustrată în Figura 1.

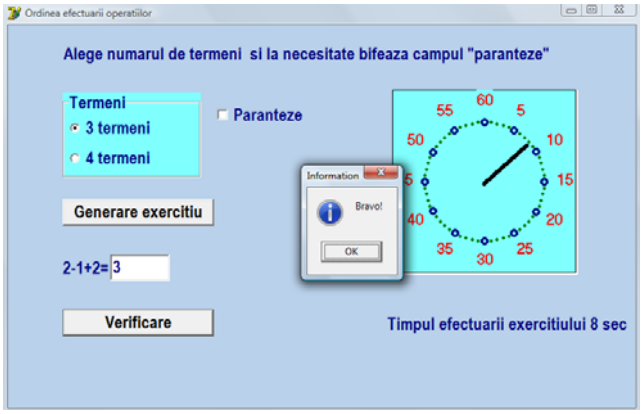

Fig.1. Interfaţa finală a aplicaţiei.

Obiectivele proiectului:

- utilizarea componentelor Label, Edit, Button, Checkbox, RadioGroup;
- proiectarea aplicaţiei;
- setarea proprietăţilor componentelor utilizate;
- prelucrarea evenimentelor necesare;
- realizarea interconexiunilor dintre componentele utilizate;
- realizarea design-ului final al aplicaţiei.

Aspecte teoretice. În cadrul proiectului au fost utilizate următoarele componente [2]:

I. Paleta ***Standard:***

Label – este un control fără fereastră folosit pentru a afişa text pe un form. Acest text poate fi folosit pentru a eticheta alte componente. Atribuirea unei valori proprietăţii *Caption* este echivalentul desenării unui text pe componenta părinte.

Edit – este un control care permite afişarea de text către utilizator sau permite preluarea textului pe care utilizatorul îl tastează folosind proprietatea *Text*.

Button – cel mai simplu tip de buton. Acesta are o etichetă definită de proprietatea *Caption.* Cele mai multe acţiuni sunt executate la apăsarea unui buton. Codul care se va executa la apăsarea unui buton trebuie înscris în interiorul handler-ului de evenimente *OnClick.*

Textul care va fi afişat în componentele ***Label***, ***Edit***, pentru realizarea unor calcule, trebuie să fie convertit în număr şi invers cu ajutorul următoarelor funcţii:

Convertirea	Text în Număr	Număr în Text
numere întregi	StrToInt();	IntToStr();
numere reale	StrToFloat();	FloatToStr();

CheckBox – este o componentă ce constă dintr-o cutie de bifare etichetată care oferă utilizatorului posibilitatea de a alege sau nu o opţiune a aplicaţiei.

RadioGroup – componenta *TRadioGroup* reprezintă un grup de butoane care funcționează împreună. Pentru a adăuga butoane la *TRadioGroup* se va utiliza proprietatea *Items***.** Prin intermediul acesteia se specifică textul ataşat fiecărui buton de radio. Contrar unei impresii de moment, proprietatea *Items* nu pointează spre elemente de tip *TRadioButton***.** Elementele ei nu sunt decât nişte simple şiruri de caractere ce identifică eticheta ataşată fiecărui buton radio. Cu ajutorul proprietății *Itemlndex* se poate determina care buton va fi activ la un moment dat. Butoanele de tip radio sunt, de obicei, grupate. Ele pot fi grupate în cinci moduri: pe un *Form*, într-un *TradioGroup*, într-un *TgroupBox*, într-un *TscrollBox*, într-un *TPanel.* Mai multe butoane radio, aflate în acelaşi grup, nu pot avea simultan statutul activ. Doar unul dintre ele se poate afla în starea de setare, celelalte fiind înactiive. Dacă fac parte din grupuri diferite fiind pe un form, două butoane se pot afla în acelaşi timp în starea de setare.

II. Paleta ***Additional:***

Image – permite afişarea unei imagini pe ecran. Această imagine este valoarea proprietății *Picture*. Poate fi icon, bitmap, metafile sau orice alt obiect grafic definit de utilizator.

III. Paleta ***System:***

Timer – timer-ele anunță utilizatorul despre scurgerea unui interval de timp. Componenta *TTimer* este o rutină care măsoară repetat scurgerea unui interval de timp. După scurgerea acestui interval de timp, a cărui valoare este indicată de către proprietatea *Interval*, sistemul este notificat prin apelarea evenimentului *OnTimer*. Înştiințarea se produce prin generarea unui mesaj. Programatorul poate intercepta acest mesaj sub forma unui eveniment. În interiorul acestui eveniment trebuie scris codul care se va executa la un interval (de obicei, regulat) de timp. Datorită faptului că timer-ul depinde de viteza cu care sunt prelucrate mesajele din coada de mesaje, intervalul scurs va fi aproximativ.

PaintBox – furnizează o componentă *TCanvas* în interiorul unui dreptunghi, prevenind desenarea în afara marginilor acestuia.

Mersul lucrării:

1. Plasarea componentelor pe *Form* se va realiza conform Figurii 2.

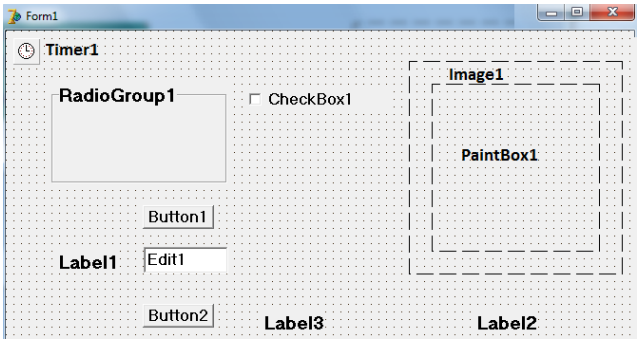

Fig.2. Interfața de lucru a aplicației.

2. Distribuția şi setarea proprietăților componentelor

Componenta	*Fereastră* ***Object Inspector/Properties***	***Valoare***
RadioGroup1	Caption	Termeni
	ItemIndex	0
	Items→Tstrings	3 termeni; 4 termeni
CheckBox1	Caption	Paranteze
	Checked	False
Button1; Button2	Caption	Generare; Verificare
Label1; Label2	Caption	Eliminarea textului din câmp
Edit1	Text	Eliminarea textului din câmp
	Visible	False
Timer1	Inteval	1000
	Enabled	False
Image1	Height; Width	225
PaintBox1	Height; Width	177

3. Declararea variabilelor şi subprogramelor suplimentare

```
Var a:array[1..4] of integer; // termeni
      op: array[1..3] of char; //operaţii pentru exerciţiu
      n,k,m,i,t,r,p :integer;
      sb:char; //simbol + - *
xc1, yc1,xi1,yi1,med:integer; // declararea variabilelor pentru desenarea ceasului
Procedure Ceas(xc,yc:integer;cs:TCanvas); //construieşte indicatorul secundomerului
Var rs :integer; s:word; fis:extended;
Begin
  cs.pen.Width:=5;
  s:=t-1;
  rs:=70;
  fis:=degtorad(6)*s;
  cs.MoveTo(xc,yc);
  cs.LineTo(xc+round(rs*sin(fis)),yc-round(rs*cos(fis)));
  t:=t+1;
end;
```

4. Prelucrarea evenimentelor

Compo-nenta	**Object Inspector / Eveniments**	**Instrucţiuni**

Form1	OnCreate	**var** u, h,x,y,R:integer;
		begin
		Image1.Canvas.Brush.Color:= $00FFFF80;
		Image1.Canvas.Rectangle(0,0,225,225);
		Label1.Caption:=' '; Label2.Caption:=' ';Label3.Caption:=' ';
		Edit1.Text:=' '; Edit1.Visible:=False; Image1.Canvas.Pen.Width:=3;
		xi1:=Image1.Width; yi1:=Image1.Height; med:=xi1 div 2; R:=med-40;
		Image1.Canvas.Font.Size:=16; Image1.Canvas.Font.Color:=clred;//n.ceas
		u:=0; h:=15;
		while u<360 **do**
		begin
		x:=med+ round(R*cos(u*2*pi/360)); y:=med-round(R*sin(u*2*pi/360));
		Image1.Canvas.MoveTo(x,y);
		If (u mod 30)=0 **then** **begin**
		Image1.Canvas.Pen.Color:=clnavy;
		Image1.Canvas.Ellipse(x-4,y-4,x+4,y+4);
		x:= med+ round((R+25)*cos(u*2*pi/360));
		y:=med-round((R+25)*sin(u*2*pi/360));
		Image1.Canvas.TextOut(x-10,y-15,IntToStr(h));
		h:=h-5; **If** h=0 **then** h:=60;
		end else begin
		Image1.Canvas.Pen.Color:=clgreen;
		Image1.Canvas.Ellipse(x-1,y-1,x+1,y+1); **end**;
		u:=u+6; **end**;
		end;

Button1	OnClick	t:=1; randomize; Timer1.Enabled:=True; Edit1.Text:=' ';Edit1.Visible:=True; Label1.Caption:=' '; Label2.Caption:=' '; Label3.Caption:=' '; **if** CheckBox1.Checked **then begin** Radiogroup1.ItemIndex:=0; n:=3; **for** i:=1 **to** n **do** a[i]:=random(5); **for** i:=1 **to** n-1 **do begin** p:=random(3)+1; **case** p **of** 1 : sb:='+'; 2 : sb:='-'; 3 : sb:='*'; **end**; op[i]:=sb; **end**;

		p:=random(2);
		case p **of**
		0: **begin** //(a+b)+c
		Label1.Caption:='('+IntToStr(a[1])+op[1]+IntToStr(a[2])+')'+op[2]+IntToStr(a[3])+'=';
		case op[1] **of**
		'*' : k:=a[1]*a[2]; '+' : k:=a[1]+a[2]; '-' : k:=a[1]-a[2]; **end**;
		case op[2] **of**
		'*' : r:=k*a[3]; '+' : r:=k+a[3]; '-' : r:=k-a[3]; **end**;
		Label2.Caption:=IntToStr(r);
		end;
		1 : **begin** //a+(b+c)
		Label1.Caption:=IntToStr(a[1])+op[1]+'('+IntToStr(a[2])+op[2]+IntToStr(a[3])+')=';
		case op[2] **of**
		'*' : k:=a[2]*a[3]; '+' : k:=a[2]+a[3]; '-' : k:=a[2]-a[3]; **end**;
		case op[1] **of**
		'*' : r:=a[1]*k; '+' : r:=a[1]+k; '-' : r:=a[1]-k; **end**;
		Label2.Caption:=IntToStr(r); **end**; **end**; **end else**
		begin case Radiogroup1.ItemIndex **of**
		0 : n:=3; //3 variabile 1 : n:=4; //4 variabile **end**;
		for i:=1 **to** n **do** a[i]:=random(5);
		for i:=1 **to** n-1 **do begin** p:=random(3)+1;
		case p **of**
		1 : sb:='+'; 2 : sb:='-'; 3 : sb:='*'; **end**;
		op[i]:=sb; **end**;
		case n **of**
		3 : **begin**
		Label1.Caption:=IntToStr(a[1])+op[1]+IntToStr(a[2])+op[2]+IntToStr(a[3])+'=';
		If op[1]='*' **then**
		begin k:=a[1]*a[2];
		case op[2] **of**
		'+' : r:=k+a[3]; '-' : r:=k-a[3]; '*' : r:=k*a[3]; **end**;
		end else if op[2]='*' **then**
		begin k:=a[2]*a[3];
		case op[1] **of**
		'+' : r:=a[1]+k; '-' : r:=a[1]-k; **end**;
		end else
		begin case op[1] **of**
		'+' : k:=a[1]+a[2]; '-' : k:=a[1]-a[2]; **end**;
		case op[2] **of**
		'+' : r:=k+a[3]; '-' : r:=k-a[3]; **end**;
		end; Label2.Caption:=IntToStr(r);
		end; **end**; **end**;
		Edit1.Left:=Label1.Left+Label1.Width+2;

Timer1	OnTimer	PaintBox1.Refresh;
		xc1:=PaintBox1.Width div 2;
		yc1:=PaintBox1.Height div 2;
		Ceas(xc1,yc1,PaintBox1.Canvas);

Button2	OnClick	Timer1.Enabled:=False;
		Label2.Caption:='Timpul efectuării exercitiului ' +IntToStr(t-2)+ ' sec';
		if Edit1.Text=' ' **then** MessageDlg('Nu ai introdus raspunsul', mtWarning, [mbOk],0) **else**
		if StrToInt(Edit1.Text)=r **then** MessageDlg('Bravo!', mtInformation,[mbOk],0)
		else
		begin
		MessageDlg('Ai gresit!',mtError,[mbOk],0);
		Label3.Caption:='Raspuns='+IntToStr(r); **end**;

5. Compilarea şi testarea aplicaţiei se va face pentru a elimina greşelile admise şi a confirma corectitudinea procedurilor realizate.

6. Codul-sursă. Pentru a evita plagiatul programului, codul nu este indicat complet.

Concluzii

Abordarea informatică în elaborarea programului educaţional „Ordinea efectuării operaţiilor" s-a bazat pe instrumentarul mediului de programare Borland Delphi 7.0. Prin utilizarea succesivă a paşilor descrişi în lucrare, studentul va putea prezenta la finele sesiunii de laborator o aplicaţie de tip Windows. Considerăm că elaborarea produselor educaţionale este astăzi una dintre cele mai importante direcţii care urmează la etapa dată a informatizării învăţământului.

Referinţe:

1. Curriculum naţional. Curriculum Şcolar, clasele I-IV, http://www.scribd.com/doc/39253666/Curriculum-Scolar-Clasele-I-IV.
2. Oltean M., Croşan C. Delphi 7.0 în 200 de aplicaţii, 2004.

Prezentat la 29.06.2012