Programare orientată pe obiecte în TurboPascal

Programare orientată pe obiecte în TurboPascal

Bibliografie

T. Bălanescu, S. Gavrilă, H. Georgescu, M. Gheorghe, L. Sofonea, I. Văduva

"Pascal şi Turbo Pascal", vol 2, Ed. Tehnică, Bucureşti, 1992

Programare orientată pe obiecte în TurboPascal

În TurboPascal corespondentul clasei este tipul declarat de utilizator **object**, a cărui declarare are sintaxa asemănătoare cu a tipului **record**. Componentele unui tip **object** pot fi atât date (reprezentând *atribute*), cât şi proceduri sau funcţii (reprezentând *metode*). Declararea unui tip **object** este permisă numai în partea de declarare a modulului (deci **nu** în corpul funcţiilor şi al procedurilor). Definiţiile metodelor asociate trebuie plasate în acelaşi domeniu după declararea tipului **object**. Referirea la componentele unei variabile având un tip **object** se face cu sintaxa

nume_variabilă.nume_componenta[(lista_de_ argumente_efective)].

Programare orientată pe obiecte în TurboPascal

Exemplu: program în care se defineşte şi se utilizează un tip **object** pentru numere complexe.

```
type complex=object
 {date}
 re, im:real;
 {declarari de metode}
 procedure comp(a, b:real);
 procedure atr(v:complex);
 function sum_re(v:complex):real;
 function sum_im(v:complex):real;
 procedure sum(v:complex; var s:complex);
 procedure citeste;
 procedure scrie;
 end;
```

Programare orientată pe obiecte în TurboPascal

Exemplu (continuare)

```
{implementarea metodelor}
procedure complex.comp(a, b:real);
begin
 re:=a;
 im:=b
end;
procedure complex.atr(v:complex);
begin
 comp(v.re, v.im)
end;
function complex.sum_re(v:complex):real;
begin
 sum_re:=re+v.re
end;
function complex.sum_im(v:complex):real;
begin
 sum_im:=im+v.im
end;
```

Programare orientată pe obiecte în TurboPascal

Exemplu (continuare)

```
function complex.sum_im(v:complex):real;
begin
 sum_im:=im+v.im
end;
procedure complex.sum(v:complex; var s:complex);
begin
s.re:=re+v.re;
s.im:=im+v.im
end;
procedure complex.citeste;
begin
read(re, im); readln
end;
```

Programare orientată pe obiecte în TurboPascal

Exemplu (continuare)

```
procedure complex.scrie;
begin
write('re=', re, ' im=', im); writeln
end;
{program principal}
var c1, c2:complex;
begin
 c1.citeste;
 c2.atr(c1);
 c2.sum(c1, c1);
 c1.scrie;
 c2.scrie;
 readln
end.
```

Programare orientată pe obiecte în TurboPascal

Alte elemente de programare orientată pe obiecte pe care TurboPascal le oferă.

- Posibilitatea ca un tip **object** să **moştenească** un alt tip **object**, declaraţia începând cu o specificare a tipurilor cu sintaxa:
  - *tip_obiect* object *tip_obiect_mostenit*
  - urmată de declararea datelor şi metodelor.
- Posibilitatea de a se **supraîncărca** metodele moştenite prin definirea în obiectul moştenitor unor metode cu acelaşi nume, dar cu un corp diferit şi, eventual, cu altă listă de argumente.

Programare orientată pe obiecte în TurboPascal

· Posibilitatea ca unele din metodele unui tip obiect, să fie declarate *virtuale* prin adăugarea cuvântului cheie **virtual** la declaraţia metodei, ca de exemplu:

procedure *nume*(*argumente*); virtual;

ceea ce obligă la declararea ca virtuale a metodelor cu acelaşi nume ale tipurilor **object** moştenitoare. Efectul acestor declararaţii este rezolvarea tuturor apelurilor procedurilor virtuale în momentul **execuţiei** şi nu al compilării, adică rezolvarea apelurilor este dinamică. Acest mecanism de apel trebuie să fie pregătit în timpul execuţiei prin apelarea înainte de apelarea oricărei metode virtuale a unei metode declarată cu cuvântul cheie **constructor** (în loc de **procedure** sau **function**).

Metodele care nu sunt virtuale se numesc *statice*. Este interzisă supraîncărcarea metodelor virtuale cu metode statice.

Programare orientată pe obiecte în TurboPascal

Se observă absenţa implementării unor concepte importante ale orientării pe obiecte cum sunt: clase abstracte, modele de clase, moştenire multiplă. Principiul incapsulării datelor şi metodelor este de asemenea incomplet implementat, lipsind elmentele care să permită limitarea şi diferenţierea accesului la componentele unui tip **object**.
Aceste lipsuri se explică prin faptul că limbajul TurboPascal a fost realizat de firma Borland prin extensia limbajului standard Pascal, care este un limbaj procedural tipic, astfel încât să fie înglobate şi celelalte tehnici de programare: programarea modulară şi programarea orientată pe obiecte. Conceptele orientării pe obiecte s-au implementat însă incomplet, probabil pentru că limbajul de bază standard Pascal are caracteristici care făceau foarte dificilă o abordare mai amplă: ar fi trebuit creat un limbaj cu totul nou, mult îndepărtat de limbajul de bază.

Programare orientată pe obiecte în TurboPascal

Situaţia este substanţial diferită în cazul limbajului C++, cu toate că şi acest limbaj a fost realizat prin extensia "limbajului de bază" C. Flexibilitatea deosebită a limbajului C a permis ca adăugarea facilităţilor de programare orientată pe obiecte, care este principala raţiune a creării limbajului C++, să se facă într-un mod mult mai complet şi eficient, fără să se piardă posibilitatea ca limbajul să fie utilizat doar ca un limbaj procedural obişnuit.