```
{ Operatii care se pot face asupra tipurilor si variabilelor
  procedurale.

  Pentru a intelege mai bine modul in care functioneaza tipurile si
  variabilele procedurale, rulati programul in timp ce il analizati. }

program Operatii_tip_procedural;

uses Crt;

type
    { Definim un tip procedural pentru proceduri fara parametru.
      Avem nevoie de acest tip de date pentru a desemna procedurile
      noastre de afisare. }
    ProceduriDeScriere = procedure;

    { Mai definim un tip procedural pentru functii fara parametru
      si care returneaza Integer. Avem nevoie de acest tip pentru a
      desemna functiile noastre de citire de date. }
    FunctiiDeCitire = function:Integer;


{ Urmeaza sa definim procedurile de scriere si functiile
  de citire. Fiecare din ele va fi folosita pentru atribuiri
  la variabile procedurale, deci e incadrata de directivele de
  compilare "$F+" si "$F-". }

{$F+}
procedure ScrieCeva;
begin
     WriteLn('Ceva');
end;
{$F-}

{$F+}
function CitesteUnNumar : Integer;
var x : Integer;
begin
     Write('Introduceti un numar: ');
     ReadLn(x);
     CitesteUnNumar := x;
end;
{$F-}

{$F+}
function ReadANumber : Integer;
var x : Integer;
begin
     Write('Please enter a number: ');
     ReadLn(x);
     ReadANumber := x;
end;
{$F-}


var
   { Avem o variabila procedurala pentru procedurile de scriere. }
   afis : ProceduriDeScriere;

   { Si mai avem doua variabile procedurale pentru functiile de citire. }
   ro : FunctiiDeCitire;
```

```
   en : FunctiiDeCitire;

   { Si mai avem si o variabila de tip Integer, pentru a retine rezultatul
     returnat de functiile de citire. }
   i : Integer;


begin
     ClrScr;

     { Dorim sa vedem cat spatiu ocupa in memorie o variabila procedurala. }
     WriteLn('Spatiu ocupat in memorie = ', SizeOf(afis));


     { In mod clasic procedura de afisare se apeleaza asa: }
     ScrieCeva;

     { Dar putem face si altfel: atribuim procedura de afisare unei variabile
       procedurale... }
     afis := ScrieCeva;

     { ...si apoi apelam procedura prin intermediul variabilei procedurale.}
     afis;


     { La fel si cu functiile de citire. In mod clasic apelam asa: }
     i := CitesteUnNumar;
     WriteLn('i din ape de functie=', i);

     { Dar putem face si altfel: atribuim functia de citire unei variabile
       procedurale... }
     ro := CitesteUnNumar;

     { ...si apoi folosim variabila procedurala intr-o expresie. Ca urmare
       se apeleaza functia spre care indica variabila. }
     WriteLn('ro=', ro);



     { Cand o functie apare intr-o expresie, ea fie se evalueaza ca tip
       procedural, fie se executa si se va evalua rezultatul pe care il
       furnizeaza.
       Spre exemplu in cazul urmator functia se evalueaza la tip procedural.
       Functia nu se executa, ci doar se atribuie adresa de start a functie
       variabilei procedurale. }
     en := ReadANumber;

     { In schimb in cazul urmator functia se executa si se evalueaza
       rezultatul pe care ea il returneaza. Acest rezultat va fi atribuit
       variabilei neprocedurale. }
     i := en;
     WriteLn('i din variabila procedurala=',i);


     { Sa fim atenti cand comparam doua variabile procedurale. Daca le
       comparam ca si mai jos, in realitate se vor executa functiile spre
       care indica cele doua variabile si se vor compara rezultatele
       returnate in urma executiei. }
     if ro = en
        then WriteLn('Functii egale.')
        else WriteLn('Functii diferite.');
```

```
     { Daca vrem sa vedem daca cele doua variabile se refera la aceeasi
       functie, trebuie sa folosim operatorul de adresare. Se vor compara
       astfel adresele de inceput ale functiilor la care se refera cele
       doua variabile. }
     if @ro = @en
        then WriteLn('Adrese de functii egale.')
        else WriteLn('Adrese de functii diferite.');

     { Putem folosi operatorul de adresare si direct asupra unei functii
       declarate in cod. }
     if @ro = @CitesteUnNumar
        then WriteLn('E functia in romana')
        else WriteLn('Nu e functia in romana');
end.
```