REFERAT

FUNCTII SI PROCEDURI

Realizat de **BUCICO LUCIANA**

Clasa **11 B**

O problema complexa poate fi rezolvata prin divizarea ei intr-un set de parti mai mici .Pentru fiecare parte se scrie o anumita secventa de instructiuni, denumita **subprogram.** In limbajul Pascal exista 2 tipuri de subprograme , si anume, FUNCTII SI PROCEDURI .

Functiile sunt subprograme care calculeaza si returneaza o valoare . Limbajul Pascal continua un set de functii predefinite, cunoscute oricarui program : sin , cos,etc. In completare , programul poate define functii proprii , care se apeleaza in acelasi mod ca functiile –standart. Prin urmare , conceptul de functie extinde notiunea de expresie PASCAL.

Textul PASCAL al unei declaratii de functie are forma:

Function f(x1,x2…,xn): tr;
D;
Begin
…
f:=e;
…
end;

f –numele functiei

(x1,x2…,xn)-lista operationala de parametric formali reprezentand argumentele functiei.

tr-tipul rezultatului; acesta trebuie sa fie numele unui tip simplu sau referinta.

Antetul este urmat de corpul functiei, format din declaratiile locale oprionale **D** si instructiunea compusa **(begin…end)**. Numele **f** al functiei apare cel putin o data in partea staga a unei intructiuni de atribuire care se executa: **f:=e**. Ultima valoare atribuita lui **f** va fi introdusa in programul principal.

Procedurile sunt subprograme care efectueaza prelucrarea datelor communicate in momentul apelului.Limbajul contine procedurile predefinite **read, write, writeln,readln etc**. In completare ,programul poate define procedure proprii , care se apeleaza in acelasi mod ca procedurile-standart. Prin urmare conceptul de procedura extinde notiunea de **instructiune** PASCAL.

Forma generala a textului unei declaratii de procedura:

Procedure p(x1,x2,…,xn) p-numele procedurii

D; **(x1,x2,…,xn)**-lista optionala de

Begin parametric formali;In corpul procedurii

… sunt incluse : D- declaratiile locale (optionale)

end; grupate dupa aceleasi reguli ca in cazul

functiilor; Begin…end-instructiune compusa , ea nu contine vreo atribuire asupra numelui procedurii. Procedura poate sa intoarca mai multe rezultate , dar nu prin numele ei, ci prin variabile desemnate special (cu prefixul var) in lista de parametric formali:

- Parametri valoare – **v1,v2,…,vk:tp** (servesc pentr transmiterea de valori din programul principal in procedura)
- Parametri variabila – **var v1,v2,…,vk:tp** (servesc pentru intoarcerea rezultatelor din procedura in programul principal)

Corpul unui program sau subprogram se numeste **bloc**. Deoarece subprogramele sunt incluse in programul principal si pot contine la randul lor alte subprogram , rezulta ca blocurile pot fi **imbricate (incluse unul in altul).** Aceasta imbricare de blocuri este denumita **structura de bloc** a programului PASCAL.

Prin Domeniul de vizibilitate al unei declaratii se intelege textul de program, in care numele introdus desemneaza obiectul specificat de declaratia in studiu. Domeniul de vizibilitate incepe imediat dupa terminarea declaratiei si de sfarseste odata cu textul blocului respectiv. Domeniul de vizibilitate al unei declaratii inclus , acopera domeniul de vizibilitate al declaratiei ce implica acelasi nume din blocul exterior.

Cunoasterea domeniilor de vizibilitate ale declaratiilor este necesara pentru determinarea obiectului current desemnat de un nume.

De exemplu, identificatorul c din intructiunea c:=chr(d) desemneaza o variabila de tip char.

Orice variabila este locala in subprogramulin care a fost declarata. O variabila este globala relative la un subprogram atunci cand ea se declara in programul sau subprogramul ce il cuprinde fara sa fie redeclarata in subprogramul , cat si in afara lui, ele pot fi folosote pentru transmiterea datelor de prelucrat si returnarea rezultatului.

Destinatia unei functii este sa intoarca ca rezultat o singura valoare. In mod obisnuit, argumentele se transmit functiei prin parametru-valoare, iar rezultatul calculate se returneaza in locul de apel prin numele functiei. In completare , limbajul Pascal permite transmiterea argumentelor prin variabile globale si parametric-variabila.

Prin effort colateral se intelege o atribuire (in corpul functiei) a unei valori la o variabila globala sau la un parametru formal variabila . Efectele colaterale pot influenta in mod neasteptat executia unui program si complica procesele de depanare.

La orice apel de subprogram , in memoria calculatorului vor fi depuse urmatoarele informatii:

- Valorile curente ale parametrilor transmisi prin valoare;
- Locatiile (adresele) parametrilor-variabila
- Adresa return , adica adresa instructiunii ce urmeaza dupa apel.

Parametrii actuali trebuie sa corespunda cu parametrii formali ca numar, ordine si tip. Transferul parametrilor se face in functie de modelul de declarare a acestora, si anume:

- Parametri transmisi prin valoare
- Parametri transmisi prin adresa (referinta)

Parametrii transmişi prin valoare sunt parametrii de intrare. În acest caz subprogramul apelant transmite spre subprogram valoarea parametrului actual, iar orice modificare a acestei valori pe parcursul subprogramului, se va pierde la sfârşitul subprogramului, astfel încât la revenire vom avea aceeaşi valoare ca şi la apel.

Parametrii transmişi prin adresă se evidenţiază prin existenţa cuvântului rezervat VAR înaintea declarării parametrului. În acest caz subprogramul apelant transmite spre subprogram valoarea parametrului actual, iar orice modificare a acestei valori pe parcursul subprogramului, se va reţine la sfârşitul subprogramului, astfel încât la revenire vom avea o altă valoare, nu cea de la apel.

Domeniul de vizibilitate a variabilelor

Exemplu de program prin **PROCEDURA**:

```
Program P1;
type tab=array[1..10] of real;
var a:tab;
   i,n:integer;
   s:real;
procedure suma(var x:tab; n:integer; var sum:real);
var i:integer;
begin
 sum:=0;
 for i:=1 to n do
 sum:=sum+x[i];
end;
begin{main}
 write('n='); readln(n);
 write('dati elementele tabloului:');
 for i:=1 to n do readln(a[i]);
 suma(a,n,s);
 write('s=',s:7:2);
end.
```

Exemplu de program prin **FUNCTIE**:

```
Program P2;
type tab=array[1..10] of real;
```

```
var a:tab;
    i,n:integer;
    s:real;
function suma(x:tab; n:integer):real;
var i:integer;
    z:real;
begin
  z:=0;
  for i:=1 to n do
  z:=z+x[i];
  suma:=z;
 end;
begin{main}
  write('n='); readln(n);
  write('dati elementele tabloului:');
  for i:=1 to n do readln(a[i]);
  s:=suma(a,n);
  write('s=',s:5:2);
end.
```