SUBPROGRAME

SUBPROGRAMUL reprezinta parti identificabile prin nume care se pot activa la cerere prin intermediul acetui nume. O parte din subprogram se contruieste ca subprogram daca un algoritm cuprinde in mai multe locuri aceiasi secventa de operatii executabila pentru aceleasi date sau pentru date diferite. In loc ca subprogramul sa cuprinda in acelasi loc, acelasi grup de instructiuni, concepand grupul de intructiuni ca subprogram, el va aparea in program o singura data si se va activa de mai multe ori. Partea respectiva de program rezolva o subproblema din cele in care se descompune problema complexa. In limbajul Pascal, avem doua tipuri de subprograme : procedurile si functiile. Deosebirea intre ele consta in numarul de valori calculate si returnate programului apelat. Procedurile calculeaza mai multe valori sau nici una, iar functiile returneaza o singura valoare asociata numelui functiei. Atat procedurile cat si functiile pot fi standard(predefinite in unitul sistem), cat si nestandard(definite de utilizator). Procedurile si functiile nestandard trebuie declarate obligatoriu inainte de a fi apelate.

O declaratie de subprograme cuprinde:

-un antet de supbrogram care precizeaza interfata subprogramului cu mediul sau, si

- blocul subprogramului care descrie functionarea lui interna.

DOMENIUL DE VIZIBILITATE AL INDENTIFICATORILOR

Prin domeniul de vizibilitate (valabilitate) se intelege zona de program in care e valabila declararea sau definirea unui identificator. Toti indentificatorii definiti sau declarati intr-un bloc sunt cunoscuti in blocul respectiv si se numesc variabile locale. Daca blocul cuprinde blocuri incluse in care identificatorii (variabile locale ale acestora) nu se definesc sau redenumesc, atunci acestea sunt cunoscute in blocurile incluse si se numesc variabile globale pentru acesta. Daca o variabila declarata intr-un bloc se redefineste atunci in blocul in care a fost redeclarata va fi variabila atribuita generata la redeclarare.

DECLARAREA SI APELUL PROCEDURILOR. PARAMETRII FORMALI SI PARAMETRII EFECTIVI

O procedura e un sunbrogram care calculeaza mai multe valori accesibile sau nu programului apelant sau efectueaza anumite operatii fara sa calculeze vreo valoare. Valorile calculate accesibile programului apelant reprezinta parametrii de iesire ai subprogramului. Acestia pot depinde de anumite valori pe care subprogramul le primeste din programul apelant, valori reprezentand parametrii de intrare. Parametrii formali sunt variabile simbolice in care lucreaza subprogramul. Ele sunt declarate in antetul subprogramului si sunt cunoscute numai in interiorul subprogramului. La apelarea procedurii se specifica parametrii efectivi sau actuali prin intermediul instructiunii procedurale. Parametrii efectivi reprezinta variabilele cu care subprogramele lucreaza efectiv in momentul activarii.

Declararea procedurii se face folosind:

PROCEDURE nume_procedura(lista parametrii)

-parametrii precizati la scrierea proedurii sunt parametrii formali si se separa prin ‘ ; ’

-pentru fiecare parametru se precizeaza numele si tipul acestuia.

Apelarea procedurii :

Pentru a executa o procedura aceasta trebuia apelata. La apel se da numele procedurii si valorile concrete ale parametrilor care se separa prin punct si virgula.

```
Ex :  procedure citire(n :integer ; k :char) ;
        Begin
        …..
```

end;

Cand se apeleaza o procedura, modulul apelant a abandonat temporar, si se executa procedura. In timpul executiei procedurii, parametrii formali sunt inlocuiti in tot corpul procedurii cu parametrii actuali (valori concrete). Dupa executarea procedurii se revine in modulul apelant la linia imediat urmatoare celei care a facut apelul. Parametrii formali si parametrii efectivi nu e obligatoriu sa aiba acelasi nume dar trebuie sa existe o concordanta de numar, tip si ordine.

DECLARAREA SI APELUL FUNCTIILOR

O functie e un subprogram care calculeaza si returneaza programului apelant o singula valoare. Aceasta valoare este asociata numelui functiei. Iar tipul poate fi simplu, string sau reper. Valoarea returnata de functie nu poate avea alt tip structurat decat string.

Declararea unei functii:

*FUNCTION nume_functie(lista parametrii formali)**:** identificator de tip**;***

-nume_functie reprezinta numele functiei, al carei tip este 'identificator de tip'

-identificator de tip = nume de tip simplu: STRING sau REPER;

Blocul functiei trebuie sa contina obligatoriu o instructiune de atribuire prin care identificatorul functiei primeste valoarea unei expresii.

Identificatorul functiei nu are voie sa apara in partea dreapta a unor atribuiri decat daca functia este recursiva.

Apelul unei functii decurge astfel:

- se intrerupe calculul expresiei in care a aparul apelul functiei ;
- se transmit parametrii, daca exista, exact ca la proceduri ;
- se executa functia;

METODA BACKTRACKING

Se aplica problemelor in care solutia poate fi reprezentata sub forma unui vector - x=(x1, x2, x3, …xk,… xn) € S, unde S este multimea solutiilor problemei si S=S1 x S2 x… x Sn, si Si sunt multimi finite avand s elemente si xi € si , (¥)i = 1..n.

Pentru fiecare problema se dau relatii intre componentele vectorului x, care sunt numite conditii interne ; solutiile posibile care satisfac conditiile interne se numesc solutii rezultat. Metoda de generare a tuturor solutiilor posibile si apoi de determinare a solutiilor rezultat prin verificarea indeplinirii conditiilor interne necesita foarte mult timp.

Metoda backtracking evita aceasta generare si este mai eficienta. Elementele vectorului x, primesc pe rand valori in oridinea crescatoare a indicilor, x[k] va primi o valoare numai daca au fost atribuite valori elementelor x1.. x[k-1]. La atribuirea valorii lui x[k] se verifica indeplinirea unor conditii de continuare referitoare la x1…x[k-1]. Daca aceste conditii nu sunt indeplinite, la pasul k, acest lucru inseamna ca orice valori i-am atribui lui x[k+1], x[k+1], .. x[n] nu se va ajunge la o solutie rezultat.

Metoda backtracking construieste un vector solutie in mod progresiv incepand cu prima componenta a vectorului si mergand spre ultima cu eventuale reveniri asupra atribuirilor anterioare.

Metoda se aplica astfel :

1) se alege prima valoare sin S1 si I se atribuie lui x1 ;

2) se presupun generate elementele x1…x[k-1], cu valori din S1..S[k-1]; pentru generarea lui x[k] se alege primul element din S[k] disponibil si pentru valoarea aleasa se testeaza indeplinirea conditiilor de continuare.
   Pot aparea urmatoarele situatii :
   a) x[k] indeplineste conditiile de continuare. Daca s-a ajuns la solutia finala (k=n) atunci se afiseaza solutia obtinuta. Daca nu s-a ajuns la solutia finala se trece la generarea elementului urmator - x[k-1] ;
   b) x[k] nu indeplineste conditiile de continuare. Se incearca urmatoarea valoare disponibila din S[k]. Daca nu se gaseste nici o valoare in S[k] care sa indelineasca conditiile de continuare, se revine la elementul x[k-1] si se reia algoritmul pentru o noua valoare a acestuia. Algoritmul se incheie cand au fost luate in considerare toate elementele lui S1.

Problemele rezolvate prin aceata metoda necesita timp mare de executie, de aceea este indicat sa se foloseasca metoda numai daca nu avem alt algoritm de rezolvare.

Daca multimile S1,S2,…Sn au acelasi numar k de elemente, timpul necesar de executie al algoritmului este k la n. Daca multimile S1, S2.. Sn nu au acelasi numar de elemente, atunci se noteaza cu ‘m’ minimul cardinalelor multimilor S1..Sn si cu ‘M’, maximul. Timpul de executie este situat in intervalul [m la n .. M la n]. metoda backtracking are complexitatea exponetiala, in cele mai multe cazuri fiind ineficienta. Ea insa nu poate fi inlocuita cu alte variante de rzolvare mai rapide in situatia in care se cere determinarea tuturor solutiilor unei probleme.

RECURSIVITATE

Prin recursivitate se intelege faptul ca un subprogram se apeleaza pe el insusi, apelul aparand atunci cand subprogramul este inca activ. Exista doua tipuri de recursivitate:

1) recursivitate directa - cand un subprogram se autoapeleaza in corpul sau ;
2) recursivitate indirecta - cand avem doua subprograme (x si y), iar x face appel la y si invers ;

Se folosesc algoritmi recursivi atunci cand calculele aferente sunt descrise in forma recursiva.

Recursivitatea este frecvent folosita in prelucrarea structurilor de date definite recursiv. Un subprogram recursiv trebuie scris astfel incat sa respecte regulile :

a) Subprogramul trebuie sa poata fi executat cel putin o data fara a se autoapela ;

b)Subprogramul recursiv se va autoapela intr-un mod in are se tinde spre ajungerea in situatia de executie fara autoapel.

Pentru a permite apelarea recursiva a subprogramelor, limbajul Pascal dispune de mecanime speciale de suspendare a executiei programului apelant, de salvare a informatiei necesare si de reactivare a programului suspendat .

Pentru implementarea recursivitatii se foloseste o zona de memorie in care se poate face salvarea temporala a unor valori. La fiecare appel recursiv al unui subprogram se salveaza in aceasta zona de memorie starea curenta a executiei sale.

Desi variabilele locale ale subprogramului apelant au aceleasi nume cu cele ale subprogramului apelat, orice referire la acesti identificatori se asociaza ultimului set de valori alocate in zona de memorie. Zona de memorie ramane alocata pe tot parcursul executie subprogramului apelat si se dealoca in momentul revenirii in programul apelat. Zona de memorie nu este gestionata explicit de programator ci de catre limbaj.

La terminarea executiei subprogramului apelat recursiv, se reface contextul programului din care s-a facut apelul. Datorita faptului ca la fiecare autoapel se ocupa o zona de memorie, recursivitatea este eficienta numai daca numarul de autoapelari nu este prea mare pentru a nu se ajunge la umplerea zonei de memorie alocata.
Recursivitatea ofera avantajunl unor solutii mai clare pentru probleme si a unei lungimi mai mici a programului. Ea prezinta insa dezavantajul unui timp mai mare de executie si a unui spatiu de memorie alocata ami mare. Este de preferat ca atunci cand programul recursiv poate fi transformat intr-unul iterativ sa se faca apel la cel din urma.