2020

Iterativitatea si recursivitatea

GORGAN BOGDAN

IPLT ,,SPIRU HARET''

Cuprins

Introducere

Pe parcursul dezvoltării informaticii s-a stabilit că multe probleme de o reală importanţă practică pot fi rezolvate cu ajutorul unor metode standard, denumite **tehnici de programare**: recursia, trierea, metoda reluării, metodele euristice ş.a. Una din cele mai răspîndite tehnici de programare este **recursia**. Amintim că recursia se defi neşte ca o situaţie în care un subprogram se autoapelează fi e direct, fi e prin intermediul altui subprogram. Tehnicile în studiu se numesc respectiv **recursia directă** şi **recursia indirectă**.

În general, elaborarea unui program recursiv este posibilă numai atunci cînd se respectă următoarea regulă de consistenţă: soluţia problemei trebuie să fi e direct calculabilă ori calculabilă cu ajutorul unor valori direct calculabile. Cu alte cuvinte, definirea corectă a unui algoritm recursiv presupune că în procesul derulării calculelor trebuie să existe:

– cazuri elementare, care se rezolvă direct;

– cazuri care nu se rezolvă direct, însă procesul de calcul în mod obligatoriu progresează spre un caz elementar.

Din cunoștințele anterioare, orice algoritm recursiv poate fi transcris într-un algoritm iterativ şi invers. Alegerea tehnicii de programare – **iterativitate sau recursivitate** – ţine, de asemenea, de competenţa programatorului. Evident, această alegere trebuie făcută luînd în considerare avantajele şi neajunsurile fi ecărei metode, care variază de la caz la caz. Pentru exemplificare, în tabelul 1. sînt prezentate rezultatele unui studiu comparativ al ambelor tehnici de programare în cazul prelucrării automate a textelor.

Nr. crt.	*Caracteristici*	*Iterativitate*	*Recursivitate*
1.	Necesarul de memorie	mic	mare
2.	Timpul de execuţie	acelaşi	
3.	Structura programului	complicată	simplă
4.	Volumul de muncă necesar pentru scrierea programului	mare	mic
5.	Testarea şi depanarea programelor	simplă	complicată

Tabelul 1

1. Aspecte teoretice

1.1 Recursivitatea

Recursivitatea, folosită cu multă eficienţă în matematică, s-a impus în programare, odata cu apariţia unor limbaje de nivel înalt, ce permit scrierea de module ce se autoapelează (PASCAL,LISP,ADA,ALGOL,C sunt limbaje recursive, spre deosebire de FORTRAN,BASIC,COBOL, nerecursive).

Recursivitatea e strâns legată de iteratie, dar dacă iteratia e execuţia repetată a unei porţiuni de program, până la îndeplinirea unei conditii (while, repeat, for din PASCAL), recursivitatea presupune executia repetata a unui modul, insa în cursul executiei lui (şi nu la sfârsit, ca în cazul iteratiei), se verifica o conditie a cărei nesatisfacere, implica reluarea executiei modulului de la inceputul sau. Atunci un program recursiv poate fi exprimat: P=M(X,P) , unde M este multimea ce contine instructiunile X şi pe P insuși.

Cateva exemple:

- **șirul lui Fibonacci**: $F_N = F_{N-1} + F_{N-2}$;

```
function Fib(n:Natural):Natural;
begin
   if n=0 then Fib:=0
   else if n=1 then Fib:=1
                else Fib:=Fib(n-1)+Fib(n-2)
end; {Fib}
```

- **ridicarea la putere**: $a^n = a * a^{n-1}$;

```
Procedure Power (X: real;N:integer;var Y: real);
Begin
   If N=0 then
      Y:= 1
   Else Begin Power(X, N-1,Y);
      Y:= Y*X;
   End ;
End.
```

- **factorialul unui numar**: $N! = N * (N - 1)!$

```
function F(n : Natural) : Natural;
begin
   if n=0 then F:=1
   else F:=n*F(n-1)
end; {F}
```

Cu alte cuvinte – funcția este recursivă, dacă definiția ei folosește o referire la ea insăși.

În PASCAL, exista doua tipuri de *parametri formali* (ce apar în antetul unei proceduri sau functii) :

- **valoare**
- **variabila** (precedat de cuvântul cheie var).

Apelul recursiv al unei proceduri (functii) face ca pentru toti *parametrii-valoare* să se creeze copii locale apelului curent (în stiva) , acestea fiind referite şi asupra lor facându-se modificarile în timpul executiei curente a procedurii (functiei). Când executia procedurii (functiei) se termina, copiile sunt extrase din stivă, astfel incit modificarile operate asupra *parametrilor-valoare* nu afecteaza parametrii efectivi de apel, corespunzatori.

De asemenea pentru toate variabilele locale se rezerva spatiu la fiecare apel recursiv. În cazul *parametrilor-variabila,* nu se creaza copii locale, ci operarea se face direct asupra spatiului de memorie creat parametrilor efectivi, de apel.

1.2 Iterativitatea

Iterativitatea este procesul prin care rezultatul este obţinut ca urmare a execuţiei repetate a unui set de operaţii, de fiecare dată cu alte valori de intrare. Numărul de iteraţii poate fi necunoscut sau cunoscut, dar determinabil pe parcursul execuţiei. Metoda de repetivitate este cunoscută sub numele de *ciclu (loop)* şi poate fi realizată prin utilizarea următoarelor structuri repetitive: ciclul cu test iniţial, ciclul cu test final, ciclul cu număr finit de paşi. Indiferent ce fel de structură iterativă se foloseşte este necesar ca numărul de iteraţii să fie finit.

Iteratia este execuţia repetată a unei porţiuni de program pînă la îndeplinirea unei condiţii(while,for etc.) Dupa cum sa vazut,orice algoritm recursiv poate fi transcris intr-un algoritm iterativ si invers.Alegerea tehnicii de programare-iterativitatea sau recursivitatea- tine de competenta programatorului.

2. Exemple de programe

2.1 ***Recursive***

a. Suma elementelor unui şir

```
var n:integer;
   Function F (N:Integer):integer;
 Begin
    If N=0 Then F:=0
      Else F:=F(N-1)+N <<Functia recursiva>>
 End;
begin
readln(n);     <<Introducerea numarului de elemente  >>
writeln(F(n));   <<Afisarea rezultatului calculat prin functia recursiva>>
 end.
```

b. Inversia unui sir de caractere

```
  Var s: string;
      n: integer;
   Function s(i:integer):string;
  Begin
    if i=0 then s:=' '
     else s:=s[i]+s(i-1);     <<Functia recursiva >>
   End;
 Begin
Writeln('Introduceti sirul'); <<Introducerea elementelor sirului >>
   readln(s);
    n:=length(s);
 Writeln('Sirul inversat este: ', s(n));<< Afisarea sirului inversat  >>
   End.
```

c. Scrierea descrescatoare a unui sir

```
   procedure row(n:integer);
 begin
    if n >=1 then begin
       write (n, ' ');
       row(n-1);       <<Functia recursiva >>
   end;
end;
begin
  readln(n);
  writeln(row(n));     << Afisarea sirului descrescator  >>
end.
```

d. Determina daca stringul este palindrom

```
  Program Palindrom;
   Function Pal(S: String):Boolean;
Begin
     If length(S)<=1
     Then Pal:=True
     Else Pal:= (S[1]=S[Length(S)]) and Pal(Copy(S, 2, Length(S)
- 2));              <<Functia recursiva >>
End;
Var S : String;
 Begin
    Write('Introduceti sirul: '); ReadLn(S);   <<Afisarea rezultatului >>
If Pal(S) Then WriteLn('Sirul este palindrom');
        Else WriteLn('Sirul nu este un palindrom');
 End.
```

e. Functia C(m, n), unde 0 <= m <= n, afla coeficientul binomului C_n^m dupa urmatoarea formula:

$$C_n^0 = C_n^n = 1; \quad C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$$

```
  function C(m, n :Byte):Longint;
 Begin
    If (m=0) or (m=n)
     Then C:=1
       Else C:=C(m, n-1)+C(m-1, n-1)  <<Functia recursiva >>
  End;
```

2.2 Iterative

a. Conversia cifrelor romane

```
var i : integer;
    c : char;
  begin
  i:=0; writeln('Introduceti una din cifrele');
   writeln('romane I, V, X, L, C, D, M');
    readln(c);
 case c of 'I' : i:=1; 'V' : i:=5; 'X' : i:=10; 'L' : i:=50;
'C' : i:=100; 'D' : i:=500; 'M' : i:=1000; end;
    if i=0 then
writeln(c, ' - nu este o cifra romana')
   else writeln(i); readln;
end.
```

b. Stergerea elementului dintr-un string

```
var s: string;
x;char; i:integer;
begin
  write ('Introduceti sirul de caractere'); readln(s);
write ('Introduceti ce urmeazasa fie sters ');
readln(x);
 i:=1;
  while i<= length(s) do
     if s[i]=x then delete(s, i, 1) else i:=i+1;

write('Sirul modificat este ', s);

end.
```

c. Inlocuirea silabei <<sau>> cu <<ori>>

```
var s: string; i:integer;
    begin
  write('Introduceti sirul de caractere ');
readln(s);
  for i:=1 to length(s)-2 do if copy(s, i, 3)='sau'
   then begin
  delete(s,i,3);
  insert('ori',s,i);
   end;
writeln('Sirul obtinut este: ',s);
end.
```

d. Suma elementelor unui numar

```
function suma(n:word):byte;
  var sum:integer;
 begin
  sum:=0;
   while n<>0 do begin
  sum:=sum + n mod 10;
 n:=n div 10;
end;
suma:=sum
end;
```

e. Determinarea solutiilor unei functii de grad. II

```
var a,b,c,x1,x2:real;
  procedure ec(a,b,c:real;var x1,x2:real);
 var d:real;
 begin
  d:=sqr(b)-4*a*c;
  if d<0 then
writeln('Multimea solutiilor este vida ');
   if d=0 then
     begin
      x1:=-b/(2*a);
      writeln('X1=X2, x= ',x1);
      end;
     if d>0 then
        begin
   x1:=(-b-sqrt(d))/(2*a);
   x2:=(-b+sqrt(d))/(2*a);
 writeln('Solutiile sunt x1= ',x1,'si',' x2= ', x2);
   end;
   end;
 begin
  writeln('Dati coeficientii ecuatiei  ') ;
  readln(a,b,c);
  ecuatie(a,b,c,x1,x2);
 end.
```

3. Concluzii

În concluzie putem spune că fiecare program necesita o abordare diferita în corespondență cu cerințele și soluțiile lui, astfel atât iterativitatea cât și recursivitatea joaca un rol important în tehnica programării. În general, algoritmii recursivi sunt recomandați în special pentru problemele ale căror rezultate sunt definite prin relații de recurență: analiza sintactică a textelor, prelucrarea structurelor dinamice de date, procesarea imaginilor, calculi recursive ș.a. Astfel de probleme au fost afisate in paginile 5-6, iar diferentele dintre caracteristicile acestor doua tipuri sunt afisate in urmatorul tabel:

Nr. crt.	*Caracteristici*	*Iterativitate*	*Recursivitate*
1.	Necesarul de memorie	mic	mare
2.	Timpul de execuţie	acelaşi	
3.	Structura programului	complicată	simplă
4.	Volumul de muncă necesar pentru scrierea programului	mare	mic
5.	Testarea şi depanarea programelor	simplă	complicată

4. Bibliografie

- https://tutoriale-pe.net/despre-recursivitate-in-informatica/
- http://www.authorstream.com/Presentation/aSGuest41792-360322-iterativitate-sau-recursivitate-rodika-guzun-science-technology-ppt-powerpoint/
- http://mojainformatika.ru/paskal/lekczii-po-pascal/111-iteraczionnye-czikly.html
- https://www.cyberforum.ru/delphi-beginners/thread2170811.html
- https://labs-org.ru/pascal-8/
- http://gov.cap.ru/home/77/obrazov/hodarygim/htmls/paskal/rekursia/Rekursia.html