Iterativitate și Recursivitate

Lucrare științifică la informatică

01.04.2020

Stegărescu Olivia cl.11C

Profesor: Guțu Maria

Un **algoritm** înseamnă în matematică și informatică o metodă sau o procedură de calcul, alcătuită din pașii elementari necesari pentru rezolvarea unei probleme sau categorii de probleme. De obicei algoritmii se implementează în mod concret prin programarea adecvată a unui calculator, sau a mai multora. Din diverse motive există și algoritmi încă neimplementați, teoretici.

În funcție de modul de implementare, un algoritm poate fi:

- Recursiv- face uz de sine însuși, în mod repetat;
- Iterativ- repetitiv;

Iterativitate – procesul obținerii informației ca urmare a execuției repetate a unui set de operații.

Câteva caracteristici de esență:

- Necesarul de memorie este mic;
- Structura programului e complicată;
- Volumul de muncă necesar pentru scrierea programului este mare;
- Testarea și depănarea programelor e simplă;

Numărul de iterații poate fi necunoscut sau cunoscut, dar determinabil pe parcursul execuției. Metoda de repetivitate este cunoscută sub numele de ciclu (loop) și poate fi realizată prin utilizarea următoarelor sctructuri repetitive: ciclu cu test inițial, cuclu cu test final, ciclu cu număr finit de pași. Indiferent ce fel de structură iterativă se folosește este necesar ca numărul de iterații de fie finit.

Recursivitate – este una dintre cel mai răspândite tehnici. Se definește ca o situație în care un subprogram se autoapelează fie direct, fie prin intermediul altui subprogram.

Soluția problemei trebuie să fie direct calculabilă, ori calculabilă cu ajutorul unor valori direct calculabile.

Caracteristici de esență:

- Necesarul de memorie este mare;
- Structura programului este simplă;
- Volumul de muncă este mic;
- Testarea programelor este complicată.

Definirea recursivă a funcției factorial Fact: N ⟶ N

Fact (n) =1, dacă n=0

N*fact(n-1)dacă n>0, deosebim:

- Cazul elementar n=0. În acest caz valoarea fact (0) este direct calculabilă și anume fact (0)=1
- Cazuri neelemantare n>0. În astfel de cazuri valorile fact (n) nu sunt direct calculabile, însă procesul de calcul progresează către cazul elementar fact (0).
  De exemplu, pentru n=3 obținem:
  Fact (3)=3 * (2)=3*2*1*(0)=3*2*1*1=6
  După cum s-a văzut orice algoritm regursiv poate fi transcris într-un algoritm iterativ și invers. Alegerea tehnicii de programare – iterativitatea sau recursivitatea- ține de competența programatorului.

Studiul comparativ al iterativității și recursivității

Nr.crt.	Caracteristici	Iterativitate	Recursivitate
1.	Necesarul de memorie	Mic	Mare
2.	Timpul de execuție	Același →	
3.	Structura programului	complicată	simplă
4.	Volumul de muncă necesar pentru scrierea programului	mare	mic
5.	Testarea și depănarea programului	simplă	Complicată

Programele/funcțiile recursive se diferențiază de cele iterare prin faptul că acestea se auto apelează. Spre exemplu, în cazul Prtogramului Fibonacci, Fib are două valori inițiale presetate, 0 pentru n= 0 și 1 pentru n=1, astfel în cât, pentru alte valori, funcția seautoapelează până n ajunge la acele două valori presetate. Respectiv pentru 3, funcția va exectua Fib(2) și Fib(1), întrucât pentru fib(1) există o valoare presetată, funcția va executa doar o apelare adițională pentru a determina Fib(2), care va fi Fib(1)+Fib(0), ambele fiind cunoscute.

```
Program Fibonacci;
     Type Natural = 0.. MaxInt;
     Var  nr, i : Natural;
     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;
Begin
Writeln ('i=');
Readln (i);
nr:= Fib(i);
writeln ('Fib(', i, ')=', nr);
end.
```

Similar are loc execuția prigramului ce afișează valoare funcției lui Ackermann. Valorile preseate fiind ecuații bazate pe alt parametru, astefel, dacă executăm funcția pentru F(0, 3) obținem F=3+1. Pentru F(2, 0), fucția se va autoapela de două ori pentru a ajunge la valoare presetată pentru parametrul m, atfel F(2,0)= F(1, 1)= F(0, 1)= 1+1. Similar va avea loc pentru cazul când ambii parametri sunt nenuli.

```
Program Ex5;
    var a, b, S : integer;

    function F(n, m : integer ): integer;
    begin
        if m=0 then
            F:=n+1
        else if n=0 then
            F:= F(m-1, 1)
        else F:= F(m-1, F(m, n-1));
    end;
begin
writeln ('dati valorile a si b pentru care calculati valoarea
functiei Ackermann');
    readln (a,b);
if a <0 then
    writeln ('functia nu admite valori negative. selectati alte
valori a, b')
else if b<0 then
    writeln ('functia nu admite valori negative. selectati alte
valori a, b')
else S:=F(a,b);
writeln ('S(a,b)=', S);
readln;
end.
```

Funcția Ackermann de asemenea demonstrează că funcțiile recursive ocupă foarte rapid memoria alocată pentru executarea programului. Pentru a evita astfel de cazuri, unele recursii pot și adaptate la o executare iterativă, pe pași, prin ntermediul instrucțiunilor `for`, `while` și `repeat`.

```
Program R7;
     Type Natural = 0..MaxInt;
     Var i, a : Natural;

     Function S(n:Natural):Natural;
     Begin
          If n=0 then S:=0
          Else S:=n+S(n-1);
     End;
Begin
Writeln ('i=');
Readln (i);
a:=S(i);
writeln ('S(', i, ')=', a);
end.
Program I7;
     Type Natural = 0..MaxInt;
     Var i, a : Natural;

     Function S(n:Natural):Natural;
          Var j, p : Natural;
     Begin
     p:=0;
     If n<>0 then do begin
          for j:=1 to n do p:= p + n;
     S:=p;
     end;
Begin
Writeln ('i=');
Readln (i);
a:=S(i);
writeln ('S(', i, ')=', a);
end.
```

R7 și I7 sunt programe ce execută aceași funcție, recursiv și, respectiv, iterativ. Diferența de bază fiind facptul că în I7, funcția S nu are o valoare de bază și nu va fi autoapelată, ci va fi executat un loop cu ajutorul instrucțiunii `for` care va executa adiția, iar suma va fi atribuită funcției S.

Funcția iterară ce calculează factorialul unui număr similar se bazează pe instrucțiunea for, care execută un loop.

```
Function F(n:Natural):Natural;
     Var I, p : Natural;
Begin
     p:= 1;
     for i:=1 to n do p:=p*1;
     F:=p;
End;
```

Bibliografie :

- http://www.authorstream.com/Presentation/aSGuest41792-360322-iterativitate-sau-recursivitate-rodika-guzun-science-technology-ppt-powerpoint/
- file:///D:/Downloads/XI_Informatica%20(in%20limba%20romana).pdf
- http://info.tm.edu.ro:8080/~junea/cls%2010/recursivitate/recursivitate.pdf
- https://prezi.com/nrwqzjmovhi8/tehnici-de-elaborare-a-algoritmilor/