IPLT “Spiru Haret “

Lucrare practică

TEMA: Iterativitate sau recursivitate

Efectua de : Grubleac
Gabriela ,eleva clasei a 11-a

Verificat de : Guțu Maria ,
profesor de informatică

Descrierea aspectelor teoretice

Recursia se defineşte ca o situaţie în care un subprogram se autoapelează fi e direct, fi e prin intermediul altei funcţii sau proceduri.

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.

Observaţii importante:

- Orice funcţie recursivă trebuie să conţină o condiţie de reluare a apelului recursiv(sau de oprire). Fără această condiţie, funcţia teoretic se reapelează la infinit, dar nu se întâmpla acest lucru, deoarece se umple segmentul de stiva alocat funcţiei şi programul se înterupe cu eroare.;
- La fiecare reapel al funcţiei se execută aceeaşi secvenţă de instruciuni;
- Ţinând seama de observaţiile anterioare, pentru a implementa o funcţie recursivă, trebuie să:
    - ✓ Identificăm relaţia de recurenţa (ceea ce se execută la un moment dat şi se reia la fiecare reapel) ;
    - ✓ Identificăm conditiile de oprire ale reapelului ;
- În cazul în care funcţia are parametrii, aceştia se memorează ca şi variabilele locale pe stiva, astfel:
    - ✓ parametrii transmişi prin valoare se memorează pe stivă cu valoarea din acel moment ;
    - ✓ pentru parametrii transmişi prin referinţă se memorează adresa lor;

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. Indiferent ce fel de structură iterativă se foloseşte este necesar ca numărul de iteraţii să fie finit. 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 fiecărei metode, care variază de la caz la caz.

Avantajele și dezavantajele metodelor

În ciuda faptului că *timpul de execuție* nu diferă drastic , *capacitea de memorie necesara* este mai mare în cazul programelor recursive . Pe de altă parte metoda de creere a *structurii programelor* recursive este mult mai rapidă și eficientă, respectiv volumul de muncă necesar al programatorului este mult mai mic . Totuși mediile actuale de programare nu asigură verificarea consistenţei algoritmilor recursivi, acest lucru revenindu-i programatorului, cea ce în cazul programelor recursive este mult mai complicat de făcut.

Utilizarea metodei iterative în rezolvarea problemelor

1. Scrieți un program care să radieze litera a dintr-un șir de caractere ,utilizâbd un subprogram.

```
var s,s1:string;
procedure sterge(s:string;c:char;var s1:string);{antetul procedurii pentru gasirea literei respective}
var p:integer;
begin p:=pos(c,s);{p este pozitia literei a in sir }
while p<>0 do {creerea unui ciclu prin care se va gasi pozitia literei a}
begin
  delete(s,p,1);
  p:= pos(c,S);
  end;
  s1:=s;{atribuirea sirului initial sirul editat pentru a reintoarce valoare in programul principal}
end;
begin
  writeln(' introdu sirul ' );readln(s);
  sterge(s,'a',s1);writeln(s1);
end.
```

2. Scrieți un program care ar calcula solutiile reale ale unuei ecuatii patrate folosind subprograme

```
var a,b,c,x1,x2:real;
procedure ecuatie(a,b,c:real;var x1,x2:real);{antetul procedurii ce va determina numarul de solutii in dunctie de delta d}
   var d:real;
 begin
   d:=sqr(b)-4*a*c;
   if d<0 then writeln('multime vida ');{cazul fara solutii reale}
   if d=0 then
   begin
     x1:=-b/(2*a);
     writeln('x1=x2= ',x1);{cazul in care este o singura solutie }
     end;
   if d>0 then
     begin
     x1:=(-b-sqrt(d))/(2*a);
     x2:=(-b+sqrt(d))/(2*a);
```

```
     writeln('x1= ',x1,' x2= ', x2);{cazul in care exista doua solutii}
     end;
    end;
 begin
   writeln(' dati coeficientii  ') ;readln(a,b,c);{creerea unui ecuatii}
   ecuatie(a,b,c,x1,x2);
 end.
```

3. Scrieți un program care să schimbe într-un șir de caractere cuvântul sau cu cuvântul ori utilizând subprograme.

```
var S,s1:STRING;

 function f(s:string):string;{antetul fuctiei ce va returna sirul modificat }
    var i:integer;p:integer;
  begin
    p:=pos('sau',s);{p este pozitia lui cuvantului sau in sir}
    while p<>0 do begin{creere unui ciclu care permite gasirea pozitiei cuvantului sau in sir de mai multe
ori }
      delete (s,p,3);
      insert('ori',s,p);
      p:=pos('sau',s);
       end;
       f:=s;
  end;
  begin write ('dati sirul '); readln(s);
  s1:=f(s);
  writeln(s1);
  end.
```

4. Scrieți un program care sa calculeze suma numerelor pare dintr-un tabel unidimensional .(în acest program nu am utilizat subprograme , dar condiția acestuia este asemănatoare cu codiția problemei 1. din programele recursive, asfel am încercat să demonstrez diferențe în utilizarea metodelor respective)

```
type tab=array[1..50]of integer;
var A:tab;
i,n,s:integer;
begin
  write('dati numarul de elemente  ');readln(n);
  writeln('introdu elem ');
for i:=1 to n do readln(a[i]);{citirea elementelor tabelului}
s:=0;{valoarea initiala a sumei}
for i:=1 to n do begin
if a[i]mod 2=0 then s:=s+a[i];end;{conditia de calcul a sumei numarelor pare}
writeln('suma elem pare este ',s);
         end.
```

5. Scrieți un program care să inverseze un șir de caractere (șir din maxim 10 caractere) .(în acest program nu am utilizat subprograme , dar condiția

acestuia este asemănatoare cu codiția problemei 3. din programele recursive, asfel am încercat să demonstrez diferențe în utilizarea metodelor respective)

```
const n=10;
var a:array [1..n]of string ;
i,j:integer;
x,inv:string;
begin writeln('introduceti sirul');
for i:=1 to n do readln(a[i]);
for i:=1 to n do
begin
  x:=a[i];inv:='';
  for j:=1 to length(x) do
    inv:=inv+x[j];a[i]:=inv;
    end;
    writeln('sirul inversat');
    for i:=1 to n do writeln(a[i]);

end.
```

Utilizarea metodei recursive în rezolvarea problemelor

1. Scrieți un program care sa calculeze suma numerelor pare dintr-un tabel unidimensional print-o funcție ,iar tabelulu să fie citit cu ajutorul unei funcții recursive

```
 type tab=array[1..50]of integer;
var A:tab;
i,k:integer;
procedure citire (var x:tab;n:integer);{antetul procedurii ce va citi tabelul prin recursive}
begin if n>0 then begin {tabelul se va citi pana cand nr.de elemente va fi egal cu 0}
  citire (x,n-1);read(x[n]);{prin autoapelare se va reusi citirea fiecarui element}
  end;
end;
function suma(x:tab; n:integer):integer;{functia va returna un singur rezultat in programul
principal}
    begin
      if n=0 then suma:=0{autoapelarea se va finisa atunci cand nr. elementelor va fi 0}
      else if x[n]mod 2=0 then suma:=suma(x,n-1)+x[n]{conditia pentru numerele pare}
      else suma:=suma(x,n-1);{suma ramane aceiasi in cazul nr. impare}
    end;
begin
  write('n= ');readln(k);
  writeln('introdu elem ');{se dau elementele tabelului }
  citire(A,k);{apelul procedurii}
  writeln('suma elem pare este ',suma (A,k));{aprlul functiei ce se poate utiliza direct in
operatie }
   end.
```

2. Scrieți un program care calculează suma numerelor unui șir natural (5,8,11...) cu ajutorul unui funcții recursive .

```
var n:integer;
function f(n:integer):integer;
begin
  if n=1 then f:=5 {functia va inceta autoapelare atunci cand n va fi 1}
  else F:=F(n-1)+(2*n+3);{autoapelare functiei}
end;
begin
  writeln('introdu n ');readln(n);{numarul de elemente din sir la care se va calcula suma }
  writeln('suma nr. ', F(n));
end.
```

3. Scrieți un program care să inverseze un șir de caractere cu ajutorul unei funcții recursive.

```
var i:integer;
s:string;
function F(i:integer):string;{functia va returna un singur rezultat in programul principal}
begin
  if i=0 then f:='' {inversarea se va opri atunci cand sirul se va sfarsi}
else f:=s[i]+f(i-1);{operatia de inversare prin recursie
end;
begin write('introdu elem sirului ');readln(s);{citirea sirului}
i:=length(s);{i este lungimea sirului}
writeln(f(i));{scrierea sirului modificat}
end.
```

4. Scrieți un program care să calculeze puterea unui numar utilizand o funcție recursivă.

```
var n,k:integer;
function F(k,n:integer):integer;{antetul functiei ce calculeaza putere unui numar }
begin
if k=0 then f:=1{autoapelarea functiei se va incheia atunci cand puterea va fi 0}
else f:=n*f(k-1,n);
end;
begin
write('dati nr.');readln(n);
write('dati puterea');readln(k);
writeln('rezultat:',f(k,n));
      end.
```

5. Scrieți un program care citește și afișează două tabele undidirecționale utilizând subprograme recursive .

```
type tab=array [1..100] of integer;
var A,B:tab;
   i,k,q:integer;
```

```
procedure citire(var x:tab;n:integer);{acesta procedura va citi tabelul}
begin
if n<>0  then {procedura se va autoapela pana cand numarul de elemente va fi 0}
begin
citire(x,n-1);
read(x[n]);
end;
end;
procedure afisare(var x:tab;n:integer);{acesta procedura va afisa la ecrat tabelul }
begin
if n<>0 then {procedura se va autoapela pana cand numarul de elemente va fi 0}
begin afisare(x,n-1);
writeln(' ',x[n]);end;
end;
begin
write('dati numarul de lemente  a primului tabel ');readln(k);
citire(A,k);{citirea tabelului A}
Writeln('tabelul A');
afisare(A,K);{afisare tabelului A}
write('dati numarul de lemente  a celui de al doilea ');readln(q);
citire(B,q);{citirea tabelului B}
Writeln('tabelul B');
afisare(B,q);{afisare tabelului B}
      end.
```

Concluzii

1. Utilizarea metodei recursive este mult mai eficientă în rezolvarea unor tipuri de probleme.
2. Nu toate problemele pot fi rezolvate prin metoda recursivă.
3. Utilizarea metodei iterative, în cazul problemelor mai complicate ,în unele cazuri este mult mai convenabilă decât cea recursivă, odata ce acestea necesita o capacitate mai mare de memorie .
4. Pentru ca programul recursiv să funcșioneze este necesar ca definiția funcției recursive să nu fie inconsistentă.

Bibliografie

- Manualul de informatică clasa a 11-a
- http://info.tm.edu.ro:8080/~junea/cls%2010/recursivitate/recursivitate.pdf
- https://prezi.com/qfmfcl_7jdpg/recursivitate-si-iterativitate/

Cuprins