Liceul teoretic Spiru Haret

Referat

Iterativitate sau recursivitate

A efectuat: **eleva cl.a 11-a "D"**

Bondari Sofia

A verificat: **Guțu Maria**

Chisinău 2020

Cuprins

1.Aspecte teoretice. Avantajele/dezavantajele metodelor

1.1Iterativitatea

Iterativitatea-este procesul prin care rezultatul este obținut ca urmare a execuției repetate a unui set de operatii, de fiecare dată cu alte valori de intrare. Numărul de iterații poate fi necunoscut sau cunoscut, dar determinabil pe parcursul execuției. [2]

Avantaje	*Dezavantaje*
• Iterativitatea minimizeaza complexitatea unor algoritmi • Iterativitatea poate înlocui recursivitatea atunci cand recursivitatea este prea adâncă sau când limbajul de programare nu permite implementarea de apeluri recursive. • Necesar de memorie mic • Testarea și depanarea programelor este simplă.	• Structura programului complicată • Volumul de muncă necesar pentru scrierea programului este mare • Deseori, variantele iterative necesita folosirea explicita a structurii de tip stiva, generand astfel un cod extrem de laborios.

[3], [6]

1.2Recursivitatea

Recursivitatea este procesul iterative prin care valoarea unei variabile se determină pe baza uneia sau a mai multora ditre propriile ei valori anterioare. Structurile recursive reprezintă o alternative de realizare a proceselor repetitive fara a utiliza cicluri.[2]

Cum funcționează?

Pentru fiecare apel al unei funcții se adaugă pe stivă o zonă de memorie în care se memorează variabilele locale și parametrii pentru apelul curent. Această zonă a stivei va exista până la finalul apelului, după care se va elibera. Dacă din apelul curent se face un alt apel, se adaugă pe stivă o nouă zonă de memorie, iar conținutul zonei anterioare este inaccesibil până la finalul acelui apel. Aceste operații se fac la fel și dacă al doilea apel este un autoapel al unei funcții recursive.[5]

Observaţii importante:

- Orice funcţie recursivă trebuie să conţină o condiţie de reluare a apeluluirecursiv(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.

Elaborarea unui algoritm recursive este posibilă numai atunci când se respect ***regula de consistență:*** soluția problemei trebuie să fie direct calculabilă ori calculabilă cu ajutorul unor valori direct calculabile, adică definirea corectă a unui algoritm recursiv presupune ca ăn procesul derulării calculelor trebuie să existeȘ

- ✓ Cazuri elementare, care se rezolva direct.
- ✓ Cazuri care nu se rezolvă direct, însă procesul de calcul în mod obligatoriu progresează spre un caz elementar. [5]

Avantaje	Dezavantaje

• Recursivitatea poate fi inlocuita prin iteratie atunci cand recursivitatea este prea adanca sau cand limbajul de programare nu permite implementarea de apeluri recursive. • Soluțiile recursive sunt mult mai clare, mai scurte și mai ușor de urmărit. • Structuraprogramului este mai simplă • Volumul de mună necesar pentru scrierea programului este mai mic • Recursivitatea este mult mai avantajoasă în cazul în care: ✓ Soluțiile problemei sunt definite recursive ✓ Cerințele problemei sunt formulate recursiv.	• Din punctul de vedere al memoriei solicitate, o varianta recursiva necesita un spatiu de stiva suplimentar pentru fiecare apel fata de varianta iterativa. • Testarea și depanarea programelor este complicată

[3],[4],[1]

2.Rezolvare de probleme

2.1Probleme la iterativitate

1) *Să scriem un program care să calculeze cel mai mare divizor comun a două numere. În acestscop, vom utiliza o funcţie, care va avea două argumente: numerele întregi a şi b. În urmaalgoritmului, a şi b se modifică, însă acest lucru nu vrem să fie transmis înapoi în programulprincipal, aşa încât nu vom folosi cuvântul rezervat var în faţa identificatorilor.*

```
program CalculCMMDC;

function CMMDC(a,b: Integer): Integer;

begin

    while a<>b do

            if a>b then a:=a-b

            else b:=b-a;

     CMMDC:=a

end;

        var x,y: Integer;

 begin

      x:=20; y:=24;

    WriteLn(CMMDC(x,y))

 end.
```

2) *Următorul exemplu va face precizări asupra variabilelor care apar în blocul principal, parametrii funcţiei şi variabilele locale ei. Se calculează produsul n! = 1×2×...×n.*

```
program Test1;
function Factorial(n: Integer): Integer;
var i,p: Integer;
begin
      p:=1; for i:=1 to n do p:=p*i;
      Factorial := p
end;
var p,i: Integer;
begin
       p:=3;
       i:=Factorial(p);
      WriteLn(i)
end.
```

Programul va afişa 6 (=3!). Ce se întâmplă? În primul rând, avem declarată o funcţie Factorial, având parametrul formal n. În interiorul funcţiei, se calculează produsul p = n!. Pentru aceasta se foloseşte încă variabilă locală i. La sfârşit, funcţia returnează valoarea lui p, iar i, p şi n dispar. Execuţia programului se desfăşoară în felul următor: variabila globală p primeşte valoarea 3. Se apelează funcţia de calcul a factorialului, cu p global drept parametru efectiv. Astfel, parametrul formal n va primi valoarea 3. Acest p global nu are nimic comun cu variabila locală funcţiei, purtând acelaşi nume. Deci, când execuţia ajunge în punctul B, p-ul global va avea valoarea 3, iar cel local valoarea 6. În funcţie apare o altă variabilă i, dar acest i nu este tot una cu variabila globală i, care primeşte valoarea factorialului lui 3, deci 6.

3) *Simplificarea unei fracţii. Să se simplifice o fracţie a/b. Vom proceda în felul următor: dându-se două numere a şi b, vom verifica dacă b este diferit de zero sau nu. În caz că nu, se va afişa un mesaj de eroare. Altfel, dacă a nu este nul, se va determina, cu o funcţie corespunzătoare, cel mai mare divizor comun a lui a şi b. Fie acesta c. Rezultatul va fi dat sub forma u/v, unde u şi v se determină împărţind pe a şi b la c. Dacă v=1, atunci vom afişa doar pe u.*

```
program SimplificareaUneiFractii;
function cmmdc(a,b: Integer): Integer;
begin
     while a<>b do if a>b then a:=a-b else b:=b-a;
     cmmdc:=a
end;
procedure Simplifica(a,b: Integer; var u,v: Integer);
var c: Integer;
begin
```

```
        c:=cmmdc(a,b); u:=a div c; v:=b div c
 end;
var a,b,u,v: Integer;
begin
        WriteLn('Simplificare'); WriteLn('Dati a si b !');
        ReadLn(a,b);
        if b=0 then WriteLn('Fractie inexistenta !')
        else if a=0 then WriteLn('Rezultat: 0')
        else begin
                    Simplifica(a,b,u,v);
                     if v=1 then WriteLn('Rezultat: ',u)
                    else WriteLn('Rezultat: ',u,'/',v)
                  end;
         ReadLn
end.
```

[6]

4) *Să se calculeze un program care va calcula printr-o funcție media aritmetică a elementelor unui vector.*

```
Program P4;
Type vector=array[1..100] of integer;
Var a:vector;
      I,n:integer
      Med:rea;
Function media(a1:vector;n1:integer;):real;
     Var s,i,k:integer;
     Begin
           S:=0;
           For i:=1 ton1 do s:=s+a1[i];
            Media:=s/n1;
       End;
 Begin
Read(n);
For i:=1 to n do readl(a[i]);
Med:=media(a,n);
Writeln('media este':',med);
End.
```

5) Calculați aria și perimetrul unui dreptunghi.

```
Program P5;
Type dreptunhi=record
   Lung,lat:real;
End;
       Var d:dreptunghi;
            S,p:real;
ProcedurePS(x:dreptunhi; var s,p:real);
Begin
    S:x.lung*x.lat;
    P:2*(x.lung+x.lat);
End;
Begin
Read(d.lung,d.lat);
      PS(d,s,p);
Writeln(s);
Writeln(p);
End.
```

2.2Probleme la recursivitate

1)*Se dă un tabel bidimensional.Calculati produsul elementelor pare prin recursie.*

```
Program P1;
Type vector=array[1..100]; of integer;
      Var x: tab;
           I,k:integer;
Procedure citire(var x:tab; n:integer);
     Begin
        If n>0 then begin
       Citire(x,n-1);
        Read(x(n));
     End;
End;
Procedure scriere(var x:tab, n:integer);
```

```
        Begin
            If n>0 then begin
           Scriere(x,n-1);
            Write(' ',x(n));
        End;
    End;
Function produs(x:tab; n:integer):integer;
        Begin
           If n=0 then begin
           If x[n] mod2=0 then produs:=x[1]else produs:=1;
           Else if x[n] mod 2=0 then produs:=produs(x,n-1)*x[n]
            Else produs:=produs(x,n-1);
End.
```

2) *Să se calculeze suma primelor numere impare*

```
Program P2;
Function suma(n:integer):longint;
   Begin
   If n=0 then suma:=0
            Else if n mod<>2 then suma:=n+suma(n-1);
    End;
Begin
    Write('n='); readln(n);
    Writeln(suma(n));
    Readln
End;
```

3) Program P1;

```
s:=1;
procedure exp(a:real, n:integer);
begin
S:=s*a;
if n>=1 then exp(a,n-1);
end;
Begin
      write(‘a=’);readln(a);
      write (‘n=’);readln(n);
       writeln(exp(a,n));
       readln;
end.
```

5)

```
Program P5;
Function cmmdc (a,b: integer) :integer ;
begin
if b=0 then
        cmmdc:=a  else
          cmmdc:=cmmdc(b, a mod b);
   end;
var a,b:integer ;
begin
    write('a='); readln(a);
    write('b='); readln(b);
    writeln('cmmdc=' ,cmmdc(a,b));
    readln;
```

end.

6) *Elaborati un program cu subprograme recursive care: Afiseaza componentele vectorului pe ecran, calculeaza suma componentelor, alculeaza suma componentelor pozitive, calculeaza produsul componentelor negative.*

```
Program P6;
type
   arr_type=array[1..100] of integer;
var a:arr_type;

procedure afiseaza(arr:arr_type;n:integer);
begin
if n=1 then
begin
write(arr[1]);
write(' ');
end
else
begin
afiseaza(arr,n-1);
write(arr[n]);
write(' ');
end;
end;

procedure invers(start,n:integer);
var temp:integer;
begin
if start<n then
```

```
begin
temp:=a[start];
a[start]:=a[n];
a[n]:=temp;
invers(start+1,n-1);
end
else
end;

function suma(arr:arr_type;n:integer):integer;
begin
if n=1 then
suma:=arr[1]
else
suma:=arr[n]+suma(arr,n-1);
end;

function suma_poz(arr:arr_type;n:integer):integer;
var s:integer;
begin
if n=1 then
begin
if arr[1]>0 then
suma_poz:=arr[1]
else
suma_poz:=0;
end
```

```
else
if arr[n]>0 then
suma_poz:=arr[n]+suma_poz(arr,n-1)
else
suma_poz:=suma_poz(arr,n-1);
end;

procedure cauta_neg(arr:arr_type;n:integer);
begin
if n=1 then
begin
if arr[1]<0 then
writeln('Exista cel putin un nr negativ:',arr[1])
else
writeln('Nu exista vreun nr negativ');
end
else
if arr[n]<0 then
writeln('Exista cel putin un nr negativ:',arr[n])
else
cauta_neg(arr,n-1);
end;

function prod_neg(arr:arr_type;n:integer):integer;
var s:integer;
begin
if n=1 then
```

```
begin
if arr[1]<0 then
prod_neg:=arr[1]
else
prod_neg:=1;
end
else
if arr[n]<0 then
prod_neg:=arr[n]*prod_neg(arr,n-1)
else
prod_neg:=prod_neg(arr,n-1);
end;

procedure cauta_val(arr:arr_type;n:integer;val:integer);
begin
if n=1 then
begin
if arr[1]=val then
writeln('Exista valoarea respectiva cel putin o data in vector')
else
writeln('Nu exista acea valoare in vector');
end
else
if arr[n]=val then
writeln('Exista valoarea respectiva cel putin o data in vector')
else
cauta_val(arr,n-1,val);
```

```
end;



var  i,n,x:integer;
Begin
writeln('Introduceti dimensiune vector: ');
readln(n);
writeln('Introduceti elementele:');
for i:=1 to n do
read(a[i]);

afiseaza(a,n);
writeln('Suma elementelor este:',suma(a,n));
writeln('Suma elementelor pozitive este:',suma_poz(a,n));
writeln('Produsul numerelor negative este:',prod_neg(a,n));
cauta_neg(a,n);

writeln('Introduceti valoarea de cautat in sir:');
readln(x);
cauta_val(a,n,x);

writeln('Elementele inversate sunt: ');
invers(1,n);
afiseaza(a,n);
End.
```
[7]

Concluzie

O dată cu dezvoltarea informaticiiapare importanța metodelor standarte de rezolvare a problemelor(tehnici de programare) recursia. În concluzie putem remarca faptul că fiecare problema trebuie abordată diferit în corespondență cu cerințele și soluțiile ei, astfel atât iterativitatea cât și recursivitatea au importanță majoră 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 ș.a. Un exemplu tipic de astfel de probleme este analiza gramaticală a programelor PASCAL, sintaxa cărora, după cum se știe,este definită prin relații de recurență.

Bibliografie

1. Gremalschi A. Manual de matematică clasa a 11 2014
2. http://www.authorstream.com/Presentation/aSGuest41792-360322-iterativitate-sau-recursivitate-rodika-guzun-science-technology-ppt-powerpoint/
3. http://fmi.unibuc.ro/ro/pdf/2017/admitere/licenta/FMI_Subprograme_si_recursivitate_2017.pdf
4. https://prezi.com/zmo8gm5nw5jl/abordari-iterative-sau-recursive/
5. http://info.tm.edu.ro:8080/~junea/cls%2010/recursivitate/recursivitate.pdf
6. https://www.edusoft.ro/pascal12/cap5_pascal12.pdf
7. http://muhaz.org/recursivitate-subprograme-recursive-simple.html