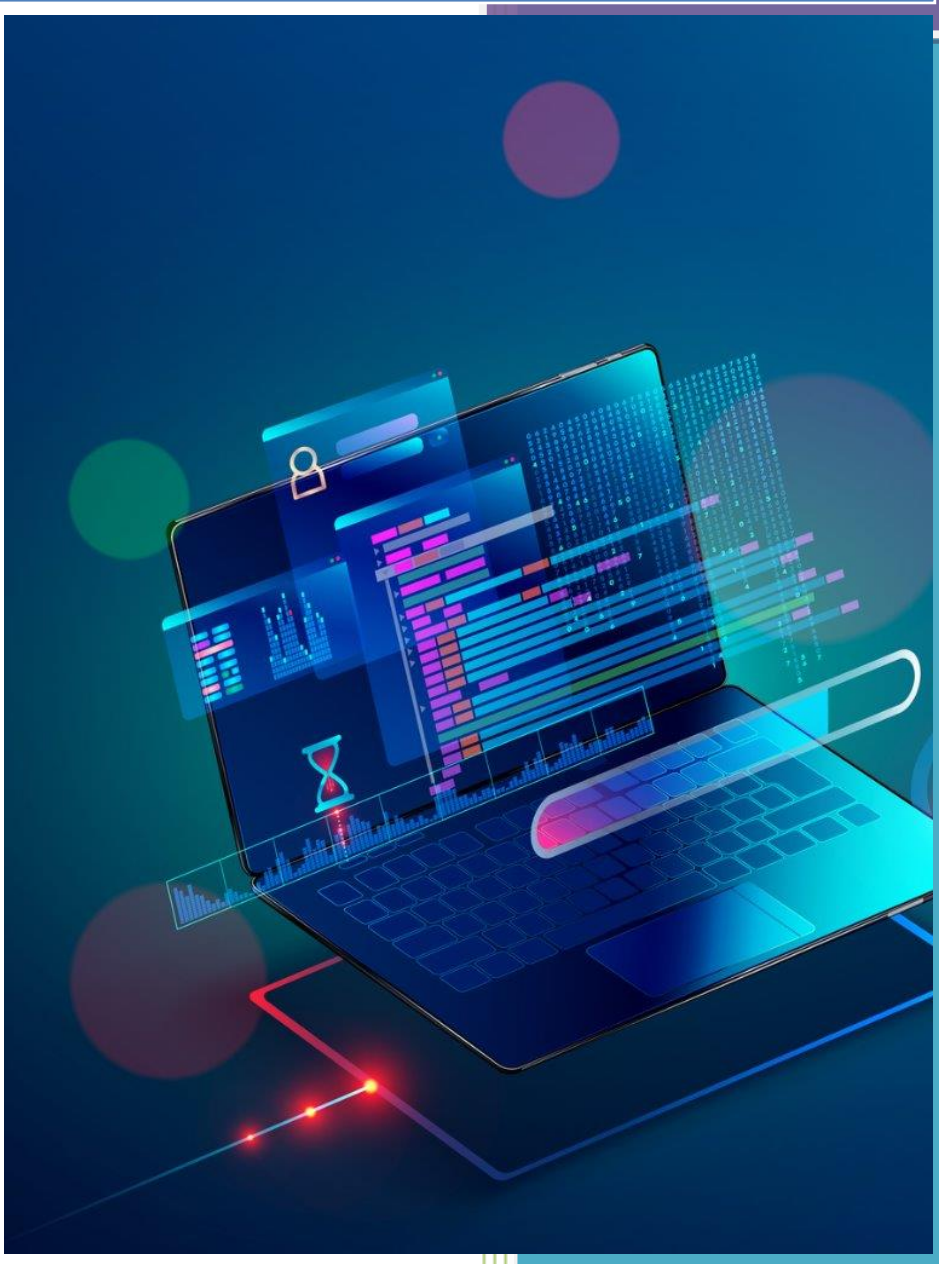
Iterativitate si recursivitate in programul PASCAL



Richicinschi Mirela

Cuprins:

Introducere **3**

1.Iterativitatea

1.1.Teorie 3

1.2.Exemple de probleme..... 4-5

2.Recursivirtatea

2.1.Teorie 5

2.2.Exemple de probleme..... 6-7

3.Relatia dintre iterativitate si recursivitate..... 8

Bibliografie..... 8

Introducere

Din cele mai vechi timpuri omenirea a folosit un set de actiuni pentru majoritatea activitatilor executate de ei. Acest model de existenta a fost descoperit a fi proeminent in tot ce ne inconjoara, mai ales in numere. Mai tarziu(secolul IX) matematicianului persan Al-Khwarizimi a introdus oficial conceptul de rezolvare a problemelor dupa un set de reguli predefinite. In cinstea sa putem acum gasi in dictionarele oricarei limbi cuvantul „algoritm”[3]

Acum una din cele mai progresiste si necesare stiinte, informatica, foloseste din plin algoritmii in metode standarte numite tehnici de programare. Daca algoritmul se autoapeleaza, atunci avem deaface cu **recursivitatea** si daca are loc repetarea sa in cicluri, atunci am facut cunostinta cu **iterativitatea**.

Totusi, metodele date nu sunt atat de simple si e nevoie de mai multa teorie asupra lor. Din fericire pentru oricine care citeste acest referat, voi continua prin a explica si exemplifica recursivitatea si iterativitatea, astfel ca pana la sfarsitul lecturii acestui document sa intelegeti mai bine aceste doua tehnici de programare.

Modulul 1

ITERATIVITATEA

1.1. Teorie

Iterativitatea este situatia in care o secventa de instructiuni sau operatii poate fi folosita de mai multe ori, de fiecare data cu alte valori de intrare(cele obtinute din precedenta executie). In acest caz, daca actiunea este iterata(repetata) o denumin un ”ciclu”(din engleza “loop”). Indiferent de ciclu si instructiuni, numarul de repetitii trebuie sa fie finit [1].

Tehnica data de rezolvare a problemei este una foarte necesara in ziua de azi, mai ales pentru software developers si creatorii de aplicatii. Prin metoda euristica(incercari si eruari) au loc crearea unor secvente in aplicatii numite iteratii. Dupa acea ele sun verificate prin a vedea daca noua iteratie este compatibila cu celelate. Metoda data ne da posibilitatea de a folosi o aplicatie care continua sa fie actualizata in timp cu adaugarea noilor iteratii. Totodata tehnica aceasta de programare simplifica

gasirea si rezolvarea problemelor, datorita faptului ca programul e impartit in subprograme. “Iterative development” este denumit si ca “circular development”, ceea ce implica conceptul de repetare [2].

1.2. Exemple de probleme

Problemele urmatoare sunt iterative. Observam aceasta prin instructiunile de repetare incluzand “while”, “for”, “case”, etc. Toate probleme de mai jos sunt luate din sursa [4]

1)
```
Program P53; { 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.
```

2)
```
Program P60; { Calcularea factorialului }
 var n, i, f : 0..MaxInt;
begin
write(’n=’); readln(n);
 f:=1;
for i:=1 to n do f:=f*i;
writeln(’n!=’, f);
readln; end.
```

3)
```
Program P67; { Tabelul functiei }
var x, y, x1, x2, deltaX : real;
 Begin
 write(’x1=’);
readln(x1);
write(’x2=’);
 readln(x2);
write(’deltaX=’);
readln(deltaX);
writeln(’x’:10, ’y’:20);
writeln; x:=x1;
```

```
while x<=x2 do begin if x>8 then y:=x+1 else y:=x-2;
writeln(x:20, y:20);
 x:=x+deltaX; end;
readln; end.
```

4)
```
Program P68; { Paritatea numerelor citite de la tastatura }
 var i : integer;
begin
writeln('Dati numere intregi:');
 repeat readln(i);
if odd(i) then writeln(i:6, ' – numar impar') else writeln(i:6, ' – numar par');
until i=0; readln;
end.
```

5)
```
Program P64; { Media aritmetica a n numere }
var x, suma, media : real; i, n : integer;
begin
 write('n=');
 readln(n);
suma:=0; writeln('Dati ', n, ' numere:');
 for i:=1 to n do
 begin
write('x='); readln(x);
 suma:=suma+x; end;
if n>0 then begin media:=suma/n;
 writeln('media=', media); end
else writeln('media=*****');
readln;
 end.
```

Modulul 2

Recursivitatea

2.1. Teorie

"Pentru a intelege recursivitatea, trebuie sa intelegi recursivitatea". Chiar daca ati zambit sau nu, fraza de mai sus este o gluma inventata de catre pasionatii de informatica si explica foarte bine esenta tehnicii date de informare[6]

Recursivitatea este situatia in care programul se autoapeleaza, o metoda repetativa fara utilizarea de cicluri. Aceasta se imaprte in doua parti: directa si indirecta. Recursivitatea directa este atunci cand solutia finala este calculabila, adica are caz elementar(ex: n=0), iar cea indirecta contine caz neelementar(ex: n<0), adica solutia nu este direct calculabila. Diferenta dintre

acestea doua este ca recursivitatea directa are o definitie consistenta, astfel e finita, iar cea indirecta contine definitie inconsistenta, adica in teorie e infinita. In practica totusi aceasta se opreste atunci cand capacitatea de memorie este depasita[5]

2.2. Exemple de problem

Pentru a rezolva o problema in mod recursiv este nevoie de a o reprezenta in subprobleme la care sa adaugam cazul elementar sau neelementar pentru a putea opri problema. Urmatoarele programe sau subprograme sunt luate din [5] si [8]

1)
```
function F(n: Natural): Natural;
    var i, p : Natural;
    begin
    p:=1;
    for i:=1 to n do p:=p*i; F:=p;
    end; {F}
```

2)
```
function Fact(n:Natural):Natural;
    begin
    if n=0 then Fact:=5
    else Fact(n+6)*Fact:=n
    end;
```

3)
```
PROCEDURE  Zig(Num: INTEGER; VAR Square: INTEGER);
    BEGIN
    WriteLn('In Zig:  Num = ', Num: 2); { Trace }
    Zag(Num - 1,  Square);
    Square := Num +  Square;
    END; { Zig }
    PROCEDURE  Zag(Num: INTEGER; VAR Square: INTEGER);
    BEGIN
    WriteLn('In Zag:  Num = ', Num: 2); { Trace }
    IF Num = 0 THEN
    Square := 0 ELSE
    Zig(Num - 1,  Square);
    END; { Zag }
    VAR TrialNum,  TopOddNum, Sq: INTEGER;
    BEGIN
    Write('Enter a  value ');
    Read(TrialNum);
    TopOddNum := 2 *  TrialNum - 1;
    Zig(TopOddNum,  Sq);
```

```
WriteLn('The  square is ', Sq:2);
END.
```

4)
```
PROGRAM  GlobalLocal;
VAR Ch: CHAR; (*  Global declaration *)
 PROCEDURE GloLo;
 (* Uses local  declaration of Ch *)
 VAR Ch: CHAR; (*  Local declaration *)
 BEGIN
 Read(Ch);
 IF Ch <>  '.' THEN BEGIN
 GloLo;
 Write(Ch);
 END;
 END; { GloLo }
 PROCEDURE  GloGlo;
 (* Uses global  declaration of Ch *)
 BEGIN
 Read(Ch);
 IF Ch <>  '.' THEN BEGIN
 GloGlo;
 Write(Ch);
 END;
 END; { GloGlo }
 BEGIN
 WriteLn('Enter a sentence ending with a period');
 GloLo;
WriteLn;
WriteLn('Enter a sentence ending with a period');
 GloGlo;
 WriteLn;
END. {  GlobalLocal }
```

5)
```
Function IterSquareFunc (Num:integer):integer;
Var square, count : integer;
Begin
Square :=0;
For count:=1 to num do
Square:=2*count-1+square;
IterSquareFunc := square;
End;
```

Modulul 3

Relatia dintre iterativitate si recursivitate

Tabelul dat este inspirat din sursa [5]

Nr.	Caracteristici	Iterativitate	Recursivitate
1	Necesarul de memorie	mic	mare
2	Timpul de executie	acelasi	
3	Structura programului	complicata	simpla
4	Volumul de munca necesar pentru scrierea programului	mare	mic
5	Testarea si depanarea programelor	simpla	complicata

Bibliografie

1. https://www.scribd.com/document/337119802/Iterativitatea?language_settings_changed=English
2. https://searchsoftwarequality.techtarget.com/definition/iterative
3. https://www.dictionary.com/
4. "Manual de informatica clasa a IX-a" de Anatol Gremalschi
5. "Manual de informatica clasa a XI-a" de Anatol Gremalschi
6. https://en.wikipedia.org/wiki/Recursion
7. https://www.geeksforgeeks.org/recursion/
8. https://www.wisdomjobs.com/e-university/pascal-programming-tutorial-168/recursion-in-pascal-7055.html
9. Imagini STOCKPHOTO.COM/ANDREW SUSLOV