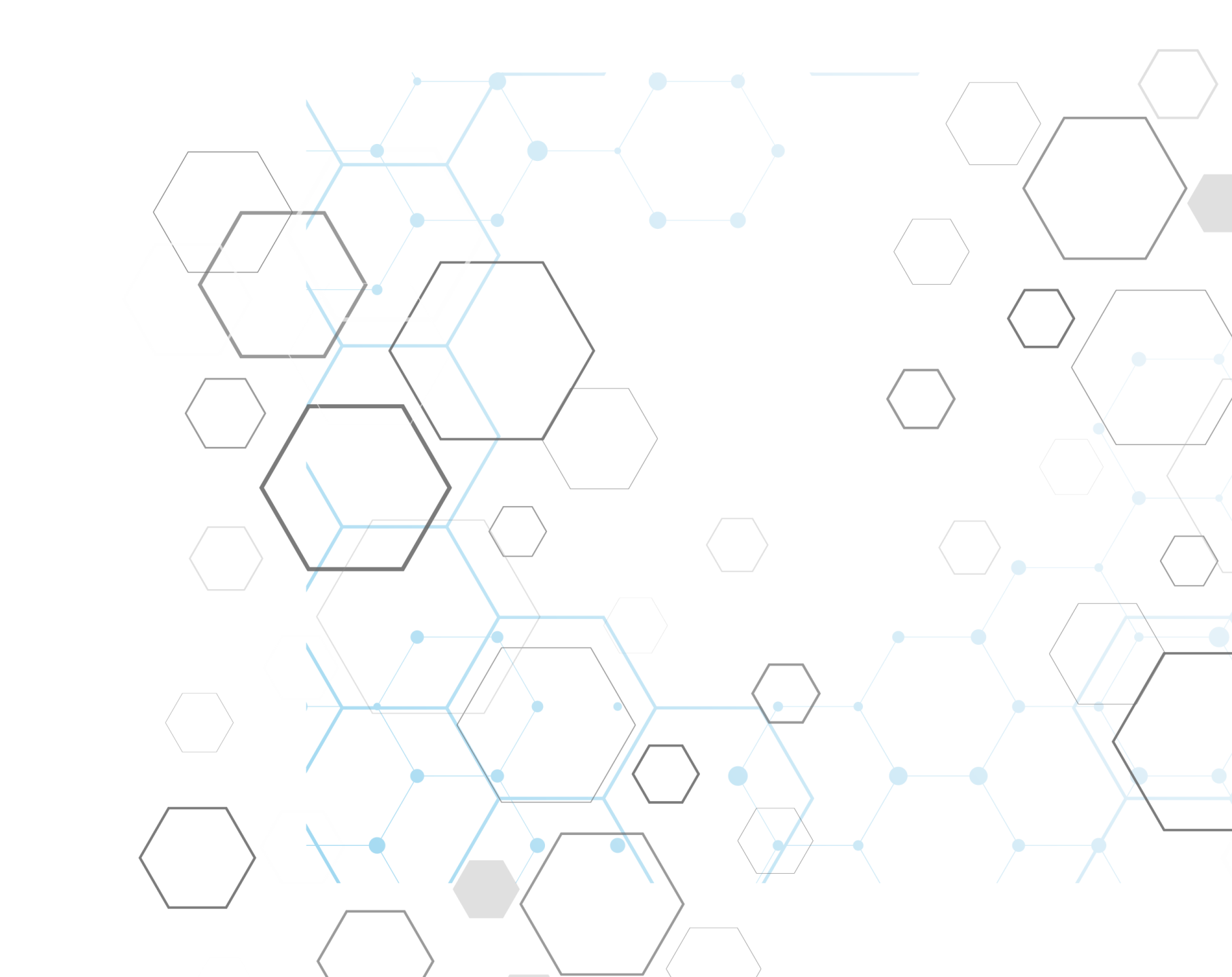
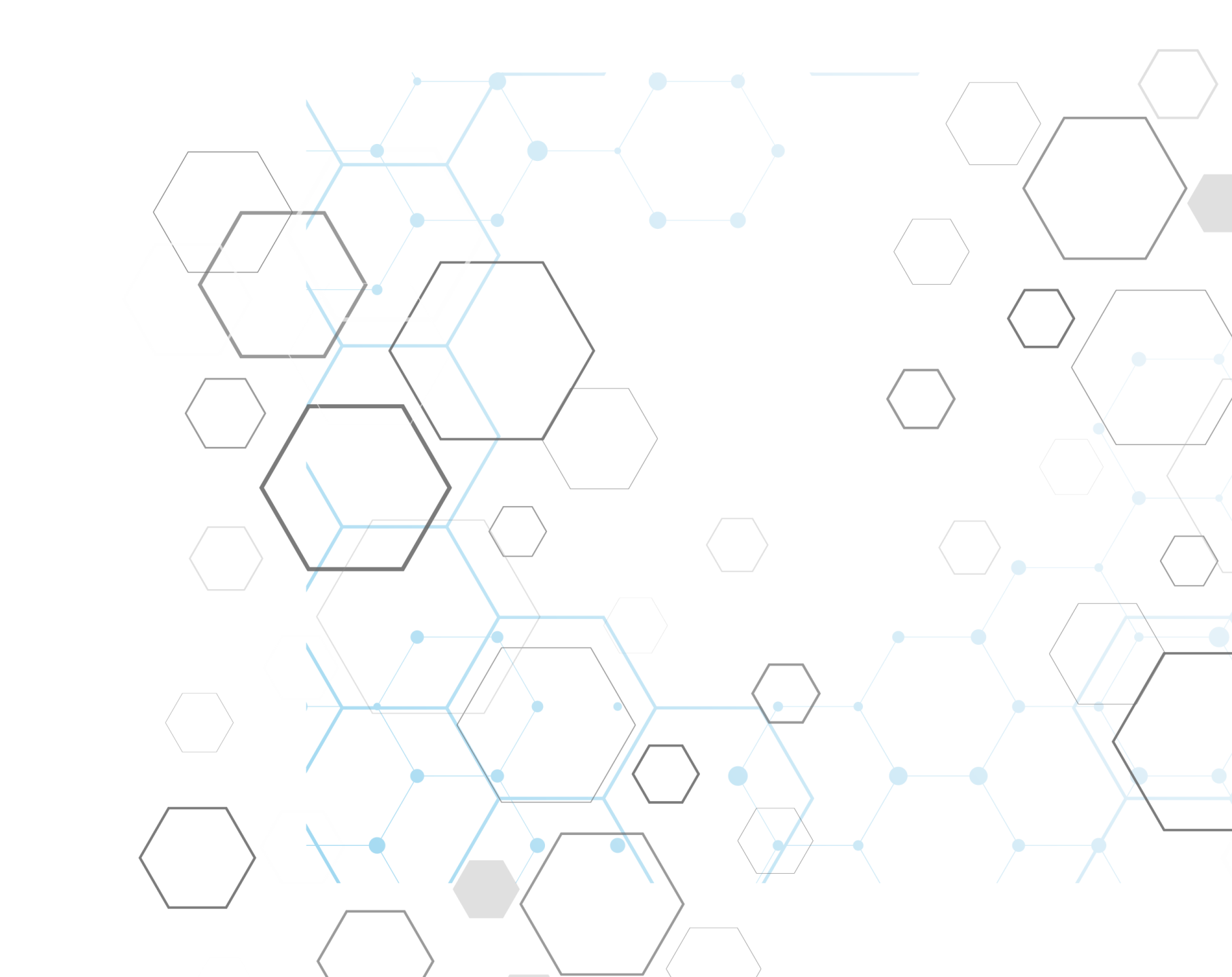
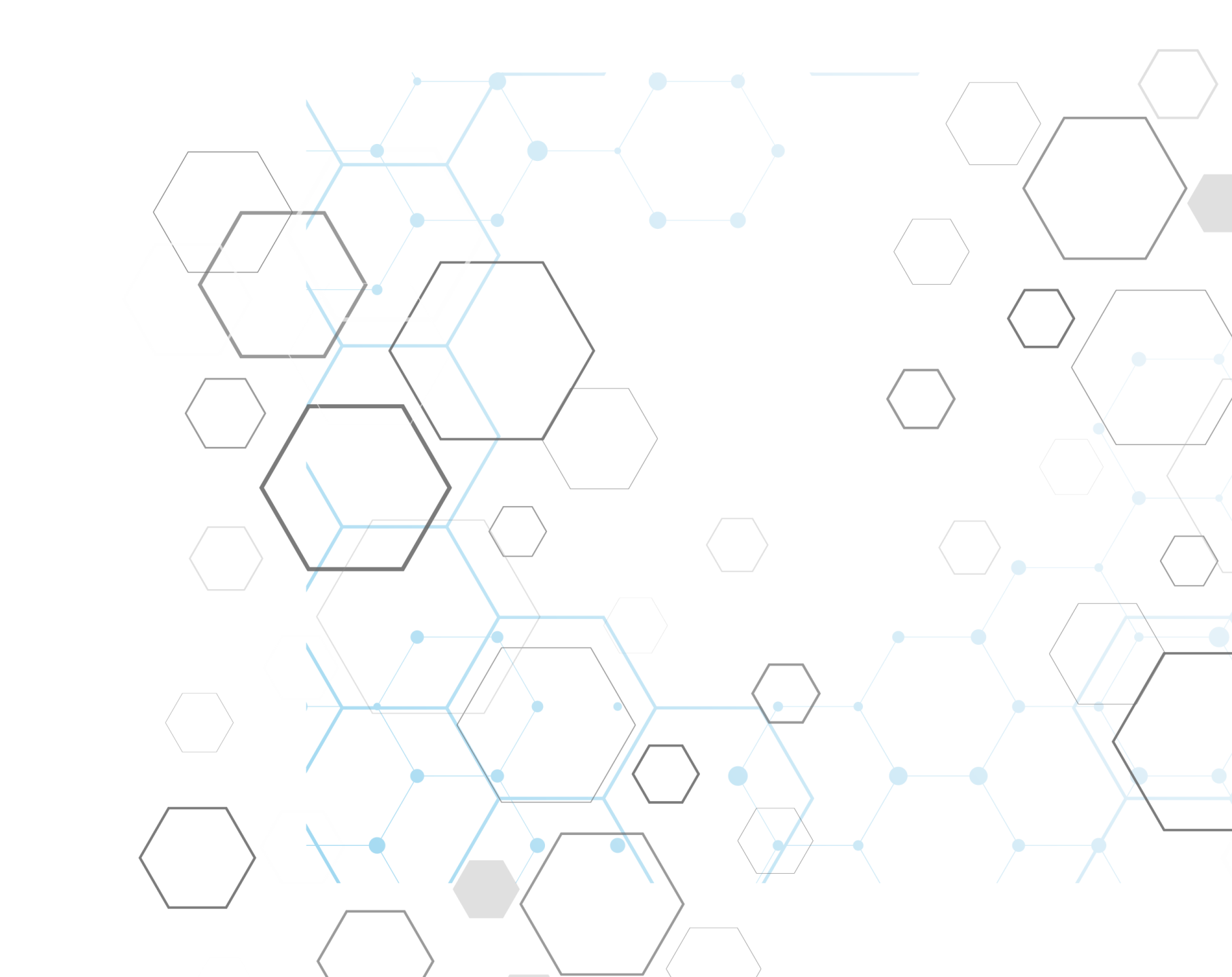
Iterativitate

sau

Recursivitate







Ciumașu Galina

cl. a XI-a ”D”

IPLT ”Spiru Haret”

Cuprins

[Iterativitate sau recursivitate? 3](#_Toc37173291)

[Recursivitatea 4](#_Toc37173292)

[Iterativitatea 5](#_Toc37173293)

[Probleme rezolvate 6](#_Toc37173294)

[Probleme recursiv 6](#_Toc37173295)

[Probleme iterativ 7](#_Toc37173296)

[Concluzii: 8](#_Toc37173297)

[Bibliografie: 8](#_Toc37173298)

Iterativitate sau recursivitate? [1]

Orice algoritm recursiv poate fi transcris într-un algoritm iterativ sau invers. Alegerea tehnicii de programare – iterativitatea sau recursivitatea – ține, de asemenea, de competența programatorului. Evident, Această alegere trebuie făcută luând în considerare avantajele și neajunsurile ficărei metode, care variază de la caz la caz.

Studiul comparativ**:** [2]

|  |  |  |
| --- | --- | --- |
| **Caracteristici** | **Iterativitate** | **Recursivitate** |
| Necesarul de memorie | Mic | Mare |
| Timpul de execuție | Același | |
| Structura programului | Complexă | Simplă |
| Volumul de muncă | Mare | Mic |
| Testare și depănare | Simplă | Complexă |

## Recursivitatea

Noțiunea de recursivitate se găsește și în matematică, unde este o modalitate de definiție a conceptelor. Numită și definiția prin inducție, este principiul care corespunde demonstrației prin inducție metematică. Recursivitatea este procesul iterativ prin care valoarea unei variabile se determină pe baza uneia sau a mai multora dintre propriile ei valori anterioare. Structurile recursive reprezintă o alternativă de realizare a proceselor repetitive fără a utiliza cicluri. În programare, recursivitatea este implementată cu ajutorul subprogramelor. Un subprogram este recursiv dacă el se autoapelează.

Tipuri de recursivitate (recursii):

1. Recursivitatea directă, care se prezintă prin autoapelul subprogramului în interiorul lui;
2. Recursivitate indirectă, în care apelul subprogramului este realizat în alt subprogram pe care acesta îl apelează. Procesul se încheie în momentul în care este îndeplinită condiția de ieșire din subprogramul care a inițiat procsul de apeluri reciproce.

Reguli pentru scrierea corectă a subprogramelor recursive:

1. Orice subprogram recursiv trebuie să se execute, cel puțin o dată, fără a se autoapela. Condiția de oprire se va scrie pentru valori extreme ale mulțimii valorilor de testat.
2. Autoapelurile trebuie să conducă spre situația în care subprogramul se execută direct (fără autoapel). [4] [5]

## 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.

***Iterația*** este execuţia repetată a unei porţiuni de program până la îndeplinirea unei condiţii(while,for etc.). După cum s-a văzut, orice algoritm recursiv poate fi transcris într-un algoritm iterativ ți invers. Alegerea tehnicii de programare-iterativitatea sau recursivitatea- ține de competenta programatorului. [3]

# Probleme rezolvate [4] [5]

## Probleme recursiv

1. Factorialul unui număr:  
   program factorial;  
   var n:byte;  
   function fact(n:byte):longint;  
   begin  
    if n=0 then fact:=1  
    else fact:=n\*fact(n-1)  
   end;  
   begin  
    write(‘n=’);  
    readln(n);  
    writeln(n,’!=’ ,fact(n));  
    readln;  
   end.
2. Suma cifrelor unui număr natural:  
   program suma;  
   var n:integer;  
   function s(n:integer):integer;  
   begin  
    if n=0 then s:=0  
    else s:=n mod 10 + s(n div 10);  
   end;  
   begin  
   write(‘n=’); readln(n);  
   writeln(‘s=’, s(n));  
   readln;  
   end.
3. Calculul celui mai mare divizor comun a două numere:  
   program cmmdc;  
   var a,b:integer;  
   function cmmdc(a,b:integer):integer;  
   begin  
    if b=0 then cmmdc:=a  
    else cmmdc:=cmmdc(b, a mod b);  
   end;  
   begin  
   write(‘a=’); readln(a);  
   write(‘b=’); readln(b);  
   writeln(‘cmmdc=’, cmmdc(a,b));  
   end.
4. **Suma unui șir**

var n:integer;  
Function F (N:Integer):Longint;  
Begin  
If N=0 Then F:=0  
Else F:=F(N-1)+N  
End;  
begin  
readln(n);  
writeln(F(n));  
end.

1. A\*b prin adunarea repetată:  
   function produs(a,b:integer):integer;  
   begin  
   if b=0 then produs:=0  
    else produs:=a+produs(a,b-1);  
   end;

## Probleme iterativ [5] [6]

1. Factorialul unui număr:  
   program factorial;  
   var n:byte;  
   function fact(n:byte):longint;  
   var i:byte; calcul:longint  
   begin   
   calcul:=1;   
   for i:=2 to n do   
   calcul:=calcul\*i;  
    fact:=calcul;   
   end;   
   begin   
   readln(n);   
   writeln('Fact=',fact(n));   
   end.
2. Suma cifrelor unui număr:  
   function suma\_cifre(n:word):byte;  
   var sum:byte;   
   begin sum:=0;   
   while n<>0 do begin   
    sum:=sum + n mod 10;  
    n:=n div 10;   
   end;   
   suma\_cifre:=sum   
   end;
3. a^b prin înmulțire repetată:  
   function putere(a,b:word):word;  
   var p:word;   
   begin p:=1;   
   while b<>0 do begin

p:=p \* a;

b:=b – 1;

end;  
putere:=p end;

1. a\*b prin adunare repetată:  
   function produs(a,b:word):word;   
   var p:word;   
   begin   
   p:=0;   
   while b<>0 do begin

p:=p + a;

b:=b – 1;   
end;   
produs:=p end;

1. CMMDC:  
   function cmmdc(a,b:word):word;   
   var rest:word;   
   begin rest:=a;   
   while rest <> 0 do   
   begin rest:=a mod b; a:=b; b:=rest;   
   end;   
   cmmdc:=a; end;

# Concluzii:

Iterativitatea și recursivitatea se bazează pe o metodă mai clară de exprimare a algoritmilor de rezolvare. Orice algoritm recursiv poate fi transcris într-un algoritm iterativ și invers deci rezultă că modul ales depinde de competența fiecarui programator în parte și alegerea personală.

# Bibliografie:

* <http://ciub.org/curse_window.php?view=curse&mod=full&id=70> [2]
* <https://www.didactic.ro/materiale-didactice/iterativitate-sau-recursivitate> [1]
* <http://www.authorstream.com/Presentation/aSGuest41792-360322-iterativitate-sau-recursivitate-rodika-guzun-science-technology-ppt-powerpoint/> [3]
* „INFORMATICĂ - manual pentru clasa a X-a” de Daniela Oprescu. Editura NICULESCU ABC,2004. [4]
* „INFORMATICĂ - manual pentru clasa a X-a” de Livia Țoca, Andreea-Ruxandra Demnco, Cristian Opincaru, Adrian Sindile. Editura NICULESCU ABC,2004. [5]
* <https://manuale.edu.ro/manuale/Clasa%20a%20XI-a/Informatica/Niculescu2/A330.pdf?fbclid=IwAR1i4WgLK-6MJ82ycy09DBqRsFNDN5cK92MfwhNqVmuaFuU2y-arQxw61WM> [6]