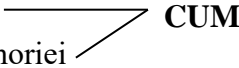


# 1. Programare și limbaje de programare

## ➤ Limbaje

- **Procedurale (imperative) – limbaje de nivel înalt**

- Fortran, Cobol, Algol, Pascal, C,...
- program – secvență de instrucțiuni
- instrucțiunea de atribuire, structuri de control – pentru controlul execuției secvențiale, ramificării și ciclării.
- rolul programatorului – “ce” și “cum”
  1. să descrie **CE** e de calculat
  2. să organizeze calculul
  3. să organizeze gestionarea memoriei
- !!! se susține că instrucțiunea de atribuire este periculoasă în limbajele de nivel înalt, așa cum instrucțiunea GO TO a fost considerată periculoasă pentru programarea structurată în anii '68.

- **Declarative (descriptive, aplicative) – limbaje de nivel foarte înalt**

- se bazează pe expresii
- expresive, ușor de înțeles (au o bază simplă), extensibile
- programele pot fi văzute ca descrieri care declară informații despre valori, mai degrabă decât instrucțiuni pentru determinarea valorilor sau efectelor.
- renunță la instrucțiuni
  1. protejează utilizatorii de la a face prea multe erori
  2. sunt generate din principii matematice - analiza, proiectarea, specificarea, implementarea, abstractizarea și raționarea (deducții ale consecințelor și proprietăților) devin activități din ce în ce mai formale.
- rolul programatorului - “ce” (nu “cum”)
- două clase de limbaje declarative
  1. **limbajele funcționale** (de exemplu Lisp, ML, Scheme, Haskell, Erlang)
    - se focalizează pe valori ale datelor descrise prin expresii (construite prin aplicări ale funcțiilor și definiții de funcții), cu evaluare automată a expresiilor
  2. **limbaje logice** (de exemplu Prolog, Datalog, Parlog), care se focalizează pe aserțiuni logice care descriu relațiile dintre valorile datelor și derivări automate de răspunsuri la întrebări, plecând de la aceste aserțiuni.
- aplicații în Inteligența Artificială – demonstrarea automată, procesarea limbajului natural și înțelegerea vorbirii, sisteme expert, învățare automată, agenți, etc.

- Limbaje multiparadigmă: **F#, Python, Scala** (imperativ, funcțional, orientat obiect)
- Interacțiuni între limbajele declarative și cele imperative – limbaje declarative care oferă interfețe cu limbaje imperative (ex C, Java): SWI-Prolog, GNUProlog, etc.

## 2. Recursivitate

- mecanism general de elaborare a programelor.
- recursivitatea a apărut din necesități practice (transcrierea directă a formulelor matematice recursive; vezi funcția lui Ackermann)
- recursivitatea este acel mecanism prin care un subprogram (funcție, procedură) se autoapelează.
- două tipuri de recursivitate: **directă** sau **indirectă**.
- două lucruri de considerat în descrierea unui algoritm recursiv: **regula recursivă** și **condiția de ieșire din recursivitate**.
- **avantaj** al recursivității: text sursă extrem de scurt și foarte clar.
- **dezavantaj** al recursivității: umplerea segmentului de stivă în cazul în care numărul apelurilor recursive, respectiv al parametrilor formali și locali ai subprogramelor recursive este mare.
  - în limbajele declarative există mecanisme specifice de optimizare a recursivității (vezi mecanismul recursivității de coadă în Prolog).