Paradigme de Programare

Conf. dr. ing. Andrei Olaru

andrei.olaru@upb.ro
Departamentul de Calculatoare

2026

Ce este o paradigmă de programare?

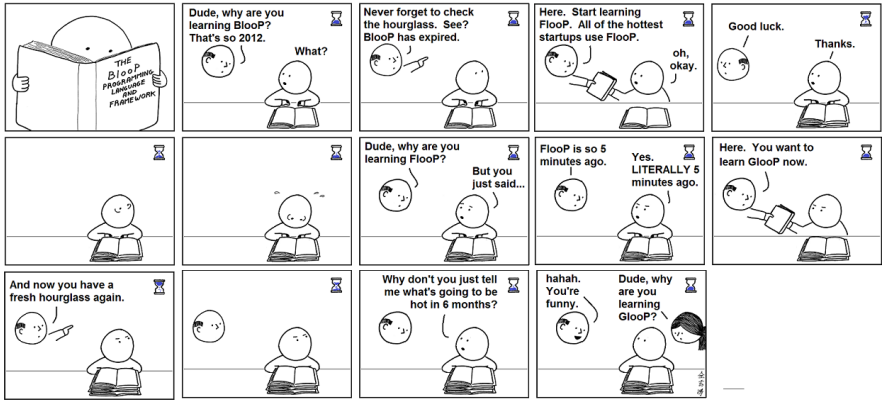
\+ ceva exemple   

\+  Racket

Cursul 1: Introducere

BlooP and FlooP and GlooP

[(CC) BY-NC abstrusegoose.com] [http://abstrusegoose.com/503]



Exemplu

Exemplu

Exemplu

Să se determine dacă un element e se regăsește într-o listă L ($e \in L$).

Să se sorteze o listă L.

Modelare funcțională (1)

Racket:

```
1  (define memList (lambda (e L)
2      (if (null? L)
3          #f
4          (if (equal? (first L) e)
5              #t
6              (memList e (rest L))
7              ))
8      ))
9
10 (define ins (lambda (x L)
11  (cond ((null? L) (list x))
12        ((< x (first L)) (cons x L))
13        (else (cons (first L) (ins x (rest L)))))))
```

Modelare funcțională (2)

Haskell

```
1 memList x [] = False
2 memList x (e:t) = x == e || memList x t
3
4 ins x [] = [x]
5 ins x l@(h:t) = if x < h then x:l else h : ins x t
```

Prolog:

```
1  memberA(E, [E|_]) :- !.
2  memberA(E, [_|L]) :- memberA(E, L).
3
4  % elementul, lista, rezultatul
5  ins(E, [], [E]).
6  ins(E, [H | T], [E, H | T]) :- E < H, !.
7  ins(E, [H | T], [H | TE]) :- ins(E, T, TE).
```

Ce studiem la PP?

Elemente pe care le vom studia

- Paradigma funcțională și discuție despre paradigme de programare în general, în contrast cu paradigma imperativă și cea orientată-obiect.
- Tipuri de date abstracte

Elemente pe care le vom studia

- Paradigma funcțională și discuție despre paradigme de programare în general, în contrast cu paradigma imperativă și cea orientată-obiect.
- Tipuri de date abstracte
- Racket: introducere în programare funcțională
- Calculul λ ca bază teoretică a paradigmei funcționale
- Racket: întârzierea evaluării și fluxuri

Elemente pe care le vom studia

- Paradigma funcțională și discuție despre paradigme de programare în general, în contrast cu paradigma imperativă și cea orientată-obiect.
- Tipuri de date abstracte
- Racket: introducere în programare funcțională
- Calculul λ ca bază teoretică a paradigmei funcționale
- Racket: întârzierea evaluării și fluxuri

- Haskell: programare funcțională cu o sintaxă avansată
- Haskell: evaluare leneșă și fluxuri
- Haskell: tipuri, sinteză de tip, și clase
- Equational Reasoning

De ce studiem această materie?

De ce?

Ne vor folosi aceste lucruri în viața reală?

The first math class.

The first math class.

[(C) Zach Weinersmith, Saturday Morning Breakfast Cereal]

[https://www.smbc-comics.com/comic/a-new-method]

De ce?

I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.

The law of instrument – Abraham Maslow

De ce?
Mai concret

· până acum ați studiat paradigma imperativă (legată și cu paradigma orientată-obiect)

→ un anumit mod de a privi procesul de rezolvare al unei probleme și de a căuta soluții la probleme de programare.

· paradigmele declarative studiate oferă o gamă diferită (complementară!) de unelte → alte moduri de a rezolva anumite probleme.

⇒ o pregătire ce permite accesul la poziții de calificare mai înaltă (arhitect, designer, etc.)

De ce?

Sunt aceste paradigme relevante?

- evaluarea lenească → prezentă în Python (de la v3), .NET (de la v4)
- funcții anonime → prezente în C++ (de la v11), C#/.NET (de la v3.0/v3.5), Dart, Go, Java (de la JDK8), JS/ES, Perl (de la v5), PHP (de la v5.0.1), Python, Ruby, Swift.
- Prolog și programarea logică sunt folosite în software-ul modern de A.I., e.g. Watson; automated theorem proving.
- În industrie sunt utilizate limbaje puternic funcționale precum Erlang, Scala, F#, Clojure.
- Limbaje multi-paradigmă → adaptarea paradigmei utilizate la necesități.

De ce?

O bună cunoaștere a paradigmelor alternative → $$$

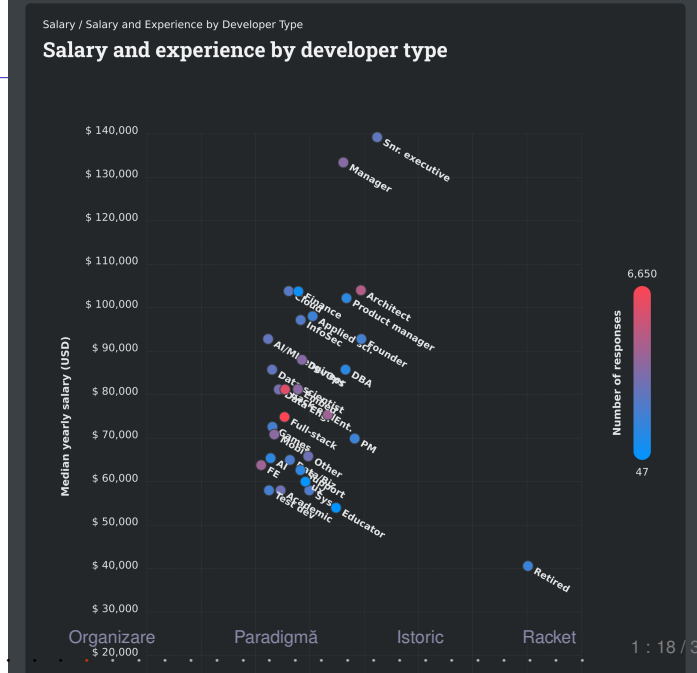
- Developer Survey 2025

  [https://survey.stackoverflow.co/2025]

- Developer Survey 2024

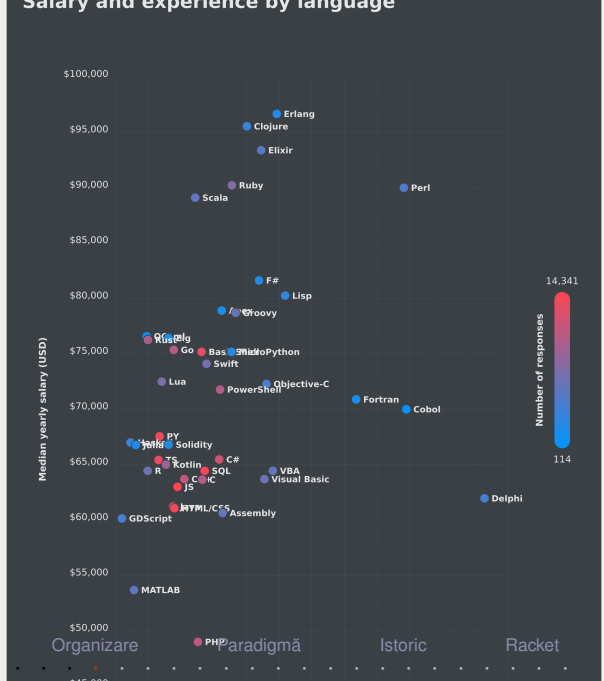
  [https://survey.stackoverflow.co/2024]

De ce?
Cine câștigă cel mai bine?



De ce?

Cine câștigă cel mai bine?



Organizare

Unde găsesc informații?

Resurse de bază

https://ocw.cs.pub.ro/courses/pp

Regulament: https://ocw.cs.pub.ro/courses/pp/--/regulament

Forumuri: Moodle → 03-ACS-L-CTI-Calculatoare-A2-S2-PP-CA-CB-CC
https://curs.upb.ro/2025/course/view.php?id=10195

Elementele cursului sunt comune la seriile CA, CB și CC.

Notare

mai multe la https://ocw.cs.pub.ro/courses/pp/--/regulament

- Laborator: 0.2p ← pentru strădunință
- Teste grilă la laborator: 0.3p ← cu bonus până la 0.4p
- Teme (2): TBD ← cu bonusuri de până la 20%
- Activitate curs: 0.5p ← se punctează întrebările bune și foarte bune
- Test din materia de laborator: TBD

punctajele pe parcurs se trunchiază la 5p

- Examen: 5p ← limbaje + teorie

Paradigma de programare

Ce înseamnă paradigma de programare

Ce diferă între paradigme?

Ce înseamnă paradigma de programare

Ce diferă între paradigme?

- diferă sintaxa ←
  - aceasta este o diferență între limbaje, dar este influențată și de natura paradigmei
  - mecanisme specifice unei paradigme aduc elemente noi de sintaxă

    e.g. funcțiile anonime

Ce înseamnă paradigma de programare

Ce diferă între paradigme?

- diferă sintaxa ←
  - aceasta este o diferență între limbaje, dar este influențată și de natura paradigmei
  - mecanisme specifice unei paradigme aduc elemente noi de sintaxă

    e.g. funcțiile anonime

Ce înseamnă paradigma de programare

Ce diferă între paradigme?

- diferă sintaxa ←
  - aceasta este o diferență între limbaje, dar este influențată și de natura paradigmei
  - mecanisme specifice unei paradigme aduc elemente noi de sintaxă

    e.g. funcțiile anonime
- diferă modul de construcție al expresiilor ← ce poate reprezenta o expresie, ce operatori putem aplica între expresii.

Ce înseamnă paradigma de programare

Ce diferă între paradigme?

- diferă sintaxa ←
  - aceasta este o diferență între limbaje, dar este influențată și de natura paradigmei
  - mecanisme specifice unei paradigme aduc elemente noi de sintaxă

    e.g. funcțiile anonime
- diferă modul de construcție al expresiilor ← ce poate reprezenta o expresie, ce operatori putem aplica între expresii.
- diferă structura programului ←
  - ce anume reprezintă programul
  - cum se desfășoară execuția programului

Ce înseamnă paradigma de programare

Ce caracterizează o paradigmă?

- valorile de prim rang
- modul de construcție a programului
- modul de tipare al valorilor
- ordinea de evaluare (generare a valorilor)
- modul de legare al variabilelor (managementul valorilor)
- controlul execuției

· Paradigma de programare este dată de stilul fundamental de construcție al structurii și elementelor unui program.

Ce vom studia?
Conținutul cursului

1. Diverse perspective conceptuale asupra noțiunii de calculabilitate efectivă → modele de calculabilitate.

2. Influența perspectivei alese asupra procesului de modelare și rezolvare a problemelor → paradigme de programare.

3. Limbaje de programare aferente paradigmelor, cu accent pe aspectul comparativ.

Modele → paradigme → limbaje

Modele de calculabilitate

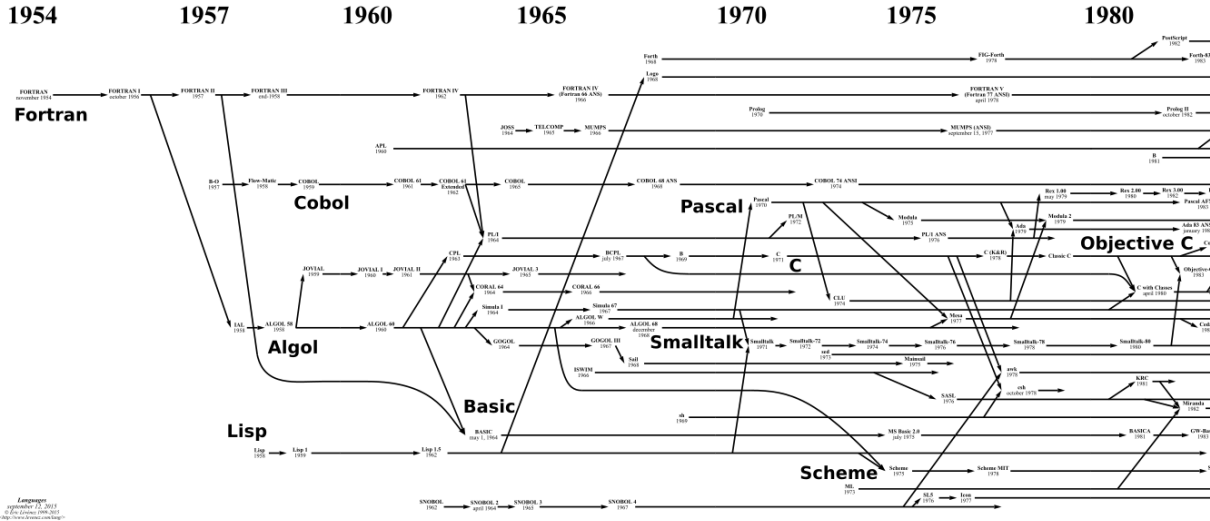
C, Pascal → procedural Java, C++, Python → orientat-obiect	→ paradigma imperativă	→ Mașina Turing
Racket, Haskell	→ paradigma funcțională	→ Mașina λ
Prolog	→ paradigma logică	→ FOL + Resolution
CLIPS	→ paradigma asociativă	→ Mașina Markov

echivalente !

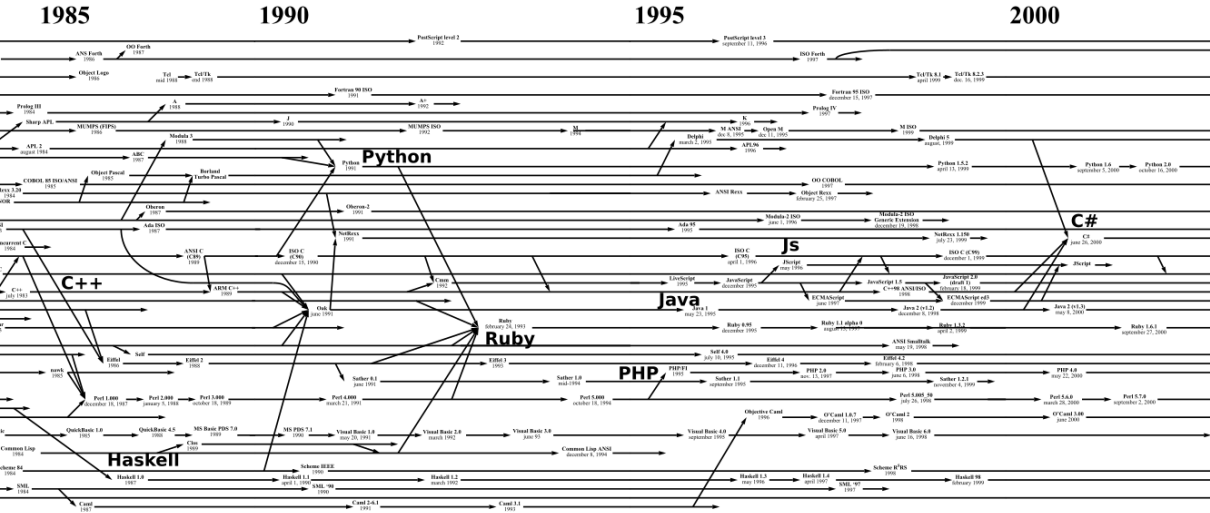
T | Teza Church-Turing: efectiv calculabil = Turing calculabil

Istoric: Paradigme și limbaje de programare

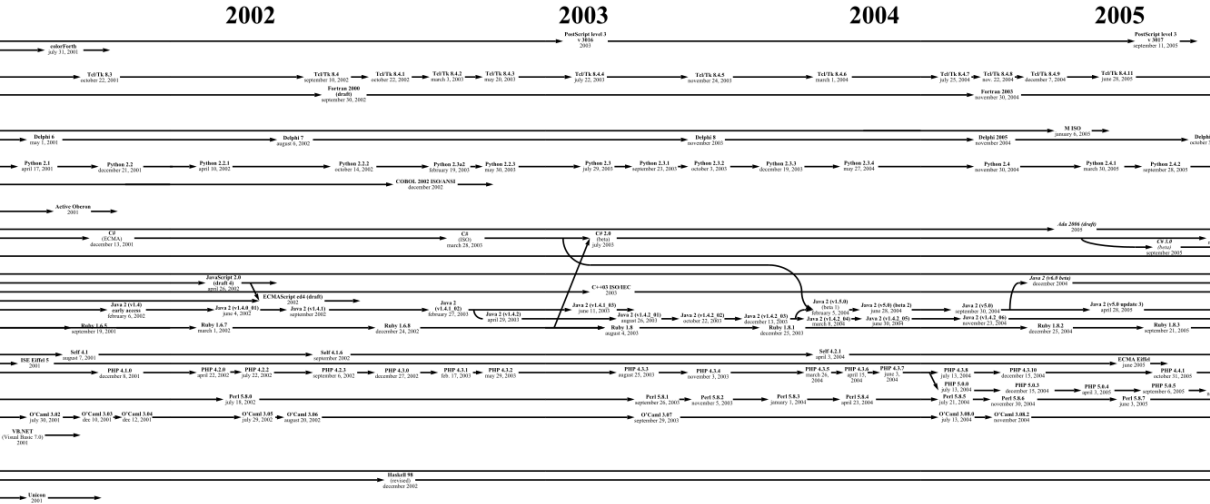
Istorie
1950-1975



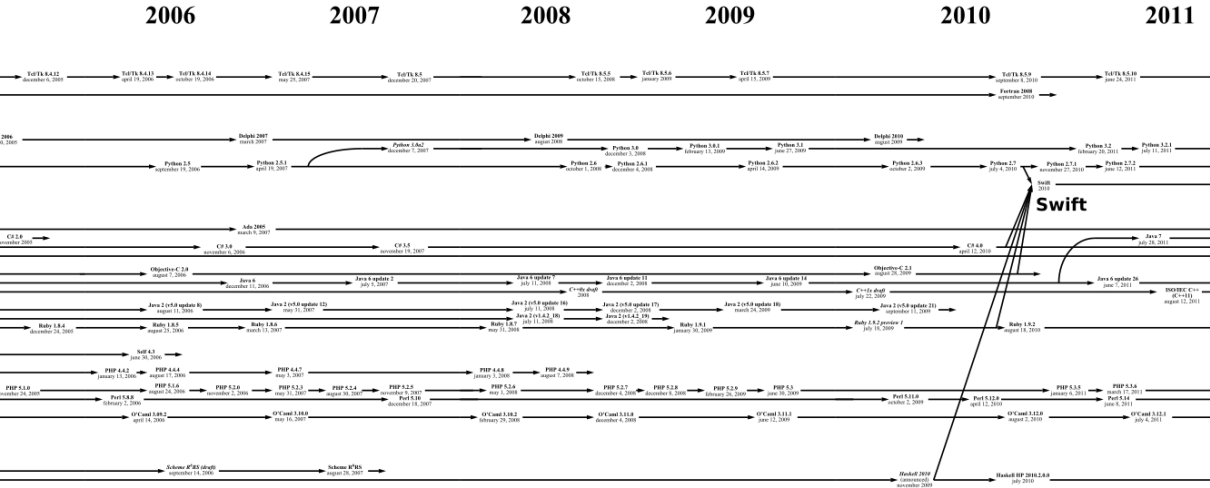
Istorie
1975-1995



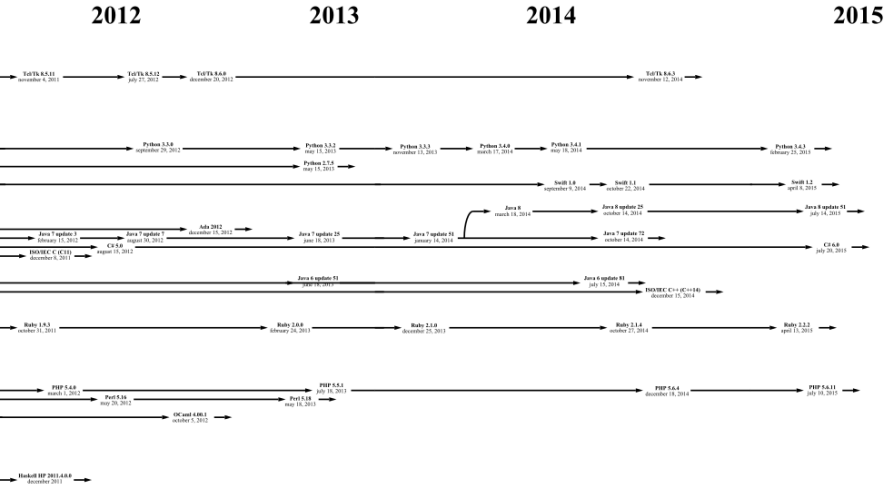
Istorie
1995-2002



Istorie
2002-2006



Istorie
2006-2013



Istorie
Resurse

- imagine navigabilă (slides precedente): [http://www.levenez.com/lang/]

- Wikipedia:

[http://en.wikipedia.org/wiki/Generational_list_of_programming_languages]

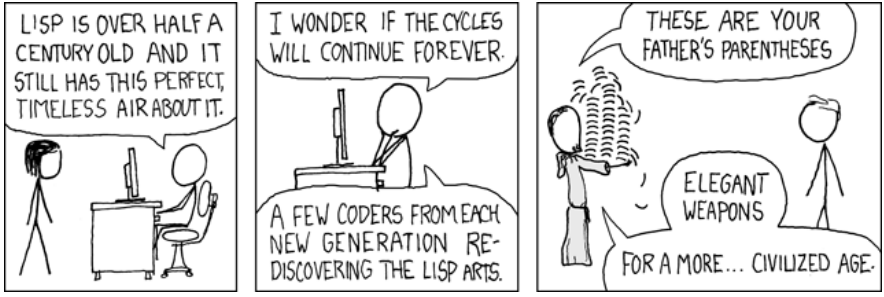
[https://en.wikipedia.org/wiki/Timeline_of_programming_languages]

[https://en.wikipedia.org/wiki/Programming_paradigm#/media/File:Programming_paradigms.svg]

Introducere în Racket

Lisp cycles

[http://xkcd.com/297/]



Racket

din 1975 (ca Scheme)

- funcțional
- dialect de Lisp
- totul este văzut ca o funcție
- constante – expresii neevaluate
- perechi / liste pentru structurarea datelor
- apeluri de funcții – liste de apelare, evaluate
- evaluare aplicativă, funcții stricte, cu anumite excepții

[http://xkcd.com/859/]

+ Dați feedback la acest curs aici:
[https://forms.gle/ETsHHPvn2nDgF7q27]