

Paradigme de Programare

Conf. dr. ing. Andrei Olaru

andrei.olaru@upb.ro | cs@andreiolaru.ro
Departamentul de Calculatoare

2021



Ce este o paradigmă de programare?

+ ceva exemple   

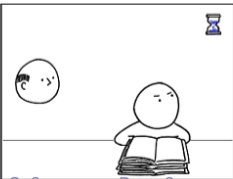
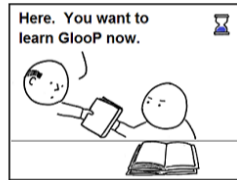
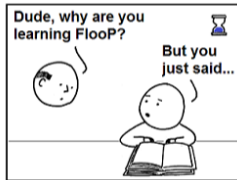
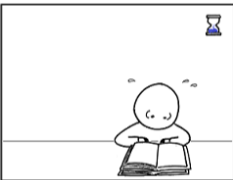
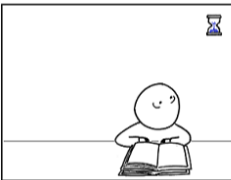
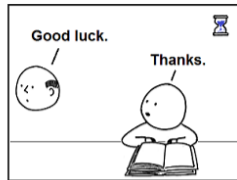
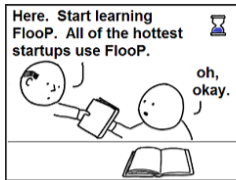
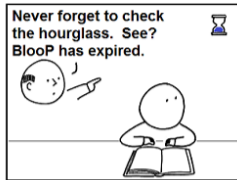
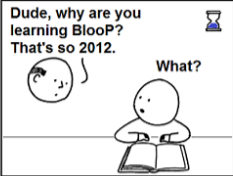
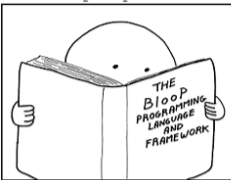
+  Racket

Cursul 1: Introducere

- 1 Exemplu
- 2 Ce studiem la PP?
- 3 De ce studiem această materie?
- 4 Organizare
- 5 Introducere în Racket
- 6 Paradigma de programare
- 7 Istoric: Paradigme și limbaje de programare

BlooP and FlooP and GloopP

[<http://abstrusegoose.com/503>]



Exemplu



Exemplu

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

Să se sorteze o listă L .

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)))))))
```

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?

- Paradigma funcțională și paradigma logică, în contrast cu paradigma imperativă.

- Paradigma funcțională și paradigma logică, în contrast cu paradigma imperativă.
- Racket: introducere în programare funcțională
- Calculul λ ca bază teoretică a paradigmei funcționale
- Racket: întârzierea evaluării și fluxuri

- Paradigma funcțională și paradigma logică, în contrast cu paradigma imperativă.
- 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

- Paradigma funcțională și paradigma logică, în contrast cu paradigma imperativă.
- 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
- Prolog: **programare logică**
- **LPOI** ca bază pentru programarea logică
- Prolog: strategii pentru controlul execuției

- Paradigma funcțională și paradigma logică, în contrast cu paradigma imperativă.
- 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
- Prolog: **programare logică**
- **LPOI** ca bază pentru programarea logică
- Prolog: strategii pentru controlul execuției
- Algorimi Markov: calcul bazat pe **reguli de transformare**

De ce studiem această materie?



The first math class.

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

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

The first math class.

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

· 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.)

Sunt aceste paradigme relevante?

- **evaluarea leneșă** → 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.

- Developer Survey 2019

[<https://insights.stackoverflow.com/survey/2019/#top-paying-technologies>]

[<https://insights.stackoverflow.com/survey/2019/#salary>]

- Developer Survey 2018

[<https://insights.stackoverflow.com/survey/2018/>

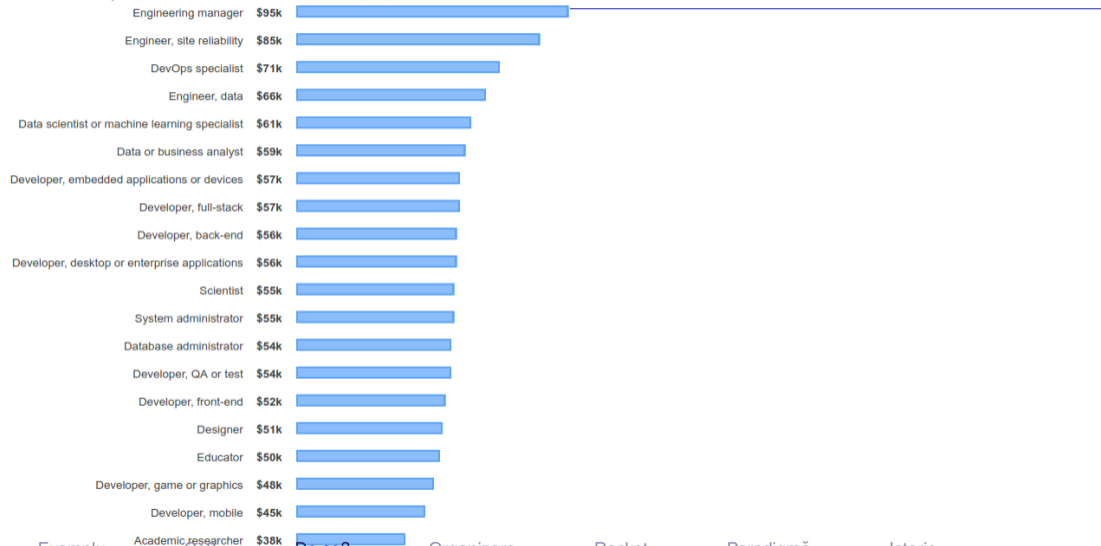
[#technology-what-languages-are-associated-with-the-highest-salaries-worldwide](#)]

- Developer Survey 2017

[<https://insights.stackoverflow.com/survey/2017/#top-paying-technologies>]

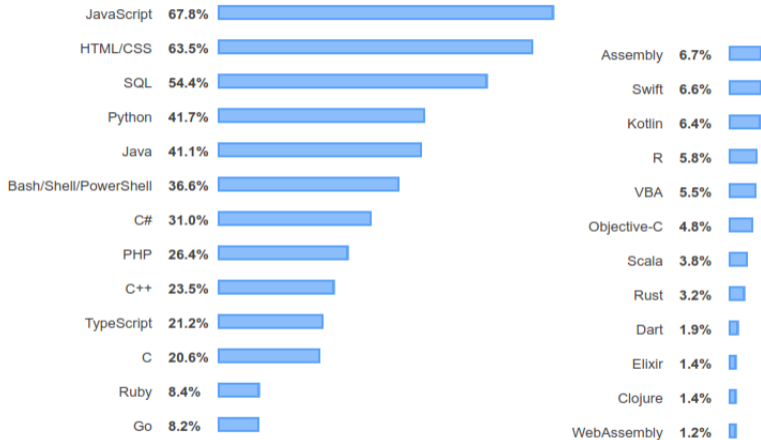
De ce?

Cine câștigă cel mai bine? (1)



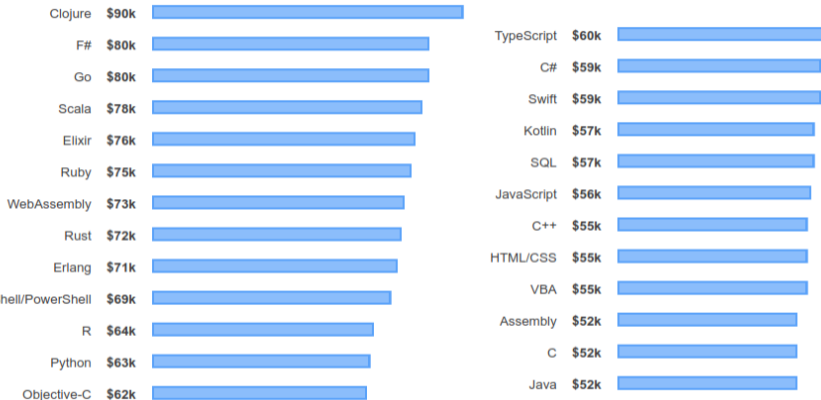
De ce?

Cine câștigă cel mai bine?



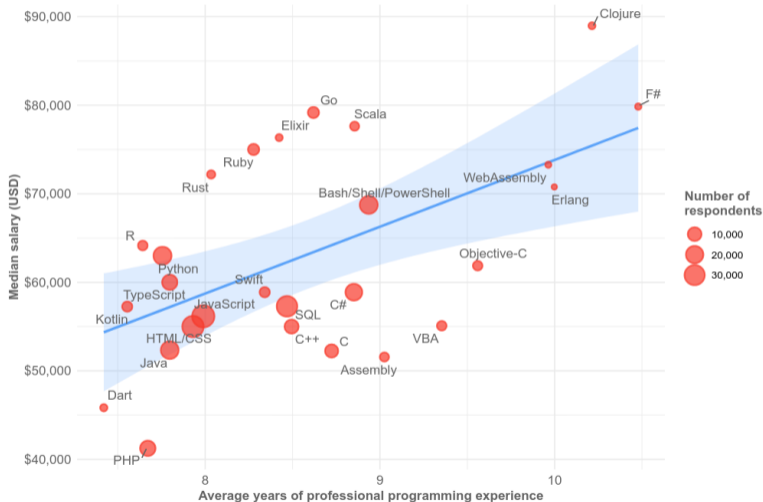
De ce?

Cine câștigă cel mai bine?



De ce?

Cine câștigă cel mai bine?



Exemplu

Ce?

De ce?

Organizare

Racket

Paradigmă

Istoric

Organizare

Exemplu

Ce?

De ce?

Organizare

Racket

Paradigmă

Istoric

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

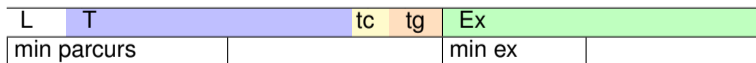
Regulament: `https://ocw.cs.pub.ro/courses/pp/21/regulament`

Forumuri: Moodle → 03-ACS-L-A2-S2-PP-CA-CC-CD

`https://curs.upb.ro/course/view.php?id=11557`

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

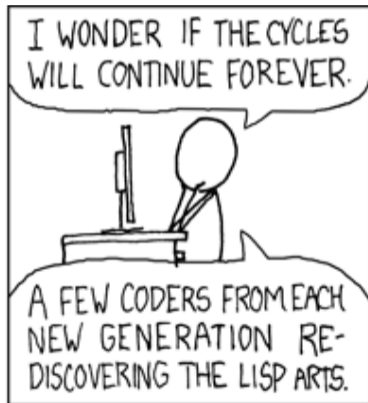
- Laborator: 1p ← activitate 0.7p + test grilă din laborator 0.3p
- Teme: 4p ($3 \times 1.33p$) ← cu bonusuri, dar în limita a maxim 6p pe parcurs
- Teste la curs: 0.5p ← punctare pe parcurs, la curs
- Test din materia de laborator: 0.5p ← test grilă, de cunoaștere a limbajelor
- Examen: 4p ← limbaje + teorie



Introducere în Racket

Lisp cycles

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



[(CC) BY-NC Randall Munroe, xkcd.com]

- 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

Paradigma de programare

- diferă sintaxa

- 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 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

- 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

- aceasta este o diferență între limbaje, dar este influențată și de natura paradigmei
- diferă sintaxa ← - 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

- 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.

- 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 paradigmatelor, cu accent pe aspectul comparativ.

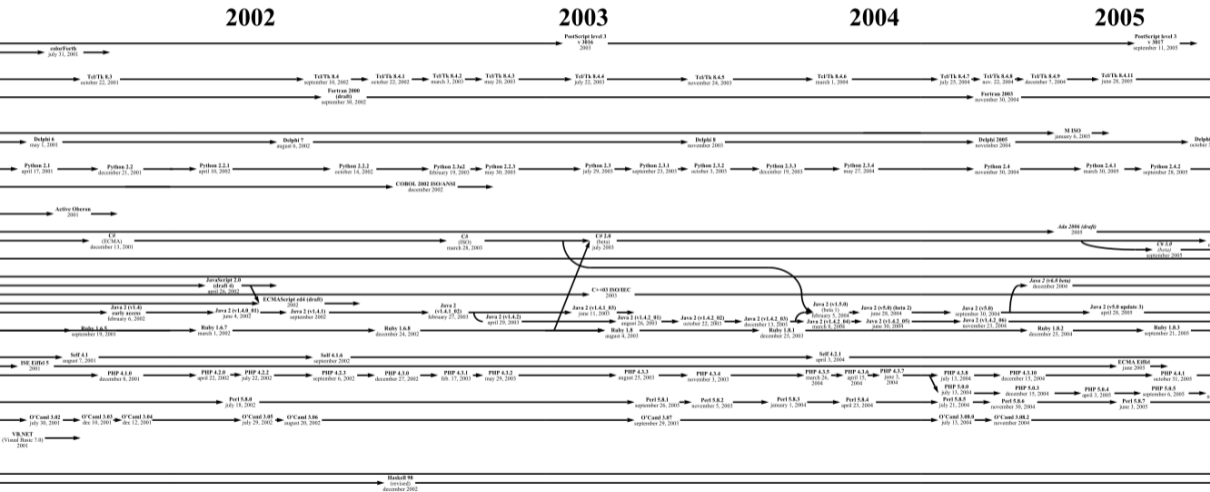
C, Pascal → procedural J, C++, Py → orientat-obiect	→ paradigma imperativă	→ Mașina Turing
Racket, Haskell	→ paradigma funcțională	→ Mașina λ
Prolog	→ paradigma logică	→ FOL + Resolution
CLIPS	→ paradigma asociativă	→ Mașina Markov

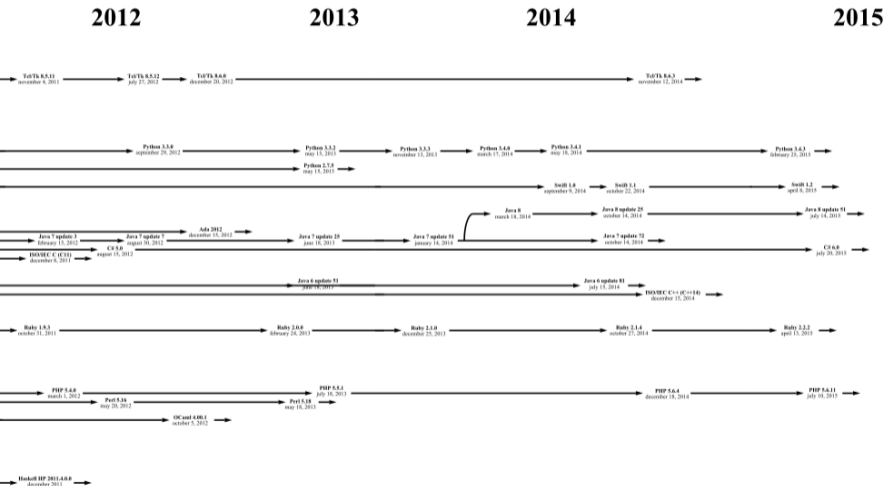
echivalente !

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

Istoric: Paradigme și limbaje de programare

Istorie 1995-2002





- 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]

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

(AN UNMATCHED LEFT PARENTHESIS
CREATES AN UNRESOLVED TENSION
THAT WILL STAY WITH YOU ALL DAY.

[(CC) BY-NC xkcd.com]

+ Dați feedback la acest curs aici:

[https://docs.google.com/forms/d/e/1FAIpQLSepDn0hMs1YqnEk79W_-6y9PWc7IzR1thr1A2bVM32Jk1cRBQ/viewform]