# Introducció als llenguatges de programació

Albert Rubio

Llenguatges de Programació, FIB, UPC

Primavera 2016

# Propietats principals

- Turing completesa.
- Paradigmes de la programació.
- Compilat vs Interpretat.
- Sistemes de tipus.

# Turing completesa

*Turing completeness* en anglès.

- Un llenguatge de programació és *Turing complet* si pot simular una Màquina de Turing (d'una sola cinta).
- Model de càlcul molt simple creat per Allan Turing que pot realitzar qualsevol còmput que un computador digital pugui realitzar.
- Una sola cinta infinita i un capçal que llegeix i modifica el contingut de la cel·la en curs. El capçal es pot desplaçar a la dreta i a l'esquerra.

# Turing completesa

Model alternatiu *lambda càlcul* d'Alonzo Church.

Model funcional molt simple equivalent a la Màquina de Turing.

*Tesi de Church-Turing* mostrada pels dos independentment:
Tot el que és algorísmicament computable és computable

- amb una Màquina de Turing o
- amb una funció en lambda càlcul.

Si tot es pot calcular amb un model tan simple, perquè hi ha tans lleguatges de programació?

- Eficiència
- Facilitat d'escriptura/llegibilitat
- ...

# Turing completesa

Alguns autors consideren només com a lleguatges de programació els llenguatges Turing complets.

Exemples de llenguatges no Turing complets:

- expressions regulars (p.e. a Perl o a AWK).
- SQL.

Per ser Turing complet només cal tenir salts condicionals (bàsicament, **if** i **goto**).

# Paradigmes de la programació

Hi ha una gran varietat de llenguatges.

- TIOBE index
- Language popularity lists

Com classificar-los?

Diferents *paradigmes o estils de programació*.

# Paradigmes de la programació

Paradigmes més generals:

- Imperatiu o procedural
- Funcional
- Declaratiu

# Paradigmes de la programació

Paradigmes més generals:

- Imperatiu o procedural
- Funcional
- Declaratiu
- Orientat a objectes (?)
- Paral·lelisme (?)

# Paradigmes de la programació

Paradigmes més generals:

- Imperatiu o procedural
- Funcional
- Declaratiu
- Orientat a objectes (?)
- Paral·lelisme (?)

Hi ha molts llenguatges multiparadigma.

# Paradigmes de la programació

## Imperatius

- noció d'estat
- canvi d'estat (efectes laterals).

Ja se n'han vistos uns quants:
C, C++, Ensamblador, ...
Útils quan, per exemple, l'eficiència és clau.

# Paradigmes de la programació

Funcionals

Són llenguatges procedurals però sense noció d'estat.

No hi ha efectes laterals.

- Més fàcil de raonar sobre corectesa.
- Útils per al prototipat, fases inicials de desenvolupament (especificacions executables i transformables).
- Tractament simbòlic.
- Sistemes de tipus potents (important per fiabilitat).

  *polimorfisme paramètric*
  *inferència de tipus*

Haskell, ML (Caml, OCaml), Erlang, XSLT (tractament XML),...

# Paradigmes de la programació

## Lògics

Llenguatges descriptius.

El programa diu que s'ha de fer, però no necessariament com.

- Prototipat d'aplicacions amb forta component simbòlica, problems combinatoris, etc.
- *Queries* en bases de dades relacionals o lògiques.
- Per especificació i raonament automàtic.

SQL (relacional), Prolog (lògica de primer ordre),...

# Paradigmes de la programació

## Orientats a objectes

Es basa en *objectes* (camps + mètodes)

Inclou principalment *polimorfisme (subtipat)* i *herència*

Poden tenir sistemes de tipus complexos.

(vist en cursos anteriors C++ i Java)

# Paradigmes de la programació

## Multiparadigma

Combinen diferents paradigmes

Python, Perl: Orientant a objectes + imperatiu + funcional
Ocaml: funcional + Orientant a objectes + imperatiu

Alguns incorporen caràcteristique d'altres paradigmes

Prolog: logic (+ imperatiu + funcional)

Altres combinacions:

Erlang: funcional, concurrent, distribuït

# Compilat vs Interpretat

Compilats: el codi és transforma en codi objecte i després es monta en un executable.
Exemples: C, C++, Ada, Haskell, ...

Interpretats: el codi s'executa direcatment o el codi es transforma en codi d'una màquina virtual, que l'executa.
Exemples: Python, JavaScript, Prolog (WAM), Java (JVM).

- Aumenten la portabilitat i l'expressabilitat (es poden fer més coses en temps d'execució).
- Disminueix l'eficiència.

Alguns interpretats, poden ser també compilats (per exemple, Prolog) i al revés (Haskell).

*Just in Time compilation*. Es compila (parcialment) en temps d'execució.

# Sistemes de tipus

- Fortament tipats. Haskell, Python, Java, ...
- Dèbilment tipats. BASIC, JavaScript, ...

- Comprovació estàtica de tipus (en compilació). Haskell, C++,...
- Comprovació dinàmica de tipus (en execució). Python, Ruby,...

- *Type safe* (o *Memory safe*). Java.
- *Type unsafe*. C i C++ (pointers), Ruby.