langage de programmation

Philippe Marquet

maison pour la science, 29 mai 2015

ordinateur = machine programmée

- ordinateur = machine universelle
  - toutes les machines informatiques ne sont pas des ordinateurs
- universelle → programmable
  - expliciter à la machine ce qu'elle doit faire
- programmable → programme *machine*
  - instruction machine
- programmeur = humain
  - exprimer un programme dans un langage de programmation

```
LCFI1:
        subq    $16, %rsp
        movl    $0, -4(%rbp)
        jmp     L2
        addl    $1, -4(%rbp)
L2:
        cmpl    $4942, -4(%rbp)
        jle     L2
        call    _getpid
        movq    (%rax), %edx
```

```
forall pixel in image {
    if pixel.luminosité() > 127
        pixel.couleur(noir)
    else
        pixel.couleur(blanc)
}
```

concepts de l’informatique

- algorithme
  - bien avant début de l’informatique
- langage
  - programme exprimé dans un langage
- information
  - numériser = abstraire / représenter objets réels
  - (math = manipuler des notions abstraites)
- machine
  - exécuter un algorithme écrit dans un langage qui traite de l’information

langage(s) informatique(s)

- langages de programmation
- autres langages
  - langages de requêtes
  - langages de description

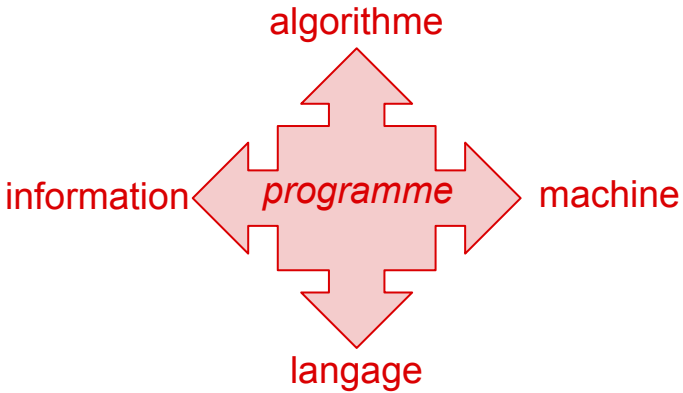
Algol · APL · ASP · Assembleur · BASIC · BCPL · Shell Unix · C · COBOL · Natural · Forth · Fortran · Go · Limbo · Lua · Modula-2 · NQC · NXC · OPL · Pascal · Perl · PHP · Rust · PL/I · Tcl / C++ · C# · CoffeeScript · D · Delphi · Eiffel · Groovy · Java · JavaScript · Lisaac · Logo · Objective-C · PHP · Python · Ruby · Scala · Simula · Smalltalk · Visual Basic / Haskell · Lisp · Common Lisp · ML · OCaml · Gallina · F# · Standard ML · Opa · Scheme · XSLT / Clips · Prolog / Ada · Erlang

- pléthore de langages de programmation, toujours en évolution

langage de programmation

- mais des notions “universelles”, “pérennes”

- apprendre un langage de programmation
  - écrire des programmes
  - central en informatique (les 4 concepts)
  - autonomie, “s’exercer”, créer...



programme

- programme = suite d'instructions
- instruction = mot / construction du langage

- programme est donc un texte
- qui respecte une syntaxe donnée
  - les règles du langage

- sémantique d'un programme
  - exécution des instructions

instruction

- syntaxe d'une instruction
  - "sucre syntaxique"
  - quels éléments
- sémantique d'une instruction
  - effet de l'exécution de l'instruction
- effet sur un environnement, sur un état
  - position / orientation du robot
  - la valeur des variables
  - ...

exemple

l'affectation

- `v := expr`
  `v = expr`
  `v ← expr`
- *variable expression*
- la valeur de la variable prend la valeur de l'expression

langage de programmation

langage de programmation défini

- par la définition de ses instructions
- par la définition d'un environnement

- exemples
  - `(avancer reculer aller-à-droite aller-à-gauche)` {*position robot*}
  - `(avancer pivoter-droite pivoter-gauche)` {*position robot*, *orientation robot*}

programme erroné — bug

- au niveau syntaxique
  - erreur lexicale
    - symbole “`;`” attendu
    - instruction “`faire-demi-tour`” inconnue
  - erreur syntaxique
    - nom de variable attendu
- au niveau sémantique
  - valeur erronée de l’environnement, de l’état
    - robot à une position impossible
    - valeur d’une variable incorrecte
- mais l’exécution peut / va continuer...

état — notion de variable

- ensemble de variables : *nom*, *valeur*
- ensemble des valeurs de chaque variable

- la valeur d'une variable x
  - dépend de l'état
  - donc dépend du temps
  - ne change pas sans qu'une instruction ne soit exécutée

(bien différent des variables x en mathématique)

programme

- suite d’instructions
- exprimées dans un langage de programmation
- dont l’exécution
- permet de passer d’un état initial à l’état final
  - (à un état final)

— ?

— . . .

— !

crédits

more

Exemple instructions

- séquence
- affectation → notion variable
-