Slide 1

Les principes des langages de programmation

Slide 2

Concevoir des algorithmes : une activité ancienne ($<$ -2500)

Transformée au milieu du XX^{e} siècle : ordinateurs

Écrire un texte lu par une machine : langage formel

Slide 3

Langage formel

Sous-ensemble de $\mathcal{L}^*$ dont on connaît une définition mathématique

e.g. $\{a, aa, aaa, aaaa, ...\}$,

ensemble des mots d'un nombre pair de lettres

OD : -1,25 (-0,50)180° OG : -1,00 (-0,25)180°

Slide 4

De nombreux langages formels :

http ://www.enseignement.polytechnique.fr/informatique/INF321/

```
... pour aborder de nouveaux
langages dans la suite de leur cursus à l'École et au delà.
</p>
<p>
Il permet d'accéder au cours
<A HREF="../INF431/">Algorithmes et Programmation : du séquentiel au
distribué</A>. </p>
<HR>
<H2>Les enseignants</H2> ...
```

Slide 5

Parmi lesquels : les langages de programmation

```
class Hypothenuse {
 static double hypothenuse (final double x,
                            final double y) {
  return Math.sqrt(x * x + y * y);}
 public static void main (String [] args) {
  System.out.println(hypothenuse(3,4));}}
```

Une double contrainte

Slide 6

Un programme doit être lisible par une machine (formel)

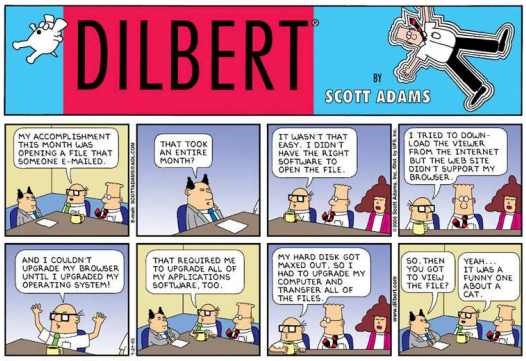
Un programme doit être lisible par un être humain (plus ou moins naturel)

Slide 7

De 6 lignes à 1 000 000 lignes

Un saut dans la complexité des objets industriels (*v.s.* locomotive à vapeur, appareil photo, ...)

Slide 8



Slide 9

Nouveaux problèmes : y a un bug, ...

LANs at JR East, media go on the fritz

Bug in update for Trend Micro antivirus software probably to blame

Kyodo News

Computer local area networks at East Japan Railway Co. and some media organizations were inaccessible Saturday morning, apparently due to a bug in antivirus software made by Trend Micro Inc.

"A bug was detected in a file (designed to detect viruses), and it is highly likely that computer networks that were updated (with the file) encountered a system failure," the software company said.

Trend Micro said it released the file at around 7:30 a.m. Saturday as part of an update for its Virus Buster software.

JR East encountered the trouble with its LAN from around 8 a.m. to noon, but train services and ticketing operations were not affected, company officials said.

View Plaza, a travel agency operated by JR East, said it suffered access failure.

Computer networks at Osaka's municipal subway system were also affected.

Kyodo News experienced LAN access failure from around 8:20 a.m. to shortly before noon, and could not file news stories via regular methods. It provided major news stories to its affiliated news organizations by fax.

The Asahi Shimbun and Yomiuri Shimbun also had trouble with their LANs at their Tokyo and Osaka bureaus, but the problems did not affect editing or printing of their evening editions.

The Nihon Keizai Shimbun's sales bureau and the Shinano Mainichi Shimbun in Nagano Prefecture encountered minor trouble.

En retour, conception d'autres objets complexes : circuits intégrés 500 000 000 transistors

Les langages de programmation

Slide 10

Plus de 2000 (de nouveaux langages avant 2012)

Principes communs : affectation, boucle, cellule, passage par référence

Langues naturelles : article, pronom, verbe, temps, mode, aspect, conjugaison, déclinaison, ...

Slide 11

Comprendre ces principes plus important que connaître les idiotismes de Java ou de Caml

Apprendre un langage ne suffit pas : comparer avec d'autres, pour dégager les principes universels

Slide 12

Objectifs

1. Apprendre à programmer en Java
2. Comprendre les principes (comparaison avec Caml et C)
3. Commencer à apprendre les outils qui permettent de décrire la signification des programmes (plus : année 3)
4. Apprendre les algorithmes de base sur les listes, les arbres et les tableaux (plus : année 2)

Slide 13

Organisation :

10 amphis + 10 séances de TD (projets) + 2 séances de tutorat

Évaluation :

TD 7 (16 juin) noté + pâle HC (6 juillet)

Note finale $=$ max$((t + 2p)/3, p)$

Slide 14

I. Le noyau impératif

Slide 15

Dans (presque) tous les langages, cinq constructions :

la déclaration, l'affectation, la séquence, le test et la boucle

L'affectation

```
x = t;
```

```
x = 3 + y;
```

Slide 16

x variable et t expression

variable : un mot d'une ou plusieurs lettres

expression : formée à partir des constantes et des variables avec des opérations (+, -, *, /, %, ...)

En Caml : `x := 3 + !y`

En C : comme en Java

Slide 17

Ce qu'il se passe quand on exécute `x = t;`

Dans un recoin de la mémoire de l'ordinateur, une case appelée `x`

On met la valeur de `t` dans cette case

La valeur antérieure est oubliée

Expressions : `3`, `5 + 3`, `y + 3`

Valeurs (nombre entier) : `3`, `8`, `10`

État : ensemble des valeurs des variables à un instant donné

Slide 18

Exemples

```
x = 4;
```

```
x = y + 2;
```

Slide 19

La déclaration

Avant de pouvoir utiliser la variable x il faut la déclarer

Associer le nom x à une des cases de la mémoire

```
{int x = t; p}
```

```
{int x = 5; x = 3 + y;}
```

Expression `t` : valeur initiale (optionnelle)

Instruction p : portée de la variable x

En Caml : `let x = ref 5 in p` En C : comme en Java

Slide 20

`int` : nombres entiers compris entre -2^{31} et $2^{31} - 1$

Outre `int` , trois autres intervalles `byte`, `short`, `long`

Autres types : `boolean` (`false`, `true`)

Flottants : `float`, `double` (`6.02E23`)

Caractères `char` (`'g'`)

Huit types primitifs

Types composites (à suivre) : `String` (`"Bonjour"`)

```
{T x = t; p}
```

Slide 21

Les variables finales et les variables mutables

Déclaration d'une variable, engagement à ne jamais l'affecter

```
{final T x = 5; p}
```

```
{final int x = 5; y = x + 3;}
```

```
{final int x = 5; x = 3;}
```
incorrecte

Variable non finale : mutable

En Caml `let x = 5 in p` et non `let x = ref 5 in p`

`y := x + 3` et non `y := !x + 3` quand x est finale

En C, `const` au lieu de `final`

Slide 22

La séquence

```
{p1 p2}
```

```
{x = 1; y = x + 1;}
```

Quand on exécute cette instruction, on exécute p1 puis p2

En Caml : `p1; p2`

En C : comme en Java

Slide 23

Le test

`if (b) p1 else p2`

```
if (x < 0) y = -1; else y = 1;
```

Quand on exécute cette instruction, on calcule la valeur de `b`

Si c'est `true` on exécute `p1` si c'est `false` on exécute `p2`

En Caml : `if b then p1 else p2`

En C : comme en Java

Slide 24

La boucle

`while (b) p`

```
while (x < 1000) x = x * 2;
```

Quand on exécute cette instruction : on calcule la valeur de `b`, si c'est `false` on a terminé, si c'est `true`, on exécute `p`, puis on

Slide 25

La boucle

```
while (b) p
```

```
while (x < 1000) x = x * 2;
```

Quand on exécute cette instruction : on calcule la valeur de b, si c'est `false` on a terminé, si c'est `true`, on exécute p, puis on calcule la valeur de b, si c'est `false` on a terminé, si c'est `true`, on exécute p, puis on calcule la valeur de b, si c'est `false` on a terminé, si c'est `true`, on exécute p, puis on calcule la valeur de b, si c'est `false` on a terminé, si c'est `true`, on exécute p, puis on calcule la valeur de b, si c'est `false` on a terminé, si c'est `true`, on exécute p, puis

Slide 26

`while (b) q` : notation finie pour une instruction infinie

```
if (b) {q if (b) {q if (b) {q if (b) ...
                                    else skip;}
                            else skip;}
                    else skip;}
        else skip;
```

avec une instruction fictive `skip;` qui ne fait rien

En Caml : `while b do p;` En C : comme en Java

Slide 27

```
{int x = 3; while (x <= 1000) x = 2;}
```

ne termine pas

Instruction infinie : potentialité de non terminaison

Slide 28

II. Le instructions d'entrée et sortie en Java

(en un transparent)

```
System.out.print(x);
System.out.println();
System.out.println(x);

Ppl.lireInt ();
```

Slide 29

III. La fonction Σ

Slide 30

Slide 31

Ce qu'il se passe quand on exécute l'instruction `i`

`x = t;` : « On met la valeur de `t` dans la case `x` »

Description en langue naturelle rapidement complexe et confuse (déjà : `while`)

On finit par donner des exemples (... qui couvrent les cas simples)

Comment écrire un interpréteur / compilateur ?

Comment raisonner à propos d'un programme ?

Comment analyser automatiquement un programme ?

Slide 32

Remplacer cette phrase par une définition mathématique

Slide 33

L'état

Ensemble des valeurs des variables à un instant donné

Ensemble infini `Var` dont les éléments sont appelés variables

Ensemble `Val` : $[-2^{31}, 2^{31} - 1] \uplus \{\texttt{false}, \texttt{true}\} \uplus \ldots$

Un état est une fonction d'une partie finie de `Var` dans `Val`

e.g. : `[x = 1, y = 4]`

Attention : pas `1 = x`

Slide 34

Exécuter une instruction `i` transforme l'état

La signification des instructions d'un langage est définie par une fonction Σ

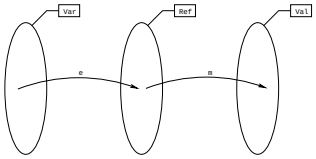
$$\Sigma(i,s) = s'$$

e.g. : $\Sigma(x=3;,[x = 1,y = 4]) = [x = 3,y = 4]$

Slide 35

La décomposition de l'état

Utile pour plus tard : `s = m ∘ e`, l'environnement et la mémoire



Ensemble intermédiaire `Ref` des références

$$\Sigma(i,e,m) = m'$$

Σ`(x=3;,[x=r1,y=r2],[r1=1,r2=4])=[r1=3,r2=4]`

Slide 36

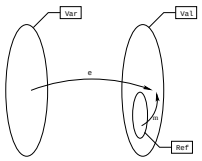
Le cas des variables finales

```
final int x = 3; p
```

au lieu d'associer `x` à `r` dans `e` et `r` à `3` dans `m`

on associe `x` à `3` dans `e` (la mémoire est ce qui peut changer)

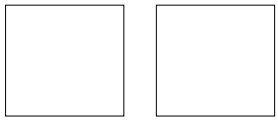
`Ref` $\subseteq$ `Val`



La représentation graphique des états

Slide 37

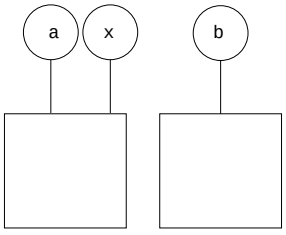
Une référence : une case



La représentation graphique des états

Slide 38

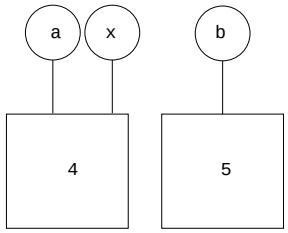
L’environnement



La représentation graphique des états

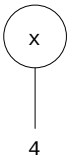
Slide 39

La mémoire



Variable associée directement à une valeur dans l'environnement

Slide 40



La valeur d'une expression

Slide 41

`x = t;`

« On met la valeur de `t` dans la case `x` »

`Θ(t,e,m)`

– `Θ(n,e,m) = n`
– `Θ(x,e,m) = m(e(x))` si x est mutable
  `Θ(x,e,m) = e(x)` si x est finale
– `Θ(t + u,e,m) = Θ(t,e,m) + Θ(u,e,m)`
– idem pour les autres opérations

La valeur d'une expression en Caml et en C

Slide 42

Caml :

– `Θ(x,e,m) = e(x)`

que x soit finale ou mutable

– `Θ( !t,e,m) = m(Θ(t,e,m))`

`x + 1` en Java `!x + 1` en Caml

C : comme Java

Slide 43

L'exécution d'une instruction : la déclaration et l'affectation

La f^n `m + (r = v)` coïncide avec `m` sauf en `r` où elle vaut `v`

– Σ({T x = t ; q},e,m) = Σ(q,e+(x=r),m+(r=v))
  `r` référence quelconque qui n'apparaît pas dans `e` et `m`
  v = Θ(t,e,m)
  Σ({final T x = t ; q},e,m) =
  Σ(q,(e+(x=v)),m)
  v = Θ(t,e,m)
– Σ(x = t ;,e,m) = (m+(e(x)=v))
  v = Θ(t,e,m)

Slide 44

L'exécution d'une instruction : la séquence et le test

– Σ({p_1 p_2},e,m) = Σ(p_2,e,Σ(p_1,e,m))
– Σ(if (b) p_1 else p_2,e,m) = Σ(p_1,e,m)
  si Θ(b,e,m) = true
  Σ(if (b) p_1 else p_2,e,m) = Σ(p_2,e,m)
  si Θ(b,e,m) = false

Slide 45

L'exécution d'une instruction : la boucle

`while (b) q`

Abréviation pour l'instruction infinie

```
if (b) {q if (b) {q if (b) {q if (b) ...
                                    else skip;}
                            else skip;}
                    else skip;}
        else skip;
```

avec Σ(`skip;,e,m`) = `m`

Slide 46

Approximations finies

p_0 = `if (b) giveup; else skip;`

p_1 = `if (b) {q if (b) giveup; else skip;} else skip;`

... p_{n+1} = `if (b) {q` p_n`} else skip;`

Σ jamais définie en (`giveup;,e,m`)

Si Σ(p_n,`e,m`) définie, alors pour tout `n'` $>$ `n`, Σ($p_{n'}$,`e,m`) définie et Σ($p_{n'}$,`e,m`) = Σ(p_n,`e,m`)

Suite Σ(p_n,`e,m`) jamais définie ou définie à partir d'un certain rang et constante sur son domaine : limite

– Σ(`while (b) q,e,m`) = $\lim_n$ Σ(p_n,`e,m`)

Slide 47

Pour résumer

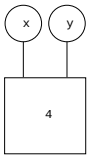
– $\Sigma(\{T\ x = t;\ q\},e,m) = \Sigma(q,e+(x=r),m+(r=v))$
  $\Sigma(\{final\ T\ x = t;\ q\},e,m) = \Sigma(q,e+(x=v),m)$
  r n'apparaît pas dans e et m et $v = \Theta(t,e,m)$
– $\Sigma(x = t;,e,m) = m+(e(x)=v)\ [v = \Theta(t,e,m)]$
– $\Sigma(\{p_1\ p_2\},e,m) = \Sigma(p_2,e,\Sigma(p_1,e,m))$
– $\Sigma(if\ (b)\ p_1\ else\ p_2,e,m) = \Sigma(p_1,e,m)$
  si $\Theta(b,e,m) = true$
  $\Sigma(if\ (b)\ p_1\ else\ p_2,e,m) = \Sigma(p_2,e,m)$
  si $\Theta(b,e,m) = false$
– $\Sigma(while\ (b)\ q,e,m) = \lim_n \Sigma(p_n,e,m)\ p_n = ...$

Slide 48

Exercices

1.5

1.9 : Dans [x = r, y = r], [r = 4]



On exécute x = 5, combien vaut y ?

Et dans [x = r1, y = r2], [r1 = 4, r2 = 4] ?

La prochaine fois : les fonctions