

Langage algorithmique: instructions de contrôle

François Barthélemy

CNAM, Département Informatique

Rappel : structure d'un algorithme

Déclaration des données (`entrées`,
`variables`, `constantes`)

Instructions : `←` (affectation), `écrire`,
`retourner`

Instructions de contrôle

- 4 nouvelles instructions : `si`, `pour`, `tant que`, `répéter`
- utilisent une condition
- contiennent une ou plusieurs suites d'instructions
- déterminent quelles instructions vont s'exécuter
- rupture de l'exécution séquentielle

Instruction si

- structure :

```
si <condition> alors
```

```
    <instruction 1>
```

```
    ...
```

```
    <instruction n>
```

```
sinon
```

```
    <instruction n+1>
```

```
    ...
```

```
↳ <instruction k>
```

- condition vraie \Rightarrow exécution 1 à n
- condition fausse \Rightarrow exécution n+1 à k

Instruction si (suite)

- un `si` est comme un aiguillage à deux branches
- choix de la branche déterminé par la condition
- utilisation du décalage pour préciser les instructions contenues dans le `si`

Exemple de si

```
entrée longueur, unité  
variable longueur_convertie  
constante rapport=0,393 701
```

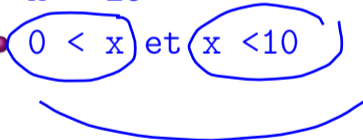
début:

```
si unité="cm" alors  
    longueur_convertie ← longueur / rapport  
sinon  
    longueur_convertie ← longueur * rapport  
retourner(longueur_convertie)
```

conditions

- comparaison égalité : $=$
- comparaison différence : \neq
- comparaisons d'ordre : $<, \leq, >, \geq$
- et logique noté **et**
- ou logique noté **ou**

Exemples de conditions

- $x = 25$
 - $x \neq 25$
 - $x < 25$
 - $0 < x$ et $x < 10$
- 

Boucle tant que

entrée nombre

variable reste, nb_chif

initialisation:

reste = nombre

nb_chif \leftarrow 0

itération:

tant que (reste > 0) faire

reste \leftarrow partie_entière(reste/10)

nb_chif = nb_chif + 1

conclusion:

retourner(nb_chif)

Boucle répéter

```
variable note, somme, nb_notes, encore  
début:
```

```
somme ← 0, nb_notes ← 0
```

```
répéter
```

```
  note ← lecture("Entrez une note")
```

```
  somme ← somme + note
```

```
  nb_notes ← nb_notes+1
```

```
  encore ← lecture("Encore?");
```

```
tant que encore = "oui"
```

```
retourner(somme / nb_notes)
```

Boucle pour

$$2^x$$

entrée x
variable deux_puis

début:

deux_puis \leftarrow 1

pour tout n entre 1 et x faire

 deux_puis \leftarrow deux_puis * 2 \rightarrow n

retourner(deux_puis)

Comparaison des boucles

- boucle répéter : exécutée au moins une fois
- boucle tant que : peut être exécutée 0 fois
- boucle pour : pour parcours d'intervalles entiers