Au menu

Au menu

Définition d'un langage de programmation L

Approche "incrémentale"

1. calculette (L0)

```
12 - 5 * (2+3)
```

2. affectations (L1)

```
X = 2 ;
Y = X + 2 ;
```

3. instructions conditionnelles et entrée-sorties (L2)

```
lire (X) ;
if X>0 then Y = X + 2 else Y = 3 ;
ecrire (Y) ;
```

4. instruction itérative (L3)
5. types, fonctions, etc.

Programmation d'un interpréteur

Rôle de l'interpréteur ?

- détecter les erreurs lexicales et syntaxique dans un programme L
  - arrêt sur la '1ère erreur ?
  - détecter toutes les erreurs ?
  - message d'erreur précis ($\neq$ `Erreur !`)
- détecter les erreurs sémantiques dans un programme L
  - calcul impossible (ex : division par 0)
  - variable non initialisée
  - erreur de type
  - autres ?

  message d'erreur précis . . .($\neq$ `Erreur !`)
- exécuter un programme L
  - mode continu *vs* mode pas-à-pas
  - menu ($\sim$ débugueur ?)
  - sauvegarde / chargement d'exécution en cours ?
  - etc.

Programmation d'un interpréteur

Rôle de l'interpréteur ?

- détecter les erreurs lexicales et syntaxique dans un programme L
  - arrêt sur la '1ère erreur ?
  - détecter toutes les erreurs ?
  - message d'erreur précis ($\neq$ `Erreur !`)
- détecter les erreurs sémantiques dans un programme L
  - calcul impossible (ex : division par 0)
  - variable non initialisée
  - erreur de type
  - autres ?

  message d'erreur précis . . .($\neq$ `Erreur !`)
- exécuter un programme L
  - mode continu *vs* mode pas-à-pas
  - menu ($\sim$ débugueur ?)
  - sauvegarde / chargement d'exécution en cours ?
  - etc.

Programmation d'un interpréteur

Rôle de l'interpréteur ?

- détecter les erreurs lexicales et syntaxique dans un programme L
  - arrêt sur la '1ère erreur ?
  - détecter toutes les erreurs ?
  - message d'erreur précis ($\neq$ `Erreur !`)
- détecter les erreurs sémantiques dans un programme L
  - calcul impossible (ex : division par 0)
  - variable non initialisée
  - erreur de type
  - autres ?

  message d'erreur précis . . .($\neq$ `Erreur !`)
- exécuter un programme L
  - mode continu *vs* mode pas-à-pas
  - menu ($\sim$ débugueur ?)
  - sauvegarde / chargement d'exécution en cours ?
  - etc.

Programmation d'un interpréteur

Rôle de l'interpréteur ?

- détecter les erreurs lexicales et syntaxique dans un programme L
  - arrêt sur la '1ère erreur ?
  - détecter toutes les erreurs ?
  - message d'erreur précis ($\neq$ `Erreur !`)
- détecter les erreurs sémantiques dans un programme L
  - calcul impossible (ex : division par 0)
  - variable non initialisée
  - erreur de type
  - autres ?

  message d'erreur précis . . .($\neq$ `Erreur !`)
- exécuter un programme L
  - mode continu *vs* mode pas-à-pas
  - menu ($\sim$ débugueur ?)
  - sauvegarde / chargement d'exécution en cours ?
  - etc.

Langage L2 : instructions conditionnelles

Exemple

```
X := 12 * 3 ;
Y := X + 5 ;
Z := 42 ;
if (Y > X) then
   Z := X-1 ;
else
   Y := X ;
   if (Y != Z) then
        Z := X + 2 ;
   fi
fi
X := Y * 2 + Z ;
```

Mise en oeuvre

- Nouveaux opérateurs
  - opérateurs de comparaison
    `OPCOMP` $= \{ <, \leq, =, \neq,$ etc. $\}$
  - opérateurs booléens [extension possible]
    `OPBOOL` $= \{$ et, ou , non $\}$
- Nouveaux mots-clés

  `if`, `then`, `else`, `fi`
- nouvelle syntaxe possible pour les instructions

  inst $\rightarrow$ `IF` condition `THEN` seq_inst `ELSE` seq_inst `FI`

  condition $\rightarrow$ eag `OPCOMP` eag
- Interprétation : avec ou sans construction de l'AST complet . . .

Grammaire “générale”

Programme = séquence d'Instructions

$$
\begin{aligned}
\text{pgm} &\rightarrow \text{seq\_inst} \\
\text{seq\_inst} &\rightarrow \text{inst suite\_seq\_inst} \\
\text{suite\_seq\_inst} &\rightarrow \texttt{SEPINST seq\_inst} \\
\text{suite\_seq\_inst} &\rightarrow \varepsilon \\
\text{inst} &\rightarrow \texttt{IDF AFF eag} \\
\text{inst} &\rightarrow \text{autres formes d'instructions } \ldots
\end{aligned}
$$

$\Rightarrow$ réutilisable pour décrire une **séquence de XXX**

Au menu

Instruction itératives

Exemple

```
lire (X) ;
R := 1 ;
while X > 1 do
   R := R * X ;
   X := X - 1 ;
od ;
ecrire (R) ;
```

Quelles modifications par rapport à L2 ?

Analyse

Lexicale : nouveaux mots-clés

`while`, `do`, `od`

Syntaxique : nouvelle instruction

$$\begin{aligned} \text{inst} &\rightarrow \texttt{WHILE}\ \text{condition}\ \texttt{DO}\ \text{seq\_inst}\ \texttt{OD} \\ \text{condition} &\rightarrow \text{eag}\ \texttt{OPCOMP}\ \text{eag} \end{aligned}$$

Interprétation

“sémantique” du while

tant que la condition est vraie
exécuter le corps de la boucle . . .

Mise en oeuvre dans l’interpréteur ?

solution 1 : avec construction **complète** de l’AST . . .
→ itération sur un noeud `N_WHILE`

solution 2 : sans construction complète de l’Ast ?
combiner “analyse syntaxique” et “exécution” ?
pb : lecture de lexèmes
→ avancer dans le fichier (**effet de bord**)
⇒ cette solution 2 est difficilement envisageable . . .

Interprétation

"sémantique" du while

tant que la condition est vraie
exécuter le corps de la boucle ...

Mise en oeuvre dans l'interpréteur ?

solution 1 : avec construction **complète** de l'AST ...
→ itération sur un noeud `N_WHILE`

solution 2 : sans construction complète de l'Ast ?
combiner "analyse syntaxique" et "exécution" ?
pb : lecture de lexèmes
→ avancer dans le fichier (**effet de bord**)
⇒ cette solution 2 est difficilement envisageable ...

Interprétation

"sémantique" du while

tant que la condition est vraie
exécuter le corps de la boucle ...

Mise en oeuvre dans l'interpréteur ?

solution 1 : avec construction **complète** de l'AST ...
→ itération sur un noeud `N_WHILE`

solution 2 : sans construction complète de l'Ast ?
combiner "analyse syntaxique" et "exécution" ?
pb : lecture de lexèmes
→ avancer dans le fichier (**effet de bord**)
⇒ cette solution 2 est difficilement envisageable ...

Au menu

Conseils pour le codage

- suivre une approche **très incrémentale !**
  (coder ; compiler ; tester)*
  $\rightarrow$ avoir toujours une vesrion **qui tourne**

- modules bien specifiés (`.h`, `.c`) ; liste des modules

- Makefile ?

- **tests ! ! !**

Ré-utilisation du langage L dans le projet ?

- analyse lexicale (mots-clés, IDF, etc.)
- analyse syntaxique : grammaire !
- AST + autres structures intermédiaires, interprétation