INF404 - Travaux pratiques - Séance 6

Instruction de lecture/ecriture et arbre abstrait d'un programme

Avant de commencer cette séance :

1. Créez un répertoire ***TP6*** dans votre répertoire ***INF404***
2. Placez-vous dans ***INF404/TP6*** et recopiez les fichiers utilisés pendant le TP5 : `cp ../TP5/* .`
3. aidez-vous des **transparents du cours 7** le cas échéant ...

L'objectif de ce TP est de compléter l'**interpreteur** commencé au TP5 sur les deux points suivants :

— ajout d'instructions d'entrée/sortie au clavier et à l'écran ;
— construction d'un arbre abstrait du programme à interpréter.

La définition du langage (lexique et syntaxe) est libre, on donne ci-dessous un exemple :

```
x = 5 ;
lire (y) ;
y = x + 1 ;
ecrire (y*2) ;
```

Etape 1 - Analyse lexicale

Il s'agit de reconnaitre les nouveaux lexèmes `LIRE` et `ECRIRE` correspondant aux suites de caractères `lire` et `ecrire`. Ces lexèmes peuvent donc être considérés comme des **mot-clès** du langage (il y en aura d'autres !). Pour les distinguer des *identificateurs* (qui sont également des séquences de lettre) vous pouvez utilisez la solution vue en cours :

1. reconnaissance d'un lexeme `IDF` comme une séquence de lettres ;
2. une fois le lexème reconnu, comparez-le avec les mots-clés du langage (`lire`, `ecrire`, etc.) et modifiez sa nature le cas échéant (ce n'est plus un `IDF` dans ce cas ...).

Modifiez le module d'analyse lexicale pour intégrer cette extension . **Testez le résultat** avec le programme **test_lexeme** avant d'aller plus loin ... !

Etape 2 - généraliser la syntaxe des instructions

Notre langage comporte maintenant 3 types d'intructions (affectation, lecture et écriture), et il y en aura d'autres à ajouter ! On doit donc "généraliser" notre grammaire du TP5 en ajoutant les instructions de lecture et écriture :

$$
\begin{array}{rcl}
\text{pgm} & \rightarrow & \text{seq\_inst} \\
\text{seq\_inst} & \rightarrow & \text{inst suite\_seq\_inst} \\
\text{suite\_seq\_inst} & \rightarrow & \texttt{SEPINST seq\_inst} \\
\text{seq\_inst} & \rightarrow & \varepsilon \\
\text{inst} & \rightarrow & \texttt{IDF AFF eag} \\
\text{inst} & \rightarrow & \texttt{LIRE PARO IDF PARF} \\
\text{inst} & \rightarrow & \texttt{ECRIRE PARO eag PARF}
\end{array}
$$

Compléter l'analyse syntaxique en modifiant `rec_inst` pour prendre en compte les deux nouvelles instructions. Les autres fonctions on déjà été écrites au TP5 ...

Vérifier sur des exemples que cette nouvelle analyse syntaxique fonctionne ...

Etape 3 - Construire l'arbre abstrait du programme

Dans la suite il sera nécessaire de mémoriser un forme intermédiaire du programme à l'aide d'un arbre abstrait (Ast).

En vous aidant des transparents du cours 7 :

1. complétez le fichier `type_ast.h` en rajoutant les nature des nouveaux noeuds de l'arbre abstrait ;
2. complétez les fichiers `ast_construction.h` et `ast_construction.c` en rajoutant les nouvelles fonctions de construction de l'arbre abstrait correspondant à ces nouveaux noeuds ;
3. complétez les fichiers `ast_parcours.h` et `ast_parcours.c` en étendant la fonction `afficher()` pour prendre en compte ces nouveaux noeuds ;
4. étendez et modifiez votre analyse syntaxique (comme vu en cours) pour y intégrer la construction de l'Ast ;

Vous pouvez alors vérifier sur des exemples que l'arbre abstrait s'affiche correctement ...

Etape 4 - Interprétez un programme

En vous aidant des transparents du cours 7, complétez les fichiers `ast_parcours.h` et `ast_parcours.c` en y ajoutant les fonctions d'interprétation (`interpreter()`, `interpreter_aff()`, etc.).

Vous pouvez maintenant tester votre interpréteur sur l'exemple donné au début de ce sujet de TP !