UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
Info0602

Travaux pratiques n°3

Présentation de `yacc`

Exercice 1 (calculatrice, en mieux)

Le but de cet exercice est d'écrire une calculatrice. L'utilisateur saisit l'opération désirée composée d'additions et/ou de soustractions d'entiers et la termine par un `'.'`.

1) Récupérez les fichiers fournis en annexes, compilez-les (ignorez les conflits) et vérifiez le comportement du programme (saisissez "`1+2.`" puis `CRTL+D` pour finir).
2) Complétez la grammaire afin que les opérations de multiplication et de division soient possibles. N'oubliez pas de gérer le cas de la division par zéro.

3. Que doit-on changer afin que la calculatrice accepte plusieurs opérations de suite ?

Exercice 2 (évitons les conflits)

En compilant les fichiers de l'exercice précédent, `yacc` nous informe que le programme comporte des conflits décalages/réductions.

1) A votre avis, quelles règles peuvent provoquer ces conflits ? Essayez l'option de compilation *-vd* et analysez le fichier généré (fichier `y.ouput`).

2. Essayez les calculs "`1+2*3`" et "`2*3+1`". Quels sont les résultats ? Pourquoi ? Donnez deux méthodes pour corriger ce problème.

3. Nous désirons ajouter la fonction du moins unaire. Quelles sont les modifications à apporter ? Cela apporte-t-il des conflits ? Testez-la en saisissant les opérations suivantes : "-2+1.", "1-2." et "1–2."

4. Aurait-il été possible de faire cette modification dans `Lex` uniquement ?

Exercice 3 (arbre d'évaluation)

Cet exercice est préparatoire au projet. Si ce n'est pas fait, terminez-le après la séance.

Dans les exercices précédents, les opérations sont calculées au fur-et-à-mesure grâce aux règles sémantiques. Nous souhaitons maintenant construire un arbre syntaxique de l'opération. Il est évalué puis supprimé uniquement lorsque toute l'opération est reconnue.

1) Proposez une structure d'arbre en C pour représenter l'arbre syntaxique d'une opération. Dans la suite, les nœuds pourront être de plusieurs types (variable, valeur ou appel à une fonction). Réfléchissez donc à une structure appropriée.

2. Écrivez les méthodes suivantes (les signatures sont données à titre indicatif) :
   - `void arbre_afficher(arbre_t *arbre)`
   - `void arbre_supprimer(arbre_t **arbre)`
   - `arbre_t *arbre_valeur(int valeur)`
   - `arbre_t *arbre_operateur(int type, arbre_t *g, arbre_t *d)`

3. Modifiez les règles sémantiques pour construire l'arbre et l'afficher.