Université de Limoges — Faculté des Sciences & Techniques

Faculté des Sciences et Techniques de Limoges

Compilation 2
Projet

Nicolas ARAGON, Neals FOURNAISE

2020-2021

Partie 1 : Lex et YACC

Vous utiliserez l'analyseur lexical Lex conjointement avec l'analyseur syntaxique YACC pour créer un programme qui traite du pseudo-code informatique. L'idée est d'utiliser l'outil Lex pour récupérer les mots-clés, lexèmes et autres symboles de votre fichier, contenant du pseudo-code, qui seront transmis à un programme YACC pour en faire l'analyse syntaxique et ainsi déterminer s'il respecte la bonne grammaire.

Soit le pseudo-code suivant :

```
1  varA, varB, N, i, temp : entiers
2  varA <- 0
3  varB <- 1
4  N <- 10
5
6  Pour i de 1 a N faire
7          Afficher(varA)
8          temp <- varB
9          varB <- varA + varB
10         varA <- temp
11 Fin Pour
```

On y aperçoit plusieurs instructions :

— Déclaration de variables (l. 1)
— Affectation de variables (l. 2, 3, 4, 8, 9, 10)
— Un appel à une fonction (l. 7)
— Une addition (l. 9)
— Une boucle (l. 6 à l. 11)

En pseudo-code, nous avons évidemment d'autres mots-clés et d'autres instructions (branchements conditionnels, autres opérations arithmétiques, etc.).

Question 1 :

Écrire une grammaire correspondant à un pseudo-code. Votre pseudo-code inclut au moins la déclaration de variables entières, l'affectation par une constante ou une autre variable, l'addition et la soustraction, une boucle ou un branchement conditionnel.

Voir le cours p. 25 pour des rappels sur les grammaires.

Exemple : On peut imaginer les règles suivantes, non suffisantes :

- **Programme → Declarations Instructions**
- ...
- **Instructions → Instructions Instruction** | / ∗ *vide* ∗ /

- ...
- **Instruction → Affectation | Declaration | Addition | Branchement**
- **Affectation →** ...
- **Branchement →** ...
- ...

où le symbole | représente le « OU » logique.

Question 2 :

Écrire un programme Lex qui récupère les mots-clés, lexèmes et autres symboles de votre pseudo-code et les renvoie à un programme YACC.

Question 3 :

Écrire la grammaire de la question 1 dans un programme YACC qui marche à l'aide de l'analyseur lexical de la question 2. Le programme validera ou non du pseudo-code donné en entrée du programme selon la grammaire.

Question 4 :

Améliorer le programme YACC pour qu'il traduise le pseudo-code en langage C.

Exemple : « var1 : entier » devient `int var1;` ou encore « Si *condition* alors *Instructions* Fin Si » devient `if (condition) instructions;`

Partie 2 : XML

On considère un analyseur de réseau simple, permettant d'obtenir une liste des hôtes présents dans un réseau.
Chaque hôte possède une adresse IP et une liste de ports, qui peut être vide. Un port est identifié par son numéro ainsi qu'un état, qui peut être :

- Ouvert
- Filtré
- Fermé

Question 1 :

Comment stocker l'ensemble de ces informations dans un fichier XML ?
Vous réaliserez un fichier d'exemple.

Question 2 :

Réalisez le fichier DTD permettant de vérifier la validité de votre fichier XML.

Question 3 :

Donnez une méthode pour compter le nombre de ports ouverts ayant un numéro donné dans le rapport d'analyse (par exemple, compter le nombre de ports 80 étant ouverts).

Le fichier XML à analyser sera issu de **nmap**, un outil d'exploration réseau.

-Vous trouverez à l'adresse suivante un extrait à utiliser :
`https://nmap.org/book/output-formats-xml-output.html`
-Un exemple de fichier XSL est disponible pour parser ces résultats :
`https://nmap.org/book/output-formats-output-to-html.html`

Rendu du projet

Vous produirez une archive contenant vos fichiers ainsi qu'un rapport expliquant vos choix. Cette archive est à envoyer en utilisant *Filesender* (**les archives envoyées directement par mail ne seront pas récupérées**) aux adresses suivantes :

— neals.fournaise@unilim.fr

— nicolas.aragon@unilim.fr

La date de rendu limite est fixée au 30/05/2021. **Le projet est à réaliser par groupes de 2 personnes maximum**.

Les documents facilitant la compilation et le test des fichiers fournis (Makefile par exemple) seront valorisés.