

# TP Compilation no 1

Au cours des 6 séances de TP Compilation vous aurez à réaliser, à l'aide de Lex et Yacc, un compilateur du langage VSL+ dont la description informelle vous a été fournie en TD. Le code produit sera une sorte d'assembleur générique appelé *code 3 adresses* que vous découvrirez bientôt en cours.

La première étape (objectif de cette première séance de TP) est la construction d'un analyseur syntaxique de VSL+.

La seconde étape, qui constitue le gros de votre travail et qui occupera 5 séances de TP, consistera pour vous à associer des actions sémantiques à votre analyseur syntaxique pour réaliser les contrôles de type et la génération du code 3 adresses.

Mais, pour commencer, nous vous proposons de vous faire découvrir les différentes étapes du TP complet sur un tout petit sous-ensemble du langage VSL+. Il est donc normal que vous ne soyez pas actuellement en mesure de comprendre certaines parties des fichiers fournis.

L'objectif de cette partie "découverte" très guidée est de vous aider au niveau de la démarche à suivre dans la conception future du compilateur de VSL+

## 1. Découverte du TP Compilation dans son ensemble

Pour cette partie "découverte", nous vous avons créé un répertoire *decouverte* composé de 2 sous-répertoires (*anasynt*, *code3a*), chacun de ces sous-répertoires correspondant à une étape du TP.

### Pour commencer :

recopier le contenu du répertoire fichier */share/mlinfo/tpcomp/decouverte*

### 1.1. Construire un analyseur lexical et un analyseur syntaxique

**Pour cette première étape, se positionner sur le répertoire *anasynt*.** Vous y trouverez les fichiers suivants:

- *parser.y* : fichier source de l'analyseur syntaxique à soumettre à yacc
- *scanner.l* : fichier source de l'analyseur lexical à soumettre à lex
- *main.c* : programme principal qui lance l'analyseur syntaxique généré
- *makefile*

a) Prendre le temps de jeter un oeil sur le contenu des fichiers *parser.y* et *scanner.l* ainsi qu'au contenu des fichiers *test1*, *test2* et *test3* qui contiennent des exemples de chaînes source correctes.

b) Pour générer l'analyseur syntaxique, utilisez la commande **yacc -v parser.y**

L'option *-v*, comme *verbose*, permet d'obtenir l'automate des items LR(0) et la table de prédiction LALR dans le fichier *y.output*.

La grammaire étant LALR, aucun conflit n'est signalé.

c) Editer le fichier *y.output* et prendre le temps de regarder le contenu de ce fichier.

d) Utiliser maintenant la commande **make** qui, en exécutant la suite de commandes figurant dans le fichier *makefile*, va permettre de générer l'analyseur lexical et l'analyseur syntaxique, de lier les deux programmes obtenus avec le programme *main.c* et de générer un exécutable de nom *vcc*.

e) La commande ***./vcc <test1*** permet de lancer l'analyseur syntaxique sur le fichier source *test1*. Vous disposez des trois fichiers source *test1*, *test2* et *test3*. Vous pouvez également utiliser la commande ***./vcc*** et taper ensuite votre chaîne source au clavier (celle-ci doit alors se terminer par ***CTRL+D***).

## 1.2. Construire un compilateur générant du code 3 adresses

### **Pour cette seconde étape, se positionner sur le répertoire code3a.**

Les fichiers *parser.y* et *scanner.l* ont été complétés par des actions sémantiques qui permettent de produire le code 3 adresses associé au programme source analysé. Vous trouverez sous ce répertoire un certain nombre de fichiers de suffixe *.o* (dans le sous-répertoire *obj*, les sources en langage C figurent dans le sous-répertoire *src*) et de fichiers de suffixe *.h* : les premiers correspondent aux exécutables de la mise en oeuvre des types abstraits introduits pour réaliser ce compilateur ; les seconds aux fichiers en-tête correspondants. Le fichier *makefile* a lui aussi été modifié pour réaliser notamment l'édition de liens avec ces fichiers *.o* précédemment cités.

a) Prendre de nouveau le temps de jeter un oeil au contenu des fichiers *scanner.l* et *parser.y*

b) Utiliser la commande **make**. On obtient un exécutable de nom *vcc*.

c) Utiliser la commande ***./vcc -3a*** pour lancer le compilateur sur les fichiers sources *test1*, *test2* et *test3*.

## 1.3. Exécuter les programmes générés sur une machine virtuelle

a) Lancer le compilateur en redirigeant la sortie dans un fichier

```
./vcc <test1 >test1.as
```

c) Produire un code exécutable par une machine d'exécution virtuelle

```
./vas <test1.as >test1.o
```

d) Exécuter le code produit

```
./vam test1.o
```

## 2. Analyseur Syntaxique de VSL+

A partir de la description informelle du langage VSL+ fournie en TD, vous avez dû rédiger une grammaire de ce langage. Il s'agit maintenant d'utiliser le logiciel Yacc pour réaliser un analyseur syntaxique de VSL+ à partir de cette grammaire. Un analyseur lexical vous est fourni sous forme d'un fichier *scanner.l*

**Pour commencer :**

recopier le contenu du répertoire */share/m1info/tpcomp/anasynt*.

Vous disposez alors des fichiers *scanner.l*, *main.c*, *makefile* et d'un sous-répertoire *test-ok* qui contient des programmes source VSL+.

**Il ne vous reste plus qu'à écrire le fichier *parser.y*.**

Les commandes à utiliser sont les mêmes que celle du paragraphe 1.1.

Vous pouvez bien sûr tester votre analyseur syntaxique sur les fichiers du répertoire *test-ok* qui cependant ne couvrent pas toutes les constructions syntaxiques de VSL+.

## 3. Informations pratiques

Cette première partie du TP est à rendre pour le **mercredi 20 octobre** au plus tard.

Vous pouvez retrouver toutes les informations utiles sur le déroulement de l'ensemble des séances de TP Compilation ainsi que la documentation en ligne sous :  
<http://etudiant.ifsic.univ-rennes1.fr/current/m1info/tpcomp/doc/>