

Python

Java

Théorie des langages de programmation

Le projet (I)

Alain Chillès – 祁冲

ParisTech Shanghai Jiao Tong
上海交大-巴黎高科卓越工程师学院

26 octobre 2020 – 2020年10月26日 – 庚子九月初十

Plan

Projet

Description

Analyse lexicale

Bilan de ce qui est demandé

Lisp

Java

C

Prolog

Forth

Ada

Pascal

APL

Lisp

Fortran

Objectif

- Écrire un compilateur pour un langage suffisamment simple permettant d'aborder les thèmes suivants :
 - Analyse lexicale
 - Analyse syntaxique
 - Analyse sémantique
 - Interprétation
 - Compilation en machine virtuelle
 - Compilation en machine réelle
- Le langage proposé est un langage inspiré du langage Forth (nous l'appellerons L_{AC}).

Plan

Projet

Description

Analyse lexicale

Bilan de ce qui est demandé

Python

Java

Prolog

Forth

Ada

Pascal

APL

Fortran

Lisp

Lexèmes

- Les identificateurs (toute chaîne de caractère d'au moins un caractère, composée à l'aide des lettres, chiffres et symboles de ponctuation, sauf le caractère blanc [noté `␣`]).
- Les chaînes de caractères (toute chaîne de caractère commençant par `␣"` (ou `"` en début de ligne), ne contenant pas le caractère `"` à l'intérieur et se terminant par le caractère `"`). On fera attention qu'elle peut être sur plusieurs lignes et donc contenir des sauts de ligne.
- Les commentaires de ligne commençant par les caractères `␣\` (ou `\` en début de ligne) et finissant en fin de ligne.
- Les commentaires multi-lignes commençant par les caractères `␣(` (ou `(` en début de ligne) et se terminant par `)`.

Première étape

- Définir le type **lexème** en C.
- Écrire un analyseur lexical qui à partir d'un fichier contenant un texte en L_{AC} :
 - Supprime les commentaires.
 - Produit une liste de lexèmes.

On produira donc trois fichiers :

- Un fichier **analex.h**
- Un fichier **analex.c**
- Un fichier principal **L_AC.c**
- Tester sur le fichier **factorielle.lac** (fourni).

Le fichier test

1 y union

Java

Fichier factorielle.lac

```
1  \ Fichier "factorielle.lac"
2
3  ( Ce fichier est un "exemple" étudié pour
   ↪ tester
4  l'analyseur lexical écrit en phase 1 du
   ↪ projet)
5
6  : fact ( n -- n!)
7      dup 0=
8      if
9          drop 1
10         else
11             dup 1- recurse *
12         then ;
```

Le fichier test

Fichier factorielle.lac

```
14 : go ( n -- )
15     " Factorielle" count type
16     dup .
17     " vaut :
18     " count type
19     fact . cr ;
20
21 6 go
```


Plan

Projet

Description

Analyse lexicale

Bilan de ce qui est demandé

Lisp

Java

C

Prolog

Forth

C++

Ada

Pascal

Lisp

APL

Fortran

Projet 1 : définition

- Les lexèmes du langage sont :
 - les entiers naturels (**N**)
 - les chaînes de caractères (nom des mots L_{AC} (**M**) et chaînes de caractères en L_{AC} (**S**))
- Ce qui sépare les lexèmes, sauf dans les chaînes de caractères est le caractère blanc \sqcup
- Tous les commentaires **doivent** être supprimés lors de l'analyse lexicale

Il faut produire une liste (ou un tableau) de lexèmes. C'est l'analyseur lexical ! Les fichiers à fournir seront

- Un fichier **analex.h**
- Un fichier **analex.c**
- Un fichier permettant de tester l'analyseur, appelé **projet1.c**

Projet 1 : exemple

Fichier factorielle.lac

```
1  : go ( n -- )
2      " Factorielle" count type
3      dup .
4      " vaut :
5      " count type
6      fact . cr ;
7
```



$M(":") \rightarrow M("go") \rightarrow S("Factorielle") \rightarrow M("count") \rightarrow M("type")$
 $\rightarrow M("dup") \rightarrow M(".") \rightarrow S("vaut : \backslash n") \rightarrow M("count")$
 $\rightarrow M("type") \rightarrow M("fact") \rightarrow M(".") \rightarrow M("cr") \rightarrow M(";")$

Projet 1 : contraintes

- L'analyse lexicale **sera faite** à l'aide d'expressions régulières.
- Les codes **doivent pouvoir être compilés** sur Linux, **sans warning ni error** avec la commande

Terminal

```
$ gcc -Wall projet1.c
```