Compilation
Année 2006-2007

Florent Bouchez
Christoph Lauter

TP n°1
Analyse lexicale

1 Présentation des TPs

Une partie des TDs de compilation seront des TPs pendant lesquels, mais pas seulement, vous allez écrire des morceaux de compilateur en `OCaml`. Vous travaillerez en binômes, et du travail personnel supplémentaire personnel sera probablement nécessaire pour terminer les TPs et rendre à certaines occasions des bouts de compilateurs fonctionnels.

Le premier bout de compilateur à compléter sera un front-end opérationnel pour un langage inclus (mais plus petit) dans le langage Pascal. Aujourd'hui, nous nous intéresserons à la partie concernant l'*a*nalyse lexicale.

2 Mise en place

2.1 Création des groupes

Choisissez un binôme, un nom stupide et rigolo par exemple "blairos" (huit caractères maximum) et créez un groupe du même nom : `http://intranet.ens-lyon.fr`, rubrique "Outils", lien "Groupe Libre service". Votez ensuite à l'unanimité dans votre groupe pour savoir qui hébergera le répertoire de travail. Vous êtes encouragés à utiliser un logiciel de *Source Control Management* si vous savez le faire (cvs, svn, mercurial...). N'oubliez pas de mettre les bonnes permissions à vos répertoires :

```
% mkdir /home/blairo1/groupe_blairos
% chgrp blairos /home/blairo1/groupe_blairos
% chmod 770 /home/blairo1/groupe_blairos
```

2.2 Travail d'aujourd'hui

Récupérez le squelette d'aujourd'hui dans mon placard :

```
% cp -r /home/fbouchez/compilation/TP1_lexer /home/blairo1/groupe_blairos
```

Dans ce répertoire vous trouverez les fichiers suivants :
- `Makefile` : basique pour le projet du jour ;
- `error_handler.mli` : exception à utiliser en cas d'erreur ;
- `parser.mli` : contient pour ce TP uniquement la liste des *token* utilisables ;
- `pretty.ml` : pour afficher facilement les *token* ;
- `main.ml` : initialise le programme et appelle le lexer ;
- `example.pas` : un exemple écrit en Pascal pour tester votre analyseur ;
- `lexer.mll` : l'analyseur lexical proprement dit. Qu'il vous reste à compléter...

Le but de ce premier TP est de compléter le fichier `lexer.mll` qui pour le moment est bien vide. Il est vivement conseillé de garder un brouteur ouvert quelque part qui pointe sur `http://caml.inria.fr/pub/docs/manual-ocaml/manual026.html`, la page de manuel d'`OCaml` qui parle du lexer `ocamllex` (et aussi du parser `ocamlyacc` mais ce n'est pas pour tout de suite).

3 Le langage Pascal

Le langage que vous allez devoir analyser est un sous-ensemble de Pascal, on se basera sur le langage `pascal-`. Ses particularités sont données sur une feuille annexe. Voici un exemple de programme :

```
Program ProgramExample;
const n=100;
type table=array[1..n] of integer;
var A : table;
  i,x : integer;
  yes : Boolean;

procedure search(value : integer; var found : Boolean; var index : integer);
var limit : integer;
begin index :=1; limit :=n;                                                    10
  while index<limit do
    if A[index]=value then limit :=index else index :=index+1;
  found :=A[index]=value
end;

begin {input table} i :=1;
  while i<=n do
  begin
    read(A[i]);
    i :=i+1                                                                    20
  end;
  {test search}
  read(x);
  while x<>0 do
  begin search(x,yes,i);
    write(x);
    if yes then
      write(i);
    read(x);
  end                                                                          30
end.
```

4 Au travail !

Modifiez maintenant le fichier `lexer.mll`. Les endroits indiqués par "..." sont à compléter. Commencez par définir les expressions régulières générales qui forment des chiffres ou des lettres. Puis complétez les règles pour reconnaitre les entiers, les mots réservés du langage (*keywords*), les opérateurs, les commentaires.

5 Améliorations facultatives

Vous pouvez vous amuser à ajouter les fonctionnalités suivantes :

- analyse des chaînes de caractères (`'entre simples quotes'`, et `"entre doubles"`) ;
- indications précises des erreurs :
  `example.pas: l.23 char.12, caractère illégal`
- gestion des commentaires imbriqués