Analyse syntaxique

Rapport de projet

SUTRA Jeremy

SY Thierno

31/12/2023

Introduction

Dans le cadre du projet d'analyse syntaxique en troisième année de licence informatique, nous avons eu à réaliser un analyseur syntaxique avec les langages flex et bisons pour du code en TPC . Pour la grammaire du langage, une grande partie a été fourni par le projet, nous avons mis en place les outils nous permettant de récupérer les informations fournies par le programme flex. Pour la traduction en arbre, les fichiers tree.h et tree.c ont été fournis. Ces derniers permettent de construire un arbre de type "fils gauche - frère droit", nous avons ajouté les éléments pour permettre l'affichage des éléments que nous voulions.

Lexique

Pour la définition des lexèmes, nous nous sommes principalement appuyés sur ce que nous avions fait en TP. Par exemple pour les commentaires sur plusieurs lignes, nous avons juste décomposé l'expression pour permettre de compter les lignes à chaque ***'\n'*** rencontré.

```
%x COM

%%
[/][*] BEGIN COM;

\/\/.* nochar += yyleng;

<COM>\n   {lineno++; nochar = 1;}

<COM>(.|\t|\r) ;

<COM>[*][/] BEGIN INITIAL;
```

Pour l'ajout de lexème concernant les caractères, nous avons fait en sorte que les caractères soient entre guillemet simple, il est aussi possible d'ajouter les caractères spéciaux comme ***'\t'***.

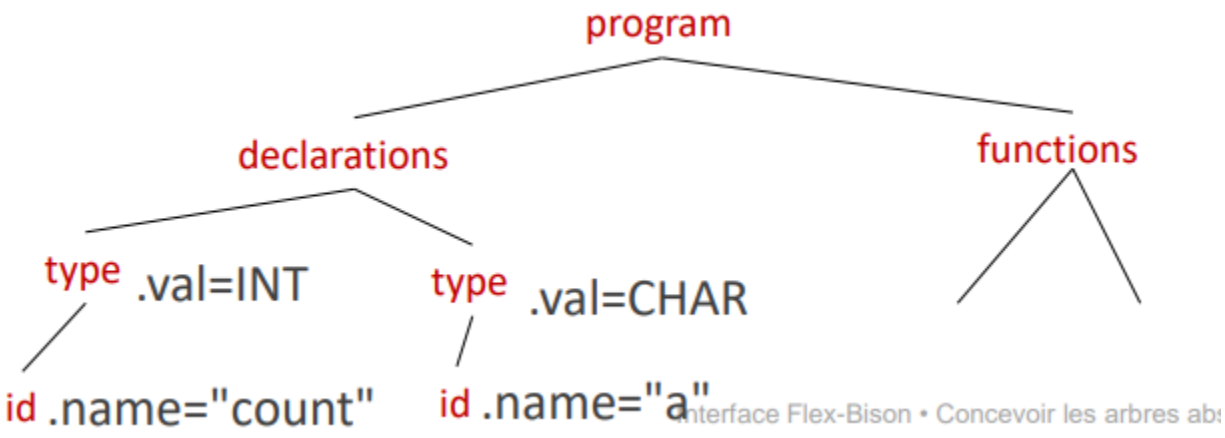
Construction de l'arbre

Pour la construction de l'arbre, les fichiers tree étaient déjà assez complet. Il fallait quand même ajouter les fonctionnalités pour pouvoir récupérer les informations comme les entiers ou encore les caractères qu'on voudrait afficher ensuite dans l'arbre. Pour se faire nous avons dans un premier temps ajouté dans le ***%union*** du bison les éléments qui nous permettent de stocker les informations à partir du flex en utilisant ***yyval.(...)***. Ensuite il a fallu ajouter dans les nœuds de l'arbre la structure nous permettant d'afficher les différents éléments. Pour cela nous avons choisi d'utiliser un union dans la structure du nœud pour pouvoir afficher l'élément qu'on veut après l'avoir enregistré à la création du nœud dans le bison. Nous avons aussi ajouté des éléments dans l'enum ***label_t*** pour faire un switch dessus au moment de l'affichage.

```
union u {
  char ident[64];
  char type[5];
  char order[3];
  char charactere[4];
  char byte;
  int num;
};

typedef struct Node {
  label_t label;
  union u type_elem;
  struct Node *firstChild, *nextSibling;
  int lineno;
} Node;
```

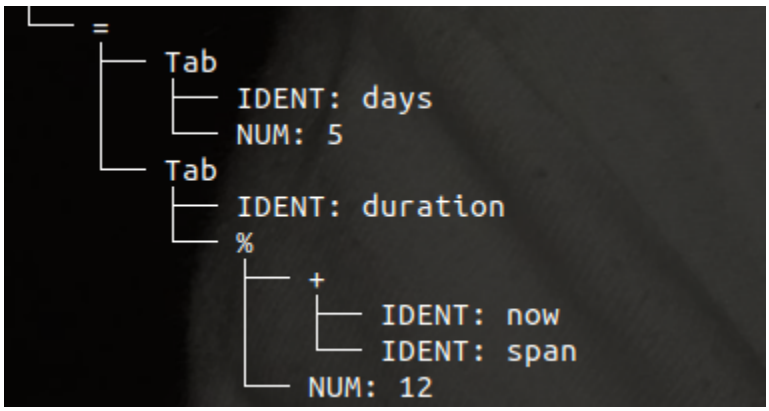
Pour le choix de l'arbre, nous avons choisi la structure du cours suivante:



Ajout des tableaux

Le projet nous imposait d'ajouter les tableaux à une dimension d'entier ou de caractère. Nous n'avons pas vu l'utilité de modifier quoi que ce soit dans le lexème car un tableau est un identifiant suivi de deux crochets. Ce qui change au final c'est ce qui est contenu entre les crochets. Effectivement selon le cas, les crochets peuvent contenir soit une expression, un entier ou alors elles sont vides. Pour chaque cas,on a ajouté une règle car un identifiant peut être suivi de rien du tout (simplement une variable) soit un tableau et dans ce cas, on doit définir ce qui est autorisé ou pas. Pour différencier un identifiant d'un tableau dans l'arbre, nous avons décidé de mettre un nœud qui marque le début du tableau. Le nœud aura comme fils l'identifiant du tableau et l'identifiant a lui-même comme frère ce qui est contenue entre les crochets.

Par exemple l'expression ***days[5]=duration[(now+span)%12];*** se traduit par:



Options et déploiement

Pour l'ajout des options, nous avons fait un fichier à part qui lit les différentes options et lance le programme selon ces dernières. Un fichier ***help.md*** est fourni avec le zip, ce fichier contient le mode d'emploie du programme. L'aide est affichée dans le terminal grâce à une commande cat. Pour le déploiement, nous avons écrit un script bash qui lance les test contenu dans le dossier test et stocke le résultat détaillé dans un fichier ***result.txt***. Un petit rapport contenant le nombre de tests réussi et échoués est affiché dans le terminal. Pour écrire le programme nous nous sommes principalement appuyés sur nos connaissances acquises en L2 durant les cours et TP de système, nous avons aussi fait pas mal de recherches sur internet pour faire certaines choses.

Conclusion

Le projet était très clair et bien expliqué, ce qui nous a permis de nous lancer assez rapidement et de ne pas perdre trop de temps à essayer de comprendre les choses. Les TPs et le cours nous ont beaucoup aidé (surtout sur la résolution de l'ambiguïté). Nous avons donc apprécié travailler sur le projet.