Université Bordeaux 1
Licence, module Compilation, 2016/2017

TD6- Tableaux dynamiques

On se concentre sur une *implémentation des tableaux* qui se conforme à la spécification du projet terminal. On considère un langage réduit TPPascal, sans fonction ni procédure, où les seuls types possibles sont des tableaux de dimension $k \geq 0$ d'entiers.
Une grammaire LALR(1) [au format BISON] et une définition du lexique [au format FLEX] se trouvent dans l'archive `AIDE_td6.tar` : `tableaux.y`, `tableaux.l`.
Des fonctions utilitaires se trouvent dans `arbre.c` et une esquisse d'interpréteur se trouve dans `interp.c`.
Chaque exercice consiste à compléter les fonctions C de ces modules, de façon à réaliser un *analyseur syntaxique*, puis un *interpréteur* de ce langage réduit.

Exercice 6.1

On souhaite que, en fin d' analyse d'un programme, la valeur de la variable globale `syntree` soit l'arbre de syntaxe abstraite et que la valeur de la variable globale `benvty`, soit la table des symboles (utilisable aussi comme environnement global par un interpréteur).
1- Compléter les règles de calcul des attributs (synthétisés) des non-terminaux `Argt`, `L_vartnn` permettant d'atteindre ces deux objectifs.
2- Tester votre analyseur sur les exemples `pexi.t` de l'archive.

Exercice 6.2

On souhaite interpréter les instructions de construction, d'affectation et d'évaluation des tableaux. On définit 3 tableaux statiques : ADR,TAL,TAS qui stockent respectivement l'adresse de début d'un tableau, la taille du tableau et les cases du tableau. Un tableau de dimension 0 est simplement un entier. Un tableau T de dimension $k+1$ est représenté par un entier t :
- un tableau nil est représente par l' entier 0 (et $ADR[0] = 0$)
- si T ne vaut pas nil, pour tout entier $i < TAL[t]$, la variable $T[i]$ est représentée par $TAS[ADR[t] + i]$.

Ainsi :
- on évalue `X[Y]` par `TAS[ADR[X] + Y]`
- on exécute `X[Y]:= Z` par `TAS[ADR[X] + Y]:= Z` (où Z lui-même sera traduit si c'est une expression comportant des crochets).

1- Compléter la fonction `semop(BILENVTY rho_gb,NOE e)` du module `interp.c` de façon à ce qu'elle exécute les arbres de la forme `Wh(e,Ca)`ou de la forme `IfThEl(e,Ca1,Ca2)`.
2- Compléter la fonction `semval(BILENVTY rho_gb,NOE e)` du module `interp.c` de façon à ce qu'elle évalue les arbres de la forme `Ind(X,Y)` (qui représentent les expressions `X[Y]`) et les arbres de la forme `NewAr(e)` (qui représentent les expressions `new array of array ...of integer[e]`).
3- Tester votre interpréteur sur les exemples `pexi.t` de l'archive.