Rapport — premiere partie du projet de compilation

Lucas Pesenti

- Idee generale : OCaml, Ocamllex, Menhir
- Description des fichiers et leur significaion :
  - main.ml : gestion generale et affichage des exceptions
  - ast.ml : arbres de syntaxe abstraites manipules par les autres composants
  - lexer.mll : lexer pour Ocamllex
  - parser.mly : parser pour Menhir
  - typer.ml : typeur
  - une variable de type est representee par un identificateur et un niveau de declaration (gestion des masquages). Ce n'est pas necessaire pour une variable classique, caracterisee par son identificateur (idem pour les identificateurs de fonction)
  - un type = int ou char ou bool ou record ident ou access ident ou null ou unit (utilise uniquement de maniere interne au compilateur, pour gerer de maniere generique fonctions et procedures ("void" peut aussi apparaitre dans les messages d'erreur)
- representation de l'environnement :
  - dec_vars : associe a chaque variable declaree dans l'environnement courant son type
  - dec_typs : associe a chaque variable de type declaree le type qu'elle denote. Si il n'y a pas de declaration anticipee, les structures sont declarees quand elles sont definies. Les types builtin comme int, character ou boolean restent egalement dans l'environnement
  - def_recs : associe a chaque variable de type structure definie une map qui associe a chaque identificateur de champ son type

  - def_funs : associe a chaque fonction definie son type de retour (eventuellement unit pour une procedure), la liste du type et du mode de chacun de ses arguments
  - const_vars : variables constantes. Une expression formee a partir d'elle n'est jamais une lvalue. Concerne les parametres In et les compteurs de boucle for.
  - for_vars : ensemble des compteurs de boucle for. Utilise pour gerer le cas suivant :
  - idents : associe a chaque identificateur un type de derniere declaration (parmi declaration de variable, declaration de variable de type, definition de structure et definition de fonction) et le niveau de cette declaration
  - return_value : type de retour de la fonction courante (si cela a un sens)
  - level : niveau d'imbrication courant
  - nb_incomplete : nombre de declarations incompletes (ie. non suivies d'une definition)

- apports par rapport a Mini-Ada : pas beaucoup, des verifications supplementaires par rapport a Ada (par exemple, il est facile de rajouter le support de pointeurs vers des choses autres que des structures, des declarations anticipes de types access, etc.) mais ils produisent normalement une erreur.

- description de ce qui se passe quand on ajoute une variable, un type, etc.

- remarques sur le masquage

- gestion de character'val, des constantes entieres et de use ...

- cas de return agressif