Pablo Rauzy <pr@up8.edu>
https://pablo.rauzy.name/teaching/ic

Interprétation et compilation

TP 9 : Démarrage projet

Pour compiler le code fourni : `dune build main.exe`.

Un compilateur très basique mais complet vous est fourni. Pour l'instant, les programmes du langage qu'il compile ne peuvent consister qu'en une unique expression qui doit être une valeur entière.

La méthodologie à suivre pour développer votre projet est de rajouter des fonctionnalités à votre langage par couches *complètes* successives. Par complète, on veut dire qui vont d'un bout à l'autre de la compilation, du lexer jusqu'à la production de code MIPS valide.

Typiquement, ajouter une couche de fonctionnalité va consister à :

1. écrire au moins un nouveau fichier de test qui utilise cette fonctionnalité ;
2. éditer le module `Ast` pour décrire dedans la fonctionnalité supplémentaire voulue ;
3. ajouter les éventuelles déclarations de tokens nécessaire dans le module `Parser` ;
4. éditer l'analyseur lexical ;
5. éditer l'analyseur syntaxique ;
6. éditer l'analyseur sémantique (et si besoin la baselib de types) ;
7. éditer le compilateur (et si besoin la baselib de builtins) ;
8. éditer l'émetteur de code MIPS ;
9. tester la fonctionnalité jusqu'à l'exécution du programme MIPS de sortie avec SPIM.

Et bien sûr, vous devez régulièrement tenter de compiler et corriger les erreurs et warnings pendant et entre chacune de ces étapes !

Attendus *minimaux* du langage que vous devez implémenter :

- types de base :
  - entiers,
  - booléens,
  - chaînes de caractères ;
- bibliothèque de base :
  - lecture et écriture sur l'entrée et la sortie standard des types de bases,
  - opérateurs logiques de base sur les booléens (et, ou, non),
  - opérateurs arithmétiques de base sur les entiers (addition, soustraction, multiplication, division, modulo) ;
- expression :
  - valeur,
  - variable,
  - appel de fonction (de la bibliothèque de base ou définie par l'utilisateur·ice) ;
- instructions :
  - déclaration de variable,
  - assignation de la valeur d'une expression à une variable,
  - renvoie de la valeur d'une expression,
  - branchement conditionnel "si *expression* alors *bloc* sinon *bloc*",
  - boucle "tant que *expression* faire *bloc*" ;
- un bloc est une séquence d'instructions ;
- un programme est une liste de définitions de fonctions (dont une s'appelle `main`) :
  - une fonction déclare son nom, les noms et types de ses arguments, son type de retour, et son corps,
  - le corps d'une fonction est un bloc.

Au boulot ! :)