Raport du projet de compilation

CHEMINADE Dorian
SENEL Yasin

6 décembre 2013

0.1 Spécification du langage

Commentaires

_ commentaire sur une ligne : __COMMENT__ <votre commentaire ici>

_ commentaire sur plusieurs lignes : __COMMENT__(<debut de votre long commentaire>
<fin de votre long commentaire>)

Types Simples

_ entier signé sur 1 octet (-128..127) : Integer <name> ;

_ entier signé sur 2 octets (-32768..32767) : BigInteger <name> ;

_ entier non signé sur 1 octet (0..255) : UnsignedInteger <name> ;

_ entier signé sur 2 octets (0..65535) : UnsignedBigInteger <name> ;

_ booléen : Boolean <name> ;

_ caractère (sur 4 octets) : Character <name> ;

_ nombre réel ($1.5 * 10^{-45}$..$3.4 * 10^{38}$) : Real <name> ;

_ énumération non gérée.

Types Complexes

_ intervalles : Non gérés

_ string (de (1..65535) caractères)) : String <name> ;

_ array : Non géré

_ pointer :
déclaration de pointeur : <type> <name> =-> <var>
obtenir l'adresse du pointeur : name
obtenir le contenue du pointeur : name->

Structure de Contrôle

_ switch
SWITCH
CASE (<condition>_1>)
<instructions>
...
CASE (<condition_n>)
<instructions>
END_SWITCH

Boucles

_ while :
WHILE (<condition>) DO
<instructions>
OD

0.2 Choix d'implémentation

0.2.1 Parseur

On utilise un parseur CUP pour parser notre grammaire.

0.2.2 Arbre de syntaxe abstraite

Nous avons utilisé un arbre de syntaxe abstraite pour produire du code 3 adresses. Nous avons choisie cette implémentation car elle reflète bien la structure du code source. On a différent type de noeud qui corresponde tous a un élément dans le programme (exemple : NodeWhile, NodeVariable, NodeArithmetic...). Chaque noeud implémente une interface (Node) qui nous fournit deux méthodes GetTac et GetValue. GetTac permet de générer le code a 3 adresse.

0.2.3 Table des symboles