Compiler C to PDL

Philippe Geraldeli Araujo e Allan Patrick

09-08-2018

1 C to PDL

Fork do miniC feito por eubnara

Este compilador tem o objetivo de converter C para PDL(Propositional Dynamic Logic).

Program := (DeclList)? (FuncList)? // DeclList FuncList ou DeclList ou FuncList
DeclList := (Declaration)+ // Declaration ou DeclList Declaration
FuncList := (Function)+
Declaration := Type IdentList
IdentList := identifier (, identifier)* // identifier ou IdentList , identifier
Identifier := id ou id [intnum] // (Note) [,] are not symbols used in regular expression
Function := Type id ((ParamList)?) CompoundStmt
ParamList := Type identifier (, Type identifier)*
Type := int ou float
CompoundStmt := (DeclList)? StmtList
StmtList := (Stmt)*
Stmt := AssignStmt ou CallStmt ou RetStmt ou WhileStmt ou ForStmt ou IfStmt ou CompoundStmt ou ;
AssignStmt :=Assign
Assign := id = Expr ou id [Expr] = Expr
CallStmt := Call ;
Call := id ((ArgList)?)
RetStmt := return (Expr)? ;
Expr := MINUS Expr | MathRel Eqltop Expr | MathRel | Call | Ids
MathRel := MathEql Relaop MathRel | MathEql
MathEql := TERM Addiop MathEql | TERM
TERM := FACTOR Multop TERM | FACTOR
FACTOR := '(' Expr ')' | FLOATNUM | INTNUM
Id := ID | ID [Expr]

So, Our miniC program doesn't follow the rule below.
1. ++, –
2. According to this rule := CompoundStmt := (DeclList)? StmtList

2 Algorithm Converter

Input = Arquivo em C
Output = Arvore/Arquivo

Algorithm 1: BuildTree(Program* head)

```
1 if headDeclaration != NULL then
2     visitDeclaration(headDeclaration);
3 if headFunction != NULL then
4     visitFunction(headFunction);
```

Algorithm 2: visitDeclaration(DECLARATION* decl)

```
1 if DeclList then
2     if Declaration then
3         insert(declarationType);
4         insert(declarationId);
5     if DeclList then
6         visitDeclaration(previousDeclaration);
```

Algorithm 3: visitFunction(FUNCTION* func)

```
1 if FunctionList then
2     if FunctionList then
3         visitFunction(previousFunction);
4     if Function then
5         insert('(');
6         if funcParameter != NULL then
7             insert(funcParameter);
8         visitCompoundStmt(FunctionCstmt);
```

Algorithm 4: visitCompoundStmt(COMPOUNDSTMT* cstmt)

```
1 if cstmtDeclaration != NULL then
2     visitDeclarationcstmtDeclaration;
3 if cstmtStatement != NULL then
4     visitStmt(cstmtStatement);
```

Algorithm 5: visitStmt(Stmt* stmt)

```
1  switch stmtS do
2      case Assign do
3          InsertSemicolon();
4          insert(stmtS_AssignID);
5          insert("=");
6          insert(stmtS_AssignExpression);
7      case Call do
8          insertSemicolon();
9          insert(stmtSCallIdentifier);
10         insert(CallArg);
11     case Return do
12         if Stmt_Return == NULL then
13             insert("return");
14         else
15             insert("return");
16             visitStmt(stmtS_Return);
17     case While do
18         if StmtSdo_while == true then
19             visitStmt(stmtS_while);
20             insert(WhileCondition);
21             visitStmt(stmtS_while);
22             insert(")*");
23         else
24             insert(WhileCondition);
25             visitStmt(stmtS_while);
26             insert(")*");
27     case For do
28         visitStmtS(StmtSAssign);
29         insert(ForCondition);
30         visitStmt(stmtS_For);
31         visitStmt(stmtS_ForInc);
32         insert(ForCondition);
33     case If do
34         insert(If_condition);
35         VisitStmt(stmtS_if);
36         if stmtSelse != NULL then
37             insert(IfCondition);
38             visitStmt(stmtSelse);
39     case CompoundStmt do
40         visitCompoundStmtstmtS;
41     case Semicolon do
42         insert(";");
```