

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 Eqktop Expr | MathRel | Call | Ids
MathRel := MathEq Relatop MathRel | MathEq
MathEq := TERM Addiop MathEq | 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    else
22      insert(WhileCondition);
23      visitStmt(stmtS_while);
24      insert(WhileCondition);
25  case For do
26    visitStmtS(StmtSAssign);
27    insert(ForCondition);
28    visitStmt(stmtS_For);
29    visitStmt(stmtS_ForInc);
30    insert(ForCondition);
31  case If do
32    insert(If_condition);
33    VisitStmt(stmtS_if);
34    if stmtSelse != NULL then
35      insert(IfCondition);
36      visitStmt(stmtSelse);
37  case CompoundStmt do
38    visitCompoundStmtstmtS;
39  case Semicolon do
40    insert(";");
```
