­­­­Edgar Ruiz

CS 411

February 29, 2016

Project 2

Program -> DeclAdd

DeclAdd -> DeclAdd Decl This was created to handle one or more Decl

Decl

Decl -> VariableDecl

FunctionDecl

VariableDecl -> Variable ;

Variable -> Type id

Type -> int

bool

string

char

FunctionDecl -> Type id ( OptFormals ) StmtBlock

void id ( OptFormals ) StmtBlock

OptFormals -> Formals

**N/A** This was created to handle Formals if wanted or not

Formals -> Variable

Formals , Variable This was created to handle one or more variables

separated by a comma.

StmtBlock -> { VariableDeclAdd StmtAdd }

{ VariableDeclAdd }

{ StmtAdd }

{ } This was created to handle none or as many of both

VariableDeclList and StmtList; VariableDeclList and

StmtList were created to handle more than one of VariableDecl and Stmt.

VarDeclAdd -> VarDeclAdd VariableDecl

VariableDecl This was created to handle multiple VariableDecl

StmtAdd -> StmtAdd Stmt

Stmt This was created to handle multiple Stmt

Stmt -> ; This was created to handle no Expr

Expr;

IfStmt

WhileStmt

ForStmt

ReturnStmt

PrintStmt

StmtBlock

IfStmt -> if ( Expr ) Stmt else Stmt

If ( Expr ) Stmt \*NoElse token\* This was created to handle no else Stmt.

WhileStmt -> while ( Expr ) Stmt

ForStmt -> for ( Expr ; ; ) Stmt This was created to handle zero or up to three occurences

for ( ; Expr ; ) Stmt of Expr.

for ( ; ; Expr ) Stmt

for ( ; ; ) Stmt

for ( Expr ; Expr ; ) Stmt

for ( Expr ; ; Expr ) Stmt

for ( ; Expr ; Expr ) Stmt

for ( Expr ; Expr ; Expr ) Stmt

ReturnStmt -> return ; This was created to handle no Expr.

return Expr ;

PrintStmt -> printf ( stringconstant, ExprList ) ;

printf ( Constant ) ;

printf ( id ) ;

ExprAdd -> ExprAdd , Expr

Expr This was created to handle 1 or more Expr.

Expr -> id = Expr

Constant

Id

Call

( Expr )

Expr + Expr  
Expr – Expr

Expr \* Expr

Expr / Expr

- Expr

Expr < Expr  
Expr <= Expr

Expr > Expr

Expr >= Expr

Expr == Expr

Expr != Expr

Expr % Expr This was created to handle things modulus equations.

Call -> id ( OptExprAdd )

OptExprAdd -> ExprAdd This was created to handle ExprList if wanted or not.

**N/A**

Constant -> intconstant

stringconstant

boolconstant

charconstant

Everything highlighted above are things that were added in order to handle specific situation in the BNF grammar. The whole structure above is the same structure implemented in my parser.y file.

I had one shift/reduce conflict which was the second rule in IfStmt where it handles when there is not an Else present. I created a NOELSE token that won’t be parsed that has a %nonassoc precedence. When I passed in the bison –v parser.y, I checked the parse.output file that was created and confirmed the handling when there is no ELSE token in the second rule of the IfStmt.

