

## Appendix A

---

# *MiniC Grammar*

In this appendix, we give a description and grammar of the source language that we call “MiniC.” **MiniC** is a simple subset of the standard C language. It does not include arrays, structs, unions, files, sets, switch statements, do statements, or many of the low level operators. The only data types permitted are int and float. A complete grammar for MiniC is shown below, and it is similar to the yacc grammar used in the compiler in Appendix B.2. Here we use the convention that symbols beginning with upper-case letters are nonterminals, and all other symbols are terminals (i.e., lexical tokens). As in BNF, we use the vertical bar | to indicate alternate definitions for a nonterminal.

|             |   |                                                                                          |
|-------------|---|------------------------------------------------------------------------------------------|
| Function    | → | Type identifier ( ArgList ) CompoundStmt                                                 |
| ArgList     | → | Arg<br>  ArgList , Arg                                                                   |
| Arg         | → | Type identifier                                                                          |
| Declaration | → | Type IdentList ;                                                                         |
| Type        | → | int<br>  float                                                                           |
| IdentList   | → | identifier , IdentList<br>identifier                                                     |
| Stmt        | → | ForStmt<br>  WhileStmt<br>  Expr ;<br>  IfStmt<br>  CompoundStmt<br>  Declaration<br>  ; |
| ForStmt     | → | for ( Expr ; OptExpr ; OptExpr ) Stmt                                                    |

```

OptExpr      →      Expr
                | ε
WhileStmt    →      while ( Expr ) Stmt

IfStmt       →      if ( Expr ) Stmt ElsePart
ElsePart     →      else Stmt
                | ε
CompoundStmt →      { StmtList }
StmtList     →      StmtList Stmt
                | ε
Expr         →      identifier = Expr
                | Rvalue
Rvalue       →      Rvalue Compare Mag
                | Mag
Compare      →      == | < | > | <= | >= | !=
Mag          →      Mag + Term
                | Mag - Term
                | Term
Term         →      Term * Factor
                | Term / Factor
                | Factor
Factor      →      ( Expr )
                | - Factor
                | + Factor
                | identifier
                | number

```

This grammar is used in Appendix B.2 as the yacc grammar for our MiniC compiler, with very few modifications. It is not unusual for a compiler writer to make changes to the given grammar (which is descriptive of the source language) to obtain an equivalent grammar which is more amenable for parsing.

MiniC is clearly a very limited programming language, yet despite its limitations it can be used to program some useful applications. For example, a MiniC program to compute the cosine function is shown in Figure A.1.

```
int main ()
{float cos, x, n, term, eps, alt;
// compute the cosine of x to within tolerance eps
// use an alternating series
  x = 3.14159;
  eps = 0.1;
  n = 1;
  cos = 1;
  term = 1;
  alt = -1;
  while (term>eps)
  {
    term = term * x * x / n / (n+1);
    cos = cos + alt * term;
    alt = -alt;
    n = n + 2;
  }
}
```

**Figure A.1** A MiniC Program to Compute the Cosine Function