Lecturer (4)

1.3.2 Symbol Table Management

A symbol table is a data structure containing all the identifiers (i.e. names of variables, procedures etc.) of a source program together with all the attributes of each identifier. For variables, typical attributes include:

- its type,
- how much memory it occupies,
- its scope.

For procedures and functions, typical attributes include:
- the number and type of each argument (if any),
- the method of passing each argument, and
- the type of value returned (if any).

The purpose of the symbol table is to provide quick and uniform access to identifier attributes throughout the compilation process. Information is usually put into the symbol table during the lexical analysis and/or syntax analysis phases.

1.3.3 Syntax Analysis

A syntax analyser or parser is a program that groups sequences of tokens from the lexical analysis phase into phrases each with an associated phrase type.

A phrase is a logical unit with respect to the rules of the source language. For example, consider:

$$a := x * y + z$$

After lexical analysis, this statement has the structure

id1 assign id2 binop1 id3 binop2 id4

Now, a syntactic rule of Pascal is that there are objects called ‘expressions’ for which the rules are (essentially):

(1) Any constant or identifier is an expression.

(2) If exp1 and exp2 are expressions then so is exp1 binop exp2.

Taking all the identifiers to be variable names for simplicity, we have:

• By rule (1) $exp_1 = id_2$ and $exp_2 = id_3$ are both phrases with phrase type 'expression';

• by rule (2) $exp_3 = exp_1\ binop_1\ exp_2$ is also a phrase with phrase type 'expression';

• by rule (1) $exp_4 = id_4$ is a phase with type 'expression';

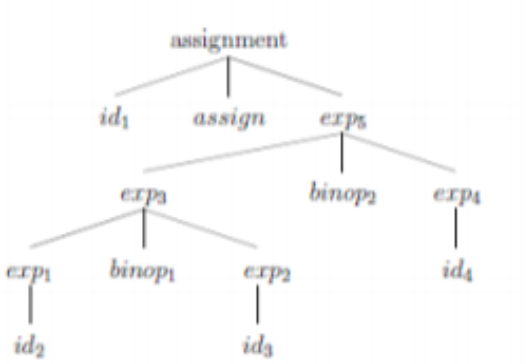
• by rule (2), $exp_5 = exp_3\ binop_2\ exp_4$ is a phrase with phrase type 'expression'.

Of course, Pascal also has a rule that says:

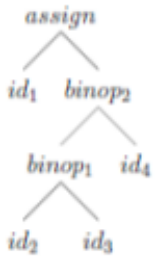
`id assign exp`

is a phrase with phrase type 'assignment', and so the Pascal statement above is a phrase of type 'assignment'.

Parse Trees and Syntax Trees. The structure of a phrase is best thought of as a parse tree or a syntax tree. A parse tree is tree that illustrates the grouping of tokens into phrases. A syntax tree is a compacted form of parse tree in which the operators appear as the interior nodes. The construction of a parse tree is a basic activity in compiler-writing. A parse tree for the example Pascal statement is:



and a syntax tree is:



Comment. The distinction between lexical and syntactical analysis sometimes seems arbitrary.

The main criterion is whether the analyser needs recursion or not:

- lexical analysers hardly ever use recursion; they are sometimes called linear nalysers since they scan the input in a 'straight line' (from from left to right).
- syntax analysers almost always use recursion; this is because phrase types are often defined in terms of themselves (cf. the phrase type 'expression' above).