```
a sub {i sup 2}
```

results in a_i^2. Grouping the operators **sub** and **sup** into tokens is part of the lexical analysis of EQN text. However, the syntactic structure of the text is needed to determine the size and placement of a box.

1.3 THE PHASES OF A COMPILER

Conceptually, a compiler operates in *phases*, each of which transforms the source program from one representation to another. A typical decomposition of a compiler is shown in Fig. 1.9. In practice, some of the phases may be grouped together, as mentioned in Section 1.5, and the intermediate representations between the grouped phases need not be explicitly constructed.

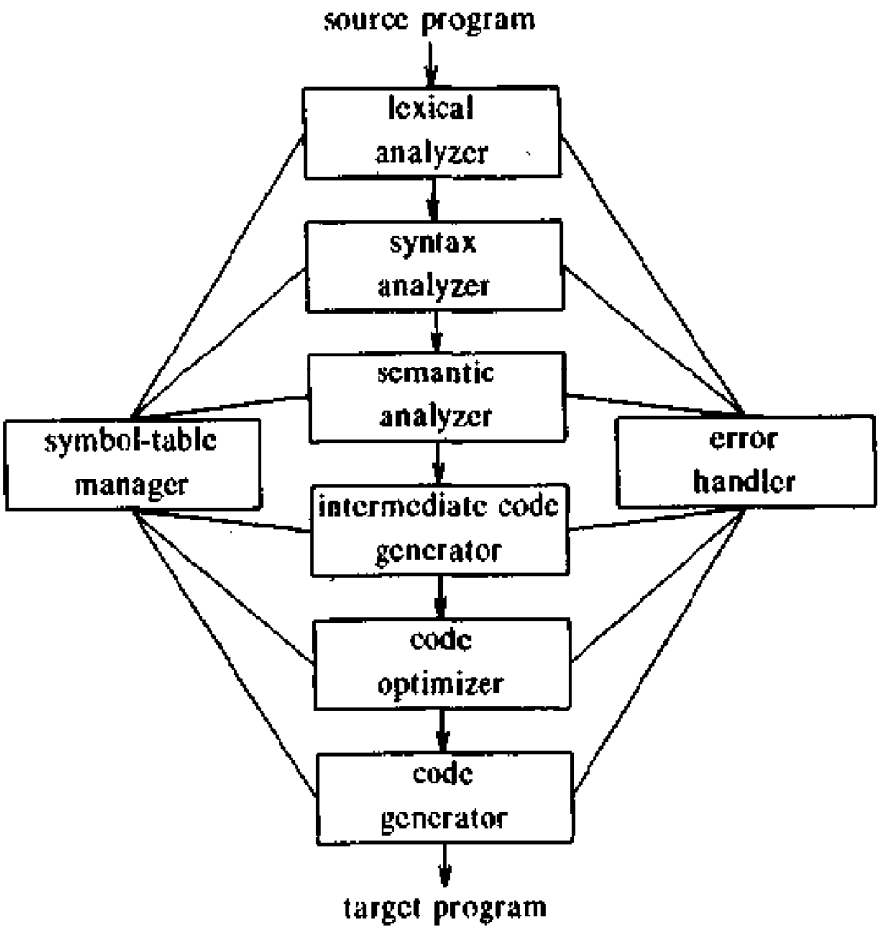

Fig. 1.9. Phases of a compiler.

The first three phases, forming the bulk of the analysis portion of a compiler, were introduced in the last section. Two other activities, symbol-table management and error handling, are shown interacting with the six phases of lexical analysis, syntax analysis, semantic analysis, intermediate code generation, code optimization, and code generation. Informally, we shall also call the symbol-table manager and the error handler "phases."