1. . Each language has its own grammar and it has undefined symbols like integer, identifier, string.

The grammar has undefined terminal symbols integer-constant, character-constant, floating- constant, identifier, string, and enumeration-constant; the typewriter style words and symbols are terminals given literally. This grammar can be transformed mechanically into input acceptable for an automatic parser-generator. Besides adding whatever syntactic marking is used to indicate alternatives in productions, it is necessary to expand the "one of" constructions, and (depending on the rules of the parser-generator) to duplicate each production with an opt symbol, once with the symbol and once without. With one further change, namely deleting the production typedef-name: identifier and making typedef-name a terminal symbol, this grammar is acceptable to the YACC parser-generator. It has only one conflict, generated by the if-else ambiguity.

A function description has this form:

The decimal numbers assigned to the individual propositions indicate the logical importance of the propositions, the stress laid on them in my exposition. The propositions n.1, n.2, n.3, etc. are comments on proposition no. n ; the propositions n.m 1, n.m 2, etc. are com- ments on proposition no. n.m ; and so on.

```
void x( );
```

2. Index.

TEST