Lexical Analysis

The role of a lexical analyser

- The first phase of a compiler
- Reads input characters
- Groups them into lexemes
- Produces as output a sequence of tokens
- The stream of tokens is sent to the parser
- When it discovers the lexeme for a new identifier it enters this lexeme into the symbol table

0.1 Lexical Analysis vs Parsing

Lexical analysis and syntax analysis are separate phases

- Simplicity of design
  - Simplify each task separately
- Improved compiler efficiency
  - Apply specialised techniques for each step
- Compiler portability
  - Different lexical analysers for specific devices

1 Tokens vs Lexemes

Token

- A notation representing the kind of lexical unit, e.g.
  - A keyword
  - An identifier
- Consists of a pair of:
  - A token name
  - An (optional) attribute value
- The tokens are given as input to the parser

Pattern of a token

- A description of the form of its lexemes, e.g.
  - For a keyword, the sequence of its characters
  - For an identifier: a description that matches many strings

The lexeme of a token

- A sequence of characters in the source program that matches the pattern of the token
- The lexical analyser identifies a lexeme as an instance of a token

2 Attributes of tokens

In cases when many lexemes match with a specific token

- The compiler must know which lexeme was found in the source program
- The lexical analyser provides to the parser
  - The token name
  - Additional information that specifies the particular lexeme represented by the token (attribute value)

Token names - influence parsing decisions

Attribute names - influence the transition of the token after the parsing (in the semantic analysis)