2. Lexical Analysis

Andrea Polini, Luca Tesei

Compilers
MSc in Computer Science
University of Camerino

ToC

Lexical Analysis

```
if (i==j)
  z=0;
else
  z=1;
```

```
\tif (i==j)\n\t\tz=0;\n\telse\n\t\tz=1;
```

Lexical Analysis

```
if (i==j)
  z=0;
else
  z=1;
```

```
\tif (i==j)\n\t\tz=0;\n\telse\n\t\tz=1;
```

Token, Pattern Lexeme

Token

A token is a pair consisting of a token name and an optional attribute value. The token names are the input symbols that the parser processes.

Pattern

A pattern is a description of the form that the lexemes of a token may take. In the case of a keyword as a token, the pattern is just the sequence of characters that form the keyword.

Lexeme

A lexeme is a sequence of characters in the source program that matches the pattern for a token and is identified by the lexical analyzer as an instance of that token.

Lexical Analysis

- Token Class (or Class)
  - In English: *Noun, Verb, Adjective, Adverb, Article, . . .*
  - In a programming language: *Identifier, Keywords, "(", ")", Numbers, . . .*

Lexical Analysis

- Token classes corresponds to sets of strings

- Identifier
  - strings of letter or digits starting with a letter
- Integer
  - a non-empty string of digits
- Keyword
  - “else”, “if”, “while”, . . .
- Whitespace
  - a non-empty sequence of blanks, newlines, and tabs

Lexical Analysis

- Token classes corresponds to sets of strings
- Identifier
  - strings of letter or digits starting with a letter
- Integer
  - a non-empty string of digits
- Keyword
  - “else”, “if”, “while”, . . .
- Whitespace
  - a non-empty sequence of blanks, newlines, and tabs

Lexical Analysis

- Token classes corresponds to sets of strings

- Identifier
  - strings of letter or digits starting with a letter
- Integer
  - a non-empty string of digits
- Keyword
  - “else”, “if”, “while”, . . .
- Whitespace
  - a non-empty sequence of blanks, newlines, and tabs

Lexical Analysis

- Token classes corresponds to sets of strings
- Identifier
  - strings of letter or digits starting with a letter
- Integer
  - a non-empty string of digits
- Keyword
  - “else”, “if”, “while”, . . .
- Whitespace
  - a non-empty sequence of blanks, newlines, and tabs

Lexical Analysis

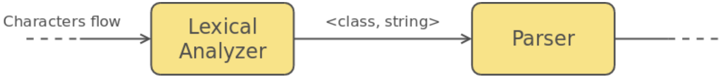
- Token classes corresponds to sets of strings

- Identifier
  - strings of letter or digits starting with a letter
- Integer
  - a non-empty string of digits
- Keyword
  - “else”, “if”, “while”, . . .
- Whitespace
  - a non-empty sequence of blanks, newlines, and tabs

Lexical Analysis

Therefore the role of the lexical analyser (Lexer) is:

- Classify program substring according to role (token class)
- communicate tokens to parser

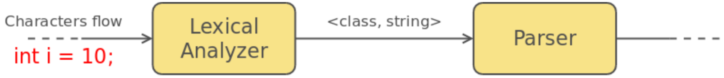


Why is not wise to merge the two components?

Lexical Analysis

Therefore the role of the lexical analyser (Lexer) is:

- Classify program substring according to role (token class)
- communicate tokens to parser

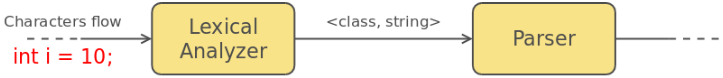


Why is not wise to merge the two components?

Lexical Analysis

Therefore the role of the lexical analyser (Lexer) is:

- Classify program substring according to role (token class)
- communicate tokens to parser



Why is not wise to merge the two components?

Lexical Analysis

Let's analyse these lines of code:

```
\tif (i==j)\n\t\tz=0;\n\telse\n\t\tz=1;
```

```
x=0;\n\twhile (x<10) {\n\tx++;\n}
```

Token Classes: Identifier, Integer, Keyword, Whitespace