CS480 Translators

Introduction to Lexical Analysis
Chap. 2
Quiz #2

Question 1
• \( S \rightarrow (S) \ S \mid \varepsilon \)

Question 3
\[-1 + 2 \times 3.0^{4.7} / 6\]

\[-1 2 3.0e 4.7e f** s>f fswap f* 6 s>f f/ s>f fswap f+\]
Lexical Analysis

• What is its purpose?
• What is the difference in a token vs. lexeme?
• Example:
  – The Brown Fox
  – if (i !=32) then j := 12
• Are spaces important?
The Role of the Scanner...

Figure 3.1: Interactions between the lexical analyzer and the parser
while (i > 0) 
  i = i - 2; 

Mini-Translator

T_WHILE
T_LPAREN
T_IDENTIFIER
T_LESS_THAN
T_INTCONSTANT
T_RPAREN
T_IDENTIFIER
T_EQUALS
T_MINUS
T_INTCONSTANT
T_SEMICOLON
What’s new in this grammar?

```
expr   →   expr + term   { print('+') }  
        |   expr - term   { print('-') }  
        |   term

term   →   term * factor  { print('*') } 
        |   term / factor  { print('/') }  
        |   factor

factor →   ( expr )  
        |   num        { print(num.value) }  
        |   id         { print(id.lexeme) }
```

Figure 2.28: Actions for translating into postfix notation
The Scanner

```java
for ( ; ; peek = next input character ) {
    if ( peek is a blank or a tab ) do nothing;
    else if ( peek is a newline ) line = line+1;
    else break;
}
```

Figure 2.29: Skipping white space

- What is the purpose of line?
- What is the purpose of peek?
Reading Ahead

• Read the next char, it is an “i”
• Could be int, if, or an identifier, so read next char, “f”
• Could be if, could still be an identifier, so read next char, “(”
• Oops, we’ve gone too far, push back “(”
Buffers

• Why is this important?
• Ways to implement:
  – Two pointers into buffer (start_char, look_ahead)
  – Push back buffer (peek)
The Lexical Analyzer

\[
\textbf{if} \ ( \text{peek holds a digit} ) \ \{ \\
\quad v = 0; \\
\quad \textbf{do} \ { \\
\quad \quad v = v \times 10 + \text{integer value of digit \textit{peek};} \\
\quad \quad \text{peek} = \text{next input character;} \\
\quad \} \ \textbf{while} \ ( \text{peek holds a digit} ); \\
\textbf{return} \ \text{token} \ \langle \text{num, v} \rangle; \\
\}
\]

Figure 2.30: Grouping digits into integers
Keywords vs. Identifiers

- \[ \text{count} = \text{count} + \text{increment}; \]

- \(<\text{id}, \text{“count”}> <=> <\text{id}, \text{“count”}> <+> <\text{id}, \text{“increment”}> <;>\]

- How do we know count is an id vs. keyword?
- Why use a hash table?
- What is in the hash table?
How to distinguish words?

```java
if ( peek holds a letter ) {
    collect letters or digits into a buffer b;
    s = string formed from the characters in b;
    w = token returned by words.get(s);
    if ( w is not null ) return w;
    else {
        Enter the key-value pair (s, ⟨id, s⟩) into words
        return token ⟨id, s⟩;
    }
}
```

Figure 2.31: Distinguishing keywords from identifiers
Reading/Assignment

• Milestone 2
• Read Chap. 2.6 - 2.7 and Chap. 3