

Lexical Analysis and Parsing

Lexical Analysis

When you read, one thing your brain does is identify the words in each sentence. The software term for this task is *lexical analysis*.

The job of a *lexel analyzer*, or *lexer* for short is to extract the *tokens* from a string of characters and identify the type of each token.

For an English sentence, the token types include “subject”, “verb”, “object”.

Jill eats food

For a computer program, the token types include “Identifier”, “Integer”, “AssignOp”, etc.

Lexical Analysis: Sample

Given the code:

```
x18      = 4 y24 =  
3+9
```

What do you think a Lexer would output for this code?

Lexical Analysis

What is Java's definition for identifier? What must one consist of? What about in our "Simple" Language?

What does an integer consist of?

Lexical Analysis: Sample


Given the code:

x18 = 4 y24 =
3+9

Lexer Output:

| <u>value</u> | <u>type</u> |
|--------------|-------------|
| x18 | Identifier |
| = | AssignOp |
| 4 | Integer |
| y24 | Identifier |
| = | AssignOp |
| 3 | Integer |
| + | PlusOp |
| 9 | Integer |

Each row is a Token,
with a type and a value



Lexical Analysis: Sample

Given the code:

4x+33y

Lexer Output:

| <u>value</u> | <u>type</u> |
|--------------|-------------|
| 4 | Integer |
| x | Identifier |
| + | PlusOp |
| 33 | Integer |
| y | Identifier |

In determining when a token ends, don't stop when you reach a space or end-line, stop when you reach a char not part of the current token.

e.g. when you see the first 3, you know you have a digit. Loop until you get a non-digit.

Note: it is not the Lexer's job to decide if things are in the right order
(Parser does that)

Parsing

Parsing is the process of analyzing a sequence of tokens to determine if they conform to a valid expression in a particular language.

“Bo likes candy” is a valid sentence in English.

“likes Bo candy” is not.

Parser uses the Lexer

Given the code:


```
x18      = 4 y24 =  
3+9
```

Lexer Output:

| <u>value</u> | <u>type</u> |
|--------------|-------------|
| x18 | Identifier |
| = | AssignOp |
| 4 | Integer |
| y24 | Identifier |
| = | AssignOp |
| 3 | Integer |
| + | PlusOp |
| 9 | Integer |

The Parser doesn't look at the program text directly.

Instead, it loops through the tokens that the Lexer has produced to see if the program is valid.



Parser

If the language consists only of assignment statements, the Parser will loop through tokens and ask:

- Is the first thing an identifier?
 parseId function
- Is the second thing an AssignOp?
 parseAssignOp function
- Is the third thing an *expression*?
 parseExpr function.

ParseExpression

What does an expression, the right-hand side of an assignment statement, consist of?

Symbol Table

The symbol table maps identifiers to memory addresses. The first identifier will be at address 0, the second at address 1, and so on.

Given the code:

x18 = 4 y24 =
3+9

Symbol Table

Lexer Output:

| <u>value</u> | <u>type</u> |
|--------------|-------------|
| x18 | Identifier |
| = | AssignOp |
| 4 | Integer |
| y24 | Identifier |
| = | AssignOp |
| 3 | Integer |
| + | PlusOp |
| 9 | Integer |

| <u>Identifier</u> | <u>Address</u> |
|-------------------|----------------|
| x18 | 0 |
| y24 | 1 |