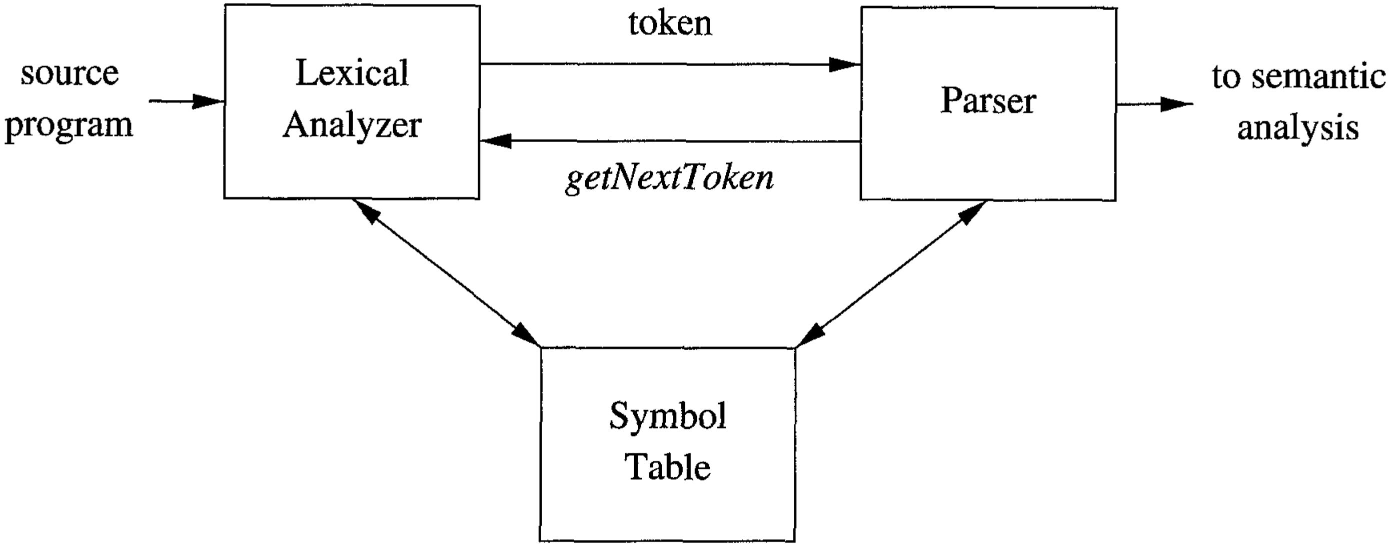
Compiler Construction

(Week 2, Lecture 1)

Interaction



Lexical Analysis

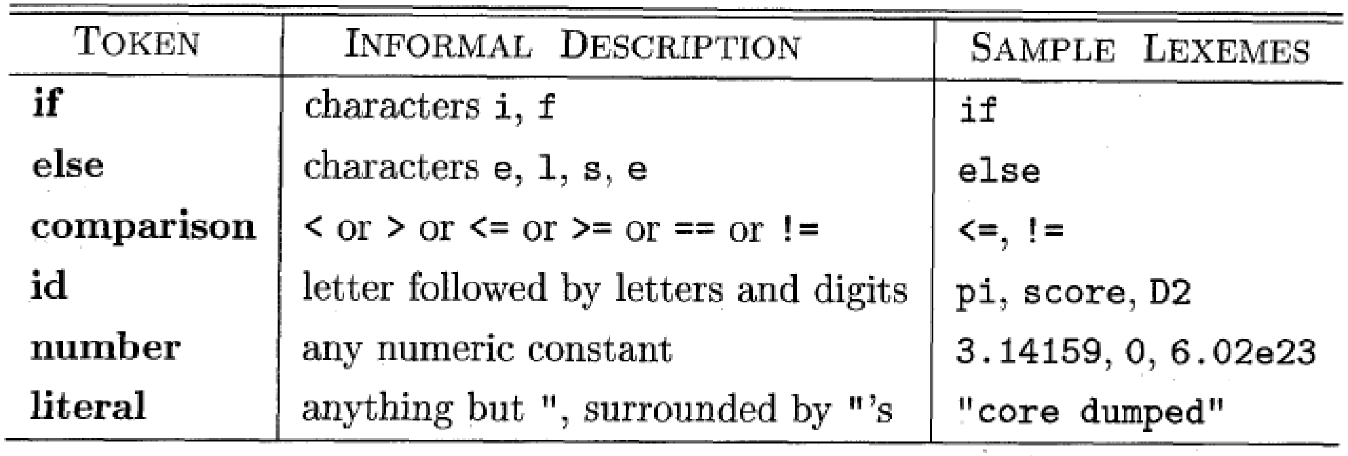
- Strips out comments and whitespace
  - Blank, newline, tab etc.
- Correlates error messages.
  - May keep track of the number of newline characters seen, so it can associate a line number with each error message.
- Does the expansion of macros.
- Produces the sequence of tokens as output.

Lexical Analysis

- Token
  - Pair consisting of a token name and an optional attribute value.
- Pattern
  - Description of the form that the lexemes of a token may take.
- Lexeme
  - Sequence of characters in the source program that matches.

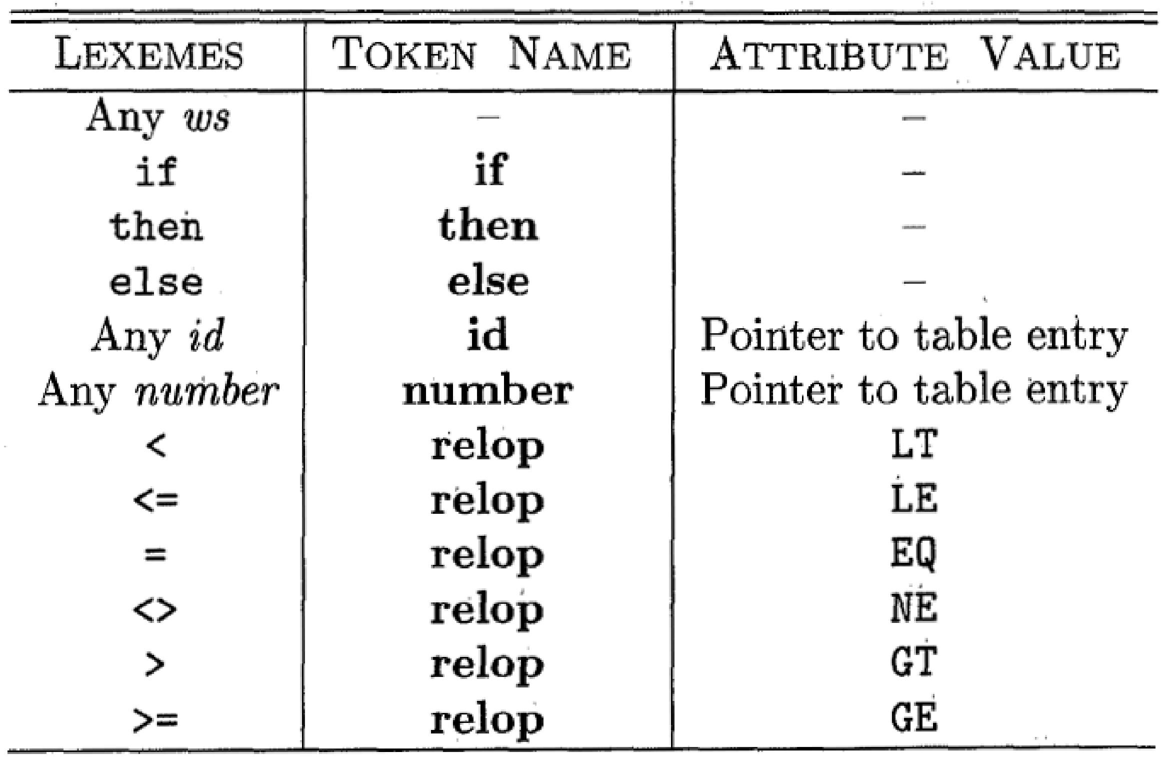
Lexical Analysis

- Examples:
  - printf ("Total = %d\n”, score);
    - both printf and score are lexemes matching the pattern for token id.
    - "Total = %d\n”” is a lexeme matching literal.



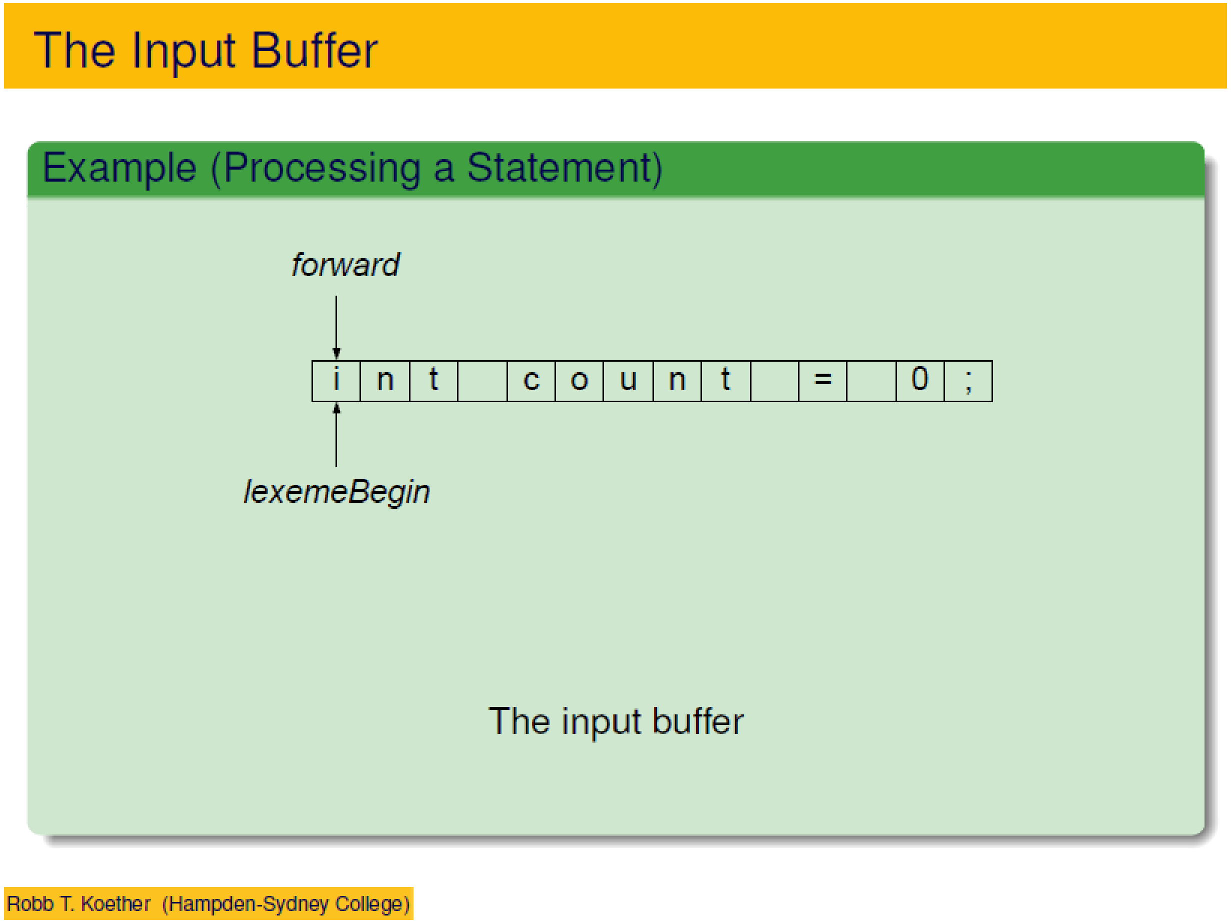
Token	Informal Description	Sample Lexemes
if	characters `i`, `f`	`if`
else	characters `e`, `l`, `s`, `e`	`else`
comparison	`<` or `>` or `<=` or `>=` or `==` or `!=`	`<=`, `!=`
id	letter followed by letters and digits	`pi`, `score`, `D2`
number	any numeric constant	`3.14159`, `0`, `6.02e23`
literal	anything but `"`, surrounded by `"`'s	`"core dumped"`

Lexical Analysis



LEXEMES	TOKEN NAME	ATTRIBUTE VALUE
Any *ws*	–	–
`if`	**if**	–
`then`	**then**	–
`else`	**else**	–
Any *id*	**id**	Pointer to table entry
Any *number*	**number**	Pointer to table entry
`<`	**relop**	`LT`
`<=`	**relop**	`LE`
`=`	**relop**	`EQ`
`<>`	**relop**	`NE`
`>`	**relop**	`GT`
`>=`	**relop**	`GE`

Lexical Analysis



The Input Buffer

Example (Processing a Statement)

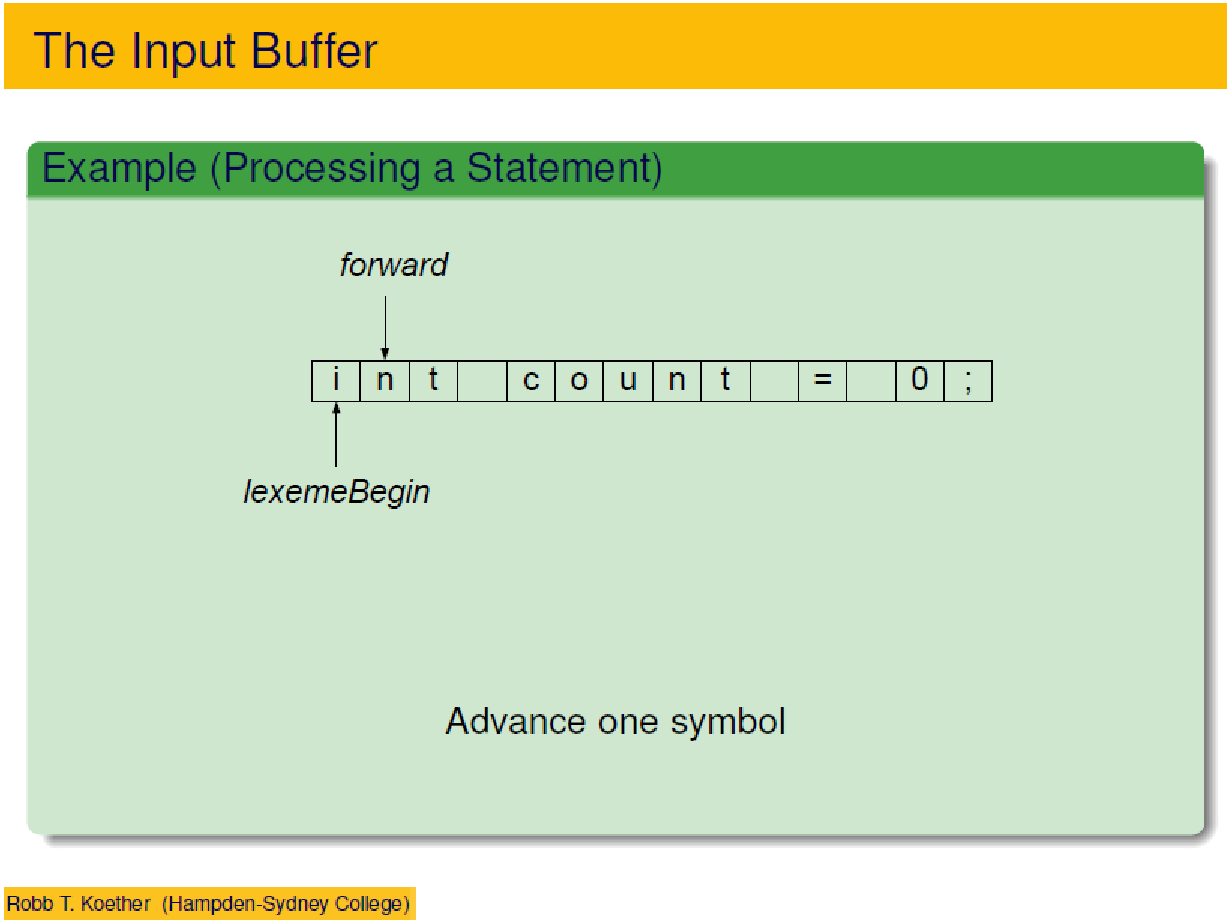
forward

i	n	t		c	o	u	n	t		=		0	;

lexemeBegin

The input buffer

Lexical Analysis



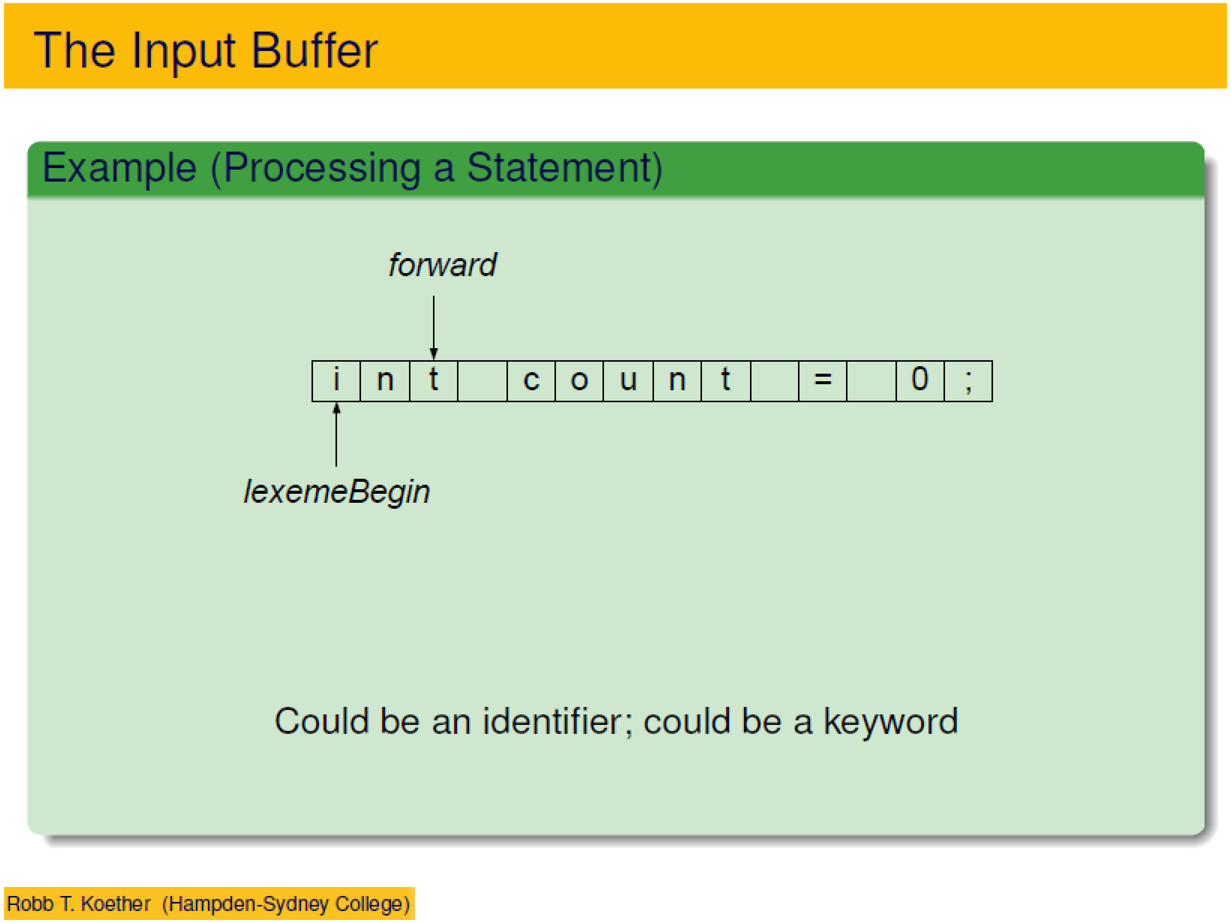
The Input Buffer

Example (Processing a Statement)

i	n	t		c	o	u	n	t		=		0	;

Advance one symbol

Lexical Analysis



The Input Buffer

Example (Processing a Statement)

forward

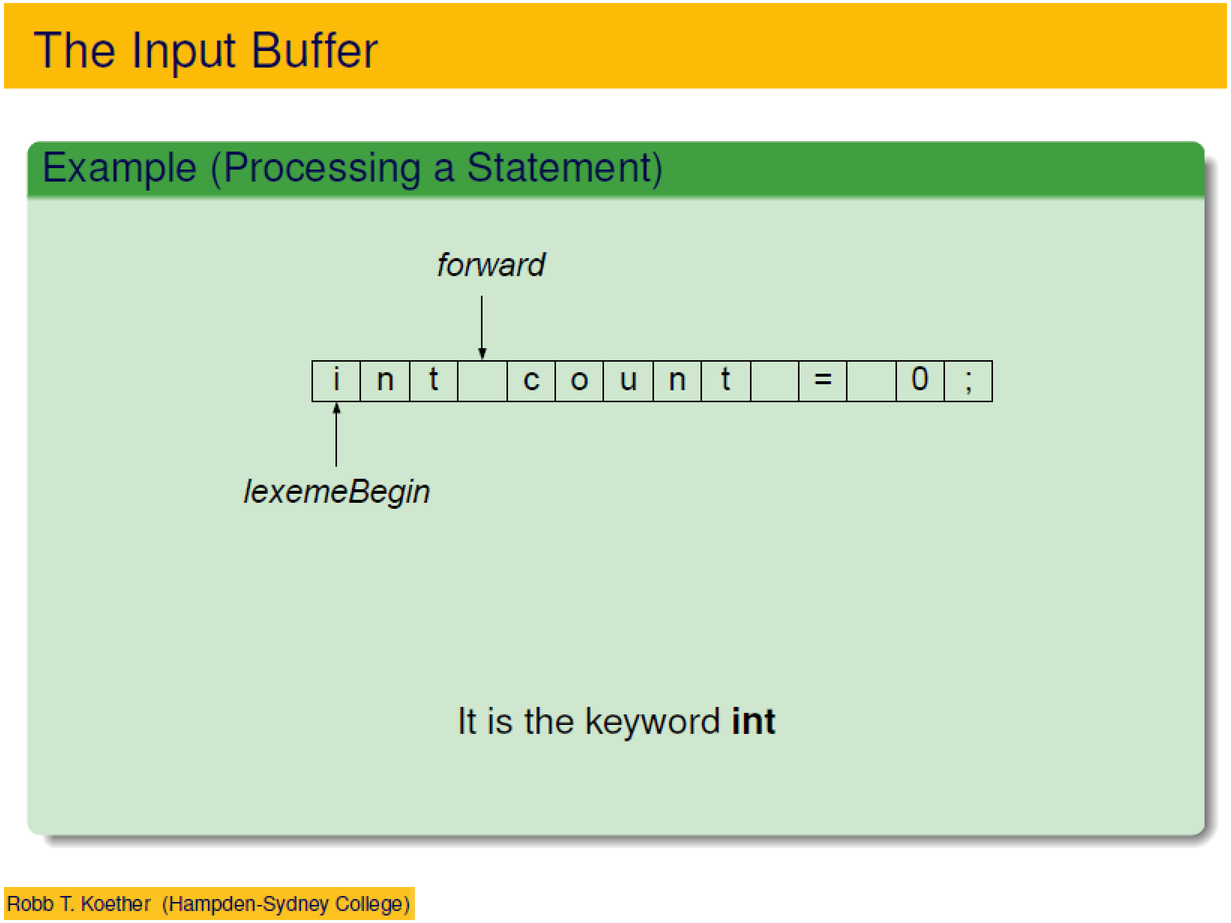
i	n	t		c	o	u	n	t		=		0	;

lexemeBegin

Could be an identifier; could be a keyword

Robb T. Koether (Hampden-Sydney College)

Lexical Analysis



The Input Buffer

Example (Processing a Statement)

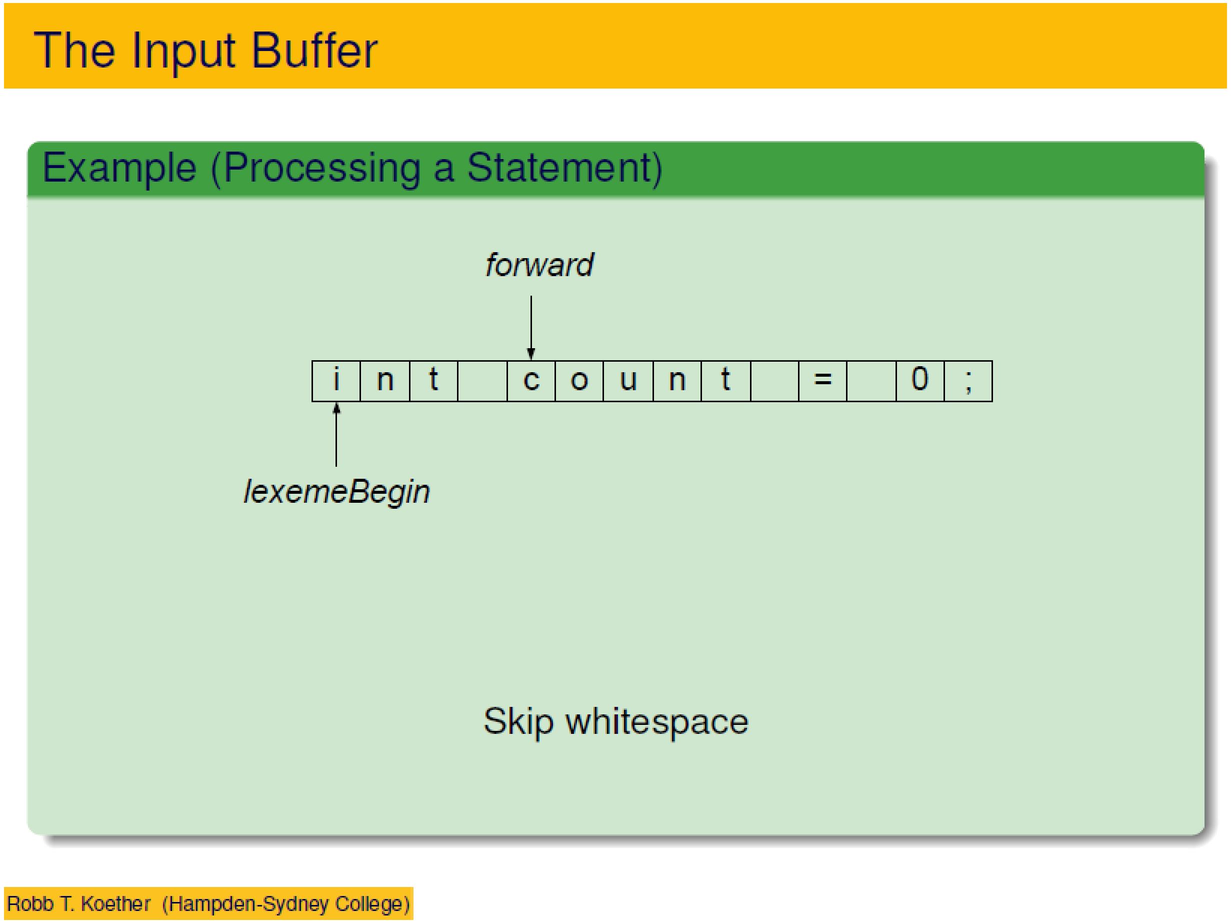
forward

i	n	t		c	o	u	n	t		=		0	;

lexemeBegin

It is the keyword **int**

Lexical Analysis



The Input Buffer

Example (Processing a Statement)

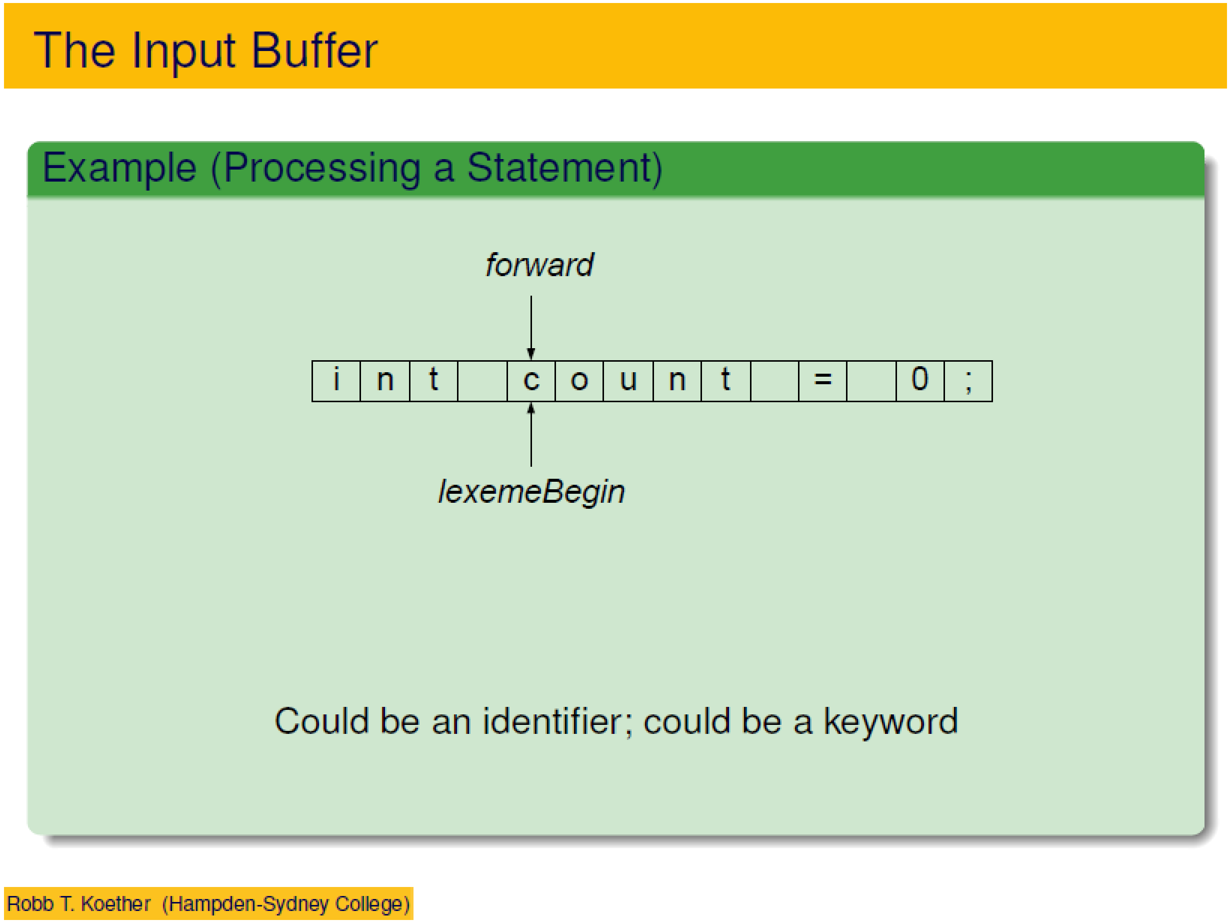
forward

i	n	t		c	o	u	n	t		=		0	;

lexemeBegin

Skip whitespace

Lexical Analysis



The Input Buffer

Example (Processing a Statement)

forward

i	n	t		c	o	u	n	t		=		0	;

lexemeBegin

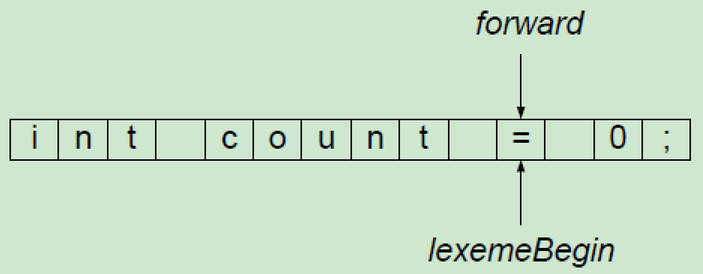
Could be an identifier; could be a keyword

Robb T. Koether (Hampden-Sydney College)

Lexical Analysis

The Input Buffer

Example (Processing a Statement)



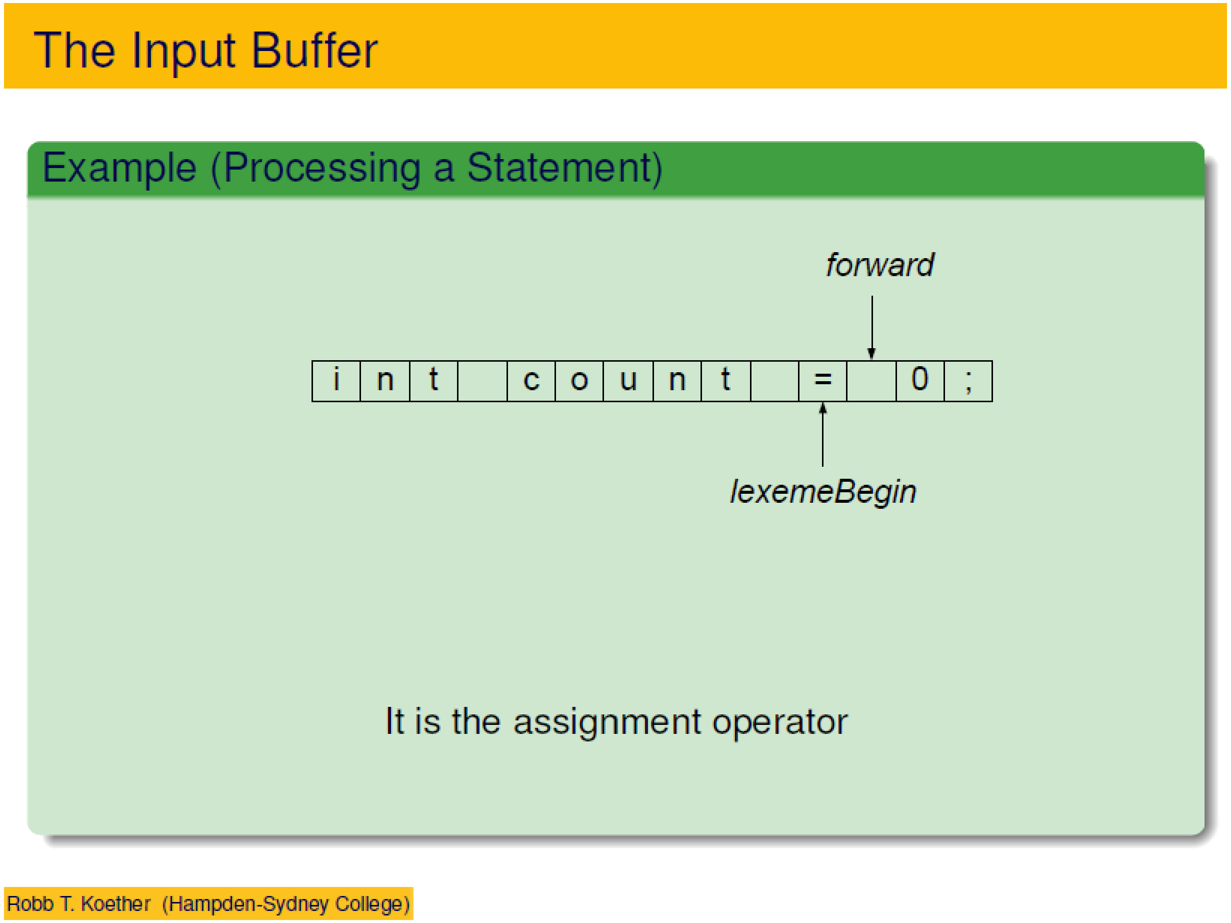
forward

i	n	t		c	o	u	n	t		=		0	;

lexemeBegin

This is an operator, but which one?

Lexical Analysis



The Input Buffer

Example (Processing a Statement)

forward

i	n	t		c	o	u	n	t		=		0	;

lexemeBegin

It is the assignment operator

Robb T. Koether (Hampden-Sydney College)

Lexical Analysis

- Classes that cover most of the tokens:
  - One token for each keyword. The pattern for a keyword is the same as the keyword itself.
  - Tokens for the operators, either individually or in classes such as the token comparison for == or !=.
  - One token representing all identifiers.
  - One or more tokens representing constants, such as numbers and literal strings.
  - Tokens for each punctuation symbol, such as left and right parentheses, comma, and semicolon.

Lexical Analysis

- Attributes for tokens:
  - More than one lexeme can match a pattern. Additional information must be provided about the particular lexeme that matched.
  - Example of the token id:
    - Need to associate with the token a great deal of information.
    - Appropriate attribute value for an identifier is a pointer to the symbol-table entry for that identifier.

Lexical Analysis

- Attributes for tokens:
  - Example of the Fortan statement E = M * C ** 2, Tokens:
    - <id, pointer to symbol-table entry for E>
    - <assign-op >
    - <id, pointer to symbol-table entry for M>
    - <mult-op>
    - <id, pointer to symbol-table entry for C>
    - <exp-op>
    - <number, integer value 2>

Lexical Analysis

- Reading ahead
  - =, ==
  - !, !=
  - <, <=
  - >, >=

Lexical Analysis

- Constants
  - Anytime a single digit appears, it seems reasonable to allow an arbitrary integer constant in its place.
  - Integer constants can be allowed either by creating a terminal symbol.
  - Numbers can be treated as single units during parsing and translation.
  - 31 + 28 + 59:
    - <num,31> <+> <num,28> <+> <num,59>

Lexical Analysis

- Recognizing Keywords and Identifiers
  - Character strings
    - Keywords: Fixed character strings such as for, do, and if to identify constructs.
    - Identifiers: Variable names, Functions.
  - Using a table:
    - Reserved words: Initializing the string table with the reserved strings and their tokens.
    - Symbol Table.