NJIT

New Jersey's Science & Technology University

THE EDGE IN KNOWLEDGE

CS 280 Programming Language Concepts

About Assignment 2

Notes for Assignment 2

- Be sure to read and understand the assignment!
- Make a list of the information that you will need to keep track of in order to do the assignment
  - How should you save the data?
- What algorithm should you use for lexical analyzer?

What are we writing

- Main test program
- getNextToken
- Code to write out tokens in trace mode

getNextToken

```
extern Tok getNextToken(
        istream& in,
        int& linenum);
```

- First argument: reference to stream to read from (might be a file, or standard input
- Second argument: reference to an int that holds the current line number

- Returns: a Tok

Tok

- Given in lex.h
- A class containing
  - Token (a value identifying how the characters have been classified)
  - Lexeme (a string with the characters that were classified
  - Line number (where in the input was the token found?)
- Constructors
- Getters
- Overloaded comparison operators

Comparison Operator

```
bool operator==(const Token token) const
{ return this->token == token; }
bool operator!=(const Token token) const
{ return this->token != token; }
```

- Allows a Tok (an instance of the class Tok) to be compared to a Token.
- Ex: `if( t == DONE )`
  - “Is this Tok the DONE token?”

Loop for getting tokens

```
int linenum = 0;
istream *in;  // make sure you initialize this!
Tok t;
while( (t=getNextToken(*in, linenum)) != DONE && t != ERR ) {
          // process the Tok
}

// keep reading tokens until DONE or ERR is returned
// loop is executed once for each new Tok that is read
```

Outline

- Set up to run (check arguments, open files, etc)
- Repeatedly call getNextToken
  - Each call returns a new Tok
  - Keep statistics on results
- Print results

- Write the pseudocode for this!

Assignment 2 pieces

- The lex.h header file is given
- You should implement the lexical analyzer function in one source file
- You should implement a test main program in another source file

- Vocareum will compile everything together

Reading from cin or a file

- The first argument to getNextToken is an istream&
- An istream& (a reference to an istream) can refer to cin, or it can refer to an ifstream
  - cin is an istream. Therefore you can pass cin as the first argument if you want to read from standard input
  - ifstream inherits from istream, so it "is a" istream. Therefore you can pass the stream as the first argument if you want to read from a file
- Keep it simple: create an istream* representing the input. Initialize it to either &cin or &the file you opened. Then just pass *that variable to getNextToken

NJIT

THE EDGE IN KNOWLEDGE