NJIT
New Jersey's Science & Technology University
THE EDGE IN KNOWLEDGE

# CS 280
# Programming Language Concepts

## NOTES for Assignment 2:
- reading characters
- conversions to numbers
- operator<<

## Reading Characters

- Assignment 2 requires you to read input
- You *could* read input into a string, a line at a time, and examine the characters in the string
  - However if a string contains several tokens, how do you keep the string intact across several calls to getToken??
  - That's possible, but it's complicated
- It's much easier to use get() and read a character at a time



## Useful stream features

- I want to check to see what the next character in the stream is, but I do not want to read it yet
  - Use the peek() method
  - The next time you call get() you will read what peek() returns
  - Example: I read in a letter. If the next character *after* the letter is a +, then I'm done recognizing a T_ID, and I do not want to read the +. I can do this by using peek() to see what the next character is

## Useful stream features

- I read a character, but I decided that I didn't want to read it after all
  - Use the putback() method
  - The next time you call get() you will read what you put back
  - Example: I'm in the middle of recognizing a T_ID, a letter followed by zero or more letters or numbers. Read a character; if it is a letter or a number, add it to the lexeme and continue. If it is NOT, then putback() the character and return T_ID
- NOTE: if you increase a line count when you see a newline, you need to decrease the line count if you putback() a newline



## How do I convert between strings and numbers?

- A string of digits is a string of characters from the set [0-9]. It is not a number.
- A number is not a string of digits
- Many ways to convert
  - write code by hand
  - use a library routine
  - use a string stream

## String to number

- Code can rely on the fact that a character that represents a digit, minus the character '0', is the numerical value of the digit (i.e. '3' − '0' is the integer 3)

```
int value = 0;
for each character in string
   value = value * 10
   value += character – '0'
```



## Number to String

- Similarly, a single digit + '0' is the character representation of that digit (i.e. 7 + '0' is '7')

```
string value;
while number != 0
   digit = number%10
   number = number/10
   value = string(digit + '0') + value
```

## Library Routines

- C
  - atoi (ascii to integer)
  - atof (ascii to float)
  - printf/sprintf
- C++
  - stoi()
  - to_string()



## sstream

- A stringstream is a stream that is connected to a string buffer
- You can << items to a stringstream and get the string that results
- You can >> items from a stringstream to parse a string
- Basically you manipulate a string as if it is an i/o device

## Printing An Object

- Java allows you to specify how an object is printed out by letting you define a toString() method
- In C++, we want to be able to print an object to an iostream, the same way that we can print numbers and strings
- Since iostream works by overloading the << operator, we need to define how << works for our object



## Printing to iostream

- iostream works by overloading the << operator
- When you write code that says, for example, `cout << 10;` the compiler generates a call to the function `operator<<(cout, 10);`
- To make your class printable on an iostream you must define your own overloaded operator<< function

```
// function must return the first argument
// so that a sequence of << operators will work

// NOTE this is NOT a member of MyClass!

ostream& operator<<(ostream& out, const MyClass& o)
{
    // format output any way you like, and
    // send it to "out"
    out << "(a,b=" << o.getA() << "," << o.getB() << ")";
    return out;
}
```