Review

Saturday, September 08, 2007
9:33 PM

- Basic Data Types (Ch 2)
  - int
  - double
  - char
  - typedef type alias
- Flow Control (App C)
  - if, else if, else
  - switch
  - while
  - do while
  - for
- Functions (App C)
  - defining your own function
    - prototype
    - body
  - function calls
  - returning a valve or no valve
  - call-by-value and call-by-reference
- Scope (App C)
  - local vs global
  - namespaces
- Operators (App C)
  - math and logical operators
  - order of precedence
- Arrays (Ch 3)
  - static vs dynamic
  - how to declare name & size
  - how to access an element
  - memory allocation & storage
  - out-of-range errors
  - multidimensional arrays
  - allocating & deallocating dynamic arrays
    - memory leaks
- Pointers (Ch 2.4)
  - how pointers differ from basic data types
  - declaring a pointer
  - assigning value to a pointer
  - accessing a pointer
    - dereferencing vs accessing directly
  - pointers and the new operation
  - pointers in functions

I/O (Ch 5.1)
- stdin & stdout
- file I/O
- formatting output
- either printf or C++ I/O manip

Strings (Ch 5.2)
- C-style strings
- C++ string class
- String operations for both
- Ch 5.3 text editor example
  - shows C++ string class used to edit a file
- Ch 5.4 pattern matching
- Ch 5.5 data encryption introduction

Recursion (Ch 10)
- base case & recursive cease
- infinite recursion
- how a recursive function call works
  - see figures on pgs 528 & 529
- run-time stack (10.3)

Structs (Ch 3.5)
- grouping variables
- syntax to create a struct
- how to declare a struct variable
- dot operator to assign values
- arrow operator for struct pointers

Classes (Ch 4)
- member variables & member functions
- encapsulation
  - public vs private sections
- declaration of a class
  - inline functions
  - function body syntax
- using separate compilation
- how to declare & use a class var
- constructors & destructors
- friend functions

Templates & Template Classes (Ch 9)
- reusability - write once, use many
- syntax for function overloading
- syntax for template classes
- using template functions & classes
- how compilers treat templates

Inheritance & Polymorphism (Ch 14)
- inheritance
  - parent us child class

public, private, protected sections
how a child can access each
polymorphism / virtual functions
dynamic birding us compile time binding
parent pointer var