

# For Wednesday

- Skim Savitch, chapters 7 and 9
- Read Savitch, chapter 11, section 1
- C++ Practice 1 due
  - Assignment is described on Blackboard.
  - Submit through Blackboard **from the Suns**.
  - Start Firefox on the Suns by typing `firefox&` at any command prompt.

# Good Function Programming

- Put prototypes at the top of the file
- DO use the names as well as the types in the prototype
- Be sure to provide a comment explaining the use of the function with the prototype

# Parameters

- Pass by value
- Pass by reference
- Const
- Important:
  - Arrays
  - Objects

# Default Arguments

- Provided ONCE
- May be omitted right to left

# Scope

- File scope (global scope) is generally appropriate only for functions and constants
- Avoid the use of global variables
- Variables are local to the block in which they are declared
- Nested scopes may result in some variables being invisible

# Random Numbers

- Not going to take time for them in class – but they will be needed for some programs
- Note that their use is explained starting on page 103

# Arrays

- Arrays in C++ are of static size.
- Declaration brackets come after the name:  
`int score[10];`  
`double tax[5];`
- May use named constants in the declarations, but NOT variables.

# Array Pitfalls

- Size must be determined at compile time
- No bounds checking – period.
- No automatic initialization.

# Initializing Arrays

# Multi-dimensional Arrays

- Actually very similar to Java
- Really an array of arrays

# Structures

- Kind of like a class, but more limited.
- Can be convenient if you really just want a piece of data that has parts.
- Everything in the struct is public by default.
- Despite what the book says – you can have methods – but just use a class if you want methods . . .

# Classes in C++

- No initial public – though you do need to declare whether inheritance is public
- Everything is private by default.
- Modifiers apply to sections – not each item.
- We do NOT normally write the code inside the class declaration. Only do that with one-line methods (getters/setters and such)
- Do NOT forget your final semi-colon

# Object Variables

- Are not, I repeat, NOT, references automatically.

# Defining Methods

- You only declare most methods in the class declaration
- So when you define the method – you have to the compiler that it belongs to a class (and which class)
- We use the scope operator (::) for this.

# Separate Compilation

- Put the class declaration in a .h file.
- Put the method definitions in a .cpp file that includes that .h file
- Also include the .h file in any file that uses your class
- When compiling on the Suns, must include all of the .cpp files in the program