

For Friday

- Read Savitch, chapter 8
- C++ Practice 2 due

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

Defining Constructors

- Mostly similar to Java
- Required to initialize any instance variables
- May use an initialization section (primarily used for inheritance and object member variables in practice)

Creating Objects

- Issues with default constructors
- Calling the constructor explicitly (generally not recommended, but occasionally appropriate)

Uses of const

- To create constants
- On parameters
 - Usually used with reference parameters
 - Important for efficiency
- On methods
 - A very good idea in general where appropriate
 - Necessary to call methods with an object passed as a const reference

Inline Functions

- Like a macro
- Much more efficient for very short functions
- Generally less efficient for longer functions

static

- Basically the same as Java for methods
- Class variables are declared similarly, but must be defined outside the class as well. Should be done in the .cpp file for the class.

Vector Class

- Rather similar to the `ArrayList` class in Java
- Basically a resizable array class

C-Strings

- Arrays ending in the null character
- Can be initialized at declaration to a string literal using =.
- Once declared, = will NOT work correctly to copy a C-string
- == does NOT work with C-strings
- If you work with C-strings – use the functions described on pages 375-376 of Savitch

String I/O

- Printing strings is very straightforward
- Reading them can be more of an issue
- `>>` will read one “word”
- To read a whole line, use `getline`
 - Member function for C-strings
 - Function that accepts a stream and a string parameter for string class strings

Character I/O

- get
- put
- putback
- peek
- ignore

Using Characters

- There are a number of built-in functions for manipulating or answering questions about characters.

string class

- Very similar to Java's in many ways
- Benefits from C++ operator overloading (=, ==, and []) all work as expected)
- Use `c_str` to convert to C-string