Structures

1-21-2002

Opening Discussion

- What did we talk about last class?
- What is a structure? What do they allow us to do?
- Connect-4 Code
  - Let's now finish the connect-4 code that we started thinking about in the last class.
  - Think of the interface that we need. What implementations could we write to get this interface to work?

Strings

- Last class we talked briefly about vectors, but not about strings. The string class is a handy way to represent character strings in C++. It is typically implemented as a vector of chars.
- I recommend that you use the string class for strings in this course, but you can feel free to use null terminated character arrays if you want also.

Reference Variables

- C++ adds the concept of reference variables that hadn't been there in C.
- They are basically like pointers only they can only be set at declaration, and they are implicitly dereferenced.
- While they aren't much use as local variables, they are commonly used for function arguments and return values.
- Should use const when possible.

Structures

- A structure is a collection of objects, potentially of different types. Elements are referred to by names instead of integers.
- Structures are helpful for when we have entities that should be grouped together in a logical way and should always stay together.
- These same abilities exist in classes so we won't use structures much, if ever.

Pointers to Structures

- Each structure in C++ is a type, and just with other types, you can have pointers to them.
- When you want to access a member of a structure that you have a pointer to you typically use the "->" notation. This is short for a dereference followed by a dot to access the member.

Copying Structures

- If you don't overload the '=' operator for a struct doing this will simply do a direct copy of the members that are actually in the structures.
- Your book refers to those members as indigenous. Members that are represented by pointers so the data for the member is outside the struct are exogenous.

Deep vs. Shallow Copying

- This default copy behavior is called a shallow copy. You should be aware of it because it can have unwanted side effects where changing something in one object will change another object that you didn't want to alter.
- Overloading operator= is typically done to produce a deep copy. This uses more memory and can have its own pitfalls.

Minute Essay

- What did we discuss today? Was everything we discussed clear to you? What have you thought about the text so far?
- Remember that you can put down any feedback you want on the minute essay.
- Next class you have your first quiz. Make sure you show up on time and that you have done the reading.