Pointer and Pass-by-Reference

10/19/2007

Opening Discussion

- Let's look at solutions to the interclass problem.
- Challenges with the maze problems. How long does it take?

Pointers

- Now we really get into the meat of C.
- Pointers are variables that store memory addresses.
- We denote a pointer type by adding a * after the name of the type. So int* is a pointer to an int.
- In memory a pointer is just a number.
- Pointers should always be initialized. A pointer that doesn't have a valid value should be set to NULL.

Pointer Operators

- We get the address of something in memory with a unary &.
- This is what you have been doing in your scanf statements all semester.
- The & operator returns a pointer to the type of the value. You can think of it as adding a * to the type.
- To tell C to look at a particular address use the unary * operator.
- This follows a pointer to what it points at. When you use this you are basically taking away a * from the type.
- Let's play with this some and look at how things are laid out in memory.

Pass-by-Reference

- For a while we have had to deal with this limitation that C only lets you return one value from a function.
- Pointers allow us to get around this.
- By passing a pointer to something instead of the thing itself we are able to change the original. This allows us to have a function effectively return multiple things.
- This is how scanf does what it does and why we need the &.
- Let's look at an example of such a function.

Minute Essay

- Pointers are very low level constructs. They expose the underlying machine. What might be some advantages and disadvantages of this?
- Interclass Problem – Write a function that asks the user for two numbers and multiplies them. The catch is that I want the function to “return” all three values back to the calling function.