Dynamic Memory

11-10-2003

Opening Discussion

- Do you have an questions about the quiz?
- What did we talk about last class?
- Do you have any questions about the assignment?
- Is there any such thing as a truly random event? Are you deterministic? Implications on free will.

Motivation

- So far the only way that we have been able to get memory is with stack variables. For single variables this works well, but when we want a collection we have to use an array and make it big enough to hold anything we might create. That often waists a lot of memory.
- Today we will look at functions that let us request memory from the computer in exactly the amount we need during runtime.

Pointers Again

- When we ask the computer for a piece of memory of a certain size, the only way it can give that to us in C is to tell us the address of that memory.
- This is the second critical use of pointers in C and where you will need to understand pointers and their syntax. Mainly you need the declarations and dereferencing.

Don't Stack It, Heap It

- When we do dynamic memory allocation the memory we get is in a very different location than when we declare a local variable. Instead of being on the stack, the memory we dynamically allocate is on the heap.
- This means it isn't lost when a function returns. But it also means that there can be some headaches in dealing with it.

malloc and free

- The way we get dynamic memory in C is with the malloc command. The function takes one argument which is the size of the memory block we want. We typically figure this out using the sizeof operation.
- malloc returns a void* which we have to cast to the type of pointer we want.
- When we are done using a piece of dynamic memory we have to call free with it. Otherwise memory "leaks" away.

Type Casts

- In C you can "turn" one type into another using a cast. This is done by putting the type you want in parentheses before the expression you want to cast.
- Here we are using it with pointers where it doesn't really change anything but the type the compiler sees. It can also be used with regular types in which case some change in form might occur.

Arrays as Pointers Again

- Remember when we first talked about arrays I said that arrays were just pointers (unfortunately that isn't as true as I'd like it to be with higher dimensional arrays).
- The way the array syntax works in C, the only requirement is that the chunk of memory we get is big enough to hold the proper number of elements.

Allocating Arrays

- If we allocate a chunk of memory big enough to hold several elements of a given type we can use that as an array. We can take that pointer name and use it with the [] syntax to access different elements of it.
- In the allocation we have to do our own work determining how big the block of memory for the array should be.

Minute Essay

- Write a little piece of code where the user inputs an int and then we dynamically allocate memory for an array of doubles with that many elements in it.
- Please look at assignment #7 and start working on it ASAP. If you have problems, but I don't hear about them until next Tuesday or Wednesday, the amount that I can help will be greatly limited.