Pointer Review (Chapter 11, Sections 1 - 3)

Easily one of the most important and powerful features in C, pointers are just a variable that points to an address in memory. A variable holds a value (like i = 2), and the same can be said about pointers. The value that pointers hold is an address to a memory location. The book has a great diagram of pointers and memory address on page 242.

Declaring a pointer variable

Pointer Variables are declared in a similar way to regular variables. Here is an example:

```
int *p;
```

This created an integer pointer called p. This p will point to a block of memory the size of one integer (the book calls it an object instead of block). Pointers can be declared on the same line as other variables of the same type.

```
float x, y, z, prices[10], *floatPointer;
```

Above we have 3 floats (x, y, and z), an array of 10 floats, and a float pointer called floatPointer. Note that the type of the pointer matters. If you create a float pointer, it should only point to floats. The pointer type (also called reference type) can be anything, including other pointers. We will talk about void types and pointers to pointers in later lectures.

Address Operator

One of the two operators designed to work with pointers is the address operator (&). If x is a variable, then &x is the address of x. When a pointer variable is declared, memory for one pointer is allocated. This does not initialize the pointer. The pointer will likely be pointing to an address that is garbage or inaccessible. Pointers do not start as NULL.

```
int i, *p;
p = &i;
```

In the above example, we set our pointer p to the address of i. This can also be done all on one line, assuming that i is declared before p.

Indirection Operator

The other operator is the indirection operator (*). Sometimes called the dereferencing operator, this will gain access to the value held within the address a pointer is pointing at. So, using the values i and p from above (where p points at i):

```
printf("The value of i (via p): %d\n", *p);
```

The above line will print the value of the address p points to, which happens to be the value of i.

Never apply the indirection operator to an uninitialized pointer variable.

Pointer Assignments Examples

Assume for the following examples we have this declaration and assignments.

```
int i, j, *p, *q;
p = &i;
q = &j;
```

So, here we 4 variables. Two integers (i and j) and two integer pointers (p and q). We assigned the address of i to p and the address of j to q.

```
*p = 5;
*q = 10;
printf("The value of i: %d\n", i); // 5
printf("The value of j: %d\n", j); // 10
```

Now we have set the values of i and j to 5 and 10 by using the indirection operator on the pointers q and p.

```
q = p
```

The line above would set q to the same address as p. Now both p and q point to the address that holds i.