Bubble Sort

■ Let's write a bubble sort program using two functions — **bubbleSort** and **swap**.

■ Function **bubbleSort** sorts the array.

■ It calls function **swap** to exchange the array elements array[j] and array[j + 1]

■ Remember that C enforces information hiding between functions, so **swap** does not have access to individual array elements in **bubbleSort**.

■ Because **bubbleSort** wants **swap** to have access to the array elements to be swapped, **bubbleSort** passes each of these elements by reference to **swap**—the address of each array element is passed explicitly.

~~~~~~

■ The following parameter appears in the function header for bubble
*int (*compare)(int a, int b)*

■ This tells bubble to expect a parameter (compare) that's a pointer to a function that receives two integer parameters and returns an integer result

■ Parentheses are needed around *compare to group the * with compare to indicate that compare is a pointer.

■ If we had not included the parentheses, the declaration would have been
*int *compare(int a, int b)*
which declares a function that receives two integers as parameters and returns a pointer to an integer.

■ The third parameter in the prototype could have been written as
int (*)(int, int);
without the function-pointer name and parameter names.

■ The function passed to bubble is called in an if statement as follows:
*if ((*compare)(work[count], work[count + 1]))*
~~~~~~

■ Just as a pointer to a variable is dereferenced to access the value of the variable, a pointer to a function is dereferenced to use the function

■ The call to the function could have been made without dereferencing the pointer as in

if (compare(work[count], work[count + 1]))

which uses the pointer directly as the function name.

■ We prefer the first method of calling a function through a pointer because it explicitly illustrates that compare is a pointer to a function that's dereferenced to call the function.

■ The second method of calling a function through a pointer makes it appear as if compare is an actual function.

■ This may be confusing to a programmer reading the code who would like to see the definition of function compare and finds that it's never defined in the file.