**The Dereference Operator**

 Bookmark this page

This operator is another source of confusion for those who are new to pointers.  The reason is because we are "overloading" the use of the \* symbol.  It is used as the pointer symbol and also as the dereference symbol.   So what is a dereference?  Consider the following code sample.  
  
    int num = 3;            // a simple variable holding the value 3  
    int \*pNum = &num;        // a pointer holding the address of num  
    cout << pNum << endl;    // output the memory address of num  
    cout << \*pNum << endl;    // output the value 3  
  
In the first line, we declare a variable called num and assign it the value 3.  Next we create a pointer \*pNum and assign it the memory address of the variable num.  The first cout statement outputs the address of num, because that is what the pointer variable pNum holds.   But the last line outputs the value 3.  Why?  Here is where the power and the danger of pointers starts to become apparent.  
  
Using the dereference operator, you can gain direct access to the underlying value in the variable num.  If you are unsure what this means, lets add another code segment to help clarify.  Copy and paste this code into a C++ app to see it working on your system if you want to test it.  
  
    int num = 3;  
    int \*pNum = &num;  
    cout << pNum << endl;  
    cout << \*pNum << endl;  
  
    \*pNum = 45;  
    cout << \*pNum << endl;  
    cout << num << endl;  
  
What is being demonstrated here is that we are using the dereference operator to change the underlying value of num, indirectly.  The two last cout lines demonstrate that the value of num and \*pNum are precisely the same value. You might be thinking at this point that we could simply have changed the value of num directly and you would be correct. But hold that thought for a moment and  we'll revisit the importance of this later in the lesson when we discuss the purpose of pointers and their uses.