**Introducing Pointers**

 Bookmark this page

In order to understand pointers we must first start with an understanding of how computers handle the data for you application.  When you declare a variable in your code, you are providing key pieces of information to the compiler for handling the data stored as a part of that variable.  The data type determines how much memory is requested for that variable.  Once the variable is declared, you don't need to worry about where in memory it is stored.  Simply referring to the variable name in code is sufficient for gaining access to the value stored there.  
  
But let's take this concept a little further and explore memory locations in your code.  Consider the following code:  
  
    int num = 3;  
  
This is a simple variable declaration of type int, called num, that stores the value 3.  The variable num actually references a location in the computer's memory.  The memory location is what gives num an identity.  During an execution of the application code, while num is in scope, it will have one memory address for the entirety of its scope. Most objects in C++ have an identity, which means that they have a unique memory address during their lifetime (scope).  
  
A pointer is simply a variable that holds the memory address of an object in C++.  To use pointers, you create a pointer variable.  As an example, if we want to, we can create a variable to point to our num variable's address in memory by using the following line of code:  
  
    int\* pNum = &num;  
  
In this line of code, int\* is referred to as a pointer to an int.  In other words, it will contain the memory address of an int data type.  The variable that will actually hold that memory address is called pNum.   After the assignment operator, we see the use of the & symbol.  the & symbol, in this instance, is known as the  address-of operator and it allows us to get at the memory address for the num variable.  
  
Depending on the documentation you read, you may also find the \* used in this manner:  
  
    int \*pNum;  
  
In this case, the author of the text is trying to show that pNum is the actual pointer.  Regardless, the concept is the same.  pNum is a pointer an int.  
  
If you compile and run the following code, you will notice the memory address of num output to the console window.  On my first run, the address for num was 001CF79C and on the second run it was 0018FD38.  Subsequent runs will offer up different memory addresses showing how the "identity" of the variable is only good for the duration of its lifetime.  The values you see may be different that these addresses returned for this execution of the program.  
  
  
    #include "stdafx.h"  
    #include <iostream>  
      
    using namespace std;  
      
    int main()  
    {  
      
        int num = 3;  
        int \*pNum = &num;  
  
        cout << pNum << endl;  
  
        return 0;  
    }  
  
NOTE: A word of caution when using pointers.  You should always initialize a pointer variable to NULL, nullptr, 0, or with a memory address.   Leaving a pointer variable uninitialized is an error that can result in difficult to find bugs and create security issues for your code.  
  
    int \*pNum;  // not recommended  
      
    cout << pNum << endl;  
  
Referring to the preceding code, some compilers will not compile the code and will issue a warning or raise an error because of the pointer not being initialized.  For compilers that do not catch this error, you could see a memory address output to the console but you will have no idea what data resides at that memory address.  
  
One final note, when declaring a pointer, ensure that the variable you will be pointing to is the same data type as the pointer you are declaring.  This is important because of the difference in the number of bytes stored in the various data types.

For more information , you can see:   
C and C++ with Visual Studio: <https://aka.ms/edx-dev210.2x-vs02>