Cody Wisniewski

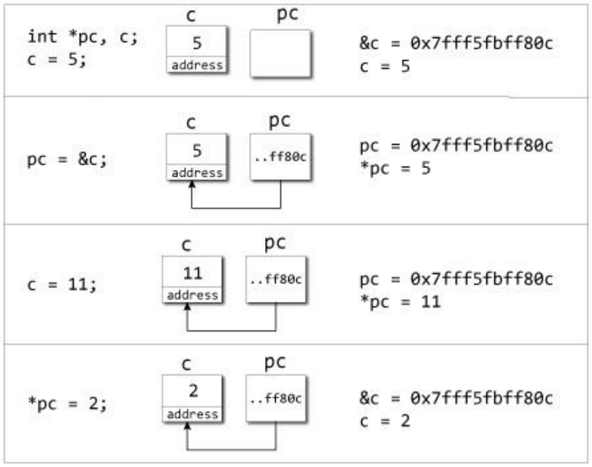
Technical Writing

Information Page Project

October 25, 2015

Pointers and Why they are Important

Pointers are a very important part of programming because it gives you much more control of the program you are writing, but also much more responsibility. Pointers are data objects that "point" to a location of memory. This is useful for many reasons, one being so that instead of passing in a large object full of tons of information into a function, you can just pass in the simple memory location of the object and then access it that way. Pointers are used to make your program faster, more efficient, and are used in many important data structures including linked lists, structs, and memory allocation.

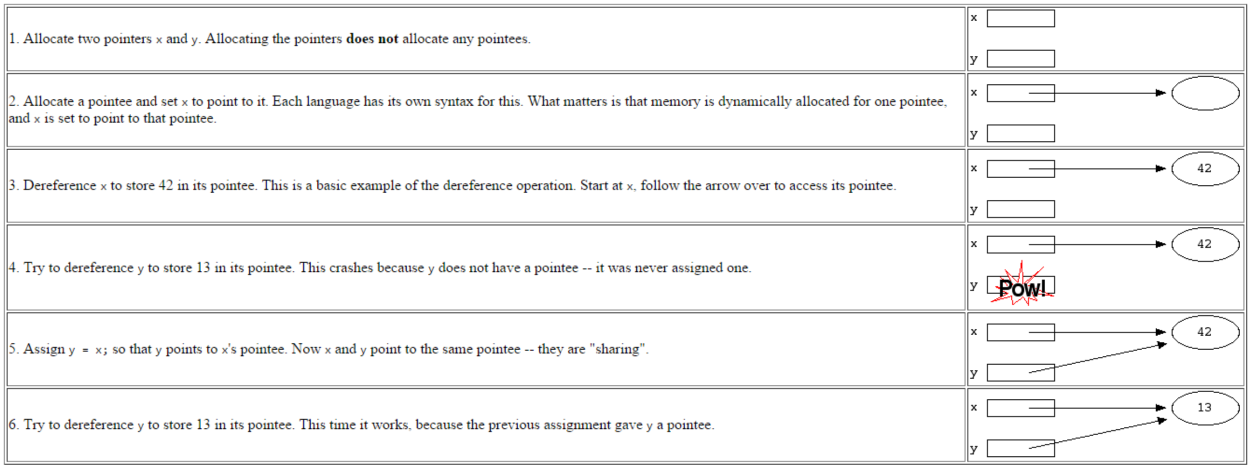


```
int *pc, c;        &c = 0x7fff5fbff80c
c = 5;             c = 5

pc = &c;           pc = 0x7fff5fbff80c
                   *pc = 5

c = 11;            pc = 0x7fff5fbff80c
                   *pc = 11

*pc = 2;           &c = 0x7fff5fbff80c
                   c = 2
```

This is some examples on how pointes are used in coding. When you see the '*' that means that variable is a pointer and it also means to dereference that pointer. When you see the '&' that means you are referencing that pointer.

So what that really means is '*' gets the value of the variable stored at the memory location, and that memory location is accessed by using '&'.

Here is a generalization on how pointers work in all languages from a high level.



1. Allocate two pointers x and y. Allocating the pointers **does not** allocate any pointees.	x y
2. Allocate a pointee and set x to point to it. Each language has its own syntax for this. What matters is that memory is dynamically allocated for one pointee, and x is set to point to that pointee.	x y
3. Dereference x to store 42 in its pointee. This is a basic example of the dereference operation. Start at x, follow the arrow over to access its pointee.	x 42 y
4. Try to dereference y to store 13 in its pointee. This crashes because y does not have a pointee -- it was never assigned one.	x 42 y Pow!
5. Assign y = x; so that y points to x's pointee. Now x and y point to the same pointee -- they are "sharing".	x 42 y
6. Try to dereference y to store 13 in its pointee. This time it works, because the previous assignment gave y a pointee.	x 13 y