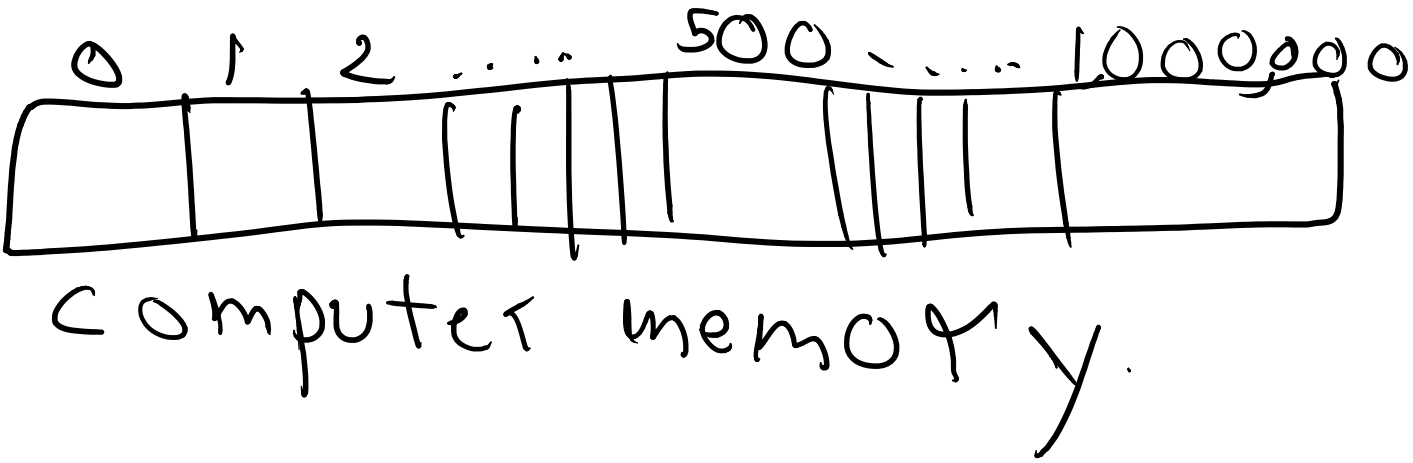
C-21 pointers

Thursday, March 14, 2013 2:41 AM

First of all think of your computer as a one dimensional array of billions of elements with each element containing some memory. Now each element has an ADDRESS (known as an index) and a value stored at that address known as the VALUE.



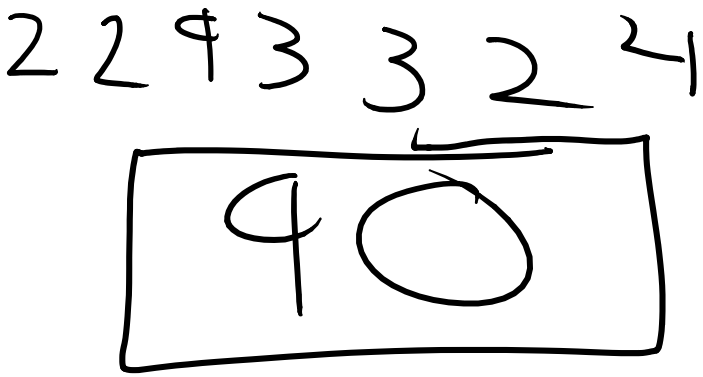
The address operator (&) returns the address of a variable for example

```
1  #include <stdio.h>
2
3  //break
4  //continue
5  int main() {
6
7      int i = 90;
8      printf("%d", &i);
9
10
11
12     return 0;
13 }
14
```

2293324

2293324 is the actual memory address of where the variable i is stored in the computer MEMORY
At that address the value of 90 (in this example) is stored.

-



So at the location of 2293324 the integer 90 Is stored

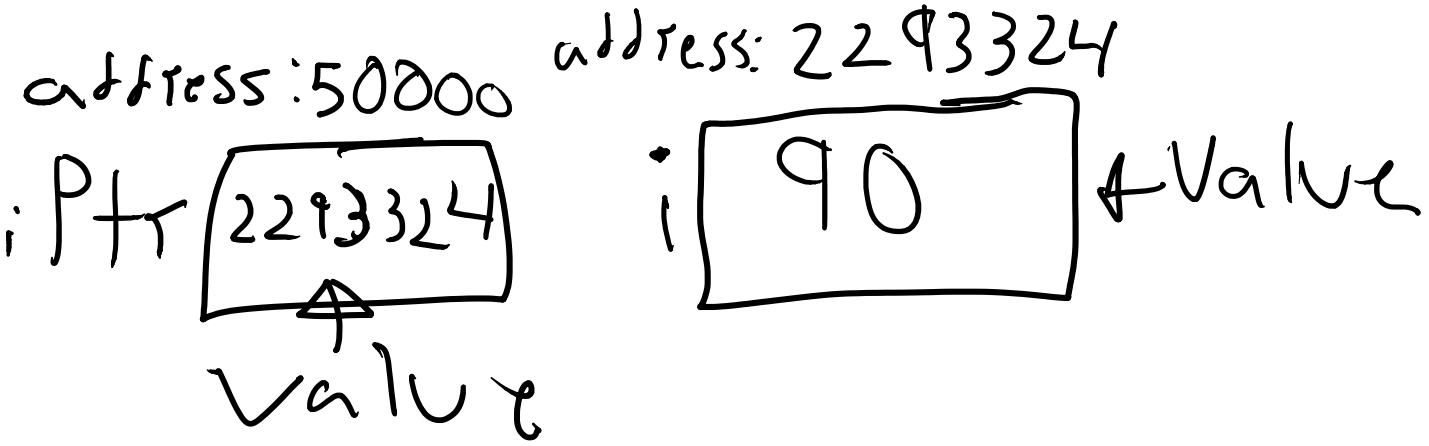
Now what a pointer does.
Pointers are variables whose values are MEMORY ADDRESSES. A pointer contains an address to a variable (for instance the address of i: 2293324).

For instance the statement below is creating a pointer (by using the asterisk, that means you are declaring a pointer!) and in that pointer variable the address of I is getting stored. Note that pointers

have types just like regular variables do because pointers can only reference addresses that contain the same type variables.

```
8    int *iPtr = &i;
```

Here is a drawing of what has happened after the above line is executed with the initialization of i



As you can see the pointer variable iPtr contains the address of i and i contains the value of 90 and has an address of 2293324.

Note that the pointer variable iPtr also has an address, just as a regular variable does. The key difference is that pointer variables (denoted by an asterisk *) contain the ADDRESS of another variable . While regular variables simply contain values.

NOTE: it is possible for a pointer to contain an address that points to ANOTHER pointer. We will get into this later as it is a little too much.

The indirection operator (*) returns the VALUE to which the pointer points. For example:

```
5  int main() {
6
7      int i = 90;
8      int *iPtr = &i;
9      printf("%d", *iPtr);
10
11
12
13     return 0;
14 }
15
```

This will print out the variable value 90 because it will print out the value contained at the address of the pointer. This is known as **deferencing .**

Note that the * and & symbols are opposites of each other meaning that they cancel each other out

*&iPtr = &*iPtr;