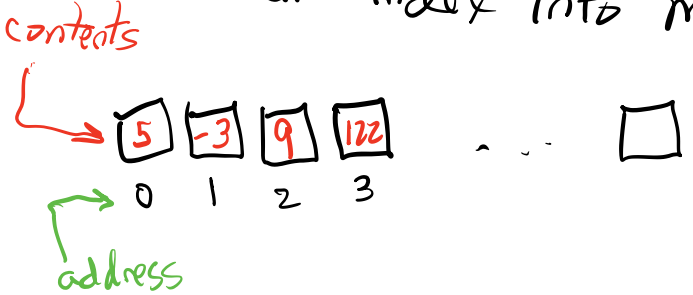
Pointers in C

A pointer is an address
- an index into memory.



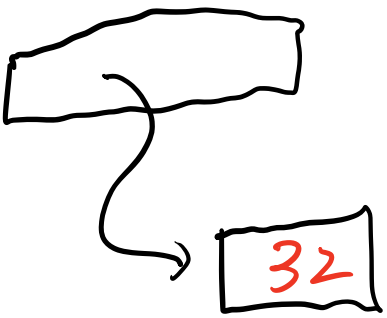

```
int x; // x is a variable (4 bytes) that
       // hold a 32-bit integer.
       // x is uninitialized, so we don't know what
       // it contains.

int *p; // p is a pointer. It can hold an
        // address (an index into memory)
        // of 4 bytes in memory where
        // an int can be stored.
        // p is uninitialized so far.
```

this is the type of p

The declaration can also be written
```
int* p; // not usually written this way.
```

Conceptually:

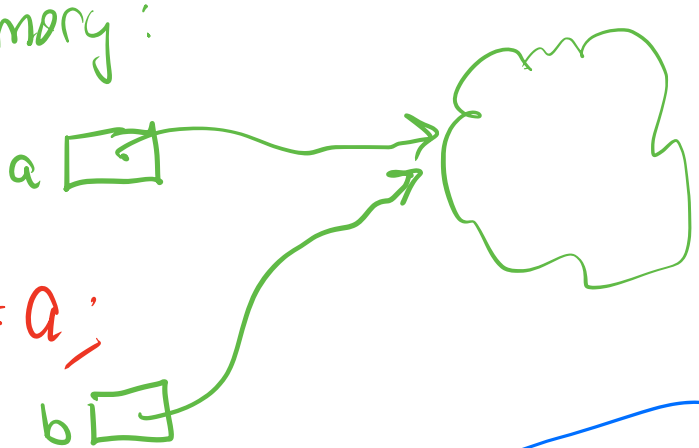
P points somewhere else.



Java has implicit pointers.

```
class Apple {...}

Apple a = new Apple();
```

In memory:




```
Apple b = a;
```

Back to C:

How do we assign pointer variables?

```
x = 7; // int variable
p = &x; //assigns the address of x to p.
```

↑ "address-of" operator

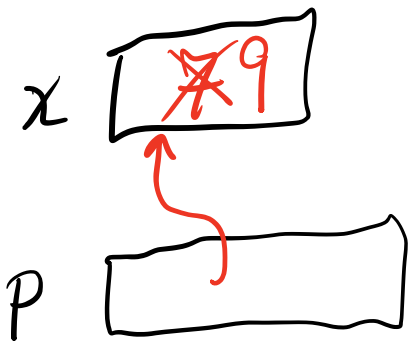
Using a pointer:

- use the "dereference" operator, *

↑ follow the pointer

"*p" represents the memory location that p points to.

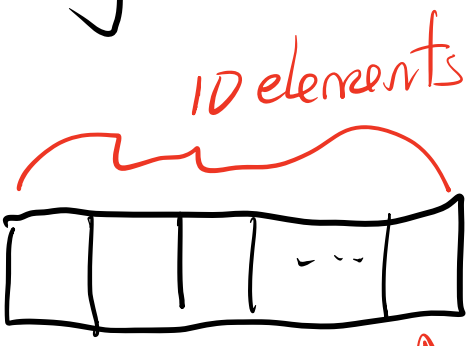
```
int x;
int *p;
x = 7;
p = &x;
```




```
printf("x = %d\n", x);   "x = 7"
printf("*p = %d\n", *p);   "*p = 7"
*p = *p + 2;   // overwrites x's memory location
printf("x = %d\n", x);   "x = 9"
```

Revisiting arrays:

```
int a[10];
```

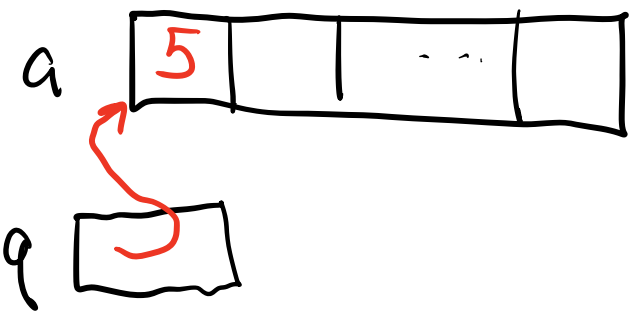


"a[3]" represents the contents of the fourth element of a.

"a" represents the address of the start of a.

- it's a constant, so "a = ..." is illegal.

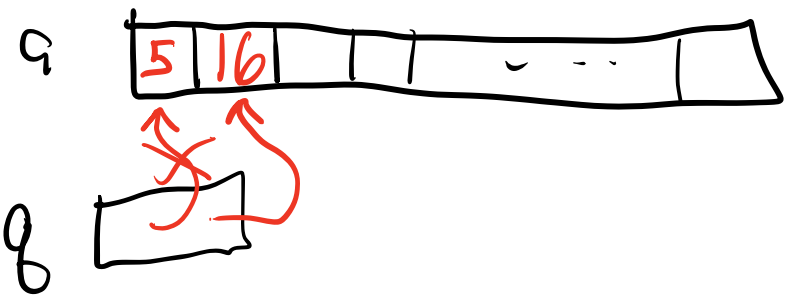
```
int *q = a;
```




```
*q = 5;
```

We can do arithmetic on pointers:

```
q=q+1; // This modifies the address
       // stored in q.
       // Sets q to point to
       // the next integer in
       // memory.
       // Address is actually increased
       // by 4.
```




```
*q=16;
```