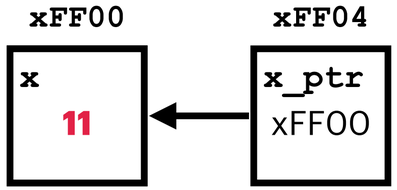
Pointer Basics

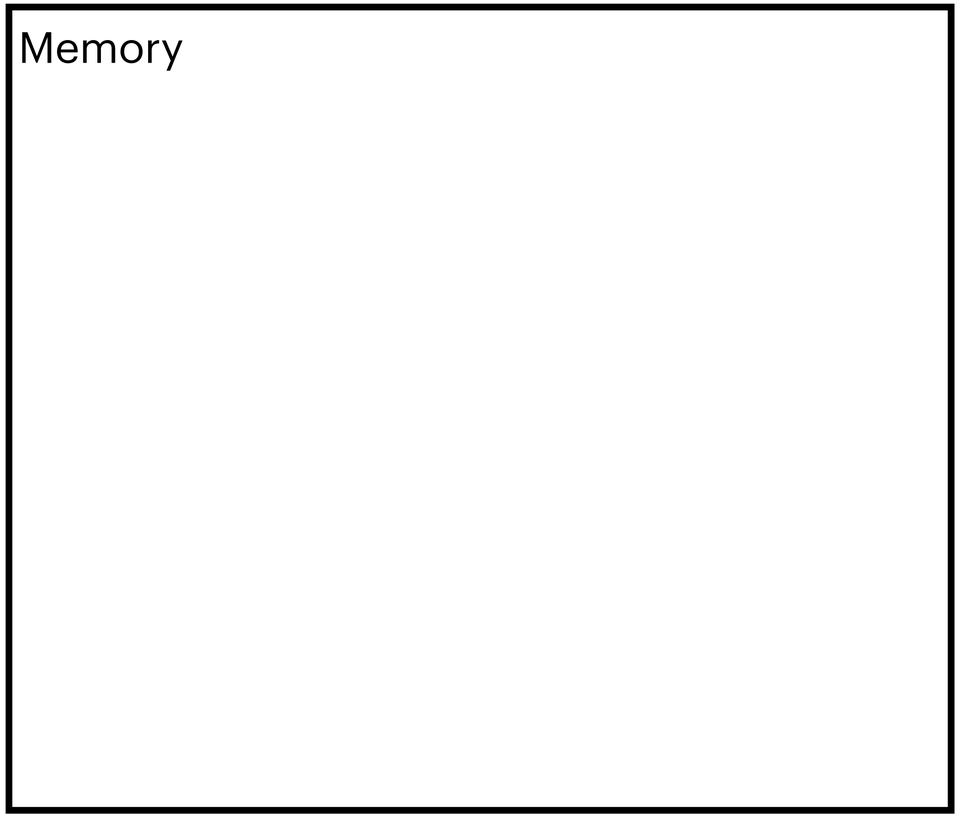
```
int x = 10;
int *x_ptr = &x;
*x_ptr = 11;
```



Pointer Syntax

What's happening in this code snippet?

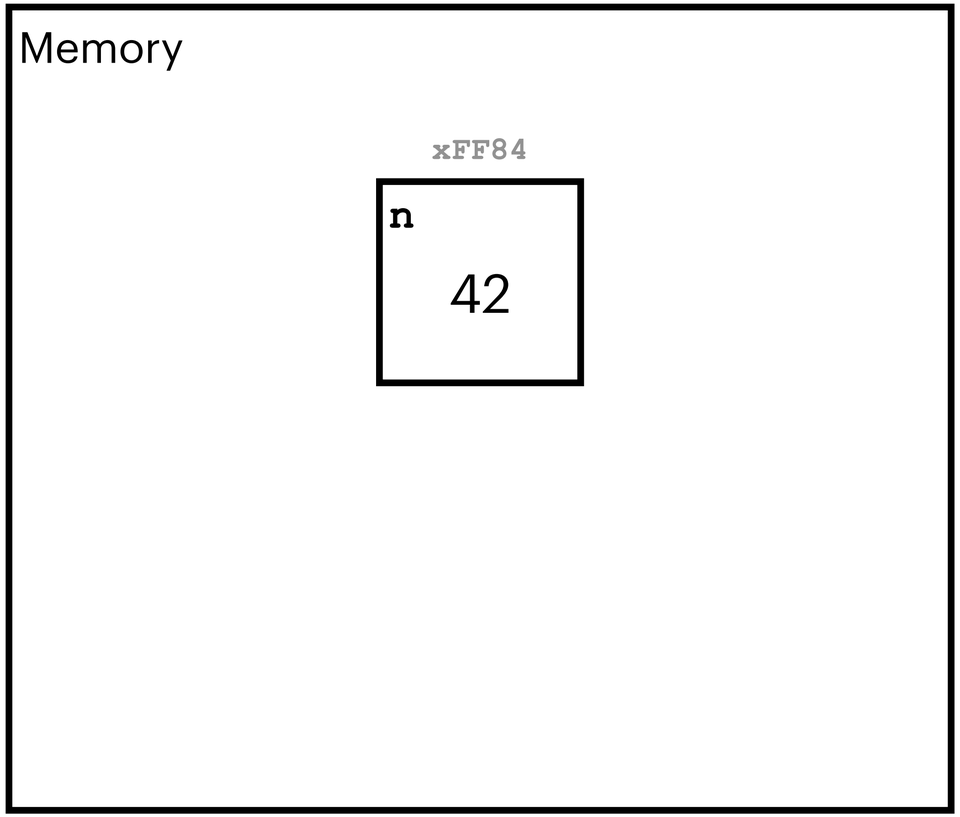
```
int n = 42;
int *p;
int *q;
p = &n;
*p = 5;
*q = 17;
q = p;
*q = 8;
```



Pointer Syntax

What's happening in this code snippet?

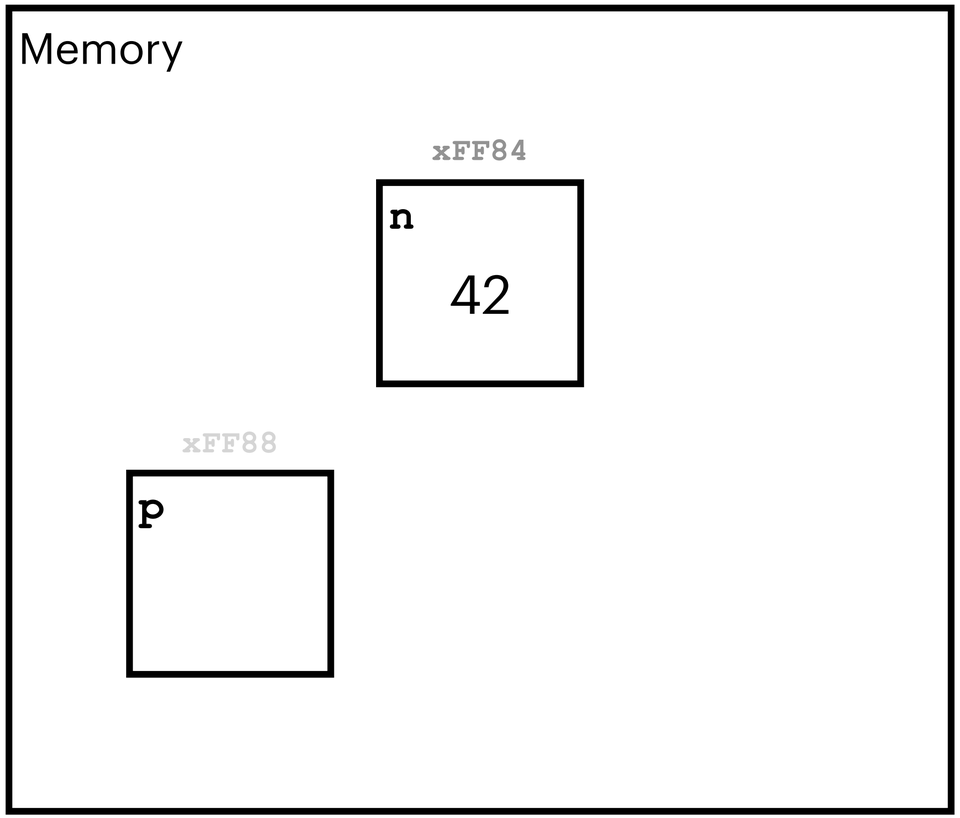
✅
```
int n = 42;
int *p;
int *q;
p = &n;
*p = 5;
*q = 17;
q = p;
*q = 8;
```



Pointer Syntax

What's happening in this code snippet?

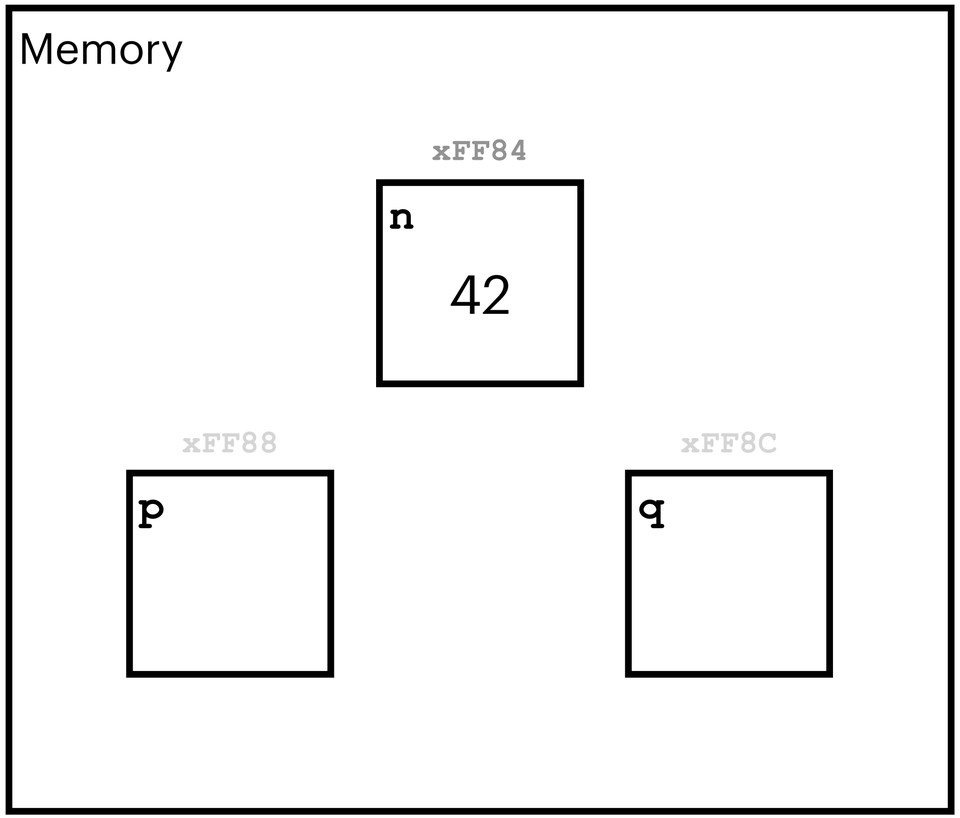
```
✅ int n = 42;
✅ int *p;
   int *q;
   p = &n;
   *p = 5;
   *q = 17;
   q = p;
   *q = 8;
```



Pointer Syntax

What's happening in this code snippet?

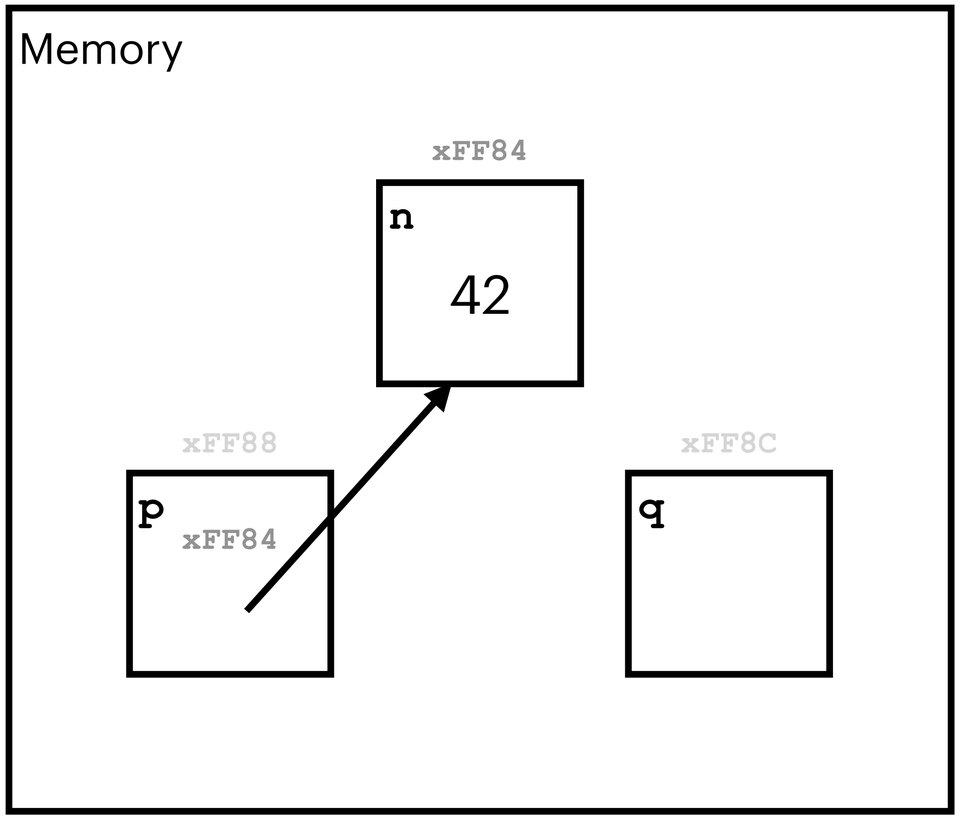
```
✅ int n = 42;
✅ int *p;
✅ int *q;
   p = &n;
   *p = 5;
   *q = 17;
   q = p;
   *q = 8;
```



Pointer Syntax

What's happening in this code snippet?

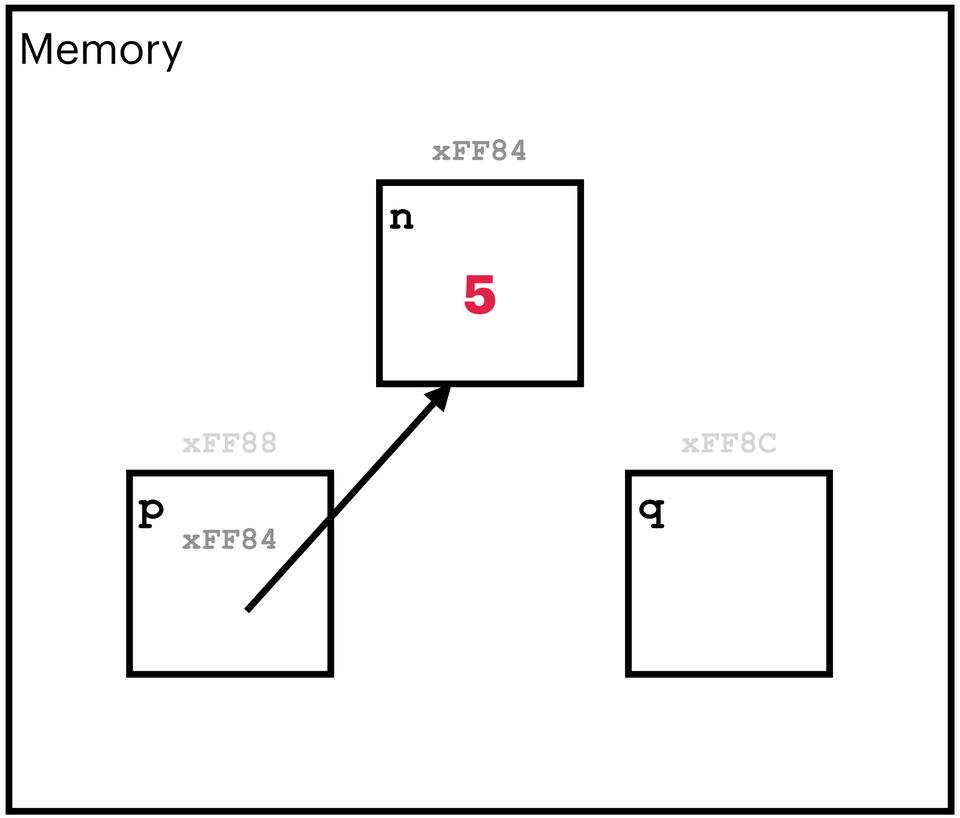
```
✅ int n = 42;
✅ int *p;
✅ int *q;
✅ p = &n;
   *p = 5;
   *q = 17;
   q = p;
   *q = 8;
```



Pointer Syntax

What's happening in this code snippet?

```
✅ int n = 42;
✅ int *p;
✅ int *q;
✅ p = &n;
✅ *p = 5;
   *q = 17;
   q = p;
   *q = 8;
```

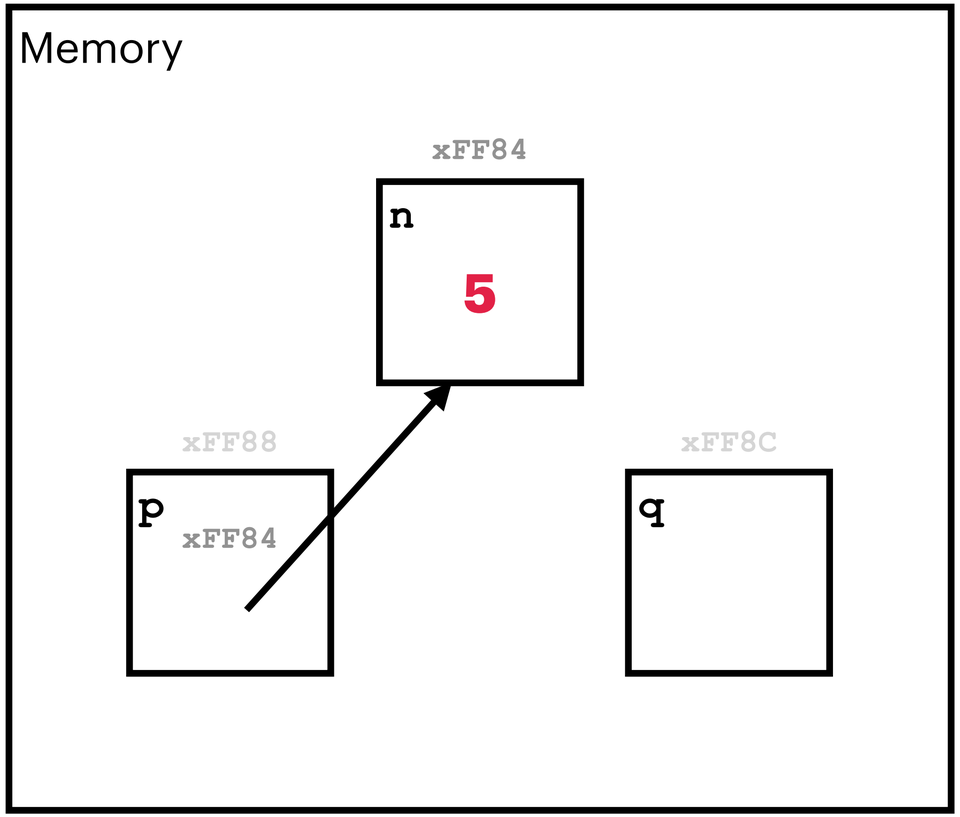


Pointer Syntax

What's happening in this code snippet?

```
✅ int n = 42;
✅ int *p;
✅ int *q;
✅ p = &n;
✅ *p = 5;
❌ *q = 17;
   q = p;
   *q = 8;
```

(q doesn't store an address yet)

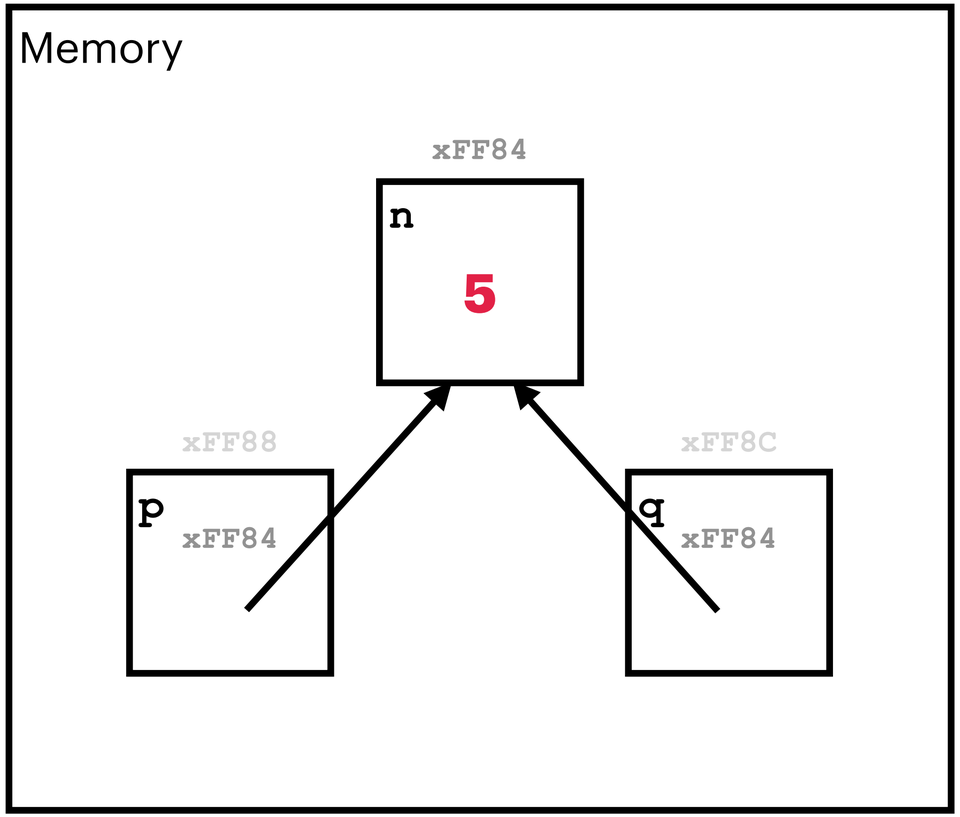


Pointer Syntax

What's happening in this code snippet?

```
✅ int n = 42;
✅ int *p;
✅ int *q;
✅ p = &n;
✅ *p = 5;
❌ *q = 17;
✅ q = p;
   *q = 8;
```

(q doesn't store an address yet)

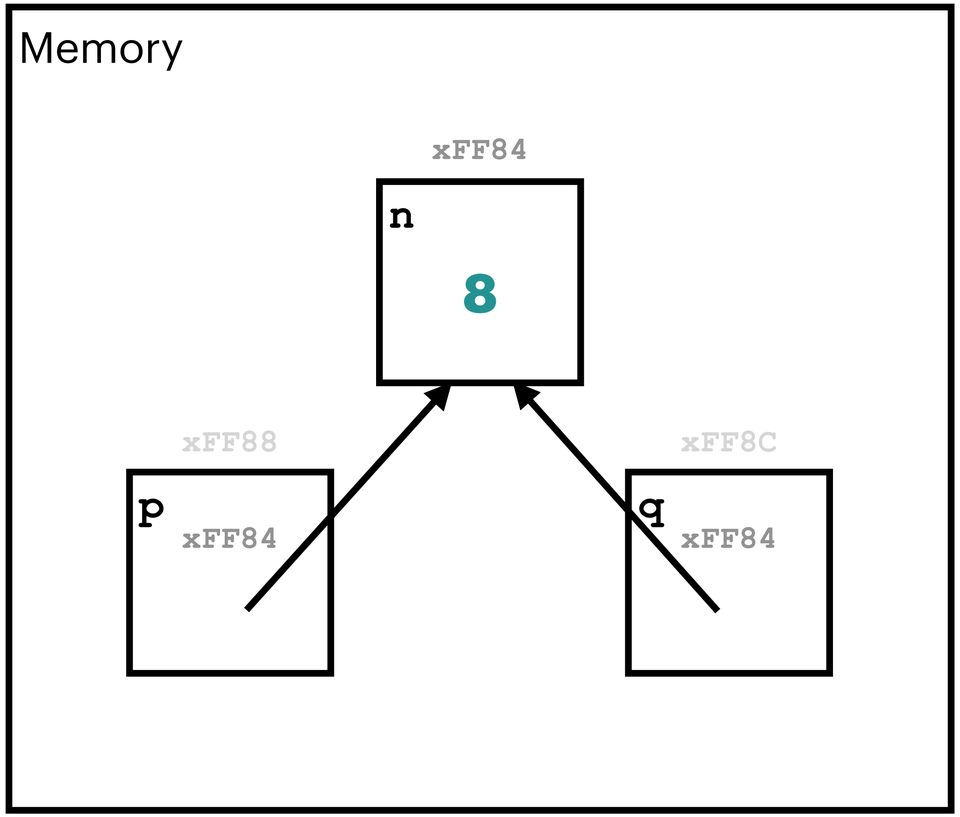


Pointer Syntax

What's happening in this code snippet?

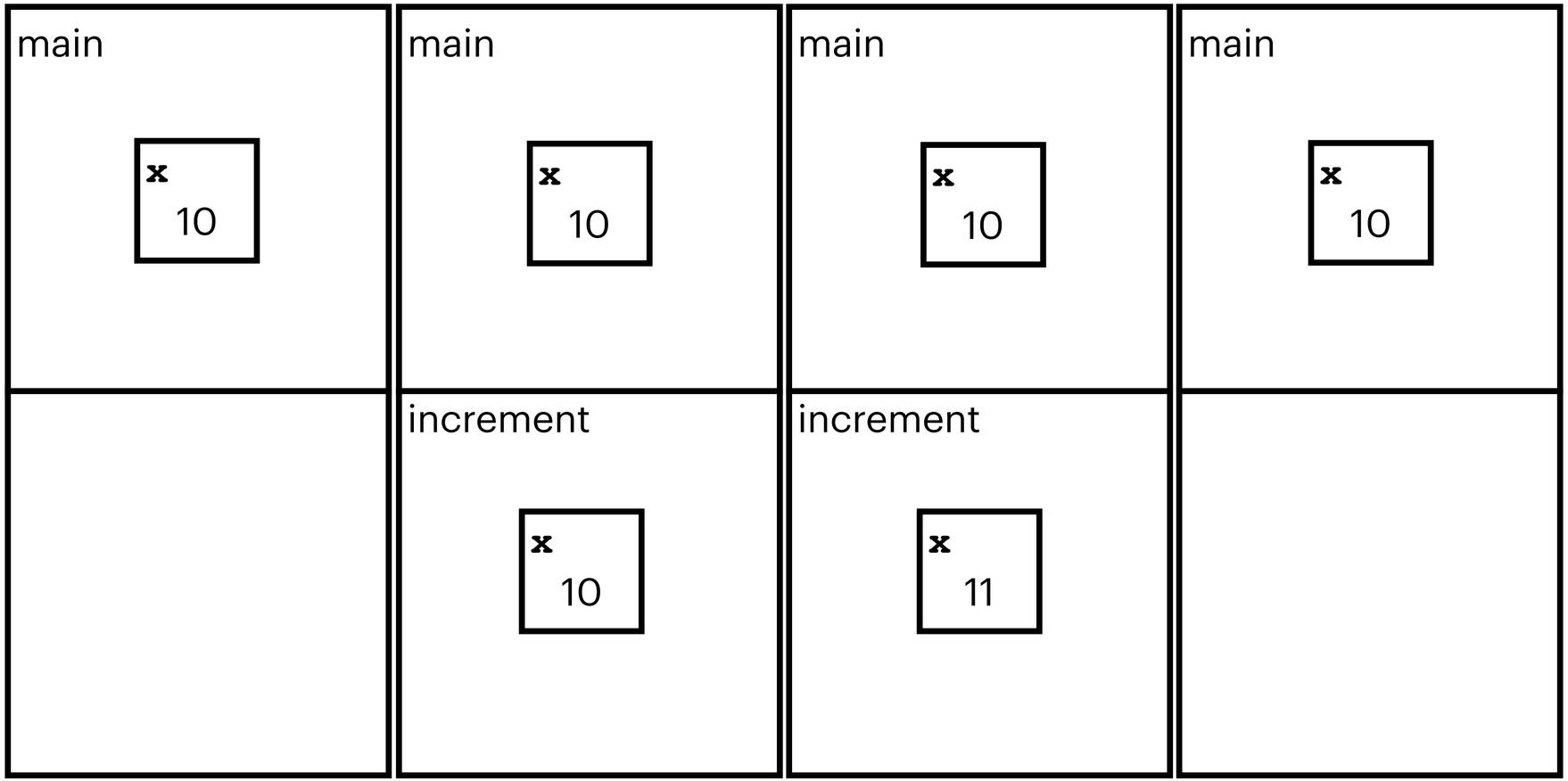
```
✅ int n = 42;
✅ int *p;
✅ int *q;
✅ p = &n;
✅ *p = 5;
❌ *q = 17;
✅ q = p;
✅ *q = 8;
```

(q doesn't store an address yet)



Writing Functions without Pointers

Why can't the variables change?



`x` is initally given the value 10

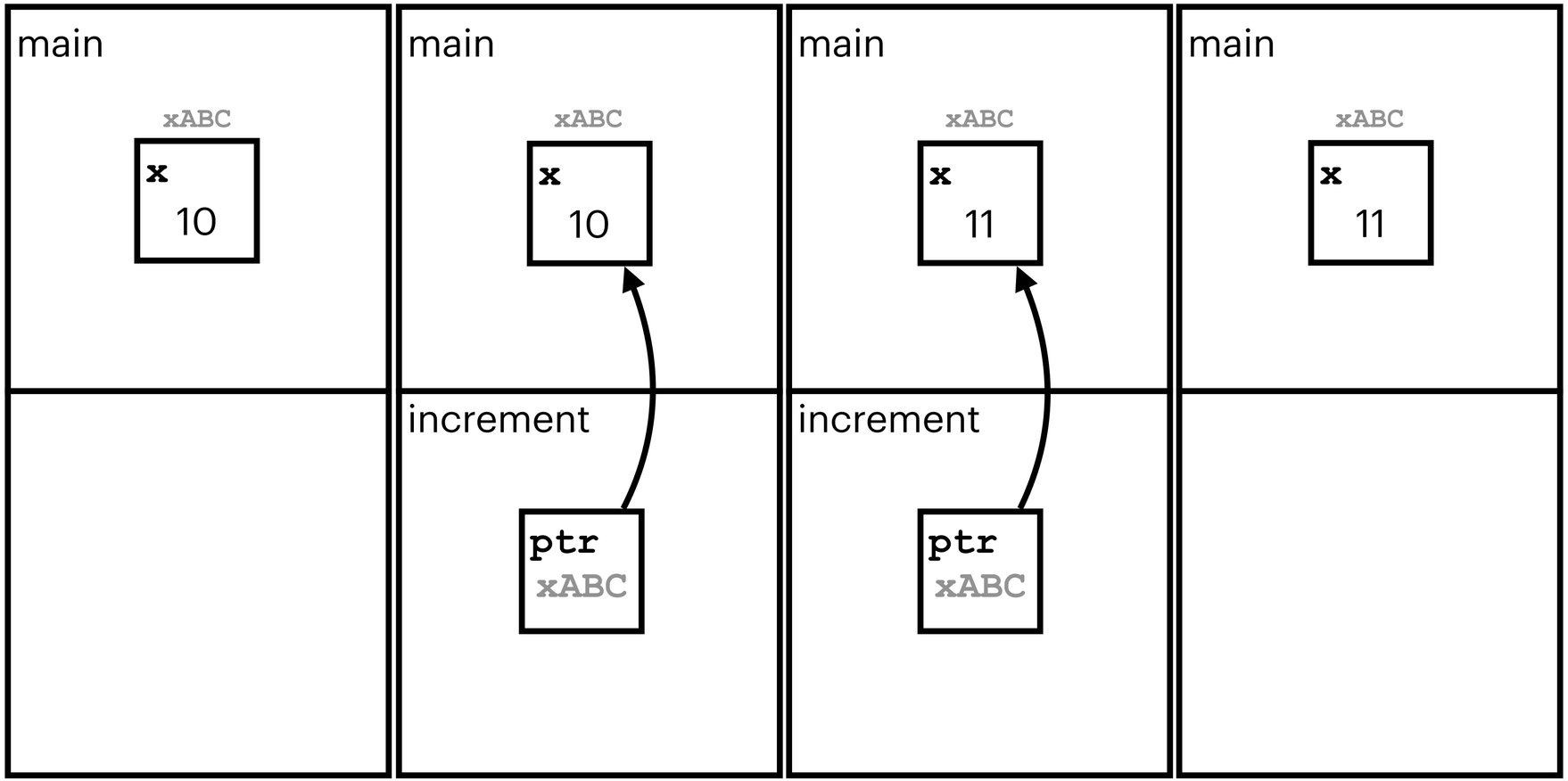
`increment` is called. The value of `x` from `main` is **copied** over into the memory for `increment`

`x` in `increment` is incremented.

`increment` is exited

Writing Functions with Pointers

Changing the value of variables through a function



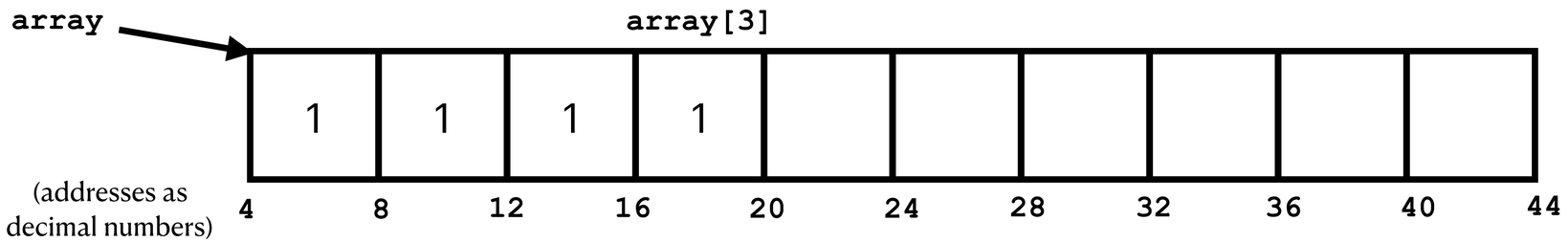
x is initally given the value 10

increment is called. The **address** of **x** from **main** is copied over into the memory for **increment**

ptr is deferenced and **x** in **main** is incremented.

increment is exited

Indexing Pointers



`array[3]` means...

1. take the address `array` (4)
2. add `3 * sizeof(int) == 12` to it (4 + 12 = 16)
3. retrieve the element at that address
   (or write to the element at that address)