Pointers + Dynamic Memory.

Reading: end of l3.pdf, l4.pdf, Prof. Li's notes.

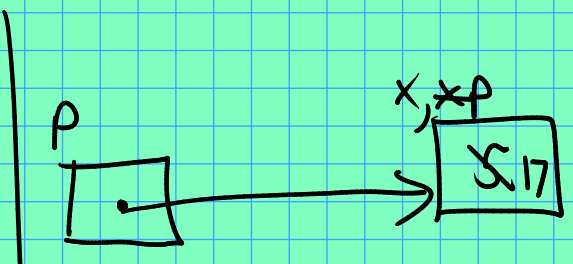
Pointers: they're just variables that store memory addresses.

Code sample:

```
int x;
x = 5;
int* p = &x;
```

`int*` — new datatype! (pointer to int)

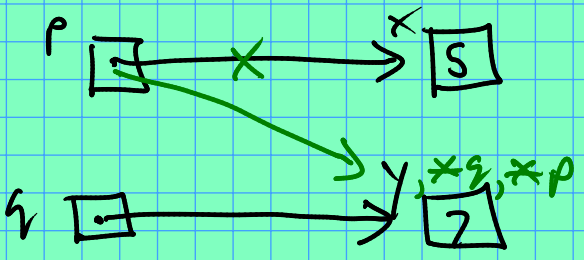
`&x` — new operator! "address of"




```
*p = 17;
cout << x; // prints 17
```

What happens if we assign one pointer to another?

```
int x = 5, y = 7;
int* p = &x;
int* q = &y;
```



p = q;

Note: p = *q wouldn't compile!

type int* type int

*p = *q ≡ x = y

Dynamic memory

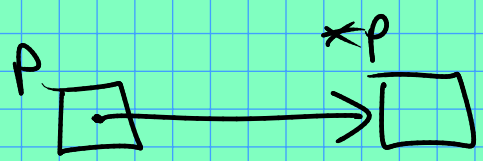
- How to allocate memory as your program is running??

(Vector does this as needed when you call push_back(...))

How can we learn this magic?

In C++, use the "new" operator:

```
int* p = new int; // creates an
                  // "anonymous" int
```



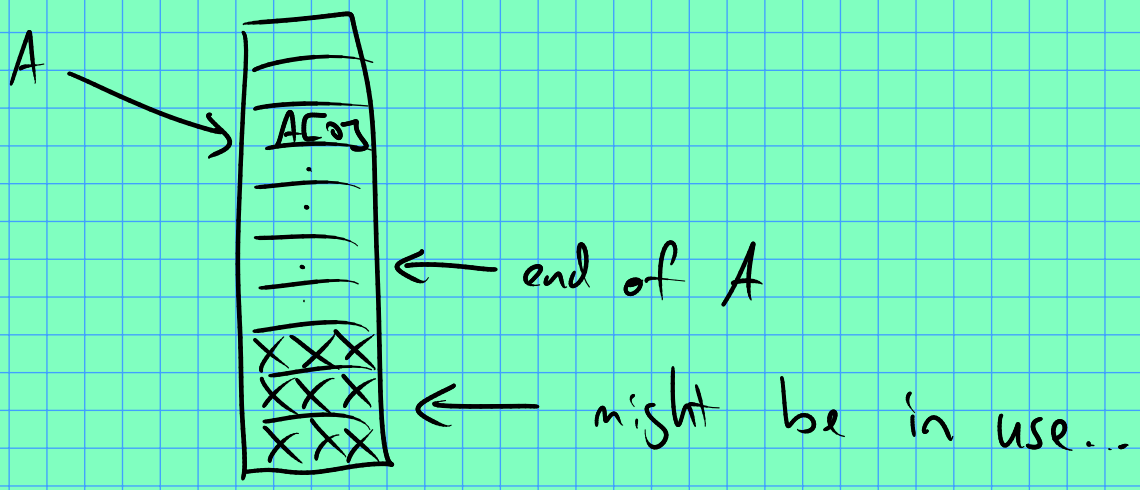
Important Note: In general, you must delete any thing that you have new'd!

```
delete p; // frees memory that was
          // used by *p.
```

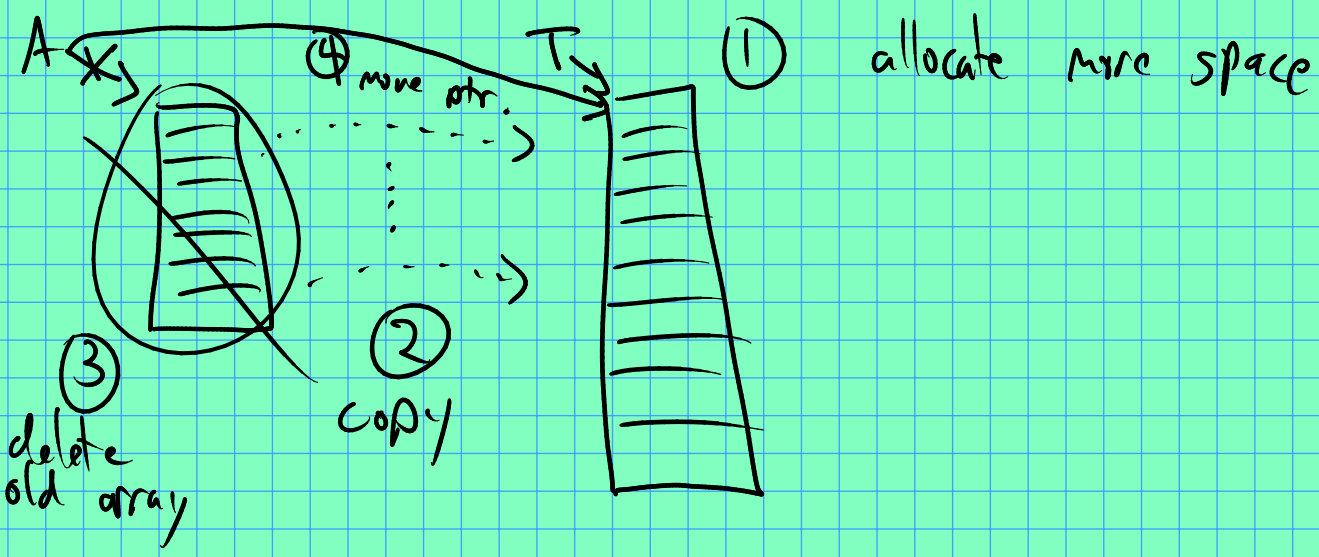
We can also allocate arrays:

```
int * A = new int[100];
// Allocates space for A[0], A[1] .. A[99].
// No A.push_back( ) :(
```

Exercise: how to "grow" an array?



Only reliable way: find A a new home...



Say A had n elements, and we want 2*n.

```
int* T = new int[2*n];  // ①
for(int i = 0; i < n; i++) {
    T[i] = A[i];
}                                  // ②
delete [] A;
A = T;  // Note! must delete A
        // before doing this!
```