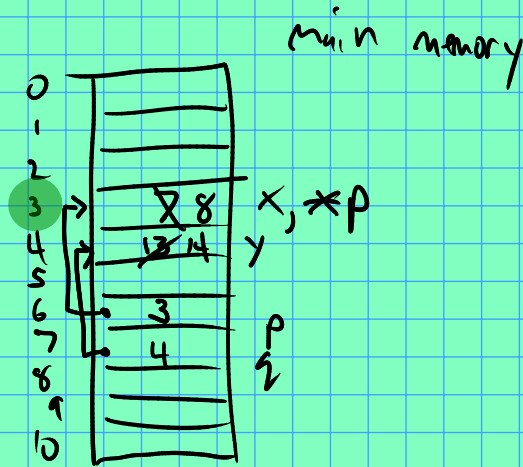


New topic: pointers + dynamic memory

Pointer variables: just a variable that stores a memory address.

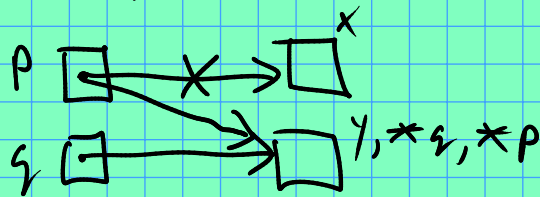
```
int x;  
x = 8;  
int *p = &x;  
(*p)++;  
cout << x; // prints 8.
```



```
int y = 13;  
int *q = &y;  
y++;  
cout << *q; // prints 14
```

Alternate picture:

what does
 $p = q$; do?




Aside: in C (not C++), there are no
by-reference parameters. Instead,
just pass pointers.

<u>C++</u>	
<u>void f(int &x);</u>	<u>C</u>
int a = 99;	void f(int *x);
f(a);	int a = 99;
	f(&a);

Dynamic Memory

How to allocate memory as a program runs?

To dynamically allocate memory in C++,
use the "new" operator:

`int* p = new int;` 

Note: in general, you must release dynamically allocated memory once you're done with it.

How? Use "delete":

`delete p;`

You can also allocate larger blocks (arrays):

`int n;`

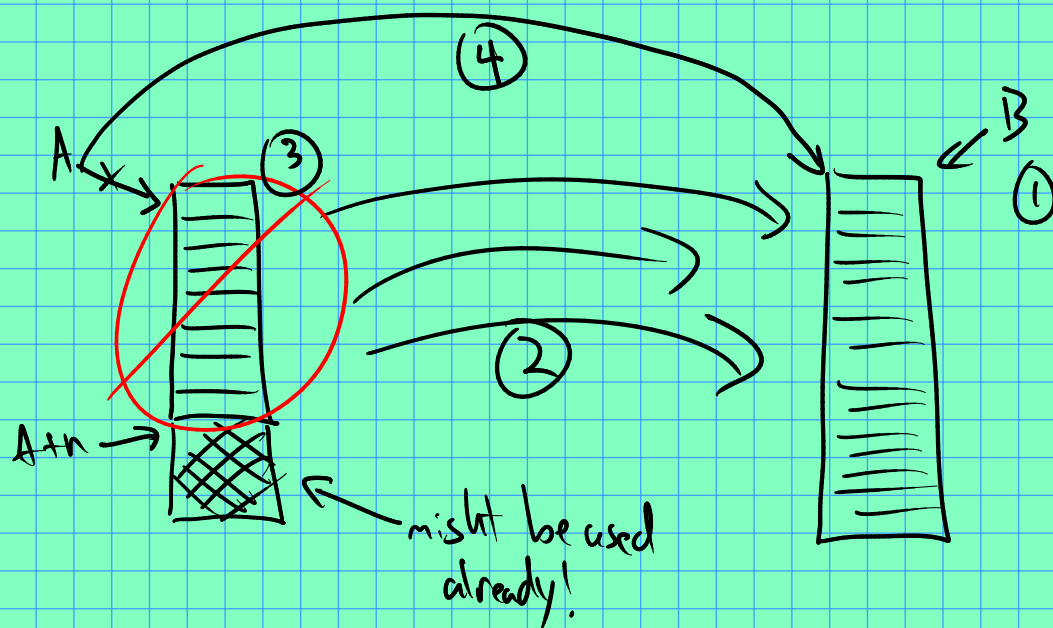
`cin >> n;`

`int* A = new int[n];` // allocates block
// of n integers.

// to recycle the array:

`delete [] A;`

Exercise: "grow" a dynamically allocated array.



High level steps:

- ① Allocate bigger array B.
- ② Copy from A into B.
- ③ recycle old A
- ④ redirect A to point to B

```
void resize(int*& A, size_t n, size_t newSize)
{
    int* B = new int[newSize]; // ①
    for(size_t i = 0; i < n; i++)
        B[i] = A[i]; // ②
    delete[] A; // ③
    A = B; // ④
}
```

← this is the reason for the
by reference `int*&`