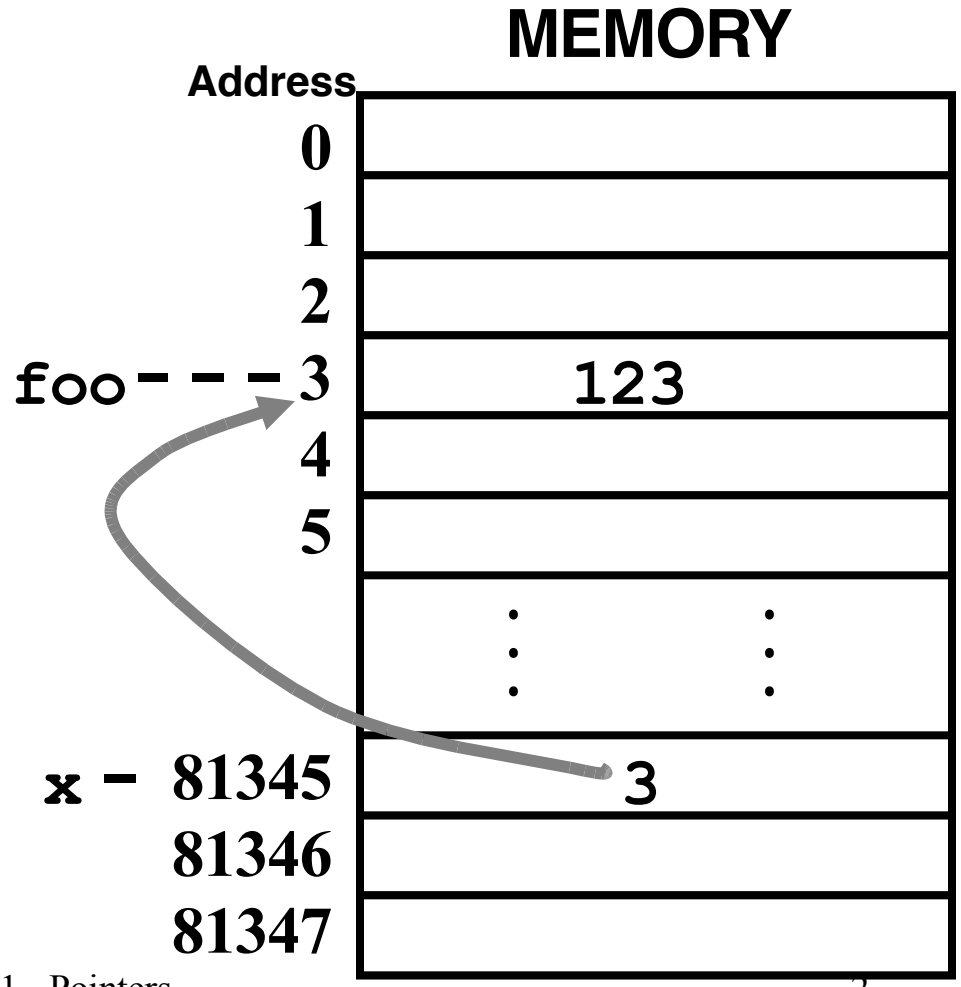
C++ Pointers and C Strings

Pointers

A pointer is a variable that holds the *address* of something else.

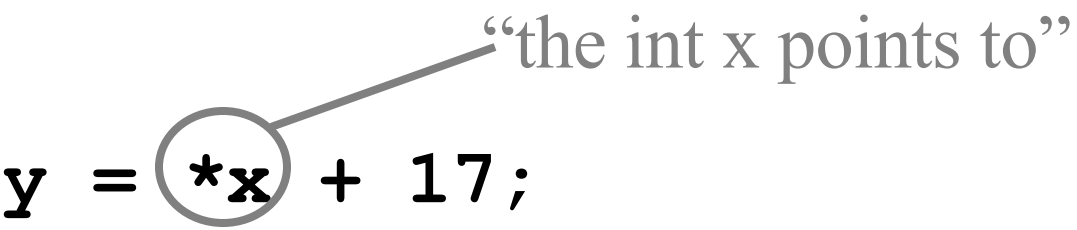
```
int foo;
int *x;

foo = 123;
x = &foo;
```



int *x;

- x is a pointer to an integer.
- You can use the integer x-points-to in a C++ expression like this:



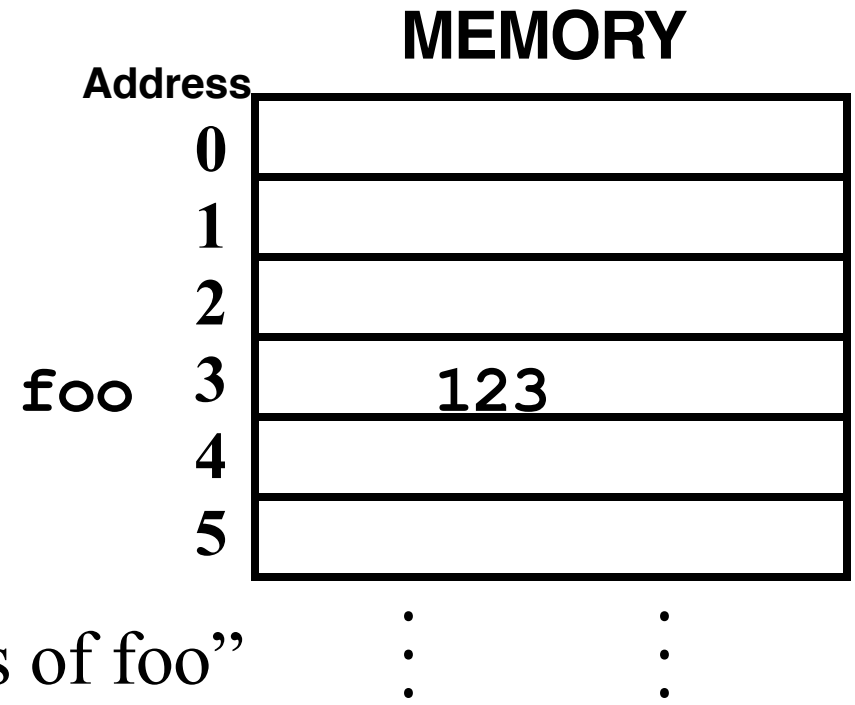

```
*x = *x +1;
```

&foo

In C++ you can get the *address* of a variable with the "&" operator.

```
int foo;

foo = 123;
x = &foo;
```



`&foo` means "the address of foo"

Assigning a value to a *dereferenced* pointer

A pointer must have a value before you can *dereference* it (follow the pointer).

```
int *x;
*x=3;
```

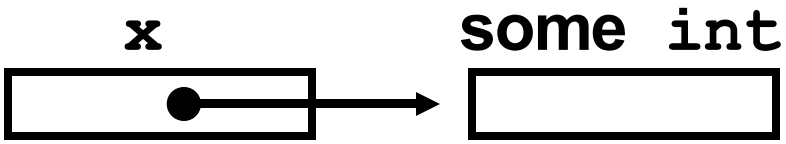
ERROR!!!
x doesn't point to anything!!!

```
int foo;
int *x;
x = &foo;
*x=3;
```

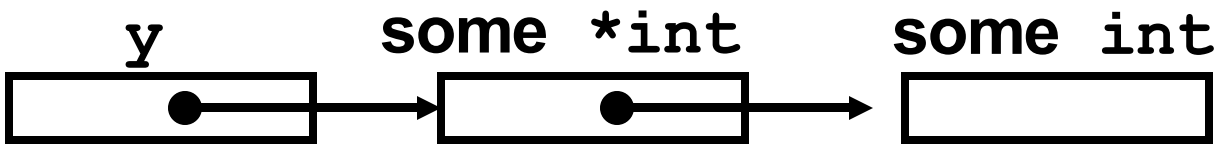
this is fine
x points to foo

Pointers to anything

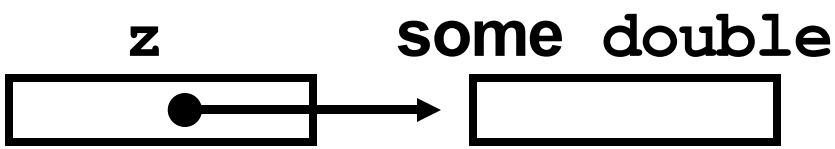
```
int *x;
```




```
int **y;
```




```
double *z;
```



Pointers and Arrays

- An array name is basically a *const* pointer.
- You can use the [] operator with a pointer:

```
int *x;
int a[10];
x = &a[2];
for (int  i=0;i<3;i++)
   x[i]++;
```

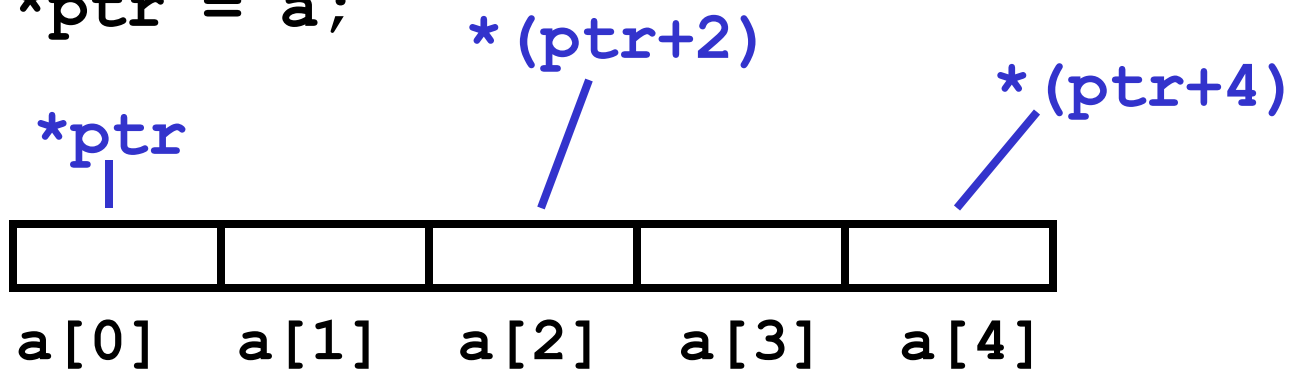
x is "the address of **a[2]** "

x[i] is the same as **a[i+2]**

Pointer arithmetic

- Integer math operations can be used with pointers.
- If you increment a pointer, it will be increased by the size of whatever it points to.

```
int *ptr = a;
```




```
int a[5];
```

printing an array

```
void print_array(int a[], int len) {
  for (int i=0;i<len;i++)
      cout << "[" << i << "] = "
           << a[i] << endl;
}
```

array version

```
void print_array(int *a, int len) {
  for (int i=0;i<len;i++)
      cout << "[" << i << "] = "
           << *a++ << endl;
}
```

pointer version

Passing pointers as parameters

```
void swap( int *x, int *y) {
  int tmp;

  tmp = *x;
  *x = *y;
  *y = tmp;
}
```

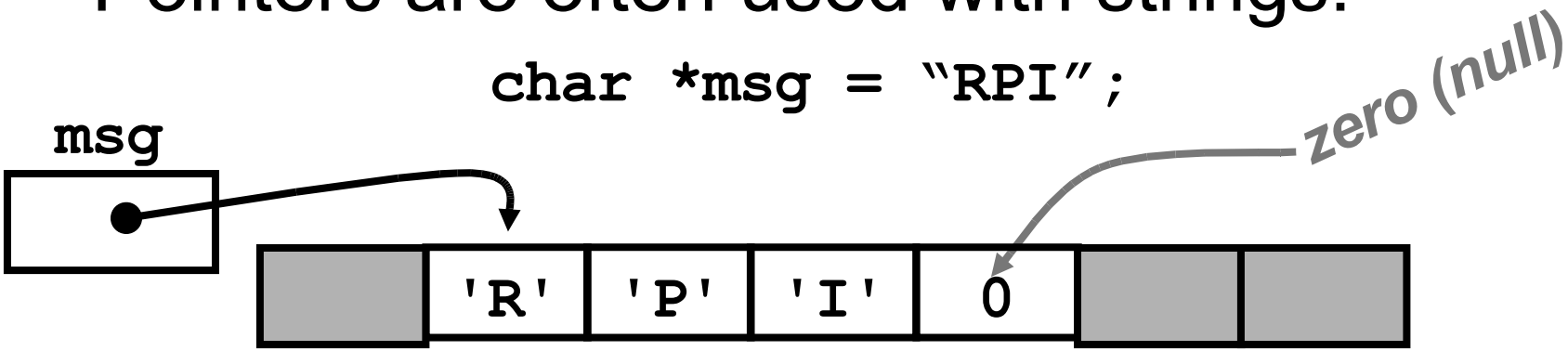
Pointer Parameters

- Pointers are passed by value (the value of a pointer is the address it holds).
- If we change what the pointer points to the caller will see the change.
- If we change the pointer itself, the caller won't see the change (we get a copy of the pointer)

C strings

- A C *string* is a *null terminated* array of characters.
  - null terminated means there is a character at the end of the the array that has the value 0 (null).
- Pointers are often used with strings:

```
char *msg = "RPI";
```



String Manipulation Functions

- C++ includes a library of C string handling functions:

```
char * strcpy(char *dst, const char *src)

char * strcat(char *dst, const char *src)
```

lots more!

String Example - Count the chars

```
int count_string( char *s) {
  int n=0;
  while (*s) {
     n++;
     s++;
  }
  return(n);
}
```

while the thing pointed to by **s** is not null

increment count

set **s** to point to the next char

Another way

```
int count_string( char *s) {
  char *ptr = s;
  while (*ptr) {
     ptr++;
  }
  return(ptr - s);
}
```

pointer arithmetic!

C String vs. C++ `string` class

- C++ string class is much easier to use
  - comparison operator, concatenation operator, no memory allocation issues...

- There are times when you have to use a C String (char *), generally you just convert from a `string` using `c_str()`.
  - we saw this when opening a file...