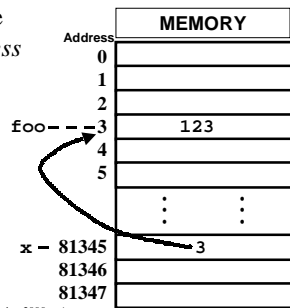


C++ Pointers and Strings

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:

```
y = (*x) + 17;
```

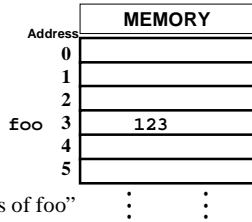
“the int x points to”

```
*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”

C++ Spring 2000 Arrays

4

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

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

this is fine
x points to foo

C++ Spring 2000 Arrays

5

Pointers to anything

```
int *x;  
int **y;  
  
double *z;
```

The diagram illustrates pointer chains. For the first example, variable 'x' (labeled 'some int') points to a memory box. For the second example, variable 'y' (labeled 'some *int') points to a memory box that points to another memory box (labeled 'some int'). For the third example, variable 'z' (labeled 'some double') points to a memory box.

C++ Spring 2000 Arrays

6

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]*

C++ Spring 2000 Arrays

7

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;
          *(ptr+2)
          *(ptr+4)
*ptr
|
+-----+
| a[0] | | a[1] | | a[2] | | a[3] | | a[4] |
+-----+
int a[5];
```

C++ Spring 2000 Arrays

8

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
```

C++ Spring 2000 Arrays

9

Passing pointers as parameters

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

C++ Spring 2000 Arrays

10

Pointer Parameters

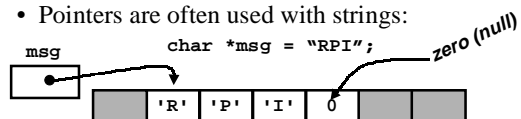
- Pointers are passed by value (the value of a pointer is the address it holds).
- If we change what the pointer points to the caller will see the change.
- If we change the pointer itself, the caller won't see the change (we get a copy of the pointer)

C++ Spring 2000 Arrays

11

C++ strings

- A *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:



C++ Spring 2000 Arrays

12

String Manipulation Functions

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

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

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

lots more!

C++ Spring 2000 Arrays

13

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

C++ Spring 2000 Arrays

14

Another way

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

pointer arithmetic!

C++ Spring 2000 Arrays

15

Exercises (for those so inclined)

- Write `strcpy`
- Write a function that compares 2 strings and returns `true` if they are the same string, `false` if they are not.
- Write a function that removes all spaces from a string.
