THE C++ STRING CLASS

#include <string>

Delroy A. Brinkerhoff

C-STRING VS. `string` OBJECTS

C-STRINGS

- Fundament (built-in) type
- Null-terminated arrays
- Array size determines the maximum length
- Little operator support
- Functions operate *on* them

`string` OBJECTS

- Programs instantiate the string class
  - `string` object ≡ instance of the `string` class
- Manage their own memory and can grow automatically
- Supports many operators
- Objects are bound to functions; we think of the object *doing* the action

OBJECT-ORIENTED PREVIEW

#include <string> using namespace std;	#include <string>	Access string features
string s1;	std::string s1;	Creates an empty string object
string s2(“hello); string s2 = “hello”;	std::string s2(“hello”); std::string s2 = “hello”;	Copies a C-string to a string object
string s3(s2); string s3 = s2;	std::string s3(s2); std::string s3 = s2;	Copies one string to another
string* s4 = new string(“Hi”); string* s4 = new string(s2);	std::string* s4 = new(“Hi”); std::string* s4 = new(s2);	Creates a string object on the heap by copying a C-string or string object

BINDING AND CALLING FUNCTIONS: THE DOT AND ARROW OPERATORS

```
cout << s1 << endl;
cout << s2 << endl;
cout << s3 << endl;
cout << *s4 << endl;

cout << s1.size() << " " << s1.capacity() << endl;
cout << s2.length() << " " << s2.capacity() << endl;
cout << s3.size() << " " << s3.length() << endl;
cout << s4->length() << " " << s4->capacity() << endl;
```

string OPERATORS

Operator	Meaning	Example
=	Assignment	`s1 = s2`
+	Concatenation	`s = s1 + s2;`
+=	Concatenation with assignment	`s += s2;`
==	Equality	`if (s1 == s2) . . .`
!=	Inequality	`if (s1 != s2) . . .`
<, <=, >, and >=	Relational	`if (s1 < s2) . . .`
[] and at()	Character access	`char c = s1[i];` `char c = s1.at(i);`
<<	Output	`cout << s2;`