The C++ string class

- classes are more complex data types than arrays
- they allow you to store specific kinds of data (what kind depends on the specific class)
- they also allow you to run special routines on variables of that type: these are called methods, and are run using the syntax

```
variablename.methodname();
```

- like functions, methods can also use parameters & return values
- we'll consider classes in more detail later, but for now we'll focus on how we can use the C++ string class

<string> library

- inclusion of the <string> library allows us to use the string class
- we can declare variables of type string, e.g.

```
string s;
string anotherstr;
```

- we can assign text to strings, and use cin, cout

```
s = “hello there!”
cout << s;
cin >> s; // reads one 'word' of user input into s
anotherstr = s; // copy content between strings
```

string methods and operators

- the length method returns the # of chars

```
s = “hello”;
int x = s.length(); // assigns 5 to x
```

- the [] operator lets us look up characters by position

```
cout << s[1];  // outputs char 'e'
```

- + acts like concatenation in strings (+= also works)

```
s = “hello” + “ world”;   // stores “hello world”
```

- there are many many many other string methods

string comparisons

- we can compare strings using ==, <, <=, etc
- the comparison logic works off the ascii tables (like with the strcmp function we looked at earlier)

```
string s1, s2;
cin >> s1 >> s2;
if (s1 < s2) {
   s1 << " comes before " << s2 << endl;
} else if (s1 > s2) {
   s2 << " comes before " << s1 << endl;
} else {
   s1 << " is the same as " << s2 << endl;
}
```

getting the char array from string

- sometimes (e.g. when using cstring functions) we need to get the content of a string but in the form of a null terminated char array
- the c_str method provides this, e.g.

```
char text[SIZE];
string s = “blahblahblah”;
strcpy(text, s.c_str());
```