C-Strings (Character Arrays)

- Character array: an array whose components are of type `char`
- C-strings are null-terminated (`'\0'`) character arrays
- Example:
  - `'A'` is the character `A`
  - `"A"` is the C-string `A`
    - `"A"` represents two characters, `'A'` and `'\0'`

C-Strings (Character Arrays) (continued)

- Consider the statement

```
char name[16];
```

- Since C-strings are null terminated and `name` has 16 components, the largest string that it can store has 15 characters
- If you store a string of length, say 10 in `name`
  - The first 11 components of `name` are used and the last five are left unused

C-Strings (Character Arrays) (continued)

- The statement

```
char name[16] = "John";
```

declares an array `name` of length 16 and stores the C-string `"John"` in it

- The statement

```
char name[] = "John";
```

declares an array `name` of length 5 and stores the C-string `"John"` in it

C-Strings (Character Arrays) (continued)

TABLE 9-1 `strcpy`, `strcmp`, and `strlen` functions

Function	Effect
`strcpy(s1, s2)`	Copies the string `s2` into the string variable `s1` The length of `s1` should be at least as large as `s2`
`strcmp(s1, s2)`	Returns a value < 0 if `s1` is less than `s2` Returns 0 if `s1` and `s2` are the same Returns a value > 0 if `s1` is greater than `s2`
`strlen(s)`	Returns the length of the string `s`, excluding the null character

String Comparison

- C-strings are compared character by character using the collating sequence of the system
- If we are using the ASCII character set
  - `"Air" < "Boat"`
  - `"Air" < "An"`
  - `"Bill" < "Billy"`
  - `"Hello" < "hello"`

EXAMPLE 9-8

Suppose you have the following statements:

```
char studentName[21];
char myname[16];
char yourname[16];
```

The following statements show how string functions work:

Statement	Effect
`strcpy(myname, "John Robinson");`	`Myname = "John Robinson"`
`strlen("John Robinson");`	Returns 13, the length of the string `"John Robinson"`
`int len;` `len = strlen("Sunny Day");`	Stores 9 into `len`
`strcpy(yourname, "Lisa Miller");` `strcpy(studentName, yourname);`	`yourname = "Lisa Miller"` `studentName = "Lisa Miller"`
`strcmp("Bill", "Lisa");`	Returns a value < 0

`strcpy(yourname, "Kathy Brown");`
`strcpy(myname, "Mark G. Clark");`
`strcmp(myname, yourname);`
	`yourname = "Kathy Brown"`
	`myname = "Mark G. Clark"`
	Returns a value > 0

Reading and Writing Strings

- Most rules that apply to arrays apply to C-strings as well
- Aggregate operations, such as assignment and comparison, are not allowed on arrays
- Even the input/output of arrays is done component-wise
- The one place where C++ allows aggregate operations on arrays is the input and output of C-strings (that is, character arrays)

String Input

- `cin >> name;` stores the next input C-string into `name`
- To read strings with blanks, use `get`:

```
cin.get(str, m+1);
```

  - Stores the next `m` characters into `str` but the newline character is not stored in `str`
  - If the input string has fewer than `m` characters, the reading stops at the newline character

String Output

- `cout << name;` outputs the content of `name` on the screen
  - `<<` continues to write the contents of name until it finds the null character
  - If `name` does not contain the null character, then we will see strange output
    - `<<` continues to output data from memory adjacent to `name` until `'\0'` is found

Specifying Input/Output Files at Execution Time

- You can let the user specify the name of the input and/or output file at execution time:

```
ifstream infile;
ofstream outfile;

char fileName[51];     //assume that the file name is at most
                       //50 characters long

cout << "Enter the input file name: ";
cin >> fileName;

infile.open(fileName);    //open the input file
.
.
.

cout << "Enter the output file name: ";
cin >> fileName;

outfile.open(fileName);  //open the output file
```

`string` Type and Input/Output Files

- Argument to the function `open` must be a null-terminated string (a `C`-string)
- If we use a variable of type `string` to read the name of an I/O file, the value must first be converted to a `C`-string before calling `open`
- Syntax:

```
strVar.c_str()
```

where `strVar` is a variable of type `string`