THIRD EDITION

A First Book of C++

From Here to There

CHAPTER 10

Strings as Character Arrays

C-string Fundamentals

- C++ has two different ways of storing and manipulating strings
  - **String class:** presented in Chapter 7
  - **Character strings (C-strings):** using an array of characters that is terminated by a sentinel value (the escape sequence `'\0'`)
- Character strings can be manipulated using standard element-by-element array-processing techniques
  - `cstring` class introduced with latest ANSI/ISO standard

C-string Fundamentals (continued)

- **String literal (string):** a sequence of characters enclosed in double quotes

```
"This is a string"
```

- Strings stored as an array of characters terminated by a special end-of-string marker called the `NULL` character
  - This character is a sentinel marking the end of the string
  - The `NULL` character is represented by the escape sequence, `\0`

C-string Fundamentals (continued)

- Individual characters in a string array can be input, manipulated, or output using standard array-handling techniques
- Array-handling techniques can use either subscripts or pointers
- The end-of-string `NULL` character is useful for detecting the end of the string

C-string Input and Output

- Inputting and displaying string requires a standard library function or class method:
  - `cin` and `cout` (standard input and output streams)
  - String and character I/O functions (Table 10.1)
    - Requires the `iostream` header file
- Character input methods not the same as methods defined for the string class having the same name
- Character output methods are the same as for `string` class

C-string Input and Output (continued)

TABLE 10.1 Some Useful C++ I/O Functions (Supplied in the `iostream` Header File)

C++ Routine	Description	Example
`cin.getline(str, n, '\n')`	Omitting input from the keyboard	`cin.getline(str, 81,'\n');`
`cin.get()`	Obtain character input from the keyboard	`nextChar = cin.get();`
`cin.peek()`	Returns the next character from the input stream without removing it from the stream.	`nextKey = cin.peek();`
`cout.put(charExp)`	Place the character on the output stream	`cout.put('A');`
`cin.putback(charExp)`	Push a character back onto the input stream.	`cin.putback(ch);`
`cin.ignore(n, char)`	Ignore a maximum of the next `n` input characters, up to and including the occurrence of `char`. If no arguments are specified, ignore the next character on the input stream.	`cin.ignore(80,'\n');` `cin.ignore();`

C-string Input and Output (continued)

Program 10.1

```
#include <iostream>
using namespace std;

int main ()
{
  const int MAXCHARS   =  81;
  char message [MAXCHARS];  // an array of characters large
                            // enough storage for a complete line
  cout << "Enter a string:\n";

  cin.getline(message, MAXCHARS, '\n');

  cout << "The string just entered is:\n"
       << message << endl;

  cin.ignore();
  return 0;
}
```


C-string Input and Output (continued)

- Program 10.1 illustrates using `cin.getline()` and `cout` to input and output a string entered at the user's terminal
  - Sample run of Program 10.1:

```
Enter a string:
This is a test input of a string of characters.
The string just entered is:
This is a test input of a string of characters.
```


C-string Input and Output (continued)

- The `cin.getline()` method in Program 10.1 continuously accepts and stores characters into character array named `message`
  - Input continues until:
    - Either 80 characters are entered
    - The ENTER key is detected