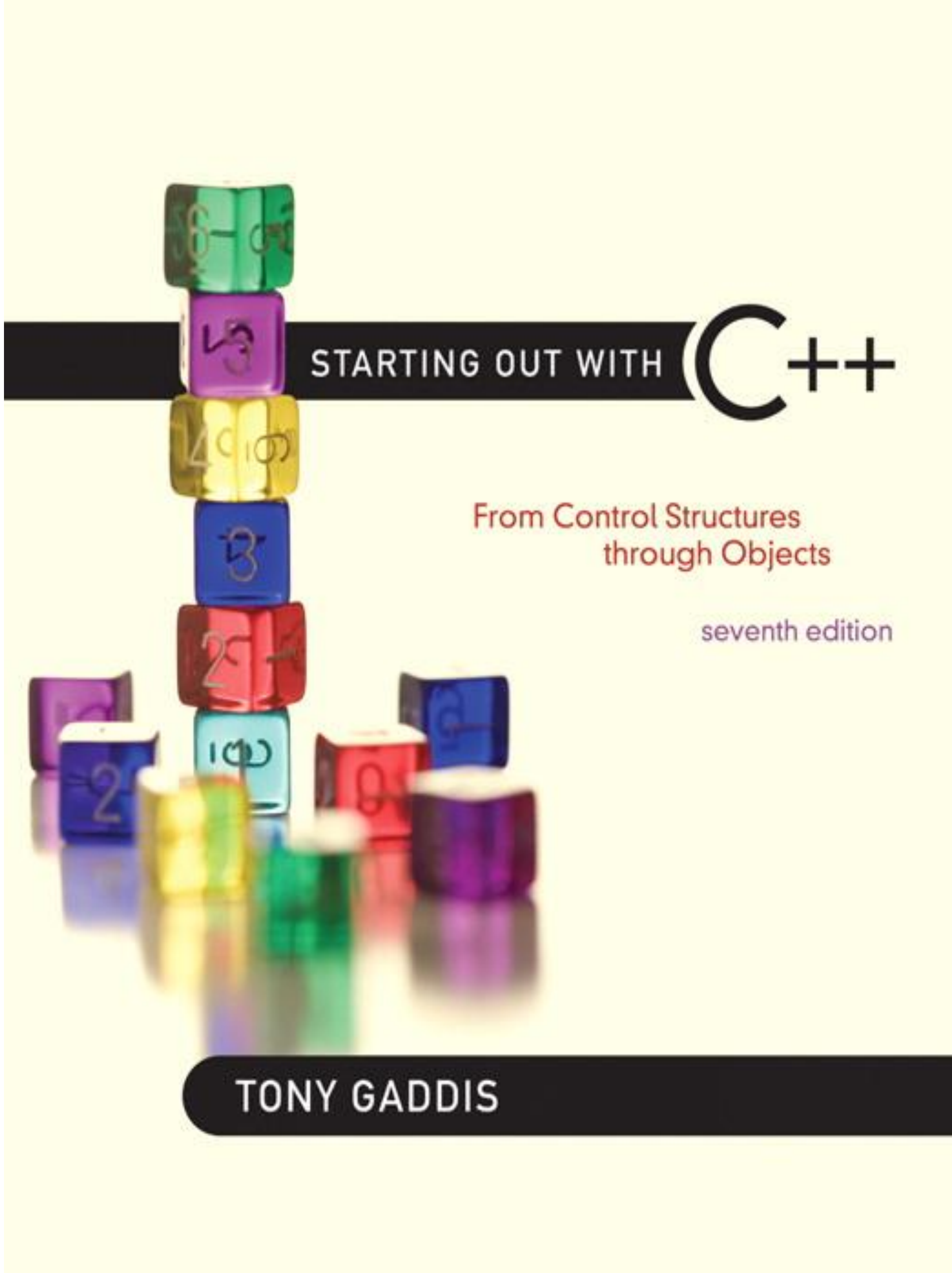
Chapter 10:

Characters, C-Strings, and More About the `string` Class



Addison-Wesley is an imprint of PEARSON

Character Testing

- require `cctype` header file

FUNCTION	MEANING
`isalpha`	`true` if arg. is a letter, `false` otherwise
`isalnum`	`true` if arg. is a letter or digit, `false` otherwise
`isdigit`	`true` if arg. is a digit 0-9, `false` otherwise
`islower`	`true` if arg. is lowercase letter, `false` otherwise
`isprint`	`true` if arg. is a printable character, `false` otherwise
`ispunct`	`true` if arg. is a punctuation character, `false` otherwise
`isupper`	`true` if arg. is an uppercase letter, `false` otherwise
`isspace`	`true` if arg. is a whitespace character, `false` otherwise

Character Case Conversion

- Require `cctype` header file
- Functions:

  `toupper`: if `char` argument is lowercase letter, return uppercase equivalent; otherwise, return input unchanged

```
char ch1 = 'H';
char ch2 = 'e';
char ch3 = '!';
cout << toupper(ch1);  // displays 'H'
cout << toupper(ch2);  // displays 'E'
cout << toupper(ch3);  // displays '!'
```

Character Case Conversion

- Functions:

  `tolower`: if `char` argument is uppercase letter, return lowercase equivalent; otherwise, return input unchanged

```
char ch1 = 'H';
char ch2 = 'e';
char ch3 = '!';
cout << tolower(ch1);  // displays 'h'
cout << tolower(ch2);  // displays 'e'
cout << tolower(ch3);  // displays '!'
```

C-Strings

- C-string: sequence of characters stored in adjacent memory locations and terminated by `NULL` character
- String literal (string constant): sequence of characters enclosed in double quotes " " :

```
"Hi there!"
```

H	i		t	h	e	r	e	!	\0

C-Strings

- Array of `char`s can be used to define storage for string:
  ```
  const int SIZE = 20;
  char city[SIZE];
  ```

- Leave room for `NULL` at end
- Can enter a value using `cin` or `>>`
  - Input is whitespace-terminated
  - No check to see if enough space
- For input containing whitespace, and to control amount of input, use `cin.getline()`

Library Functions for Working with C-Strings

Functions:

- `strlen(str)`: returns length of C-string `str`

```
char city[SIZE] = "Missoula";
cout << strlen(city); // prints 8
```

- `strcat(str1, str2)`: appends `str2` to the end of `str1`

```
char location[SIZE] = "Missoula, ";
char state[3] = "MT";
strcat(location, state);
// location now has "Missoula, MT"
```

Library Functions for Working with C-Strings

Functions:

– `strcpy(str1, str2)`: copies `str2` to `str1`

```
const int SIZE = 20;
char fname[SIZE] = "Maureen", name[SIZE];
strcpy(name, fname);
```

Note: `strcat` and `strcpy` perform no bounds checking to determine if there is enough space in receiving character array to hold the string it is being assigned.

C-string Inside a C-string

Function:

- `strstr(str1, str2)`: finds the first occurrence of `str2` in `str1`. Returns a pointer to match, or `NULL` if no match.

```
char river[] = "Wabash";
char word[] = "aba";
cout << strstr(state, word);
// displays "abash"
```

String/Numeric Conversion Functions

- require `cstdlib` header file

FUNCTION	PARAMETER	ACTION
`atoi`	C-string	converts C-string to an `int` value, returns the value
`atol`	C-string	converts C-string to a `long` value, returns the value
`atof`	C-string	converts C-string to a `double` value, returns the value
`itoa`	`int`, C-string, `int`	converts 1st `int` parameter to a C-string, stores it in 2nd parameter. 3rd parameter is base of converted value

String/Numeric Conversion Functions

```
int iNum;
long lNum;
double dNum;
char intChar[10];
iNum = atoi("1234"); // puts 1234 in iNum
lNum = atol("5678"); // puts 5678 in lNum
dNum = atof("35.7"); // puts 35.7 in dNum
itoa(iNum, intChar, 8); // puts the string
   // "2322" (base 8 for 1234$_{10}$) in intChar
```

String/Numeric Conversion Functions - Notes

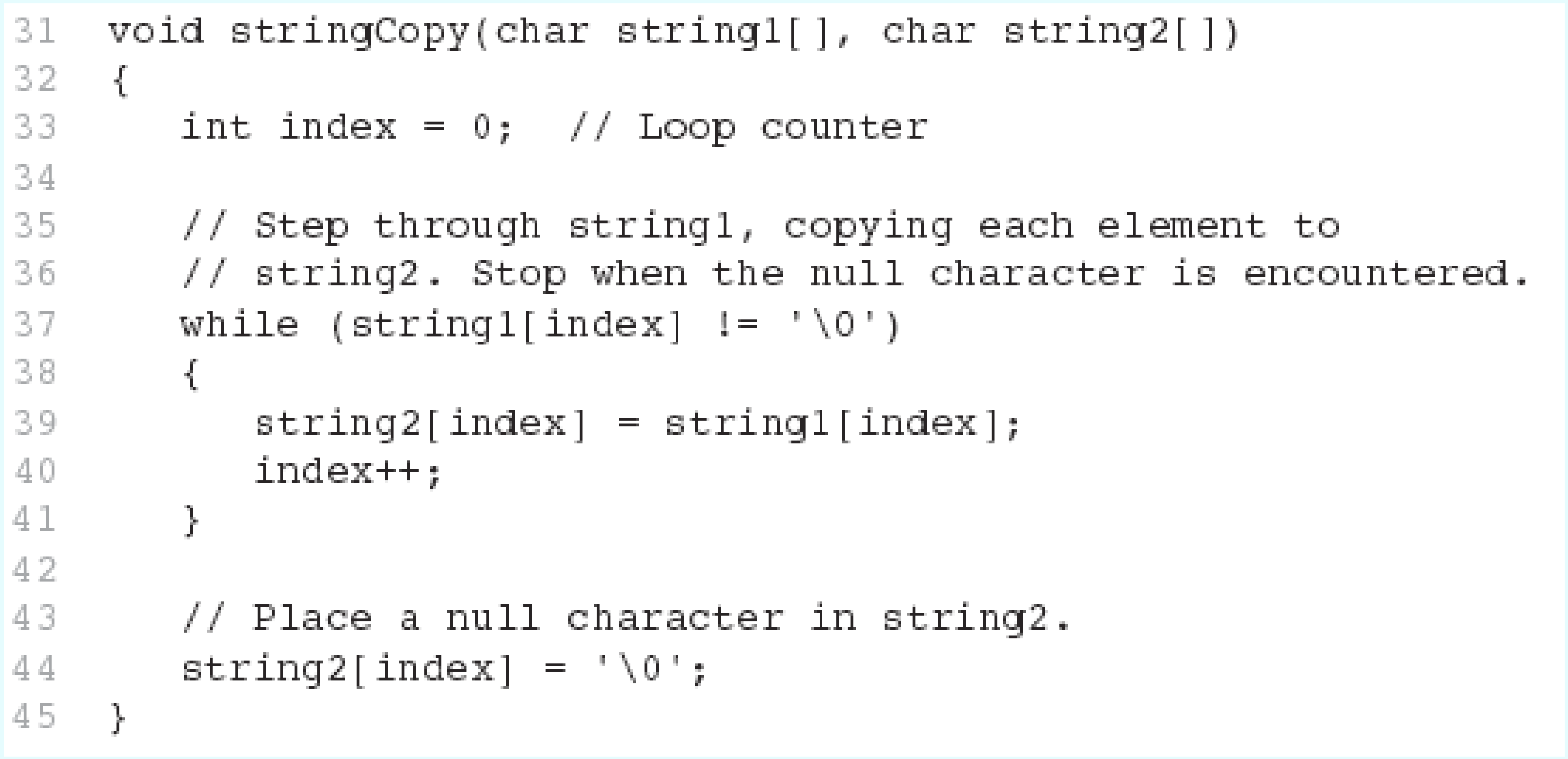
- if C-string contains non-digits, results are undefined
  - function may return result up to non-digit
  - function may return 0

- `itoa` does no bounds checking – make sure there is enough space to store the result

Writing Your Own C-String Handling Functions

- Designing C-String Handling Functions
  - can pass arrays or pointers to `char` arrays
  - Can perform bounds checking to ensure enough space for results
  - Can anticipate unexpected user input

From Program 10-9



```cpp
void stringCopy(char string1[], char string2[])
{
   int index = 0;  // Loop counter

   // Step through string1, copying each element to
   // string2. Stop when the null character is encountered.
   while (string1[index] != '\0')
   {
      string2[index] = string1[index];
      index++;
   }

   // Place a null character in string2.
   string2[index] = '\0';
}
```

The C++ `string` Class

- Special data type supports working with strings
- `#include <string>`
- Can define `string` variables in programs:

```
string firstName, lastName;
```

- Can receive values with assignment operator:

```
firstName = "George";
lastName = "Washington";
```

- Can be displayed via `cout`

```
cout << firstName << " " << lastName;
```

Input into a `string` Object

- Use `cin >>` to read an item into a string:

```
string firstName;
cout << "Enter your first name: ";
cin >> firstName;
```

Input into a `string` Object

- Use `getline` function to put a line of input, possibly including spaces, into a string:

```
string address;
cout << "Enter your address: ";
getline(cin,address);
```

`string` Comparison

- Can use relational operators directly to compare string objects:

```
string str1 = "George",
       str2 = "Georgia";
if (str1 < str2)
   cout << str1 << " is less than "
        << str2;
```

- Comparison is performed similar to `strcmp` function. Result is `true` or `false`

Other Definitions of C++ `string`s

Definition	Meaning
`string name;`	defines an empty string object
`string myname("Chris");`	defines a string and initializes it
`string yourname(myname);`	defines a string and initializes it
`string aname(myname, 3);`	defines a string and initializes it with first 3 characters of `myname`
`string verb(myname,3,2);`	defines a string and initializes it with 2 characters from `myname` starting at position 3
`string noname('A', 5);`	defines string and initializes it to 5 `'A'`s

`string` Operators

OPERATOR	MEANING
`>>`	extracts characters from stream up to whitespace, insert into string
`<<`	inserts string into stream
`=`	assigns string on right to string object on left
`+=`	appends string on right to end of contents on left
`+`	concatenates two strings
`[]`	references character in string using array notation
`>, >=, <, <=, ==, !=`	relational operators for string comparison. Return `true` or `false`

`string` Operators

```
string word1, phrase;
string word2 = " Dog";
cin >> word1; // user enters "Hot Tamale"
              // word1 has "Hot"
phrase = word1 + word2; // phrase has
                        // "Hot Dog"
phrase += " on a bun";
for (int i = 0; i < 16; i++)
     cout << phrase[i]; // displays
               // "Hot Dog on a bun"
```

`string` Member Functions

- Are behind many overloaded operators
- Categories:
  - assignment: `assign, copy, data`
  - modification: `append, clear, erase, insert, replace, swap`
  - space management: `capacity, empty, length, resize, size`
  - substrings: `find, substr`
  - comparison: `compare`
- See Table 10-7 for a list of functions

`string` Member Functions

```
string word1, word2, phrase;
cin >> word1;                // word1 is "Hot"
word2.assign(" Dog");
phrase.append(word1);
phrase.append(word2);   // phrase has "Hot Dog"
phrase.append(" with mustard relish", 13);
         // phrase has "Hot Dog with mustard"
phrase.insert(8, "on a bun ");
cout << phrase << endl; // displays
         // "Hot Dog on a bun with mustard"
```