Q: Do I have to use the names `argc` and `argv` for `main`'s parameters? [p. 302]

A: No. Using the names `argc` and `argv` is merely a convention, not a language requirement.

Q: I've seen `argv` declared as `**argv` instead of `*argv[]`. Is this legal?

A: Certainly. When declaring a parameter, writing `*a` is always the same as writing `a[]`, regardless of the type of `a`'s elements.

Q: We've seen how to set up an array whose elements are pointers to string literals. Are there any other applications for arrays of pointers?

A: Yes. Although we've focused on arrays of pointers to character strings, that's not the only application of arrays of pointers. We could just as easily have an array whose elements point to any type of data, whether in array form or not. Arrays of pointers are particularly useful in conjunction with dynamic storage allocation.

dynamic storage allocation ➤17.1

Exercises

Section 13.3

1. The following function calls supposedly write a single new-line character, but some are incorrect. Identify which calls don't work and explain why.

```
(a) printf("%c", '\n');      (g) putchar('\n');
(b) printf("%c", "\n");      (h) putchar("\n");
(c) printf("%s", '\n');      (i) puts('\n');
(d) printf("%s", "\n");      (j) puts("\n");
(e) printf('\n');            (k) puts("");
(f) printf("\n");
```

W 2. Suppose that `p` has been declared as follows:

```
char *p = "abc";
```

Which of the following function calls are legal? Show the output produced by each legal call, and explain why the others are illegal.

```
(a) putchar(p);
(b) putchar(*p);
(c) puts(p);
(d) puts(*p);
```

*3. Suppose that we call `scanf` as follows:

```
scanf("%d%s%d", &i, s, &j);
```

If the user enters `12abc34 56def78`, what will be the values of `i`, `s`, and `j` after the call? (Assume that `i` and `j` are `int` variables and `s` is an array of characters.)

W 4. Modify the `read_line` function in each of the following ways:

(a) Have it skip white space before beginning to store input characters.

(b) Have it stop reading at the first white-space character. *Hint:* To determine whether or not a character is white space, call the `isspace` function.

isspace function ➤23.5

(c) Have it stop reading at the first new-line character, then store the new-line character in the string.
(d) Have it leave behind characters that it doesn't have room to store.

Section 13.4

5. (a) Write a function named `capitalize` that capitalizes all letters in its argument. The argument will be a null-terminated string containing arbitrary characters, not just letters. Use array subscripting to access the characters in the string. *Hint:* Use the `toupper` function to convert each character to upper-case.

`toupper` function ➤23.5

(b) Rewrite the `capitalize` function, this time using pointer arithmetic to access the characters in the string.

W 6. Write a function named `censor` that modifies a string by replacing every occurrence of `foo` by `xxx`. For example, the string `"food fool"` would become `"xxxd xxxl"`. Make the function as short as possible without sacrificing clarity.

Section 13.5

7. Suppose that `str` is an array of characters. Which one of the following statements is not equivalent to the other three?

(a) `*str = 0;`
(b) `str[0] = '\0';`
(c) `strcpy(str, "");`
(d) `strcat(str, "");`

W *8. What will be the value of the string `str` after the following statements have been executed?

```
strcpy(str, "tire-bouchon");
strcpy(&str[4], "d-or-wi");
strcat(str, "red?");
```

9. What will be the value of the string `s1` after the following statements have been executed?

```
strcpy(s1, "computer");
strcpy(s2, "science");
if (strcmp(s1, s2) < 0)
  strcat(s1, s2);
else
  strcat(s2, s1);
s1[strlen(s1)-6] = '\0';
```

W 10. The following function supposedly creates an identical copy of a string. What's wrong with the function?

```
char *duplicate(const char *p)
{
  char *q;

  strcpy(q, p);
  return q;
}
```

11. The Q&A section at the end of this chapter shows how the `strcmp` function might be written using array subscripting. Modify the function to use pointer arithmetic instead.

12. Write the following function:

```
void get_extension(const char *file_name, char *extension);
```

`file_name` points to a string containing a file name. The function should store the extension on the file name in the string pointed to by `extension`. For example, if the file name is `"memo.txt"`, the function will store `"txt"` in the string pointed to by `extension`. If the file name doesn't have an extension, the function should store an empty string (a single null character) in the string pointed to by `extension`. Keep the function as simple as possible by having it use the `strlen` and `strcpy` functions.

13. Write the following function:

```
void build_index_url(const char *domain, char *index_url);
```

`domain` points to a string containing an Internet domain, such as `"knking.com"`. The function should add `"http://www."` to the beginning of this string and `"/index.html"` to the end of the string, storing the result in the string pointed to by `index_url`. (In this example, the result will be `"http://www.knking.com/index.html"`.) You may assume that `index_url` points to a variable that is long enought to hold the resulting string. Keep the function as simple as possible by having it use the `strcat` and `strcpy` functions.

Section 13.6

*14. What does the following program print?

```
#include <stdio.h>

int main(void)
{
  char s[] = "Hsjodi", *p;

  for (p = s; *p; p++)
    --*p;
  puts(s);
  return 0;
}
```

W *15. Let `f` be the following function:

```
int f(char *s, char *t)
{
  char *p1, *p2;

  for (p1 = s; *p1; p1++) {
    for (p2 = t; *p2; p2++)
      if (*p1 == *p2) break;
    if (*p2 == '\0') break;
  }
  return p1 - s;
}
```

(a) What is the value of `f("abcd", "babc")`?
(b) What is the value of `f("abcd", "bcd")`?
(c) In general, what value does `f` return when passed two strings `s` and `t`?

W 16. Use the techniques of Section 13.6 to condense the `count_spaces` function of Section 13.4. In particular, replace the `for` statement by a `while` loop.

17. Write the following function:

```
bool test_extension(const char *file_name,
                    const char *extension);
```

`file_name` points to a string containing a file name. The function should return `true` if the file's extension matches the string pointed to by `extension`, ignoring the case of letters. For example, the call `test_extension("memo.txt", "TXT")` would return `true`. Incorporate the "search for the end of a string" idiom into your function. *Hint:* Use the `toupper` function to convert characters to upper-case before comparing them.

`toupper` function ➤23.5

18. Write the following function:

```
void remove_filename(char *url);
```

`url` points to a string containing a URL (Uniform Resource Locator) that ends with a file name (such as `"http://www.knking.com/index.html"`). The function should modify the string by removing the file name and the preceding slash. (In this example, the result will be `"http://www.knking.com"`.) Incorporate the "search for the end of a string" idiom into your function. *Hint:* Have the function replace the last slash in the string by a null character.

Programming Projects

W 1. Write a program that finds the "smallest" and "largest" in a series of words. After the user enters the words, the program will determine which words would come first and last if the words were listed in dictionary order. The program must stop accepting input when the user enters a four-letter word. Assume that no word is more than 20 letters long. An interactive session with the program might look like this:

```
Enter word: dog
Enter word: zebra
Enter word: rabbit
Enter word: catfish
Enter word: walrus
Enter word: cat
Enter word: fish

Smallest word: cat
Largest word: zebra
```

Hint: Use two strings named `smallest_word` and `largest_word` to keep track of the "smallest" and "largest" words entered so far. Each time the user enters a new word, use `strcmp` to compare it with `smallest_word`; if the new word is "smaller," use `strcpy` to save it in `smallest_word`. Do a similar comparison with `largest_word`. Use `strlen` to determine when the user has entered a four-letter word.

2. Improve the `remind.c` program of Section 13.5 in the following ways:
   (a) Have the program print an error message and ignore a reminder if the corresponding day is negative or larger than 31. *Hint:* Use the `continue` statement.
   (b) Allow the user to enter a day, a 24-hour time, and a reminder. The printed reminder list should be sorted first by day, then by time. (The original program allows the user to enter a time, but it's treated as part of the reminder.)
   (c) Have the program print a one-*year* reminder list. Require the user to enter days in the form *month/day*.

3. Modify the `deal.c` program of Section 8.2 so that it prints the full names of the cards it deals:

```
Enter number of cards in hand: 5
Your hand:
Seven of clubs
Two of spades
Five of diamonds
Ace of spades
Two of hearts
```

Hint: Replace `rank_code` and `suit_code` by arrays containing pointers to strings.

W 4. Write a program named `reverse.c` that echoes its command-line arguments in reverse order. Running the program by typing

```
reverse void and null
```

should produce the following output:

```
null and void
```

5. Write a program named `sum.c` that adds up its command-line arguments, which are assumed to be integers. Running the program by typing

```
sum 8 24 62
```

should produce the following output:

```
Total: 94
```

atoi function ➤26.2

Hint: Use the `atoi` function to convert each command-line argument from string form to integer form.

W 6. Improve the `planet.c` program of Section 13.7 by having it ignore case when comparing command-line arguments with strings in the `planets` array.

7. Modify Programming Project 11 from Chapter 5 so that it uses arrays containing pointers to strings instead of `switch` statements. For example, instead of using a `switch` statement to print the word for the first digit, use the digit as an index into an array that contains the strings `"twenty"`, `"thirty"`, and so forth.

8. Modify Programming Project 5 from Chapter 7 so that it includes the following function:

```
int compute_scrabble_value(const char *word);
```

The function returns the SCRABBLE value of the string pointed to by `word`.

9. Modify Programming Project 10 from Chapter 7 so that it includes the following function:

```
int compute_vowel_count(const char *sentence);
```

The function returns the number of vowels in the string pointed to by the `sentence` parameter.

10. Modify Programming Project 11 from Chapter 7 so that it includes the following function:

```
void reverse_name(char *name);
```

The function expects `name` to point to a string containing a first name followed by a last name. It modifies the string so that the last name comes first, followed by a comma, a space, the first initial, and a period. The original string may contain extra spaces before the first name, between the first and last names, and after the last name.

11. Modify Programming Project 13 from Chapter 7 so that it includes the following function:

```
double compute_average_word_length(const char *sentence);
```

The function returns the average length of the words in the string pointed to by `sentence`.

12. Modify Programming Project 14 from Chapter 8 so that it stores the words in a two-dimensional `char` array as it reads the sentence, with each row of the array storing a single word. Assume that the sentence contains no more than 30 words and no word is more than 20 characters long. Be sure to store a null character at the end of each word so that it can be treated as a string.

13. Modify Programming Project 15 from Chapter 8 so that it includes the following function:

    ```
    void encrypt(char *message, int shift);
    ```

    The function expects `message` to point to a string containing the message to be encrypted; `shift` represents the amount by which each letter in the message is to be shifted.

14. Modify Programming Project 16 from Chapter 8 so that it includes the following function:

    ```
    bool are_anagrams(const char *word1, const char *word2);
    ```

    The function returns `true` if the strings pointed to by `word1` and `word2` are anagrams.

15. Modify Programming Project 6 from Chapter 10 so that it includes the following function:

    ```
    int evaluate_RPN_expression(const char *expression);
    ```

    The function returns the value of the RPN expression pointed to by `expression`.

16. Modify Programming Project 1 from Chapter 12 so that it includes the following function:

    ```
    void reverse(char *message);
    ```

    The function reverses the string pointed to by `message`. *Hint:* Use two pointers, one initially pointing to the first character of the string and the other initially pointing to the last character. Have the function reverse these characters and then move the pointers toward each other, repeating the process until the pointers meet.

17. Modify Programming Project 2 from Chapter 12 so that it includes the following function:

    ```
    bool is_palindrome(const char *message);
    ```

    The function returns `true` if the string pointed to by `message` is a palindrome.

18. Write a program that accepts a date from the user in the form *mm/dd/yyyy* and then displays it in the form *month dd, yyyy*, where *month* is the name of the month:

    ```
    Enter a date (mm/dd/yyyy): 2/17/2011
    You entered the date February 17, 2011
    ```

    Store the month names in an array that contains pointers to strings.