For the exclusive use of adopters of the book C++ How to Program, 5th Edition,
by Deitel and Deitel. ISBN 0-13-185757-6.

18

Class `string` and String Stream Processing

The difference between the almost-right word and the right word is really a large matter — it's the difference between the lightning bug and the lightning.
—Mark Twain

I have made this letter longer than usual, because I lack the time to make it short.
—Blaise Pascal

Mum's the word.
—Miguel de Cervantes

Suit the action to the word, the word to the action; with this special observance, that you o'erstep not the modesty of nature.
—William Shakespeare

OBJECTIVES

In this chapter you will learn:

- To use class `string` from the C++ standard library to treat strings as full-fledged objects.
- To assign, concatenate, compare, search and swap `string`s.
- To determine `string` characteristics.
- To find, replace and insert characters in a `string`.
- To convert `string`s to C-style strings and vice versa.
- To use `string` iterators.
- To perform input from and output to `string`s in memory.

Self-Review Exercises

18.1 Fill in the blanks in each of the following:

a) Header _____ must be included for class `string`.

ANS: `<string>`.

b) Class `string` belongs to the _____ `namespace`.

ANS: `std`.

c) Function _____ deletes characters from a `string`.

ANS: `erase`.

d) Function _____ finds the first occurrence of any character from a `string`.

ANS: `find_first_of`.

18.2 State which of the following statements are *true* and which are *false*. If a statement is *false*, explain why.

a) Concatenation of `string` objects can be performed with the addition assignment operator, `+=`.

ANS: True.

b) Characters within a `string` begin at index `0`.

ANS: True.

c) The assignment operator, `=`, copies a `string`.

ANS: True.

d) A C-style string is a `string` object.

ANS: False. A `string` is an object that provides many different services. A C-style string does not provide any services. C-style strings are null terminated; `string`s are not necessarily null terminated. C-style strings are pointers and `string`s are not.

18.3 Find the error(s) in each of the following, and explain how to correct it (them):

a)
```
string string1( 28 ); // construct string1
string string2( 'z' ); // construct string2
```

ANS: Constructors for class `string` do not exist for integer and character arguments. Other valid constructors should be used—converting the arguments to `string`s if need be.

b)
```
// assume std namespace is known
const char *ptr = name.data(); // name is "joe bob"
ptr[ 3 ] = '-';
cout << ptr << endl;
```

ANS: Function `data` does not add a null terminator. Also, the code attempts to modify a `const char`. Replace all of the lines with the code:

```
cout << name.substr( 0, 3 ) + "-" + name.substr( 4 ) << endl;
```

Exercises

18.4 Fill in the blanks in each of the following:

a) Class `string` member functions _____ and _____ convert `string`s to C-style strings.

ANS: `data`, `c_str`, `copy`

b) Class `string` member function _____ is used for assignment.

ANS: `assign`

c) _____ is the return type of function `rbegin`.

ANS: `string::reverse_iterator`

d) Class `string` member function _____ is used to retrieve a substring.

ANS: `substr`

18.5 State which of the following statements are *true* and which are *false*. If a statement is *false*, explain why.

a) `strings` are always null terminated.

ANS: False. `strings` are not necessarily null terminated.

b) Class `string` member function `max_size` returns the maximum size for a `string`.

ANS: True.

c) Class `string` member function `at` can throw an `out_of_range` exception.

ANS: True.

d) Class `string` member function `begin` returns an `iterator`.

ANS: True (`string::iterator` is more precise).

18.6 Find any errors in the following and explain how to correct them:

a)
```
std::cout << s.data() << std::endl; // s is "hello"
```

ANS: The array returned by `data` is not null terminated.

b)
```
erase( s.rfind( "x" ), 1 ); // s is "xenon"
```

ANS: Function `erase` is a `string` class member function (i.e., `erase` must be called by an object of type `string`).

c)
```
string& foo()
{
   string s( "Hello" );
   ...    // other statements
   return;
} // end function foo
```

ANS: A value is not being returned from the function (i.e., the `return` statement should be `return s;`). The `return` type should be `string` not `string&`—reference returns are dangerous

18.7 (*Simple Encryption*) Some information on the Internet may be encrypted with a simple algorithm known as "rot13," which rotates each character by 13 positions in the alphabet. Thus, `'a'` corresponds to `'n'`, and `'x'` corresponds to `'k'`. rot13 is an example of **symmetric key encryption**. With symmetric key encryption, both the encrypter and decrypter use the same key.

a) Write a program that encrypts a message using rot13.

b) Write a program that decrypts the scrambled message using 13 as the key.

c) After writing the programs of part (a) and part (b), briefly answer the following question: If you did not know the key for part (b), how difficult do you think it would be to break the code? What if you had access to substantial computing power (e.g., supercomputers)? In Exercise 18.26 we ask you to write a program to accomplish this.

ANS:

```
1  // Exercise 18.7 Part A Solution: Ex18_07.cpp
2  // When solving Part B of this exercise, you might find it more
3  // convenient to only use uppercase letters for you input.
4  #include <iostream>
5  using std::cin;
6  using std::cout;
7  using std::endl;
8
9  #include <string>
10 using std::string;
11 using std::getline;
12
13 int main()
```

```
{
   string m; // to store input
   int key = 13; // Our key for encryption

   cout << "Enter a string: ";
   getline( cin, m );

   string::iterator mi = m.begin(); // using function begin

   // loop through the string
   while ( mi != m.end() )
   {
      *mi += key;
      mi++;
   } // end while

   cout << "\nEncypted string is: " << m << endl;
   return 0; // indicates successful termination
} // end main
```

```
Enter a string: JAMES BOND IS 007

Encypted string is: WNZR`-O\[Q-V`-==D
```

```
// Exercise 18.7 Part B Solution: Ex18_07.cpp
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

#include <string>
using std::string;
using std::getline;

int main()
{
   string m; // to store input
   int key = 13; // Our key for decryption

   cout << "Enter encrypted string: ";
   getline( cin, m );

   // define an iterator
   string::iterator mi = m.begin();

   // loop through the string
   while ( mi != m.end() )
   {
      *mi -= key;
      mi++;
   } // end while

   cout << "\nDecypted string is: " << m << endl;
```

```
30      return 0; // indicates successful termination
31   } // end main
```

```
Enter encrypted string: WNZR`-O\[Q-V`-==D

Decypted string is: JAMES BOND IS 007
```

18.8 Write a program using iterators that demonstrates the use of functions `rbegin` and `rend`.
ANS:

```
1   // Exercise 18.8 Solution: Ex18_08.cpp
2   // Program demonstrates rend and rbegin.
3   #include <iostream>
4   using std::cout;
5   using std::endl;
6
7   #include <string>
8   using std::string;
9
10  int main()
11  {
12     string s( "abcdefghijklmnopqrstuvwxyz" ); // declare string s
13
14     // re is set to the end of the reverse sequence of s
15     string::reverse_iterator re = s.rend();
16
17     // rb is set to the beginning of the reverse sequence of s
18     string::reverse_iterator rb = s.rbegin();
19
20     cout << "Using rend() string is: ";
21
22     // print from the end of the reversed string to the beginning
23     while ( re >= s.rbegin() )
24     {
25        cout << *re;
26        re--;
27     } // end while
28
29     cout << "\nUsing rbegin() string is: ";
30
31     // print from the beginning of the reversed string
32     while ( rb != s.rend() )
33     {
34        cout << *rb;
35        rb++;
36     } // end while
37
38     cout << endl;
39
40     return 0; // indicates successful termination
41  } // end main
```

```
Using rend() string is: ²abcdefghijklmnopqrstuvwxyz
Using rbegin() string is: zyxwvutsrqponmlkjihgfedcba
```

18.9 Write a program that reads in several `string`s and prints only those ending in "`r`" or "`ay`". Only lowercase letters should be considered.

ANS:

```
1  // Exercise 18.9 Solution: Ex18_09.cpp
2  // Program determines if string ends in 'r' or "ay".
3  #include <iostream>
4  using std::cin;
5  using std::cout;
6  using std::endl;
7
8  #include <string>
9  using std::string;
10
11 int main()
12 {
13    string s[ 5 ]; // declare a string array
14
15    // loop to get user input
16    for ( int i = 0; i < 5; i++ )
17    {
18       cout << "Enter a word: ";
19       cin >> s[ i ];
20    } // end for
21
22    // use function rfind to find occurrences of "ay" or 'r'
23    for ( int j = 0; j < 5; j++ )
24    {
25       // does the word end in "ay" or 'y'?
26       if ( ( ( s[ j ].rfind( "ay" ) == s[ j ].length() - 2 ) )
27           || ( s[ j ].rfind( "r" ) == s[ j ].length() - 1 ) )
28          cout << s[ j ] << endl; // if match, display it
29    } // end for
30
31    return 0; // indicates successful termination
32 } // end main
```

```
Enter a word: bicycle
Enter a word: car
Enter a word: tree
Enter a word: canary
Enter a word: iron
car
```

18.10 Write a program that demonstrates passing a `string` both by reference and by value.
ANS:

```
1  // Exercise 18.10 Solution: Ex18_10.cpp
2  // Program passes a string by value and
3  // passes a string by reference.
4  #include <iostream>
5  using std::cout;
6  using std::endl;
7
8  #include <string>
9  using std::string;
10
11 // prototypes
12 void byValue( string );
13 void byReference( string& );
14
15 int main()
16 {
17    string s = "Standard C++ draft standard";
18    cout << "Original string: " << s;
19
20    // call to function byValue
21    byValue( s );
22    cout << "\nAfter calling byValue: " << s;
23
24    // call to function byReference
25    byReference( s );
26    cout << "\nAfter calling byReference: " << s << endl;
27
28    return 0; // indicates successful termination
29 } // end main
30
31 // demonstrates passing by value
32 void byValue( string s )
33 {
34    // call function erase to take out 4 characters from the string
35    s.erase( 0, 4 );
36 } // end function byValue
37
38 // demonstrates passing by reference
39 void byReference( string& sRef )
40 {
41    // erasing 9 characters from the string passed in
42    sRef.erase( 0, 9 );
43 } // end function byReference
```

```
Original string: Standard C++ draft standard
After calling byValue: Standard C++ draft standard
After calling byReference: C++ draft standard
```

18.11 Write a program that separately inputs a first name and a last name and concatenates the two into a new `string`.

ANS:

```
// Exercise 18.11 Solution: Ex18_11.cpp
// Program reads a first name and last name and concatenates the two.
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

#include <string>
using std::string;

int main()
{
   // declare two strings
   string first;
   string last;

   cout << "Enter first name: ";
   cin >> first;

   cout << "Enter last name: ";
   cin >> last;

   // use function append to insert space and string last
   first.append( " " ).append( last );
   cout << "The full name is: " << first << endl;
   return 0; // indicates successful termination
} // end main
```

```
Enter first name: John
Enter last name: Green
The full name is: John Green
```

18.12 Write a program that plays the game of Hangman. The program should pick a word (which is either coded directly into the program or read from a text file) and display the following:

```
Guess the word:    XXXXXX
```

Each `X` represents a letter. The user tries to guess the letters in the word. The appropriate response `yes` or `no` should be displayed after each guess. After each incorrect guess, display the diagram with another body part filled. After seven incorrect guesses, the user should be hanged. The display should look as follows:

```
 O
/|\
 |
/ \
```

After each guess, display all user guesses. If the user guesses the word correctly, the program should display

```
Congratulations!!! You guessed my word. Play again? yes/no
```

ANS:

```
// Exercise 18.12 Solution: Ex18_12.cpp
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

#include <string>
using std::string;

#include <iomanip>
using std::setw;

int main()
{
   string response; // "yes"/"no" input from user
   int w = 0; // index for current word
   const int WORDS = 4; // total number of words

   // loop will construct all necessary variables and begin game
   do
   {
      const char body[] = " o/|\\|/\\";  // body parts

      string words[ WORDS ] = {
         "MACAW", "SADDLE", "TOASTER", "XENOCIDE" };
      string xword( words[ w ].length(), '?' ); // masked display

      string::iterator i;
      string::iterator ix = xword.begin();

      char letters[ 26 ] = { '\0' }; // letters guessed

      int n = 0; // index variable
      int xcount = xword.length();
      bool found = false;
      bool solved = false;
      int offset = 0;
      int bodyCount = 0;
      bool hung = false;

      cout << "Guess the word: ";

      // display the word in Xs
      for ( unsigned loop = 0; loop < words[ w ].length(); loop++ )
         cout << "X";

      // loop to begin game
      do
```

```
49          {
50             cout << "\n\nGuess a letter (case does not matter): "
51                << xword << "\n?";
52
53             char temp;
54             cin >> temp; // letter guessed
55
56             if ( !isalpha( temp ) ) // validate for letters
57             {
58                cout << "\nLETTERS ONLY PLEASE\n";
59                continue; // next iteration of do/while
60             } // end if
61
62             letters[ n ] = toupper( temp ); // convert to uppercase
63
64             // seach word for letters
65             i = words[ w ].begin(); // initialize iterator to beginning
66             found = false; // assume letter is not found in word
67             offset = 0; // initial position set to 0
68
69             // replace letter in mask string in all the necessary
70             // places. decrement count of characters masked such
71             // that we know when word is solved.
72             while ( i != words[ w ].end() )
73             {
74                if ( *i == letters[ n ] )
75                {
76                   *( ix + offset ) = *i;
77                   found = true;
78
79                   if ( --xcount == 0 )
80                      solved = true;
81                } // end if
82
83                i++;
84                offset++;
85             } // end while
86
87             if ( !found ) // if the letter was not found
88                bodyCount++; // increment our count of incorrect guesses.
89
90             bool newline = false;
91
92             // graphically draw the pieces of the body
93             // based upon the number of incorrect answers.
94             for ( int q = 1; q <= bodyCount; q++ )
95             {
96                if ( q == 1 || q == 5 || q == 7 )
97                {
98                   newline = true;
99                   cout << body[ 0 ]; // output space
100               } // end if
101               else if ( q == 4 )
102                  newline = true;
103               else
```

```
104                newline = false;
105
106             cout << body[ q ];
107
108             if ( newline )
109                cout << '\n';
110          } // end for
111
112          // test to see if guesses were exceeded.
113          if ( bodyCount == 7 )
114          {
115             cout << "\n\n...GAME OVER...\n";
116             hung = true;
117             break;
118          } // end if
119
120          cout << "\nYour guesses:\n";
121
122          // display all guesses. note we did not provide
123          // the code that would politely refuse duplicates
124          for ( int k = 0; k <= n; k++ )
125             cout << setw( 2 ) << letters[ k ];
126
127          n++;
128       } while ( !solved ); // end do...while
129
130       cout << "\n\nWord: " << words[ w ] << "\n\n";
131
132       if ( !hung )
133          cout << "\nCongratulations!!! You guessed "
134             << "my word.\n";
135
136       // if we are out of words, then time to exit loop
137       if ( w++ >= WORDS )
138          break;
139
140       // prompt user if they want to play again
141       cout << "Play again (yes/no)? ";
142       cin >> response;
143
144    } while ( !response.compare( "yes" ) ); // end do...while
145
146    cout << "\nThank you for playing hangman." << endl;
147    return 0; // indicates successful termination
148 } // end main
```

```
Guess the word: XXXXX

Guess a letter (case does not matter): ?????
?a

Your guesses:
 A

Guess a letter (case does not matter): ?A?A?
?e
 o

Your guesses:
 A E

Guess a letter (case does not matter): ?A?A?
?i
 o
/
Your guesses:
 A E I

Guess a letter (case does not matter): ?A?A?
?o
 o
/|
Your guesses:
 A E I O

Guess a letter (case does not matter): ?A?A?
?u
 o
/|\

Your guesses:
 A E I O U

Guess a letter (case does not matter): ?A?A?
?w
 o
/|\

Your guesses:
 A E I O U W

Guess a letter (case does not matter): ?A?AW
?x
 o
/|\
 |

Your guesses:
 A E I O U W X
```

```
Guess a letter (case does not matter): ?A?AW
?k
 o
/|\
 |
/
Your guesses:
 A E I O U W X K

Guess a letter (case does not matter): ?A?AW
?l
 o
/|\
 |
/ \


...GAME OVER...


Word: MACAW

Play again (yes/no)? no

Thank you for playing hangman.
```

18.13 Write a program that inputs a `string` and prints the `string` backward. Convert all uppercase characters to lowercase and all lowercase characters to uppercase.

ANS:

```
1  // Exercise 18.13 Solution: Ex08_13.cpp
2  #include <iostream>
3  using std::cin;
4  using std::cout;
5  using std::endl;
6
7  #include <string>
8  using std::getline;
9  using std::string;
10
11 int main()
12 {
13    string s;
14
15    cout << "Enter a string: ";
16    getline( cin, s, '\n' );
17
18    // r is set to the beginning of the reverse sequence from s
19    string::reverse_iterator r = s.rbegin();
20
21    // loop till the reversed end if reached
22    while ( r != s.rend() )
23    {
24       // convert all characters to its opposites
```

```
      *r = ( isupper( *r ) ? tolower( *r ): toupper( *r ) );
      cout << *( r++ ); // next character
   } // end loop

   cout << endl;
   return 0; // indicates successful termination
} // end main
```

```
Enter a string: The sinking of HMS Titanic
CINATIt smh FO GNIKNIS EHt
```

18.14 Write a program that uses the comparison capabilities introduced in this chapter to alphabetize a series of animal names. Only uppercase letters should be used for the comparisons.

ANS:

```
// Exercise 18.14 Solution: Ex18_14.cpp
// NOTE: The problem description should have asked
// the programmer to use a quicksort.
#include <iostream>
using std::cout;
using std::endl;

#include <string>
using std::string;

// prototypes
void output( const string *, const int );
void quickSort( string [], int, int );

int main()
{
   const int SIZE = 19;

   // an array of strings containing animal names
   string animals[] = { "Macaw", "Lion", "Tiger", "Bear", "Toucan",
      "Zebra", "Puma", "Cat", "Yak", "Boar",  "Fox", "Ferret",
      "Crocodile", "Alligator", "Elk", "Ox",  "Horse", "Eagle", "Hawk" };

   cout << "before:";
   output( animals, SIZE ); // call output to display string array
   quickSort( animals, 0, SIZE ); // sort them in order

   cout << "\nafter:";
   output( animals, SIZE ); // call output to display array of animal
   return 0; // indicates successful termination
} // end main

// function to print out each string in the array
void output( const string * const ani, const int length )
{
   // loop through the array with the given length
   for ( int j = 0; j < length; ++j )
      cout << ( j % 10 ? ' ': '\n' ) << ani[ j ];
```

```
39
40       cout << endl;
41    } // end function output
42
43    // function to sort the array
44    void quickSort( string a[], int first, int last )
45    {
46       // call function partition
47       int partition( string [], int, int );
48       int currentLocation;
49
50       if ( first >= last )
51          return;
52
53       currentLocation = partition( a, first, last );
54
55       // recursive calls to quickSort to continue the search
56       quickSort( a, first, currentLocation - 1 );
57       quickSort( a, currentLocation + 1, last );
58    } // end function quickSort
59
60    int partition( string b[], int left, int right )
61    {
62       int pos = left;
63
64       // while loop
65       while ( true )
66       {
67          // move through the array from left to right
68          while ( b[ pos ] <= b[ right ] && pos != right )
69             right--;
70
71          // if the right is reached, return that position
72          if ( pos == right )
73             return pos;
74
75          // if the element from the left is greater, swap the positions
76          if ( b[ pos ] > b[ right ] )
77          {
78             b[ pos ].swap( b[ right ] );
79             pos = right;
80          } // end if
81
82          // compare from the beginning to the pos index
83          while ( b[ left ] <= b[ pos ] && pos != left )
84             left++;
85
86          if ( pos == left )
87             return pos;
88
89          if ( b[ left ] > b[ pos ] )
90          {
91             b[ pos ].swap( b[ left ] );
92             pos = left;
93          } // end if
```

```
94      } // end while
95   } // end function partition
```

```
before:
Macaw Lion Tiger Bear Toucan Zebra Puma Cat Yak Boar
Fox Ferret Crocodile Alligator Elk Ox Horse Eagle Hawk

after:
Alligator Bear Boar Cat Crocodile Eagle Elk Ferret Fox Hawk
Horse Lion Macaw Ox Puma Tiger Toucan Yak Zebra
```

18.15 Write a program that creates a cryptogram out of a `string`. A cryptogram is a message or word in which each letter is replaced with another letter. For example the `string`

```
The bird was named squawk
```

might be scrambled to form

```
cin vrjs otz ethns zxqtop
```

Note that spaces are not scrambled. In this particular case, `'T'` was replaced with `'x'`, each `'a'` was replaced with `'h'`, etc. Uppercase letters become lowercase letters in the cryptogram. Use techniques similar to those in Exercise 18.7.

ANS:

```
1  // Exercise 18.15 Solution: Ex18_15.cpp
2  // Program creates a cryptogram from a string.
3  #include <iostream>
4  using std::cin;
5  using std::cout;
6  using std::endl;
7
8  #include <string>
9  using std::getline;
10 using std::string;
11
12 #include <cstdlib>
13 using std::rand;
14 using std::srand;
15
16 #include <ctime>
17 using std::time;
18
19 // prototype
20 void convertToLower( string::iterator, string::iterator );
21
22 int main()
23 {
24    string s;
25    string alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
26    string::iterator is;
27    string::iterator is2;
28    string::iterator is3;
29
```

```
30
31    srand( time( 0 ) ); // random generator
32
33    cout << "Enter a string: ";
34    getline( cin, s, '\n' ); // allow white space to be read
35    cout << "Original string:      " << s;
36
37    is = s.begin(); // is points to the beginning of string s
38
39    // function convertToLower runs through the end
40    convertToLower( is, s.end() );
41
42    string s2( s ); // instantiate s2
43
44    is3 = s2.begin(); // is3 points to the beginning of s2
45
46    do
47    {
48       is2 = is3; // position location on string s2
49
50       // do not change spaces
51       if ( *is == ' ' )
52       {
53          ++is;
54          continue;
55       } // end if
56
57       int x = rand() % alpha.length(); // pick letter
58       char c = alpha.at( x ); // get letter
59       alpha.erase( x, 1 ); // remove picked letter
60
61       // iterate along s2 doing replacement
62       while ( is2 != s2.end() )
63       {
64          if ( *is2 == *is )
65             *is2 = c;
66
67          ++is2;
68       } // end while
69
70       ++is3; // position to next element
71       ++is++; // position to next element
72    } while ( is != s.end() );
73
74    is3 = s2.begin();
75    convertToLower( is3, s2.end() ); // change s2 to lowercase
76    cout << "\nCryptogram of string: " << s2 << endl; // output string
77    return 0; // indicates successful termination
78 } // end main
79
80 // convert strings to lowercase characters
81 void convertToLower( string::iterator i, string::iterator e )
82 {
83    // until the end is reached
84    while ( i != e )
```

```
85     {
86        *i = tolower( *i );
87        ++i;
88     } // end while
89  } // end function convertToLower
```

```
Enter a string: a COLD hard Rain fell
Original string:      a COLD hard Rain fell
Cryptogram of string: h fsjq thlq lhze dbjj
```

18.16 Modify Exercise 18.15 to allow the user to solve the cryptogram. The user should input two characters at a time: The first character specifies a letter in the cryptogram, and the second letter specifies the replacement letter. If the replacement letter is correct, replace the letter in the cryptogram with the replacement letter in uppercase.

18.17 Write a program that inputs a sentence and counts the number of palindromes in it. A palindrome is a word that reads the same backward and forward. For example, "tree" is not a palindrome, but "noon" is.

18.18 Write a program that counts the total number of vowels in a sentence. Output the frequency of each vowel.

18.19 Write a program that inserts the characters "******" in the exact middle of a string.

18.20 Write a program that erases the sequences "by" and "BY" from a string.

ANS:

```
1  // Exercise 18.20 Solution: Ex18_20.cpp
2  // Program erases "by" or "BY" from strings.
3  #include <iostream>
4  using std::cin;
5  using std::cout;
6  using std::endl;
7
8  #include <string>
9  using std::string;
10
11 void deleteBy( string&, string ); // prototype
12
13 int main()
14 {
15    string s;
16
17    cout << "Enter a word: ";
18    cin >> s;
19
20    deleteBy( s, "by" ); // call function deleteBy to get rid of
21    deleteBy( s, "BY" ); // any occurrences of "by" and "BY"
22
23    cout << s << endl;
24    return 0; // indicates successful termination
25 } // end main
26
27 // function to look for and get rid of "by" and "BY"
```

```
28  void deleteBy( string& sRef, string z )
29  {
30     int x = sRef.find( z ); // use member function find of class string
31
32     // until the end of the string is reached
33     while ( x <= sRef.length() )
34     {
35
36        sRef.erase( x, 2 ); // erase the occurrence of "by" or "BY"
37        x = sRef.find( z ); // find location of occurrence
38     } // end while
39  } // end function deleteBy
```

```
Enter a word: firstBYsecondby
firstsecond
```

18.21 Write a program that inputs a line of text, replaces all punctuation marks with spaces and uses the C-string library function `strtok` to tokenize the `string` into individual words.

18.22 Write a program that inputs a line of text and prints the text backwards. Use iterators in your solution.

ANS:

```
1  // Exercise 18.22 Solution: Ex18_22.cpp
2  // Program prints a string backwards.
3  #include <iostream>
4  using std::cin;
5  using std::cout;
6  using std::endl;
7
8  #include <string>
9  using std::getline;
10 using std::string;
11
12 int main()
13 {
14    string s;
15
16    cout << "Enter a string: ";
17    getline( cin, s, '\n' );
18
19    // reverse_iterator rd points to the beginning
20    // of the reversed string
21    string::reverse_iterator rb = s.rbegin();
22
23    // go to the end of the string
24    while ( rb != s.rend() )
25    {
26       cout << *rb; // dereference and print
27       ++rb; // advanced one position
28    } // end while
29
30    cout << endl;
```

```
   return 0; // indicates successful termination
} // end main
```

```
Enter a string: print this backwards
sdrawkcab siht tnirp
```

18.23 Write a recursive version of Exercise 18.22.

ANS:

```
// Exercise 18.23 Solution: Ex18_23.cpp
// Program recursively prints a string backwards.
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

#include <string>
using std::getline;
using std::string;

void printBackwards( const string::reverse_iterator,
   string::reverse_iterator ); // prototype

int main()
{
   string s;

   cout << "Enter a string: ";
   getline( cin, s );

   // reverse_iterator r points
   // one location beyond the end of the reverse string
   string::reverse_iterator r = s.rend();

   // call recursive function printBackwards
   printBackwards( s.rbegin(), r - 1 );
   cout << endl;
   return 0; // indicates successful termination
} // end main

// function to print the reverse string
void printBackwards( const string::reverse_iterator s,
   string::reverse_iterator rb )
{
   // if the end is reached, return
   if ( rb == s - 1 )
      return;

   // recursive call to go through the string
   printBackwards( s, rb - 1 );
   cout << *rb;
} // end function printBackwards
```

```
Enter a string: print this backwords
sdrowkcab siht tnirp
```

18.24 Write a program that demonstrates the use of the `erase` functions that take `iterator` arguments.

18.25 Write a program that generates the following from the `string` `"abcdefghijklmnopqrstuvwxyz{"`:

```
                    a
                   bcb
                  cdedc
                 defgfed
                efghihgfe
               fghijkjihgf
              ghijklmlkjihg
             hijklmnonmlkjih
            ijklmnopqponmlkji
           jklmnopqrsrqponmlkj
          klmnopqrstutsrqponmlk
         lmnopqrstuvwvutsrqponml
        mnopqrstuvwxyxwvutsrqponm
       nopqrstuvwxyz{zyxwvutsrqpon
```

ANS:

```
1  // Exercise 18.25 Solution: Ex18_25.cpp
2  // Program prints a pyramid from a string.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <string>
8  using std::string;
9
10 int main()
11 {
12    string alpha = "abcdefghijklmnopqrstuvwxyz{";
13    string::const_iterator x = alpha.begin();
14    string::const_iterator x2;
15
16    for ( int p = 1; p <= 14; p++ )
17    {
18       int w; // index variable
19       int count = 0; // set to 0 each iteration
20
21       // output spaces
22       for ( int k = 13; k >= p; k-- )
23          cout << ' ';
24
25       x2 = x; // set starting point
26
27       // output first half of characters
28       for ( int c = 1; c <= p; ++c )
```

```
29         {
30            cout << *x2;
31            x2++; // move forwards one letter
32            count++; // keep count of iterations
33         } // end for
34
35         // output back half of characters
36         for ( w = 1, x2 -= 2; w < count; w++ )
37         {
38            cout << *x2;
39            x2--; // move backwards one letter
40         } // end for
41
42         x++; // next letter
43         cout << '\n';
44      } // end for
45
46      return 0; // indicates successful termination
47   } // end main
```

```
            a
           bcb
          cdedc
         defgfed
        efghihgfe
       fghijkjihgf
      ghijklmlkjihg
     hijklmnonmlkjih
    ijklmnopqponmlkji
   jklmnopqrsrqponmlkj
  klmnopqrstutsrqponmlk
 lmnopqrstuvwvutsrqponml
mnopqrstuvwxyxwvutsrqponm
nopqrstuvwxyz{zyxwvutsrqpon
```

18.26 In Exercise 18.7, we asked you to write a simple encryption algorithm. Write a program that will attempt to decrypt a "rot13" message using simple frequency substitution. (Assume that you do not know the key.) The most frequent letters in the encrypted phrase should be replaced with the most commonly used English letters (a, e, i, o, u, s, t, r, etc.). Write the possibilities to a file. What made the code breaking easy? How can the encryption mechanism be improved?

18.27 Write a version of the selection sort routine (Fig. 8.28) that sorts `string`s. Use function `swap` in your solution.

ANS:

```
1  // Exercise 18.27 Solution: Ex18_27.cpp
2  #include <iostream>
3  using std::cout;
4  using std::endl;
5
6  #include <string>
7  using std::string;
8
9  // prototypes
```

```
10 void output( const string *, const int );
11 void selectionSort( string [], const int );
12 void swap( string * const, string * const );
13
14 int main()
15 {
16    const int SIZE = 19;
17
18    string animals[ SIZE ] = { "Macaw", "Lion", "Tiger", "Bear", "Toucan",
19       "Zebra", "Puma", "Cat", "Yak", "Boar", "Fox", "Ferret",
20       "Crocodile", "Alligator", "Elk", "Ox", "Horse", "Eagle", "Hawk" };
21
22    cout << "before:";
23    output( animals, SIZE );
24
25    selectionSort( animals, SIZE ); // sort string
26
27    cout << "\nafter:";
28    output( animals, SIZE );
29
30    return 0; // indicates successful termination
31 } // end main
32
33 // function output to print array of animal names
34 void output( const string * ani, const int length )
35 {
36    for ( int j = 0; j < length; j++ )
37       cout << ( j % 10 ? ' ': '\n' ) << ani[ j ];
38
39    cout << endl;
40 } // end function output
41
42 // function to sort array
43 void selectionSort( string animals[], const int size )
44 {
45    int smallest; // index of smallest element
46
47    // loop over size - 1 elements
48    for ( int i = 0; i < size - 1; i++ )
49    {
50       smallest = i; // first index of remaining vector
51
52       // loop to find index of smallest (or largest) element
53       for ( int index = i + 1; index < size; index++ )
54          if ( animals[ smallest ] > animals[ index ] )
55             smallest = index;
56
57       swap( &animals[ smallest ], &animals[ i ] );
58    } // end if
59 } // end function selectionSort
60
61 // swap values at memory locations to which
62 // element1Ptr and element2Ptr point
63 void swap( string * const element1Ptr, string * const element2Ptr )
64 {
```

```
65      string hold = *element1Ptr;
66      *element1Ptr = *element2Ptr;
67      *element2Ptr = hold;
68   } // end function swap
```

```
before:
Macaw Lion Tiger Bear Toucan Zebra Puma Cat Yak Boar
Fox Ferret Crocodile Alligator Elk Ox Horse Eagle Hawk

after:
Alligator Bear Boar Cat Crocodile Eagle Elk Ferret Fox Hawk
Horse Lion Macaw Ox Puma Tiger Toucan Yak Zebra
```

18.28 Modify class `Employee` in Figs. 13.6–13.7 by adding a `private` utility function called `isValidSocialSecurityNumber`. This member function should validate the format of a social security number (e.g., `###-##-####`, where `#` is a digit). If the format is valid, return `true`; otherwise return `false`.