14 Strings

REVIEW QUESTIONS

1. The two basic techniques for storing a stream of characters are fixed-length strings and variable-length strings.

a. true

3. What is the difference between a C++ strings and C strings?

C++ strings are implemented as a class object while C strings are implemented as a null-delimited array of characters.

5. C++ strings are delimiter controlled.

b. false

7. The standard defines a C++ string as a class object.

a. true

9. Can we use the assignment operator to copy a C++ string into another C++ string?

Yes. The string class is overloaded for the assignment operator.

11. Can we use the relational operators to compare two C++ strings?

Yes. The string class is overloaded for the relational operators.

13. To find the length of a C++ string, we use ____ method; to use the length of a C string, we use ____ function.

a. the *length* or *size* method

b. the *strlen* function

15. To concatenate C++ strings, we use the ____ operators or the ____ method; to concatenate C strings, we use the ____ function.

a. plus (+) or plus-assign (+=) although the latter is technically appending.

b. *append* (although this is technically appending)

c. *strcat* or *strncat*

17. To extract a substring from a C++ string, we use ____ method; to extract a substring from a C string, we use ____.

    a. *substr*

    b. *strstr*

19. To search a C++ string for a character in a set, we use ____ methods; To search a C string for a character in a set, we use ____ functions.

    a. *find_first_of*, *find_last_of*, *find_first_not_of*, or *find_last_not_of*

    b. *strstr* or *strcspn*

21. To replace a substring with another substring in a C++ string, we use ____ method.

    The solution requires two methods, *find* to locate the string and *insert* to replace it.

23. To erase a C++ string, we use ____ method.

    erase

25. A C++ string can be converted to a C string using ____ method; a C string can be converted to a C++ string using ____.

    a. *c_str*

    b. assignment (=)

EXERCISES

27. The following would be printed:

```
Good Evening!
```

29. The following would be printed:

```
0
```

31. The following would be printed:

```
Hello Hello
```

33. The following would be printed:

```
5-1
```

35. There are no compile errors in the code. However, we should get a warning that *x* is used before it is initialized. In this case, we have a classic pointer logic error, we did not allocate memory to read the string into. The result is that part of memory is destroyed and the program may fail or give invalid results.

37. The following would be printed:

```
2
1
1
0
```

39. If *str* is a C++ string, write a code fragment to print the length of *str*.

```
cout << "The length of str is: " << str.length();
```

41. The following code fragment compares the first eight characters in one string to the last eight characters in a second string.

```
int str2End = str2.length() - 8;
if (str1.substr(0, 8) == str2.substr(str2End, 8))
    cout << "strings are equal";
 else
    cout << "strings are not equal";
```

43. The following code extracts the last 6 characters of *str*.

```
str.substr(str.length() - 6, 6)
```

45. The following code finds the second occurrence of the first 4 characters of *str1* in *str2*.

```
loc = str2.find(str1.substr(0, 4));
loc = str2.find(str1.substr(0, 4), loc + 1);
```

47. The following code finds the first occurrence of any characters in *str1* in *str2*.

```
loc = str2.find_first_of(str1);
```

49. The following code finds the first occurrence of any character in *str2* that is not *str1*.

```
loc = str2.find_first_not_of(str1);
```

PROBLEMS

51.

```
/* Delete last character of a C++ string.
      Pre:  Nothing
      Post: last character deleted
*/
void delLast (string& s1)
{
   s1.erase(s1.length() - 1, 1);
   return;
}  // delLast
```

53.

```
/* Delete first character of a C++  string.
      Pre:  Nothing
      Post: first character deleted
*/
void delFirst (string& s1)
{
   s1.erase(0, 1);
   return;
}  // delFirst
```

55.

```
/* Delete trailing spaces in a C++ string.
      Pre:  Nothing
      Post: trailing spaces deleted
*/
void delTailSps (string& str)
{
   int start = str.find_last_not_of(' ');
   if (start < str.length())
       str.erase(start + 1, str.length()
              - (start + 1));
   return;
}  // delTailSps
```

57.

```
/* Count times char found in a C++ string.
      Pre:  Nothing
      Post: count returned
*/
int countChar (string& str, char a)
{
   int count = 0;
   int loc   = str.find_first_of(a, 0);
   while (loc < str.length())
      {
       count++;
```

```
          loc   = str.find_first_of(a, loc + 1);
         } // while
      return count;
   }  // countChar
```

59.

```
/* ================== palindrome ================
   Check a string to see if it is a palindrome.
      Pre  str is a ponter to the string.
      Post if str is a palindrome, returns true
           if it is not, returns false
*/
bool palindrome (const string str)
   {
   string buffer;
   for (int i = 0; i < str.length(); i++)
       if (isalpha (str.at(i)))
           buffer.insert(buffer.length(),
                         1, toupper(str.at(i)));

   string reffub;               // buffer backwards
   for (int i = 0; i < buffer.length(); i++)
        reffub.insert(0, 1, buffer.at(i));
   return buffer == reffub;
}  // palindrome
```