PROGRAMMING EXAMPLE: Code Detection

When a message is transmitted in secret code over a transmission channel, it is usually sent as a sequence of bits, that is, `0`s and `1`s. Due to noise in the transmission channel, the transmitted message may become corrupted. That is, the message received at the destination is not the same as the message transmitted; some of the bits may have been changed. There are several techniques to check the validity of the transmitted message at the destination. One technique is to transmit the same message twice. At the destination, both copies of the message are compared bit by bit. If the corresponding bits are the same, the message received is error-free.

Let's write a program to check whether the message received at the destination is error-free. For simplicity, assume that the secret code representing the message is a sequence of digits (`0` to `9`) and the maximum length of the message is `250` digits. Also, the first number in the message is the length of the message. For example, if the secret code is:

```
7 9 2 7 8 3 5 6
```

then the actual message is `7` digits long.

The above message is transmitted as:

```
7 9 2 7 8 3 5 6 7 9 2 7 8 3 5 6
```

Input A file containing the secret code and its copy

Output The secret code, its copy, and a message—if the received code is error-free—in the following form:

```
Code Digit    Code Digit Copy
    9                 9
    2                 2
    7                 7
    8                 8
    3                 3
    5                 5
    6                 6
Message transmitted OK.
```

PROBLEM ANALYSIS AND ALGORITHM DESIGN

Because we have to compare the corresponding digits of the secret code and its copy, we first read the secret code and store it in an array. Then we read the first digit of the copy and compare it with the first digit of the secret code, and so on. If any of the corresponding digits are not the same, we indicate this fact by printing a message next to the digits. Because the maximum length of the message is `250`, we use an array of size `250`. The first number in both the secret code and the copy of the secret code indicates the length of the code. This discussion translates into the following algorithm:

1. Open the input and output files.
2. If the input file does not exist, exit the program.
3. Read the length of the secret code.
4. If the length of the secret code is greater than `250`, terminate the program because the maximum length of the code in this program is `250`.
5. Read and store the secret code into an array.
6. Read the length of the copy.
7. If the length of the secret code and its copy are the same, compare the codes and output an appropriate message. Otherwise, print an error message.

To simplify the function `main`, let us write a function, `readCode`, to read the secret code and another function, `compareCode`, to compare the codes.

readCode This function first reads the length of the secret code. If the length of the secret code is greater than `250`, a `bool` variable `lenCodeOk`, which is a reference parameter, is set to `false` and the function terminates. The value of `lenCodeOk` is passed to the calling function to indicate whether the secret code was read successfully. If the length of the code is less than `250`, the `readCode` function reads and stores the secret code into an array. Because the input is stored into a file and the file was opened in the function `main`, the input stream variable corresponding to the input file must be passed as a parameter to this function. Furthermore, after reading the length of the secret code and the code itself, the `readCode` function must pass these values to the function `main`. Therefore, this function has four parameters: an input file stream variable, an array to store the secret code, the length of the code, and the `bool` parameter `lenCodeOk`. The definition of the function `readCode` is as follows:

```
void readCode(ifstream& infile, int list[], int& length,
              bool& lenCodeOk)
{
    lenCodeOk = true;

    infile >> length;  //get the length of the secret code

    if (length > MAX_CODE_SIZE)
    {
        lenCodeOk = false;
        return;
    }

        //Get the secret code.
    for (int count = 0; count < length; count++)
        infile >> list[count];
}
```

compareCode This function compares the secret code with its copy. Therefore, it must have access to the array containing the secret code and the length of the secret code. The copy of the secret code and its length are stored in the input file. Thus, the input stream variable corresponding to the input file must be passed as a parameter to this function. Also, the **compareCode** function compares the secret code with the copy and prints an appropriate message. Because the output will be stored in a file, the output stream variable corresponding to the output file must also be passed as a parameter to this function. Therefore, the function has four parameters: an input file stream variable, an output file stream variable, the array containing the secret code, and the length of the secret code. This discussion translates into the following algorithm for the function **compareCode**:

a. Declare the variables.
b. Set a `bool` variable **codeOk** to `true`.
c. Read the length of the copy of the secret code.
d. If the length of the secret code and its copy are not the same, output an appropriate error message and terminate the function.
e. For each digit in the input
   e.1. Read the next digit of the copy of the secret code.
   e.2. Output the corresponding digits from the secret code and its copy.
   e.3. If the corresponding digits are not the same, output an error message and set the `bool` variable **codeOk** to `false`.
f. If the `bool` variable **codeOk** is `true`

   Output a message indicating that the secret code was transmitted correctly.

   else

   Output an error message.

Following this algorithm, the definition of the function **compareCode** is:

```
void compareCode(ifstream& infile, ofstream& outfile,
                 const int list[], int length)
{
        //Step a
    int length2;
    int digit;
    bool codeOk;

    codeOk = true;                                    //Step b

    infile >> length2;                                //Step c
```

```
    if (length != length2)                                  //Step d
    {
        cout << "The original code and its copy "
             << "are not of the same length."
             << endl;
        return;
    }

    outfile << "Code Digit    Code Digit Copy"
            << endl;

    for (int count = 0; count < length; count++)            //Step e
    {
        infile >> digit;                                    //Step e.1
        outfile << setw(5) << list[count]
                << setw(17) << digit;                       //Step e.2

        if (digit != list[count])                           //Step e.3
        {
            outfile << "  code digits are not the same"
                    << endl;
            codeOk = false;
        }
        else
            outfile << endl;
    }

    if (codeOk)                                             //Step f
        outfile << "Message transmitted OK."
                << endl;
    else
        outfile << "Error in transmission. "
                << "Retransmit!!" << endl;
}
```

The following is the algorithm for the function `main`:

MAIN ALGORITHM

1. Declare the variables.
2. Open the files.
3. Call the function `readCode` to read the secret code.
4. `if` (length of the secret code `<= 250`)

   Call the function `compareCode` to compare the codes.
5. `else`

   Output an appropriate error message.

COMPLETE PROGRAM LISTING

```
//*************************************************************
// Author: D.S. Malik
//
// Program: Check Code
// This program determines whether a code is transmitted
// correctly.
//*************************************************************

#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

const int MAX_CODE_SIZE = 250;

void readCode(ifstream& infile, int list[],
              int& length, bool& lenCodeOk);
void compareCode(ifstream& infile, ofstream& outfile,
                 const int list[], int length);

int main()
{
        //Step 1
    int codeArray[MAX_CODE_SIZE]; //array to store the secret
                                  //code
    int codeLength;               //variable to store the
                                  //length of the secret code
    bool lengthCodeOk;  //variable to indicate if the length
                        //of the secret code is less than or
                        //equal to 250

    ifstream incode;    //input file stream variable
    ofstream outcode;   //output file stream variable

    char inputFile[51]; //variable to store the name of the
                        //input file
    char outputFile[51];    //variable to store the name of
                            //the output file

    cout << "Enter the input file name: ";
    cin >> inputFile;
    cout << endl;

        //Step 2
    incode.open(inputFile);
    if (!incode)
```

```
    {
        cout << "Cannot open the input file." << endl;
        return 1;
    }

    cout << "Enter the output file name: ";
    cin >> outputFile;
    cout << endl;

    outcode.open(outputFile);

    readCode(incode, codeArray, codeLength,
             lengthCodeOk);                          //Step 3

    if (lengthCodeOk)                                //Step 4
       compareCode(incode, outcode, codeArray,
                   codeLength);
    else
        cout << "Length of the secret code "
             << "must be <= " << MAX_CODE_SIZE
             << endl;                                //Step 5

    incode.close();
    outcode.close();

    return 0;
}

//Place the definitions of the functions readCode and
//compareCode, as described previously, here.
```

Sample Run: In this sample run, the user input is shaded.

```
Enter the input file name: Ch8_SecretCodeData.txt

Enter the output file name: Ch8_SecretCodeOut.txt
```

Input File Data: (Ch8_SecretCodeData.txt)

```
7 9 2 7 8 3 5 6 7 9 2 7 8 3 5 6
```

Output File Data: (Ch8_SecretCodeOut.txt)

```
Code Digit    Code Digit Copy
    9                 9
    2                 2
    7                 7
    8                 8
    3                 3
    5                 5
    6                 6

Message transmitted OK.
```

PROGRAMMING EXAMPLE: Text Processing

(Line and letter count) Let us now write a program that reads a given text, outputs the text as is, and also prints the number of lines and the number of times each letter appears in the text. An uppercase letter and a lowercase letter are treated as being the same; that is, they are tallied together.

Because there are 26 letters, we use an array of `26` components to perform the letter count. We also need a variable to store the line count.

The text is stored in a file, which we will call `textin.txt`. The output will be stored in a file, which we will call `textout.out`.

Input A file containing the text to be processed.

Output A file containing the text, number of lines, and the number of times a letter appears in the text.

PROBLEM ANALYSIS AND ALGORITHM DESIGN

Based on the desired output, it is clear that we must output the text as is. That is, if the text contains any whitespace characters, they must be output as well. Furthermore, we must count the number of lines in the text. Therefore, we must know where the line ends, which means that we must trap the newline character. This requirement suggests that we cannot use the extraction operator to process the input file. Because we also need to perform the letter count, we use the `get` function to read the text.

Let us first describe the variables that are necessary to develop the program. This will simplify the discussion that follows.

VARIABLES

We need to store the line count and the letter count. Therefore, we need a variable to store the line count and 26 variables to perform the letter count. We will use an array of `26` components to perform the letter count. We also need a variable to read and store each character in turn, because the input file is to be read character by character. Because data is to be read from an input file and output is to be saved in a file, we need an input stream variable to open the input file and an output stream variable to open the output file. These statements indicate that the function `main` needs (at least) the following variables:

```
int lineCount;          //variable to store the line count
int letterCount[26];    //array to store the letter count
char ch;                //variable to store a character
ifstream infile;        //input file stream variable
ofstream outfile;       //output file stream variable
```

In this declaration, `letterCount[0]` stores the `A` count, `letterCount[1]` stores the `B` count, and so on. Clearly, the variable `lineCount` and the array `letterCount` must be initialized to `0`.

The algorithm for the program is as follows:

1. Declare the variables.
2. Open the input and output files.
3. Initialize the variables.
4. While there is more data in the input file:
   - 4.1 For each character in a line:
     - 4.1.1 Read and write the character.
     - 4.1.2 Increment the appropriate letter count.
   - 4.2 Increment the line count.
5. Output the line count and letter counts.
6. Close the files.

To simplify the function **main**, we divide it into four functions:

- Function **initialize**
- Function **copyText**
- Function **characterCount**
- Function **writeTotal**

The following sections describe each of these functions in detail. Then, with the help of these functions, we describe the algorithm for the function **main**.

initialize

This function initializes the variable **lineCount** and the array **letterCount** to **0**. It, therefore, has two parameters: one corresponding to the variable lineCount and one corresponding to the array **letterCount**. Clearly, the parameter corresponding to **lineCount** must be a reference parameter. The definition of this function is:

```
void initialize(int& lc, int list[])
{
    lc = 0;

    for (int j = 0; j < 26; j++)
        list[j] = 0;
} //end initialize
```

copyText

This function reads a line and outputs the line. After reading a character, it calls the function **characterCount** to update the letter count. Clearly, this function has four parameters: an input file stream variable, an output file stream variable, a **char** variable, and the array to update the letter count.

Note that the **copyText** function does not perform the letter count, but we still pass the array **letterCount** to it. We take this step because this function calls

the function **characterCount**, which needs the array **letterCount** to update the appropriate letter count. Therefore, we must pass the array **letterCount** to the **copyText** function so that it can pass the array to the function **characterCount**.

```
void copyText(ifstream& intext, ofstream& outtext, char& ch,
              int list[])
{
    while (ch != '\n')       //process the entire line
    {
        outtext << ch;       //output the character

        characterCount(ch, list);   //call the function
                                    //character count
        intext.get(ch);      //read the next character
    }
    outtext << ch;           //output the newline character
} //end copyText
```

characterCount This function increments the letter count. To increment the appropriate letter count, it must know what the letter is. Therefore, the **characterCount** function has two parameters: a `char` variable and the array to update the letter count.

In pseudocode, this function is:

a. Convert the letter to uppercase.
b. Find the index of the array corresponding to this letter.
c. If the index is valid, increment the appropriate count. At this step, we must ensure that the character is a letter. We are counting only letters, so other characters—such as commas, hyphens, and periods—are ignored.

Following this algorithm, the definition of this function is:

```
void characterCount(char ch, int list[])
{
    int index;

    ch = toupper(ch);                          //Step a

    index = static_cast<int>(ch)
          - static_cast<int>('A');             //Step b

    if (0 <= index && index < 26)              //Step c
        list[index]++;
} //end characterCount
```

writeTotal This function outputs the line count and the letter count. It has three parameters: the output file stream variable, the line count, and the array to output the letter count. The definition of this function is:

```
void writeTotal(ofstream& outtext, int lc, int list[])
{
    outtext << endl;
    outtext << "The number of lines = " << lc << endl;

    for (int index = 0; index < 26; index++)
        outtext << static_cast<char>(index
                                     + static_cast<int>('A'))
                << " count = " << list[index] << endl;
} //end writeTotal
```

We now describe the algorithm for the function `main`.

MAIN ALGORITHM

1. Declare the variables.
2. Open the input file.
3. If the input file does not exist, exit the program.
4. Open the output file.
5. Initialize the variables, such as `lineCount` and the array `letterCount`.
6. Read the first character.
7. While (not end of input file):
   - 7.1 Process the next line; call the function `copyText`.
   - 7.2 Increment the line count. (Increment the variable `lineCount`.)
   - 7.3 Read the next character.
8. Output the line count and letter counts. Call the function `writeTotal`.
9. Close the files.

COMPLETE PROGRAM LISTING

```
//***********************************************************
// Author: D.S. Malik
//
// Program: Line and Letter Count
// This programs reads a text, outputs the text as is, and also
// prints the number of lines and the number of times each
// letter appears in the text. An uppercase letter and a
// lowercase letter are treated as being the same; that is,
// they are tallied together.
//***********************************************************
```

```
#include <iostream>
#include <fstream>
#include <cctype>

using namespace std;

void initialize(int& lc, int list[]);
void copyText(ifstream& intext, ofstream& outtext, char& ch,
              int list[]);
void characterCount(char ch, int list[]);
void writeTotal(ofstream& outtext, int lc, int list[]);

int main()
{
        //Step 1; Declare variables
    int lineCount;
    int letterCount[26];
    char ch;
    ifstream infile;
    ofstream outfile;

    infile.open("textin.txt");                          //Step 2

    if (!infile)                                        //Step 3
    {
        cout << "Cannot open the input file."
             << endl;
        return 1;
    }

    outfile.open("textout.out");                        //Step 4

    initialize(lineCount, letterCount);                 //Step 5

    infile.get(ch);                                     //Step 6

    while (infile)                                      //Step 7
    {
        copyText(infile, outfile, ch, letterCount); //Step 7.1
        lineCount++;                                    //Step 7.2
        infile.get(ch);                                 //Step 7.3
    }

    writeTotal(outfile, lineCount, letterCount);    //Step 8

    infile.close();                                     //Step 9
    outfile.close();                                    //Step 9

    return 0;
}
```

```
void initialize(int& lc, int list[])
{
    lc = 0;

    for (int j = 0; j < 26; j++)
        list[j] = 0;
} //end initialize

void copyText(ifstream& intext, ofstream& outtext, char& ch,
              int list[])
{
    while (ch != '\n')      //process the entire line
    {
        outtext << ch;      //output the character

        characterCount(ch, list);   //call the function
                                    //character count
        intext.get(ch);     //read the next character
    }
    outtext << ch;          //output the newline character
} //end copyText

void characterCount(char ch, int list[])
{
    int index;

    ch = toupper(ch);                       //Step a

    index = static_cast<int>(ch)
            - static_cast<int>('A');        //Step b

    if (0 <= index && index < 26)           //Step c
        list[index]++;
} //end characterCount

void writeTotal(ofstream& outtext, int lc, int list[])
{
    outtext << endl;
    outtext << "The number of lines = " << lc << endl;

    for (int index = 0; index < 26; index++)
        outtext << static_cast<char>(index
                                     + static_cast<int>('A'))
                << " count = " << list[index] << endl;
} //end writeTotal
```

Sample Run (textout.out):

```
The first device known to carry out calculations was the abacus.
The abacus was invented in Asia but was used in ancient Babylon,
China, and throughout Europe until the late middle ages. The aba-
cus uses a system of sliding beads in a rack for addition and sub-
traction. In 1642, the French philosopher and mathematician Blaise
Pascal invented the calculating device called the Pascaline. It
had eight movable dials on wheels and could calculate sums up to
eight figures long. Both the abacus and Pascaline could perform
only addition and subtraction operations. Later in the 17th cen-
tury, Gottfried von Leibniz invented a device that was able to add,
subtract, multiply, and divide.

The number of lines = 13
A count = 62
B count = 16
C count = 29
D count = 32
E count = 54
F count = 7
G count = 9
H count = 24
I count = 45
J count = 0
K count = 2
L count = 30
M count = 8
N count = 43
O count = 30
P count = 10
Q count = 0
R count = 19
S count = 33
T count = 51
U count = 25
V count = 9
W count = 6
X count = 0
Y count = 6
Z count = 1
```

EXERCISES

The number in parentheses at the end of an exercise refers to the learning objective listed at the beginning of the chapter.

1. Mark the following statements as true or false.
   a. A `double` type is an example of a simple data type. (1)
   b. A one-dimensional array is an example of a structured data type. (1)
   c. The size of an array is determined at compile time. (1, 6)
   d. Given the declaration:

      ```
      int list[10];
      ```

      the statement:

      ```
      list[5] = list[3] + list[2];
      ```

      updates the content of the fifth component of the array `list`. (2)
   e. If an array index goes out of bounds, the program always terminates in an error. (3)
   f. The only aggregate operations allowable on `int` arrays are the increment and decrement operations. (5)
   g. Arrays can be passed as parameters to a function either by value or by reference. (6)
   h. A function can return a value of type array. (6)
   i. In C++, some aggregate operations are allowed for strings. (11, 12, 13)
   j. The declaration:

      ```
      char name[16] = "John K. Miller";
      ```

      declares `name` to be an array of 15 characters because the string `"John K. Miller"` has only 14 characters. (11)
   k. The declaration:

      ```
      char str = "Sunny Day";
      ```

      declares `str` to be a string of an unspecified length. (11)
   l. As parameters, two-dimensional arrays are passed either by value or by reference. (15, 16)

2. Consider the following declaration: (1, 2)

   ```
   double currentBalance[91];
   ```

   In this declaration, identify the following:
   a. The array name
   b. The array size

c. The data type of each array component

d. The range of values for the index of the array

e. What are the indices of the first, middle, and the last elements?

3. Identify error(s), if any, in the following array declarations. If a statement is incorrect, provide the correct statement. (1, 2)

a.
```
int primeNum[99];
```

b.
```
int testScores[0];
```

c.
```
string names[60];
```

d.
```
int list100[0..99];
```

e.
```
double[50] gpa;
```

f.
```
const double LENGTH = 26;
double list[LENGTH - 1];
```

g.
```
const long SIZE = 100;
int list[2 * SIZE];
```

4. Determine whether the following array declarations are valid. If a declaration is invalid, explain why. (1, 2)

a.
```
int list[61];
```

b.
```
strings names[20];
```

c.
```
double gpa[];
```

d.
```
double[-50] ratings[];
```

e.
```
string flowers[35];
```

f.
```
int SIZE = 10;
double sales[2 * SIZE];
```

g.
```
int MAX_SIZE = 50;
double sales[100 - 2 * MAX_SIZE];
```

5. What would be a valid range for the index of an array of size 65? What are the indices of the first, middle, and the last elements? (1, 3)

6. Write C++ statement(s) to do the following: (1, 2)

a. Declare an array `alpha` of `50` components of type `int`.

b. Initialize each component of `alpha` to `-1`.

c. Output the value of the first component of the array `alpha`.

d. Set the value of the 25th component of the array `alpha` to `62`.

e. Set the value of the 10th component of `alpha` to three times the value of the 50th component of `alpha` plus `10`.

f. Use a `for` loop to output the value of a component of `alpha` if its index is a multiple of `2` or `3`.

g. Output the value of the last component of `alpha`.

h. Output the value of the `alpha` so that 15 components per line are printed.

i. Use a `for` loop to increment every other element (the even indexed elements).

j. Create a new array, `diffAlpha`, whose elements are the differences between consecutive elements in `alpha`. What is the size of `diffAlpha`?

7. What is the output of the following program segment? (2)

```
double list[5];

for (int i = 0; i < 5; i++)
    list[i] = pow(i, 3) + i / 2.0;

cout << fixed << showpoint << setprecision(2);

for (int i = 0; i < 5; i++)
    cout << list[i] << " ";
cout << endl;

list[0] = list[4] - list[2];
list[2] = list[3] + list[1];

for (int i = 0; i < 5; i++)
    cout << list[i] << " ";
cout << endl;
```

8. What is the output of the following C++ code? (2)

```
int alpha[8];

for (int i = 0; i < 4; i++)
{
    alpha[i] = i * (i + 1);
    if (i % 2 == 0)
        alpha[4 + i] = alpha[i] + i;
    else if (i % 3 == 0)
        alpha[4 + i] = alpha[i] - i;
    else if (i > 0)
        alpha[4 + i] = alpha[i] - alpha[i - 1];
}

for (int i = 0; i < 8; i++)
    cout << alpha[i] << " ";
cout << endl;
```

9. What is stored in `list` after the following C++ code executes? (2)

```
int list[8];

list[0] = 1;
list[1] = 2;
```

```
for (int i = 2; i < 8; i++)
{
    list[i] = list[i - 1] * list[i - 2];
    if (i > 5)
        list[i] = list[i] - list[i - 1];
}
```

10. What is stored in `myList` after the following C++ code executes? (2)

```
double myList[6];

myList[0] = 2.5;

for (int i = 1; i < 6; i++)
{
    myList[i] = i * myList[i - 1];
    if (i > 3)
        myList[i] = myList[i] / 2;
}
```

11. Correct the following code so that it correctly sets the value of each element of `myList` to the index of the element. (2, 3)

```
int myList[10];

for (int i = 1; i > 10; i++)
    myList[i] = i;
```

12. Correct the following code so that it correctly initializes and outputs the elements of the array `intList`. (2, 3)

```
int intList [5];

for (int i = 0; i > 5; i--)
    cin >> intList [i];

for (int i = 0; i < 5; i--)
    cout << intList << " ";
cout << endl;
```

13. What is array index out-of-bound? Does C++ checks for array indices within bound? (3)

14. Suppose that points is an array of 10 components of type `double`, and `points = {9.9, 9.6, 8.5, 8.5, 7.8, 7.7, 6.5, 5.8, 5.8, 4.6}`

The following is supposed to ensure that the elements of `points` are in nonincreasing order. What is the output of this code? There are errors in the code. Find and correct the errors. (1, 2, 3)

```
for (int i = 0; i < 10; i++)
    if (points[i + 1] >= points[i])
        cout << "points[" << i << "] and points[" << (i + 1)
             << "] are out of order." << endl;
```

15. Write C++ statements to define and initialize the following arrays. (4)

    a. Array `heights` of 10 components of type `double`. Initialize this array to the following values: `5.2, 6.3, 5.8, 4.9, 5.2, 5.7, 6.7, 7.1, 5.10, 6.0`.

    b. Array `weights` of 7 components of type `int`. Initialize this array to the following values: `120, 125, 137, 140, 150, 180, 210`.

    c. Array `specialSymbols` of type `char`. Initialize this array to the following values: `'$', '#', '%', '@', '&', '!', '^'`.

    d. Array `seasons` of 4 components of type string. Initialize this array to the following values: `"fall"`, `"winter"`, `"spring"`, `"summer"`.

16. Determine whether the following array declarations are valid. If a declaration is valid, determine the size of the array. (4)

    a. `int list[] = {18, 13, 14, 16};`

    b. `int x[10] = {1, 7, 5, 3, 2, 8};`

    c. `double y[4] = {2.0, 5.0, 8.0, 11.0, 14.0};`

    d. `double lengths[] = {8.2, 3.9, 6.4, 5.7, 7.3};`

    e. `int list[7] = {12, 13, , 14, 16, , 8};`

    f. `string name[8] = {"John", "Lisa", "Chris", "Katie"};`

17. Suppose that you have the following declaration: (4)

```
int alpha[5] = {3, 12, -25, 72};
```

    If this declaration is valid, what is stored in each of the five components of `alpha`.

18. Consider the following declaration. (2)

```
int list[] = {3, 8, 10, 13, 6, 11};
```

    a. Write a C++ code that will output the value stored in each component of list.

    b. Write a C++ code that will set the values of the first five components of `list` as follows: The value of the the *i*th component is the value of the ith component minus three times the value of the $(i+1)$th component.

19. What is the output of the following C++ code? (2)

```
#include <iostream>

using namespace std;

int main()
{
    int alpha[6] = {5};
```

```
    for (int i = 1; i < 6; i++)
    {
        alpha[i] = i * alpha[i - 1];
        alpha[i - 1] = alpha[i] - 2 * alpha[i - 1];
    }

    for (int i = 0; i < 6; i++)
        cout << alpha[i] << " ";
    cout << endl;

    return 0;
}
```

20. What is the output of the following C++ code? (2)

```
#include <iostream>

using namespace std;

int main()
{
    int alpha[10];
    int beta[15];

    for (int i = 0; i < 5; i++)
    {
        alpha[i] = 2 * i + 1;
        alpha[5 + i] = 3 * i - 1;
        beta[i] = 5 * i - 2;
    }

    cout << "alpha: ";
    for (int i = 0; i < 10; i++)
        cout << alpha[i] << " ";
    cout << endl;

    for (int i = 5; i < 10; i++)
    {
        beta[i] = alpha[9 - i] + beta[9 - i];
        beta[i + 5] = beta[9 - i] + beta[i];
    }

    cout << "beta: ";
    for (int i = 0; i < 15; i++)
        cout << beta[i] << " ";
    cout << endl;

    return 0;
}
```

21. Consider the following overloaded function headings: (6)

```
void printList(int list[], int size);
void printList(string sList[], int size);
```

and the declarations:

```
int ids[50];
double unitPrice[100];
string birds[70];
```

Which of the following function calls is valid, that is, will not cause syntax or run time error?

a. `printList(ids, 50);`

b. `printList(birds, 70);`

c. `printList(unitPrice, 100);`

d. `printList(ids, 75);`

e. `printList(birds, 50);`

22. Suppose that you have the following function definition. (6)

```
int find(int x, int y)
{
    return (x + y - x * y);
}
```

Consider the following declarations:

```
int list1[10], list2[10], list3[10];
int u, v;
```

In the following statements, which function call is valid?

a. `u = find(list1[0], v);`

b. `cout << find(list1[0], list2[9]) << endl;`

c. `cout << find(list1, list2) << endl;`

d.
```
for (int i = 0; i < 10; i++)
    list3[i] = find(list1[i], list2[i]);
```

23. What is the output of the following C++ code? (2)

```
double salary[5] = {35700, 96800, 55000, 72500, 87700};
double raise = 0.02;

cout << fixed << showpoint << setprecision(2);

for (int i = 0; i < 5; i++)
    cout << (i + 1) << " " << salary[i] << " "
         << salary[i] * raise << endl;
```

24. A car dealer has 10 salespersons. Each salesperson keeps track of the number of cars sold each month and reports it to the management at the end of the month. The management keeps the data in a file and assigns a number, 1 to 10, to each salesperson. The following statement declares an array, `cars`, of 10 components of type `int` to store the number of cars sold by each salesperson:

```
int cars[10];
```

Write the code to store the number of cars sold by each salesperson in the array **cars**, output the total numbers of cars sold at the end of each month, and output the salesperson number selling the maximum number of cars. (Assume that data is in the file **cars.dat**, and that this file has been opened using the **ifstream** variable **inFile**.) (2)

25. What is the output of the following program? (2)

```
#include <iostream>

using namespace std;

int main()
{
    int list[5];

    list[4] = 10;
    for (int i = 3; i >= 0; i--)
    {
        list[i] = 3 * list[i + 1];
        list[i + 1] = i * list[i];
    }

    cout << "list: ";
    for (int i = 0; i < 5; i++)
        cout << list[i] << " ";
    cout << endl;

    return 0;
}
```

26. What is the output of the following program? (2)

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int quantity[5] = {3, 5, 2, 8, 1 };
    double unitCost[5] = {15.00, 20.00, 5.00, 3.00, 75.00};
    double price[5];

    double billingAmount = 0;

    for (int i = 0; i < 5; i++)
    {
        price[i] = quantity[i] * unitCost[i];
        billingAmount = billingAmount + price[i];
    }

    cout << fixed << showpoint << setprecision(2);
```

```
    cout << setw(7) << "Quantity" << " " << setw(9)
         << "Unit Cost" << "   " << setw(6)
         << "Amount" << endl;

    for (int i = 0; i < 5; i++)
        cout << setw(4) << quantity[i] << "    " << setw(8)
             << unitCost[i] << "  " << setw(9) << price[i]
             << endl;

    cout << "Total due:          $" << billingAmount << endl;

    return 0;
}
```

27. What is the output of the following C++ code? (2, 4)

```
const double PI = 3.14159;
double cylinderRadii[5] = {3.5, 7.2, 10.5, 9.8, 6.5};
double cylinderHeights[5] = {10.7, 6.5, 12.0, 10.5, 8.0};
double cylinderVolumes[5];

cout << fixed << showpoint << setprecision(2);

for (int i = 0; i < 5; i++)
    cylinderVolumes[i] = 2 * PI * cylinderRadii[i]
                         * cylinderHeights[i];

for (int i = 0; i < 5; i++)
    cout << (i + 1) << " " << cylinderRadii[i] << " "
         << cylinderHeights[i] << " " << cylinderVolumes[i]
         << endl;
```

28. When an array is passed as an actual parameter to a function, what is actually being passed? (6)

29. In C++, as an actual parameter, can an array be passed by value? (6)

30. Sort the following list using the selection sort algorithm as discussed in this chapter. Show the list after each iteration of the outer `for` loop. (8)

```
12, 50, 68, 30, 46, 5, 92, 10, 38
```

31. What is the output of the following C++ program segment? (9, 10)

```
int list[] = {15, 18, 3, 65, 11, 32, 60, 55, 9};

for (auto num: list)
    cout << num % 2 << " ";
cout << endl;
```

32. What is the output of the following C++ program segment? (9, 10)

```
string names[] = {"Blair, Cindy", "Johnson, Chris",
                  "Mann, Sheila"};
string str1, str2;
char ch = ',';
int pos, length;
```

```
for (auto &str: names)
{
    pos = str.find(ch);
    length = str.length();
    str1 = str.substr(0, pos);
    str2 = str.substr(pos + 2, length - pos - 1);
    str = str2 + ' ' + str1;
}

for (auto str: names)
    cout << str << endl;
```

33. Consider the following function heading. (9, 10)

```
void modifyList(int list[], int length)
```

In the definition of the function `modifyList`, can you use a range-based for loop to process the elements of list? Justify your answer.

34. Given the declaration:

```
char name[30];
```

mark the following statements as valid or invalid. If a statement is invalid, explain why. (11)

a. `name = "Bill William";`

b. `strcmp(name, "Tom Jackson");`

c. `strcpy(name, "Jacksonville");`

d. `cin >> name;`

e. `name[0] = 'K';`

f. `bool flag = (name >= "Cynthia");`

35. Given the declaration:

```
char str1[20];
char str2[15] = "Fruit Juice";
```

mark the following statements as valid or invalid. If a statement is invalid, explain why. (11, 12)

a. `strcpy(str1, str2);`

b.
```
if (strcmp(str1, str2) == 0)
   cout << " str1 is the same as str2" << endl;
```

c.
```
if (strlen(str1) >= strlen(str2))
   str1 = str2;
```

d.
```
if (str1 > str2)
   cout << "str1 > str2." << endl;
```

36. Given the declaration:

```
char name[8] = "Shelly";
```

mark the following statements as "Yes" if they output **Shelly**. Otherwise, mark the statement as "No" and explain why it does not output **Shelly**. (11)

a.
```
cout << name;
```

b.
```
for (int j = 0; j < 6; j++)
    cout << name[j];
```

c.
```
int j = 0;
while (name[j] != '\0')
    cout << name[j++];
```

d.
```
int j = 0;
while (j < 8)
    cout << name[j++];
```

37. Given the declaration: (11, 12)

```
char myStr[26];
char yourStr[26] = "Arrays and Strings";
```

a. Write a C++ statement that stores "**Summer Vacation**" in myStr.

b. Write a C++ statement that outputs the length of **yourStr**.

c. Write a C++ statement that copies the value of **yourStr** into **myStr**.

d. Write a C++ statement that compares **myStr** with **yourStr** and stores the result into an **int** variable **compare**.

38. Assume the following declarations: (11, 12, 13)

```
char name[21];
char yourName[21];
char studentName[31];
```

Mark the following statements as valid or invalid. If a statement is invalid, explain why.

a.
```
cin >> name;
```

b.
```
cout << studentName;
```

c.
```
yourName[0] = '\0';
```

d.
```
yourName = studentName;
```

e.
```
if (yourName == name)
  studentName = name;
```

f.
```
int x = strcmp(yourName, studentName);
```

g.
```
strcpy(studentName, name);
```

h.
```
for (int j = 0; j < 21; j++)
  cout << name[j];
```

39. Define a two-dimensional array named **matrix** of 4 rows and 3 columns of type double such that the first row is initialized to 2.5, 3.2, 6.0; the second row is initialized to 5.5, 7.5, 12.6; the third row is initialized to 11.25, 16.85, 13.45; and the fourth row is initialized to 8.75, 35.65, 19.45. (15)

40. Suppose that array **matrix** is as defined in Exercise 39. Write C++ statements to accomplish the following: (15)

    a. Input numbers in the first row of **matrix**.
    b. Output the contents of the last column of **matrix**.
    c. Output the contents of the first row and last column element of **matrix**.
    d. Add 13.6 to the last row and last column element of **matrix**.

41. Consider the following declarations: (15)

```
const int CAR_TYPES = 5;
const int COLOR_TYPES = 6;

double sales[CAR_TYPES][COLOR_TYPES];
```

    a. How many components does the array **sales** have?
    b. What is the number of rows in the array **sales**?
    c. What is the number of columns in the array **sales**?
    d. To sum the sales by **CAR_TYPES**, what kind of processing is required?
    e. To sum the sales by **COLOR_TYPES**, what kind of processing is required?

42. Write C++ statements that do the following: (15)

    a. Declare an array **alpha** of 10 rows and 20 columns of type int.
    b. Initialize the array **alpha** to 0.
    c. Store 1 in the first row and 2 in the remaining rows.
    d. Store 5 in the first column, and make sure that the value in each subsequent column is twice the value in the previous column.
    e. Print the array **alpha** one row per line.
    f. Print the array **alpha** one column per line.

43. Consider the following declaration: (15)

```
int beta[3][3];
```

What is stored in **beta** after each of the following statements executes?

    a.
```
for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
        beta[i][j] = 0;
```

b.
```
for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
        beta[i][j] = i + j;
```

c.
```
for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
        beta[i][j] = i * j;
```

d.
```
for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
        beta[i][j] =  2 * (i + j) % 4;
```

e.
```
for (int i = 2; i >= 0; i--)
    for (int j = 0; j < 3; j++)
        beta[i][j] =  (i * j) % 3;
```

44. Suppose that you have the following declarations: (15)

```
int flowers[28][10];
int animals[15][10];
int trees[100][10];
int inventory[30][10];
```

a. Write the definition of the function **readIn** that can be used to input data into these arrays. Also write C++ statements that call this function to input data into these arrays.

b. Write the definition of the function **sumRow** that can be used to sum the elements of each row of these arrays. Also write C++ statements that call this function to find the sum of the elements of each row of these arrays.

c. Write the definition of the function **print** that can be used to output the contents of these arrays. Also write C++ statements that call this function to output the contents of these arrays.

PROGRAMMING EXERCISES

1. Write a C++ program that declares an array **alpha** of **50** components of type **double**. Initialize the array so that the first 25 components are equal to the square of the index variable, and the last 25 components are equal to three times the index variable. Output the array so that 10 elements per line are printed.

2. Write a C++ function, **smallestIndex**, that takes as parameters an **int** array and its size and returns the index of the first occurrence of the smallest element in the array. Also, write a program to test your function.

3. Write a C++ function, **lastLargestIndex**, that takes as parameters an **int** array and its size and returns the index of the last occurrence of the largest element in the array. Also, write a program to test your function.

4. Write a program that reads a file consisting of students' test scores in the range `0-200`. It should then determine the number of students having scores in each of the following ranges: `0-24, 25-49, 50-74, 75-99, 100-124, 125-149, 150-174,` and `175-200`. Output the score ranges and the number of students. (Run your program with the following input data: `76, 89, 150, 135, 200, 76, 12, 100, 150, 28, 178, 189, 167, 200, 175, 150, 87, 99, 129, 149, 176, 200, 87, 35, 157, 189.`)

5. Write a program that prompts the user to input a string and outputs the string in uppercase letters. (Use a character array to store the string.)

6. The history teacher at your school needs help in grading a True/False test. The students' IDs and test answers are stored in a file. The first entry in the file contains answers to the test in the form:

   ```
   TFFTFFTTTTFFTFTFTFTT
   ```

   Every other entry in the file is the student ID, followed by a blank, followed by the student's responses. For example, the entry:

   ```
   ABC54301 TFTFTFTT TFTFTFFTTFT
   ```

   indicates that the student ID is `ABC54301` and the answer to question 1 is True, the answer to question 2 is False, and so on. This student did not answer question 9. The exam has 20 questions, and the class has more than 150 students. Each correct answer is awarded two points, each wrong answer gets one point deducted, and no answer gets zero points. Write a program that processes the test data. The output should be the student's ID, followed by the answers, followed by the test score, followed by the test grade. Assume the following grade scale: `90%-100%, A; 80%-89.99%, B; 70%-79.99%, C; 60%-69.99%, D;` and `0%-59.99%, F`.

7. Write a program that allows the user to enter the last names of five candidates in a local election and the number of votes received by each candidate. The program should then output each candidate's name, the number of votes received, and the percentage of the total votes received by the candidate. Your program should also output the winner of the election. A sample output is:

   ```
   Candidate          Votes Received      % of Total Votes

   Johnson                  5000                25.91
   Miller                   4000                20.73
   Duffy                    6000                31.09
   Robinson                 2500                12.95
   Ashtony                  1800                 9.33
   Total                   19300

   The Winner of the Election is Duffy.
   ```

8. Consider the following function **main**:

```
int main()
{
    int alpha[20];
    int beta[20];
    int matrix[10][4];
    .
    .
    .
}
```

   a. Write the definition of the function inputArray that prompts the user to input 20 numbers and stores the numbers into **alpha**.

   b. Write the definition of the function **doubleArray** that initializes the elements of beta to two times the corresponding elements in **alpha**. Make sure that you prevent the function from modifying the elements of **alpha**.

   c. Write the definition of the function **copyAlphaBeta** that stores **alpha** into the first five rows of **matrix** and **beta** into the last five rows of **matrix**. Make sure that you prevent the function from modifying the elements of **alpha** and **beta**.

   d. Write the definition of the function **printArray** that prints any one-dimensional array of type **int**. Print 15 elements per line.

   e. Write a C++ program that tests the function main and the functions discussed in parts a through d. (Add additional functions, such as printing a two-dimensional array, as needed.)

9. Write a program that uses a two-dimensional array to store the highest and lowest temperatures for each month of the year. The program should output the average high, average low, and the highest and lowest temperatures for the year. Your program must consist of the following functions:

   a. Function **getData**: This function reads and stores data in the two-dimensional array.

   b. Function **averageHigh**: This function calculates and returns the average high temperature for the year.

   c. Function **averageLow**: This function calculates and returns the average low temperature for the year.

   d. Function **indexHighTemp**: This function returns the index of the highest high temperature in the array.

   e. Function **indexLowTemp**: This function returns the index of the lowest low temperature in the array.

   These functions must all have the appropriate parameters.

10. Programming Exercise 10 in Chapter 6 asks you find the mean and standard deviation of five numbers. Extend this programming exercise to

find the mean and standard deviation of up to 100 numbers. Suppose that the mean (average) of n numbers $x_1, x_2, \ldots, x_n$ is x. Then the standard deviation of these numbers is:

$$s = \sqrt{\frac{(x_1 - x)^2 + (x_2 - x)^2 + \cdots + (x_i - x)^2 + \cdots + (x_n - x)^2}{n}}$$

11. (**Adding Large Integers**) In C++, the largest `int` value is `2147483647`. So, an integer larger than this cannot be stored and processed as an integer. Similarly, if the sum or product of two positive integers is greater than `2147483647`, the result will be incorrect. One way to store and manipulate large integers is to store each individual digit of the number in an array. Write a program that inputs two positive integers of, at most, 20 digits and outputs the sum of the numbers. If the sum of the numbers has more than 20 digits, output the sum with an appropriate message. Your program must, at least, contain a function to read and store a number into an array and another function to output the sum of the numbers. (*Hint*: Read numbers as strings and store the digits of the number in the reverse order.)

12. Jason, Samantha, Ravi, Sheila, and Ankit are preparing for an upcoming marathon. Each day of the week, they run a certain number of miles and write them into a notebook. At the end of the week, they would like to know the number of miles run each day, the total miles for the week, and average miles run each day. Write a program to help them analyze their data. Your program must contain parallel arrays: an array to store the names of the runners and a two-dimensional array of five rows and seven columns to store the number of miles run by each runner each day. Furthermore, your program must contain at least the following functions: a function to read and store the runners' names and the numbers of miles run each day; a function to find the total miles run by each runner and the average number of miles run each day; and a function to output the results. (You may assume that the input data is stored in a file and each line of data is in the following form: `runnerName milesDay1 milesDay2 milesDay3 milesDay4 milesDay5 milesDay6 milesDay7`.)

13. Write a program to calculate students' average test scores and their grades. You may assume the following input data:

```
Johnson 85 83 77 91 76
Aniston 80 90 95 93 48
Cooper 78 81 11 90 73
Gupta 92 83 30 69 87
Blair 23 45 96 38 59
Clark 60 85 45 39 67
Kennedy 77 31 52 74 83
Bronson 93 94 89 77 97
Sunny 79 85 28 93 82
Smith 85 72 49 75 63
```

Use three arrays: a one-dimensional array to store the students' names, a (parallel) two-dimensional array to store the test scores, and a parallel one-dimensional array to store grades. Your program must contain at least the following functions: a function to read and store data into two arrays, a function to calculate the average test score and grade, and a function to output the results. Have your program also output the class average.

14. Write a program that prompts the user to enter 50 integers and stores them in an array. The program then determines and outputs which numbers in the array are sum of two other array elements. If an array element is the sum of two other array elements, then for this array element, the program should output all such pairs.

15. Redo Programming Exercise 14 by first sorting the array before determining the array elements that are sum of two other elements. Use selection sort algorithm, discussed in this chapter to sort the array.

16. (Pick 5 Lotto) Write a program to simulate a pick-5 lottery game. Your program should generate and store 5 distinct numbers between 1 and 9 (inclusive) into an array. The program prompts the user to enter five distinct between 1 and 9 and stores the number into another array. The program then compares and determines whether the two arrays are identical. If the two arrays are identical, then the user wins the game; otherwise the program outputs the number of matching digits and their position in the array.

    Your program must contain a function that randomly generates the pick-5 lottery numbers. Also, in your program, include the function sequential search to determine if a lottery number generated has already been generated.

17. A company hired 10 temporary workers who are paid hourly and you are given a data file that contains the last name of the employees, the number of hours each employee worked in a week, and the hourly pay rate of each employee. You are asked to write a program that computes each employee's weekly pay and the average salary of all the workers. The program then outputs the weekly pay of each employee, the average weekly pay, and the names of all the employees whose pay is greater than or equal to the average pay. If the number of hours worked in a week is more than 40, then the pay rate for the hours over 40 is 1.5 times the regular hourly rate. Use two parallel arrays: a one-dimensional array to store the names of all the employees, and a two-dimensional array of 10 rows and 3 columns to store the number of hours an employee worked in a week, the hourly pay rate, and the weekly pay. Your program must contain at least the following functions—a function to read the data from the file into the arrays, a function to determine the weekly pay, a function to output the names of all the employees whose pay is greater than or equal to the average weekly pay, and a function to output each employee's data.

18. Children often play a memory game in which a deck of cards containing matching pairs is used. The cards are shuffled and placed face down on a table. The players then take turns and select two cards at a time. If both cards match, they are left face up; otherwise, the cards are placed face down at the same positions. Once the players see the selected pair of cards and if the cards do not match, then they can memorize the cards and use their memory to select the next pair of cards. The game continues until all the cards are face up. Write a program to play the memory game. Use a two-dimensional array of 4 rows and 4 columns for a deck of 16 cards with 8 matching pairs. You can use numbers 1 to 8 to mark the cards. (If you use a 6 by 6 array, then you will need 18 matching pairs, and so on.) Use random number generators to randomly store the pairs in the array. Use appropriate functions in your program, and the main program should be merely a call to functions.

19. (**Airplane Seating Assignment**) Write a program that can be used to assign seats for a commercial airplane. The airplane has 13 rows, with six seats in each row. Rows 1 and 2 are first class, rows 3 through 7 are business class, and rows 8 through 13 are economy class. Your program must prompt the user to enter the following information:

    a. Ticket type (first class, business class, or economy class)

    b. Desired seat

    Output the seating plan in the following form:

```
        A   B   C   D   E   F
Row 1   *   *   X   *   X   X
Row 2   *   X   *   X   *   X
Row 3   *   *   X   X   *   X
Row 4   X   *   X   *   X   X
Row 5   *   X   *   X   *   *
Row 6   *   X   *   *   *   X
Row 7   X   *   *   *   X   X
Row 8   *   X   *   X   X   *
Row 9   X   *   X   X   *   X
Row 10  *   X   *   X   X   X
Row 11  *   *   X   *   X   *
Row 12  *   *   X   X   *   X
Row 13  *   *   *   *   X   *
```

    Here, `*` indicates that the seat is available; `X` indicates that the seat is occupied. Make this a menu-driven program; show the user's choices and allow the user to make the appropriate choices.

20. The program in Example 8-7 outputs the average speed over the intervals of length 10. Modify the program so that the user can store the distance traveled at the desired times, such as times 0, 10, 16, 20, 30, 38, and 45. The program then computes and outputs the average speed of the object over the successive time intervals specified by the time when the distance was recorded. For example, for the previous list of times, the average speed is computed over the time intervals 0 to 16, 16 to 20, 20 to 30, 30 to 38, and 38 to 45.

21. A positive integer n is called prime if $n > 1$ and the only factors of n are 1 and n. It is known that a positive integer $n > 1$ is prime if n is not divisible by any prime integer $m \leq \sqrt{n}$. The 1230th prime number is 10,007. Let t be an integer such that $2 \leq t \leq 100{,}000{,}000$. Then t is prime if either t is equal to one of the first 1,230 prime numbers or t is not divisible by any of the first 1,230 prime numbers. Write a program that declares an array of size 1,230 and stores the first 1,230 prime numbers in this array. The program then uses the first 1,230 prime numbers to determine if a number between 2 and 100,000,000 is prime. If a number is not prime, then output at least one of its prime factors.

22. A positive integer m is called **composite** if $m = ab$, where a and b are positive integers such that a $\neq 1$ and b $\neq 1$. If m is composite, then m can be written as a product of prime numbers. Let m be an integer such that $2 \leq m \leq 100{,}000{,}000$. Modify the program in Exercise 21 so that if m is not prime, the program outputs m as a product of prime numbers.

23. Write a program that uses a 3×3 array and randomly place each integer from 1 to 9 into the nine squares. The program calculates the magic number by adding all the numbers in the array and then dividing the sum by 3. The 3×3 array is a magic square if the sum of each row, each column, and each diagonal is equal to the magic number. Your program must contain at least the following functions: a function to randomly fill the array with the numbers and a function to determine if the array is a magic square. Run these functions for some large number of times, say 1,000, 10,000, or 1,000,000, and see the number of times the array is a magic square.

24. Write a program that randomly generates a 20×20 two-dimensional array, `board`, of type `int`. An element `board[i][j]` is a peak (either a maximum or a minimum) if all its neighbors (there should be either 3, 5, or 8 neighbors for any cell) are less than `board[i][j]`, or greater than `board[i][j]`. The program should output all elements in `board`, with their indices, which are peak. It should also output if a peak is a maximum or a minimum.