EXERCISE ANSWERS

Exam Preparation Exercises

1. Loops are segments of code that are repeated from 0 to many times until some stopping condition is met. Branches are segments of code that may be executed 0 or 1 times, depending on evaluation of the logical expression in the If statement.

2. The loop prints the integers 2 through 11, one number per line.

5. In the output below, the first two digits on each line are values of j, and the third digit is the value of i.

\[
\begin{align*}
2 & 1 & 4 \\
2 & 1 & 3 \\
2 & 1 & 2 \\
2 & 1 & 1 \\
\end{align*}
\]

6. a. 4 6 8 10 12 14 16 18 20 and so on

b. The two flaws are the initial value of n and the While condition. n is initially set to 2, but the 2 is not printed before n is incremented to 4. This flaw can be fixed by incrementing n after the output statement instead of before it. The loop never stops because n never equals 15; n always contains an even number. This may be corrected by changing the While condition to \( n < 15 \). The corrected code is:

\[
\begin{align*}
n & = 2; \\
\text{while} & \ (n < 15) \\
\{ & \text{cout} \ll n \ll \text{'}\text{';} \\
& n = n + 2; \\
\} \\
\end{align*}
\]

7. a. The output is BCDE, followed by an echo of the newline character. The 'A' that is input by the priming read is discarded because the statements in the loop body are in reverse order.

b. Corrected code:

\[
\begin{align*}
cin. & \text{get(inChar);} \\
\text{while} & \ (\text{inChar} != \text{'\text{n}'}) \\
\{ & \text{cout} \ll \text{inChar;} \\
& \text{cin.get(inChar);} \\
\} \\
\end{align*}
\]

8. One (and only one) priming read is needed. It precedes the outer While loop:

\[
\begin{align*}
cin. & \text{get(letter);} \\
\text{while} & \ (\text{cin}) \\
\end{align*}
\]

This input statement primes not only the outer loop but also the inner loop.

10. a. After exit from the loop, sum contains 18 and number contains 0.

b. No, the given data set does not fully test the program. Another set needs to be tested in which the value of i reaches its limit, another in which the first data item is the sentinel value 0, and yet another in which all data values before the sentinel are less than zero.

12. 1 3 6 10 15


**Programming Warm-up Exercises**

3. ```
    line = 1;
    while (line <= 4)
    {
        number = 1;
        while (number <= line)
        {
            cout << number << " ";
            number++;
        }
        cout << endl;
        line++;
    }
```

4. Below, sum, count, and score are of type int, and average is of type float.

```
    count = 0;
    sum = 0;
    scoreFile >> score;
    while (scoreFile)
    {
        sum = sum + score;
        count++;
        scoreFile >> score;
    }
    if (count > 0)
        average = float(sum) / float(count);
```

8. ```
    hour = 1;
    tenMinute = 0;
    am = true;
    done = false;
    while (!done)
    {
        cout << setw(2) << hour << ":" << tenMinute << "0";
        if (am)
            cout << " A.M. ";
        else
            cout << " P.M. ";
        tenMinute++;
        if (tenMinute > 5)
        {
            cout << endl;
            tenMinute = 0;
            hour++;
            if (hour == 13)
                hour = 1;
            else if (hour == 12)
                am = !am;
        }
        if (hour == 1 && tenMinute == 0 && am)
            done = true;
    }
```

9. ```
    numStrings = 0;
    numWithE = 0;
    cout << fixed << showpoint << setprecision(1);
```
cin >> aString;
while (aString != "End")
{
    numStrings++;
    if (aString.find('e') != string::npos)
        numWithE++;
    cin >> aString;
}
ocout << "Number of strings: " << numStrings << endl;
ocout << "Number of strings containing 'e': " << numWithE << endl;
if (numStrings > 0)
ocout << "Percentage of strings containing 'e': 
    " << float(numWithE) / float(numStrings) * 100.0 << endl;

10. Below, modifications to the Exercise 9 solution are shown in boldface.
cout << fixed << showpoint << setprecision(1);

    inFile >> aString;
    while (inFile)
    {
        numStrings = 0;
        numWithE = 0;
        while (aString != "End")
        {
            numStrings++;
            if (aString.find('e') != string::npos)
                numWithE++;
inFile >> aString;
        }
ocout << "Number of strings: " << numStrings << endl;
ocout << "Number of strings containing 'e': " << numWithE << endl;
if (numStrings > 0)
ocout << "Percentage of strings containing 'e': 
    " << float(numWithE) / float(numStrings) * 100.0 << endl;
inFile >> aString;
    }

11. Below, modifications to the Exercise 10 solution are shown in boldface.

totalStrings = 0;
totalWithE = 0;
cout << fixed << showpoint << setprecision(1);

    inFile >> aString;
    while (inFile)
    {
        numStrings = 0;
        numWithE = 0;
        while (aString != "End")
        {
            numStrings++;
            if (aString.find('e') != string::npos)
                numWithE++;
inFile >> aString;
        }
ocout << "Number of strings: " << numStrings << endl;
cout << "Number of strings containing 'e': " << numWithE << endl;
if (numStrings > 0)
    cout << "Percentage of strings containing 'e': " << float(numWithE) / float(numStrings) * 100.0 << endl;
totalStrings = totalStrings + numStrings;
totalWithE = totalWithE + numWithE;
inFile >> aString;
}
cout << endl;
cout << "Total number of strings: " << totalStrings << endl;
cout << "Total number of strings containing 'e': " << totalWithE << endl;
if (totalStrings > 0)
cout << "Percentage of strings containing 'e': "
    << float(totalWithE) / float(totalStrings) * 100.0 << endl;

Case Study Follow-Up

1. Below, changes to the original program are shown in boldface.

    int main()
    {
        char sex;          // Coded 'F' = female, 'M' = male
        .
        string fileName;     // External name of file
        int badDataSets;    // Number of erroneous data sets
        .
        incFile >> sex >> amount;
        femaleCount = 0;
        femaleSum = 0.0;
        maleCount = 0;
        maleSum = 0.0;
        badDataSets = 0;

        while (incFile)
        {
            cout << "Sex: " << sex << " Amount: " << amount << endl;

            if (amount < 0.0)           // Check for invalid salary
            {
                cout << "** Bad data—negative salary **" << endl;
                badDataSets++;
            }
            else
            if (sex == 'F')
            {
                femaleCount++;
                femaleSum = femaleSum + amount;
            }
            else if (sex == 'M')
            {
                maleCount++;
                maleSum = maleSum + amount;
            }
            else // Reject invalid sex code
            {
cout << "** Bad data—invalid sex code **" << endl;
    badDataSets++;
}
    incFile >> sex >> amount;
}
            if (femaleCount <= 0)
                cout << "No females" << endl; // Avoid division by zero
else
    {
        femaleAverage = femaleSum / float(femaleCount);
        cout << "For " << femaleCount << " females, the average "
            << "income is " << femaleAverage << endl;
    }
    if (maleCount <= 0)
        cout << "No males" << endl;     // Avoid division by zero
else
    {
        maleAverage = maleSum / float(maleCount);
        cout << "For " << maleCount << " males, the average "
            << "income is " << maleAverage << endl;
    }
    cout << "No. of erroneous data sets: " << badDataSets << endl;
return 0;
}

3. To compute the highest and lowest incomes for each gender, the following changes need to be made:

   a. Add the following #include directive:

      #include <cfloat>  // For FLT_MAX

   b. Add the following declarations:

      int femaleHighest;  // Highest income for females
      int femaleLowest;   // Lowest income for females
      int maleHighest;    // Highest income for males
      int maleLowest;     // Lowest income for males

   c. Add the following initializations:

      femaleHighest = 0.0;
      femaleLowest = FLT_MAX;
      maleHighest = 0.0;
      maleLowest = FLT_MAX;

   d. Modify the loop as follows:

      while (incFile)
      {
        cout << "Sex: " << sex << " Amount: " << amount << endl;
        if (sex == 'F')
        {
            femaleCount++;
            femaleSum = femaleSum + amount;
            if (amount > femaleHighest)
                femaleHighest = amount;
            if (amount < femaleLowest)
                femaleLowest = amount;
        }
Chapter 6  Looping

...

} else {
    maleCount++;
    maleSum = maleSum + amount;
    if (amount > maleHighest)
        maleHighest = amount;
    if (amount < maleLowest)
        maleLowest = amount;
}
incFile >> sex >> amount;
}

e. Finally, change the output statements at the end of the program to:

cout << "For " << femaleCount << " females, the average "
    << "income is " << femaleAverage
    << ", the highest income is " << femaleHighest
    << ", and the lowest income is " << femaleLowest << endl;
cout << "For " << maleCount << " males, the average "
    << "income is " << maleAverage
    << ", the highest income is " << maleHighest
    << ", and the lowest income is " << maleLowest << endl;

4. To test the modified Incomes program, we use the following set of data:

    M 10000.00
    F 30000.00
    M 30000.00
    F 50000.00
    M 20000.00
    F 40000.00

Expected output:

For 3 females, the average income is 40000.00, the highest income is 50000.00, and the lowest income is 30000.00

For 3 males, the average income is 20000.00, the highest income is 30000.00, and the lowest income is 10000.00