1. An array allows you to store and work with multiple values of the same data type.
2. int hours[6];

The name of this array is hours. The number inside the brackets is the array’s size declarator. It indicates the number of elements, or values, the array can hold. The hours array can store six elements, each one an integer.

1. The individual elements of an array are assigned unique subscripts. These subscripts are used to access the elements.
2. Here is an example of a statement that stores the number 20 in the ﬁrst element of the array:

hours[0] = 20;

1. If an array holding numeric values is deﬁned globally, all of its elements are initialized to zero by default. Local arrays, however, have no default initialization value.
2. It is important to understand the difference between the array size declarator and a subscript. The number inside the brackets in an array deﬁnition is the size declarator. It speciﬁes how many elements the array holds. The number inside the brackets in an assignment statement or any statement that works with the contents of an array is a subscript. It speciﬁes which element is being accessed.
3. The following are all legal assignment statements.

doctorA[0] = 31; // doctorA[0] now holds 31.

doctorA[1] = 40; // doctorA[1] now holds 40.

doctorA[2] = doctorA[0]; // doctorA[2] now also holds 31.

doctorB[0] = doctorA[1]; // doctorB[0] now holds 40.

1. Even though most C++ compilers require the size declarator of an array deﬁnition to be a constant or a literal, subscript numbers can be stored in variables. This makes it possible to use a loop to “cycle through” an entire array, performing the same operation on each element
2. In many circumstances you will need to read data from a ﬁle and store it in an array. The process is straightforward, and in most cases is best done with a loop. Each iteration of the loop reads an item from the ﬁle and stores it in an array element.
3. // This program reads employee work hours from a file and stores them 2 // in an int array. It uses one for loop to input the hours and another 3 // for loop to display them.

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

const int NUM\_EMPLOYEES = 6; // Sets number of employees 11

int hours[NUM\_EMPLOYEES]; // Holds each employee's hours 12

int count;

datafile.open("work.dat");

if (!datafile)

cout << "Error opening data file\n";

else 20

{

for (count = 0; count < NUM\_EMPLOYEES; count++)

datafile >> hours[count];

datafile.close();

cout << "The hours worked by each employee are\n";

for (count = 0; count < NUM\_EMPLOYEES; count++)

{

cout << "Employee " << count+1 << ": ";

cout << hours[count] << endl;

}

}

return 0;

}

1. Writing the contents of an array to a ﬁle is also a straightforward matter. First open an output ﬁle pointed to by an ofstream object, as you learned to do in Chapter 3. Then simply use a loop to step through each element of the array, as we did in lines 27 through 30 of Program 8-3, and direct the output to the ﬁle instead of to the computer screen.

Start from 493 ( no bound checking in c++)