C# Programming: From Problem Analysis to Program Design

Chapter 7

1. a. all have the same type.
2. d. integer
3. d. int [ ] temp = new int [7];
4. b. for(x = 0; x < 4; ++x) num [x] = 100;
5. b. specifies the size of the array as five
6. a. index of the largest of the first num elements of array anArray
7. b. counts the number of elements of array anArray less than its initial element
8. c. b
9. e. none of the above
10. b. x[4] += 42;
11. b. 3
12. d. 0 2 4 6 8
13. d. anArray [2] \*= 2;
14. a. replace the 800 amount with 810.
15. c. replace the 600 with 200.
16. b. display 600
17. c. a copy of the value in the element.
18. b. the address of the array.
19. a. Write(result(anArray, 3));
20. e. all are correct
21. a. 5

b. 67

c. 89

d. 3026

e. 155

1. foreach(int ar in bArray)

{

Console.WriteLine(ar);

}

1. for (int i = 0; i < bArray.Length; i ++)

{

bArray[i] += 5;

}

1. int location = Array.IndexOf(bArray, 14);
2. Array.Sort(bArray);
3. Array.Reverse(bArray);
4. Create array declarations for the following problem specifications.
5. string [ ] fontName = new string [ ] {"Arial", "Times New Roman", "Courier", "Century", "Helvetica"};
6. string [ ] stateName = new string[ ] {"Florida", "Georgia", "Alabama", "Tennessee", "Kentucky", "South Carolina", "North Carolina", "Texas", "Mississippi", "Oklahoma", "Ohio", "Indiana"};
7. char [ ] middleInitial = new char [10];
8. bool [ ] examKey = new bool[15];
9. int [ ] checkNumber = new int[100];

string [ ] checkDate = new string[100];

decimal [ ] checkAmount = new decimal [100];

1. Arrays are passed by reference. The array identifier memory location does not actually contain the values, but rather an address indicating the location of the elements in the array. When you pass an array to a method, by default, you pass a reference to the address of the array elements. The importance of this feature is the fact that if the method changes one or more of the elements, the changes are made to the actual data. Thus, any changes made to array elements in a called method change the same array elements created in the calling method. Not true for pass by value. Under pass by value, a copy is made of the current value. That current value is sent to the method and placed in a different memory location. Any changes made to array elements in a called method DO NOT change the same array elements created in the calling method.