Program 9-17 (continued)

```cpp
97    cout << *(arr[count]) << ");
98    cout << endl;
99 }
```

**Program Output**
The donations, sorted in ascending order, are:
5 5 5 5 5 10 10 10 10 15 25 25 100 100
The donations, in their original order, are:
5 100 25 10 5 25 5 5 100 10 15 10 5 10

---

**Review Questions and Exercises**

**Short Answer**

1. What does the indirection operator do?
2. Look at the following code.
   ```cpp
   int x = 7;
   int *iptr = &x;
   ```
   What will be displayed if you send the expression *iptr to cout? What happens if you send the expression ptr to cout?
3. So far you have learned three different uses for the * operator. What are they?
4. What math operations are allowed on pointers?
5. Assuming that ptr is a pointer to an int, what happens when you add 4 to ptr?
6. Look at the following array definition.
   ```cpp
   int numbers[] = {2, 4, 6, 8, 10};
   ```
   What will the following statement display?
   ```cpp
   cout << *(numbers + 3) << endl;
   ```
7. What is the purpose of the new operator?
8. What happens when a program uses the new operator to allocate a block of memory, but the amount of requested memory isn't available? How do programs written with older compilers handle this?
9. What is the purpose of the delete operator?
10. Under what circumstances can you successfully return a pointer from a function?
11. What is the difference between a pointer to a constant and a constant pointer?
12. What are two advantages of declaring a pointer parameter as a constant pointer?

**Fill-in-the-Blank**

13. Each byte in memory is assigned a unique ________.
14. The ________ operator can be used to determine a variable’s address.

---

**Algorithm Workbench**

24. Look at the following code.
   ```cpp
   double value = 29.7;
   double *ptr = &value;
   ```
   Write a cout statement that uses the ptr variable to display the contents of the value variable.

25. Look at the following array definition.
   ```cpp
   int set[10];
   ```
   Write a statement using pointer notation that stores the value 99 in set[7];

26. Write code that dynamically allocates an array of 20 integers, then uses a loop to allow the user to enter values for each element of the array.

27. Assume that temptab1 is a pointer that points to a dynamically allocated array. Write code that releases the memory used by the array.

28. Look at the following function definition.
   ```cpp
   void getNumber(int &n)
   {
       cout << "Enter a number: ";
       cin >> n;
   }
   ```
   In this function, the parameter n is a reference variable. Rewrite the function so that n is a pointer.

29. Write the definition of ptr, a pointer to a constant int.
30. Write the definition of ptr, a constant pointer to an int.

**True or False**

31. T  F Each byte of memory is assigned a unique address.
32. T  F The * operator is used to get the address of a variable.
33. T  F Pointer variables are designed to hold addresses.
The symbol is called the indirect operator.

When the indirect operator is used with a pointer variable, you are actually working with the value the pointer is pointing to.

Array names cannot be dereferenced with the indirect operator.

When you add a value to a pointer, you are actually adding that number times the size of the data type referenced by the pointer.

The address operator is not needed to assign an array's address to a pointer.

You can change the address that an array name points to.

Pointers may be compared using the relational operators.

When used as function parameters, reference variables are much easier to work with than pointers.

The new operator dynamically allocates memory.

A pointer variable that has not been initialized is called a null pointer.

The address 0 is generally considered unusable.

In using a pointer with the delete operator, it is not necessary for the pointer to have been previously used with the new operator.

Find the Error

Each of the following definitions and program segments has errors. Locate as many as you can.

```c
48. int ptr;
49. int x, *ptr;
   *ptr = 100; // Store 100 in x
50. int k, *ptr;
51. int x, *ptr;
   *ptr = &x;
52. int numbers[] = {10, 20, 30, 40, 50};
   cout << "The third element in the array is ";
   cout << *numbers + 3 << endl;
53. int values[20], *ipter;
   ipter = values + 2;
54. float level;
   int *fpct = &level;
55. int *ipter = &ivalue;
   int *ivalue;
56. void doubleVal(int val)
   { *val = 2; }
```

```c
57. int *pint;
   new pint;
58. int *pint;
   pint = new int;
   if (pint == NULL)
      *pint = 100;
   else
      cout << "Memory allocation error\n";
59. int *pint;
   pint = new int[100]; // Allocate memory
```

Code that processes the array:

```c
   delete pint; // Free memory
60. int getNum()
   { int wholeNum;
   }
```

```c
   cout << "Enter a number: ";
   cin >> wholeNum;
   return wholeNum;
61. const int arr[] = {1, 2, 3};
   int *ptr = arr;
62. void doSomething(int * const ptr)
   { int localArray[] = {1, 2, 3};
   }
```

Programming Challenges

Visit www.myprogramminglab.com to complete many of these Programming Challenges online and get instant feedback.

1. Array Allocator

Write a function that dynamically allocates an array of integers. The function should accept an integer argument indicating the number of elements to allocate. The function should return a pointer to the array.

2. Test Scores #1

Write a program that dynamically allocates an array large enough to hold a user-defined number of test scores. Once all the scores are entered, the array should be passed to a function that sorts them in ascending order. Another function should be called that calculates the average score. The program should display the sorted list of scores and averages with appropriate headings. Use pointer notation rather than array notation whenever possible.

Input Validation: Do not accept negative numbers for test scores.