Sample Array Codes

1. C Program to Calculate Sum & Average of an Array

This is C Program to calculate the sum & average of an array.

Problem Description

This program will declare an array and then add all the array elements and finds the average of the array.

Problem Solution

1. Create an array of fixed size (maximum capacity), let's say 10.
2. Take n, a variable which stores the number of elements of the array, less than maximum capacity of array.
3. Iterate via for loop to take array elements as input, and print them.
4. Iterate via for loop to access each element of array to get sum (sum of positive integers and negative integers are separate) and overall sum to calculate average.
5. Sum of positive and negative numbers are shown.
6. Average is calculated by dividing overall sum to the number of elements in array.

Program/Source Code

Here is source code of the C program to calculate the sum & average of an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1.    /*
2.     * C program to read N integers into an array A and
3.     * a) Find the sum of negative numbers
4.     * b) Find the sum of positive numbers
5.     * c) Find the average of all numbers
6.     * Display the results with suitable headings
7.    */
8.
9.#include  <stdio.h>
10.#define MAXSIZE 10
11.
```

```
12. void main()
13. {
14.
15.         int array[MAXSIZE];
16.        int i, num, negative_sum = 0, positive_sum = 0;
17.        float total = 0.0, average;
18.
19.         printf ("Enter the value of N \n");
20.        scanf("%d", &num);
21.
22.         printf("Enter %d numbers (-ve, +ve and zero) \n", num);
23.
24.         for (i = 0; i < num; i++)
25.         {
26.                 scanf("%d", &array[i]);
27.        }
28.
29.         printf("Input array elements \n");
30.
31.         for (i = 0; i < num; i++)
32.         {
33.                 printf("%+3d\n", array[i]);
34.         }
35.
36.         /*  Summation starts */
37.
38.         for (i = 0; i < num; i++)
39.         {
40.
41.                 if (array[i] < 0)
42.                 {
43.                         negative_sum = negative_sum + array[i];
44.                 }
45.                 else if (array[i] > 0)
46.                 {
47.                         positive_sum = positive_sum + array[i];
48.                 }
49.                 else if (array[i] == 0)
50.                 {
51.                         ;
52.                 }
53.                 total = total + array[i] ;
54.
```

```
55.         }
56.
57.         average = total / num;
58.
59.         printf("\n Sum of all negative numbers =  %d\n", negative_sum);
60.
61.         printf("Sum of all positive numbers =  %d\n", positive_sum);
62.
63.         printf("\n Average of all input numbers =  %.2f\n", average);
64.
65. }
```

Program Explanation

1. Create an array of integer of some certain maximum capacity (10, in this case).
2. From users, take a number N as input, which will indicate the number of elements in the array (N <= maximum capacity).
3. Iterating through for loops (from [0 to N)), take integers as input from user and print them. These inputs are the elements of the array.
4. Now, start summation by iterating through all the elements and adding positive and negative number separately to print as well as combined to calculate average.
5. To calculate average, the overall sum (sum of positive + negative integers) is divided by total number of elements in the array.
6. Print the average calculated.

Runtime Test Cases

```
Enter the value of N
10
Enter 10 numbers (-ve, +ve and zero)
-8
9
-100
-80
90
45
-23
-1
0
16
Input array elements
 -8
 +9
-100
```

```
-80
+90
+45
-23
 -1
 +0
+16

Sum of all negative numbers =  -212
Sum of all positive numbers =  160

Average of all input numbers =  -5.20
```

2. C Program to Find the Largest Two Numbers in a given Array

This is a C Program to calculate the largest of two numbers in a given Array.

Problem Description

This program will implement a one dimensional array, then compares the elements and finds which is the largest two elements in a given array.

Problem Solution

1. Create an array and define the elements of the array.
2. Considering the first element of the array to be the largest number and second element of the array to be the second largest element.
3. Interchange these two numbers if required.
4. Now run the loop from third element of the array to the last element.
5. Scan element of the array, comparing array elements with first largest and second largest numbers, changing both or one if required.
6. At the end, after for loop, we will be getting the actual first largest and second largest number in the array.

Program/Source Code

Here is source code of the C program to calculate the largest of two numbers in a given array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
/*
     * C program to read elements into an array and find the
     * largest two elements in a given array.
*/
#include <stdio.h>
int main (void)
{
 int array[10], n = 0, i = 0, largest1 = 0, largest2 = 0, temp = 0;

        printf ("Enter the size of the array\n");
        scanf ("%d", &n);

        printf ("Enter the elements\n");
        for (i = 0; i < n; i++)
        {
                scanf ("%d", &array[i]);
        }

        printf ("The array elements are : \n");
        for (i = 0; i < n; i++)
        {
                printf ("%d\t", array[i]);
        }

        printf ("\n");

        largest1 = array[0];
        largest2 = array[1];

        if (largest1 < largest2)
        {
                temp = largest1;
                largest1 = largest2;
                largest2 = temp;
        }

```

```
37.         for (int i = 2; i < n; i++)
38.         {
39.                 if (array[i] > largest1)
40.                 {
41.                         largest2 = largest1;
42.                         largest1 = array[i];
43.                 }
44.                 else if (array[i] > largest2 && array[i] != largest1)
45.                 {
46.                         largest2 = array[i];
47.                 }
48.         }
49.
50.         printf ("The FIRST LARGEST = %d\n", largest1);
51.         printf ("THE SECOND LARGEST = %d\n", largest2);
52.
53.         return 0;
54. }
```

Program Explanation

1. Declare an array of some fixed size, say 4.
2. Using for loop, define the elements of the array.
3. Consider the first element of array to be the first largest number (store it in a variable, largest1).
4. Consider the second element of array to be the second largest number (store it in a variable, largest2).
5. Now interchange both if largest1 is smaller than largest2.
6. Run a for loop from third element of the array till the last element of the array, wherein each element will be compared to the largest1.
    - If the array element is greater than largest1, then largest1 is assigned the array element. And the value of largest1 gets transferred to largest2.
    - If the array element is smaller than largest1 and but greater than largest2, then only largest2 is assigned the array element.
7. Running loop till end, will give us the actual first largest and second largest number.

8. Exit

Runtime Test Cases

```
Enter the size of the array
5
Enter the elements
2
4
5
8
7
The array elements are :
2       4       5       8       7
The FIRST LARGEST = 8
THE SECOND LARGEST = 7

Enter the size of the array
6
Enter the elements
2
1
1
2
1
2
The array elements are :
2       1       1       2       1       2
The FIRST LARGEST = 2
THE SECOND LARGEST = 1
```

3. C Program to Find the Second Largest & Smallest Elements in an Array

This C Program finds second largest & smallest elements in an Array.

Problem Description

The program will implement a one-dimensional array and sort the array in descending order. Then it finds the second largest and smallest element in an array and also find the average of these two array elements. Later it checks if the resultant average number is present in a given array. If found, display appropriate message.

Problem Solution

1. Create a one dimensional array and fill its content to its size.

2. Arrange the array elements in the descending order.
3. The second largest number would be the 2nd element of array, whereas the second smallest number would be the last second element of array.
4. Take average of these two numbers.
5. Now, look for this average number in the array.

Program/Source Code

Here is source code of the C program to find the second largest & smallest elements in an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1.
2.    /*
3.     * C program to accept a list of data items and find the second largest
4.     * and smallest elements in it. Compute the average of both and search
5.     * for the average value if it is present in the array.
6.     * Display appropriate message on successful search.
7.    */
8.
9.    #include <stdio.h>
10.    void main ()
11.    {
12.
13.        int number[30];
14.        int i, j, a, n, counter, average;
15.
16.        printf("Enter the value of N\n");
17.        scanf("%d", &n);
18.
19.        printf("Enter the numbers \n");
20.        for (i = 0; i < n; ++i)
21.            scanf("%d", &number[i]);
22.
23.        for (i = 0; i < n; ++i)
24.        {
25.            for (j = i + 1; j < n; ++j)
26.            {
27.                if (number[i] < number[j])
28.                {
29.                    a = number[i];
30.                    number[i] = number[j];
```

```
                    number[j] = a;
                }
            }

        }

        printf("The numbers arranged in descending order are given below \n"
    );

        for (i = 0; i < n; ++i)
        {
            printf("%d\n", number[i]);
        }

        printf("The 2nd largest number is  = %d\n", number[1]);
        printf("The 2nd smallest number is = %d\n", number[n - 2]);

        average = (number[1] + number[n - 2]) / 2;
        counter = 0;

        for (i = 0; i < n; ++i)
        {
            if (average == number[i])
            {
                ++counter;
            }
        }

        if (counter == 0 )
            printf("The average of %d  and %d is = %d is not in the array \n
    ",
            number[1], number[n - 2], average);

        else
            printf("The average of %d  and %d in array is %d in numbers \n",
            number[1], number[n - 2], counter);
    }
```

Program Explanation

1. Declare an array, a of some fixed capacity, 30.
2. Take the size of the array as input from users.

3. Define all the elements of the array using for loop.
4. Sort the element of the array using Insertion Sort in descending order.
5. The second largest number would be the second element of array (array[1]) and second smallest number would be the second last element of array (array[n-2]).
6. Take the average of these two numbers by adding both and dividing by 2.
7. Using for loop, scan each and every element of the array to check whether the element equals the average.
8. If the average is present in the array, then print appropriate message.
9. Exit

Runtime Test Cases

```
Enter the value of N
4
Enter the numbers
450
340
120
670
The numbers arranged in descending order are given below
670
450
340
120
The 2nd largest number is  = 450
The 2nd smallest number is = 340
The average of 450  and 340 is = 395 is not in the array
```

4. C Program to Sort the Array in an Ascending Order

This is a C Program to sort an array in ascending order.

Problem Description

This program will implement a one-dimensional array of some fixed size, filled with some random numbers, then will sort all the filled elements of the array.

Problem Solution

1. Create an array of fixed size (maximum capacity), lets say 10.
2. Take n, a variable which stores the number of elements of the array, less than maximum capacity of array.

3. Iterate via for loop to take array elements as input, and print them.
4. The array elements are in unsorted fashion, to sort them, make a nested loop.
5. In the nested loop, each element will be compared to all the elements below it.
6. In case the element is greater than the element presents below it, then they are interchanged.
7. After executing the nested loop, we will obtain an array in ascending order arranged elements.

Program/Source Code

Here is source code of the C program to sort the array in an ascending order. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1.
2.    /*
3.     * C program to accept N numbers and arrange them in an ascending order
4.     */
5.
6.    #include <stdio.h>
7.    void main()
8.    {
9.
10.        int i, j, a, n, number[30];
11.        printf("Enter the value of N \n");
12.        scanf("%d", &n);
13.
14.        printf("Enter the numbers \n");
15.        for (i = 0; i < n; ++i)
16.            scanf("%d", &number[i]);
17.
18.        for (i = 0; i < n; ++i)
19.        {
20.
21.            for (j = i + 1; j < n; ++j)
22.            {
23.
24.                if (number[i] > number[j])
25.                {
```

```
26.
27.                     a =  number[i];
28.                     number[i] = number[j];
29.                     number[j] = a;
30.
31.                 }
32.
33.             }
34.
35.         }
36.
37.         printf("The numbers arranged in ascending order are given below \n")
    ;
38.         for (i = 0; i < n; ++i)
39.             printf("%d\n", number[i]);
40.
41.     }
```

Program Explanation

1. Declare an array of some fixed capacity, let's say 30.
2. From users, take a number N as input, which will indicate the number of elements in the array (N <= maximum capacity)
3. Iterating through for loops (from [0 to N)), take integers as input from user and print them. These inputs are the elements of the array.
4. Now, create a nested for loop with i and j as iterators.
5. Start the sorting in ascending order by extracting each element at position i of outer loop.
6. This element is being compared to every element from position i+1 to size-1 (means all elements present below this extracted element)
7. In case any of the extracted element is greater than the element below it, then these two interchanges their position, else the loop continues.
8. After this nested loop gets executed, we get all the elements of the array sorted in ascending order.

Runtime Test Cases

```
Enter the value of N
6
Enter the numbers
3
78
90
456
780
200
The numbers arranged in ascending order are given below
3
78
90
200
456
780
```