

This screenshot shows a C program in a code editor. The program defines an array of 5 integers: {20, 30, 40, 50, 100}. It then iterates through the array, comparing each element to a variable 'max' which is initialized to the first element (20). The maximum value found is 100, which is printed to the console. The output window shows the result and execution time.

```
1 #include<stdio.h>
2 main()
3 {
4     int arr[5]={20,30,40,50,100};
5     int max=arr[0];
6     int i;
7     for(i=0;i<5;i++)
8     {
9         if(arr[i]>max)
10        {
11            max=arr[i];
12        }
13    }
14    printf("maximum number%d",max);
15 }
16 }
```

maximum number100

Process exited after 0.03657 seconds with return value 17
Press any key to continue . . .

This screenshot shows a C program that finds the largest and second largest elements in an array. It uses a bubble sort algorithm to sort the array in descending order. The first and second elements of the sorted array are the largest and second largest elements, respectively. The program prompts the user to enter the number of elements and the elements themselves. The output shows the sorted array and the identified largest and second largest elements.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int n, i, j, temp;
6     printf("Enter the number of elements in the array: ");
7     scanf("%d", &n);
8     int arr[n];
9     printf("Enter the elements of the array:\n");
10    for(i=0; i<n; i++) {
11        scanf("%d", &arr[i]);
12    }
13    // Sorting the array in descending order using bubble sort
14    for(i=0; i<n-1; i++) {
15        for(j=0; j<n-i-1; j++) {
16            if(arr[j] < arr[j+1]) {
17                temp = arr[j];
18                arr[j] = arr[j+1];
19                arr[j+1] = temp;
20            }
21        }
22    }
23    printf("The first largest element in the array is %d\n", arr[0]);
24    // Finding the second largest element in the array
25    for(i=1; i<n; i++) {
26        if(arr[i] < arr[0]) {
27            printf("The second largest element in the array is %d\n", arr[i]);
28            break;
29        }
30    }
31    return 0;
32 }
```

Enter the number of elements in the array: 5
Enter the elements of the array:
1 3 4 5 6
The first largest element in the array is 1728389152
The second largest element in the array is 6487504

Process exited after 15.28 seconds with return value 0
Press any key to continue . . .

This screenshot shows a C program that finds the average of the two largest elements in an array. It uses a bubble sort algorithm to sort the array in descending order. The first and second elements of the sorted array are the largest and second largest elements, respectively. The program calculates the average of these two elements and checks if it is present in the array. The output shows the sorted array, the average, and whether it is present in the array.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int n, i, j, temp;
6     printf("Enter the number of elements in the array (maximum 10): ");
7     scanf("%d", &n);
8     int arr[n];
9     printf("Enter the elements of the array:\n");
10    for(i=0; i<n; i++) {
11        scanf("%d", &arr[i]);
12    }
13    // Sorting the array in descending order using bubble sort
14    for(i=0; i<n-1; i++) {
15        for(j=0; j<n-i-1; j++) {
16            if(arr[j] < arr[j+1]) {
17                temp = arr[j];
18                arr[j] = arr[j+1];
19                arr[j+1] = temp;
20            }
21        }
22    }
23    printf("The sorted array in descending order is:\n");
24    for(i=0; i<n; i++) {
25        printf("%d ", arr[i]);
26    }
27    printf("\n");
28    // Find second largest and smallest elements
29    int second_largest = arr[1];
30    int second_smallest = arr[n-2];
31    printf("The second largest element is: %d\n", second_largest);
32    printf("The second smallest element is: %d\n", second_smallest);
33    // calculate the average of the two elements
34    avg = (second_largest + second_smallest) / 2.0;
35    // check if the average is present in the array
36    for(i=0; i<n; i++) {
37        if(arr[i] == avg) {
38            printf("The average number %.2f is found in the array.\n", avg);
39            break;
40        }
41    }
42    if(i == n) {
43        printf("The average number %.2f is not found in the array.\n", avg);
44    }
45    return 0;
46 }
```

Enter the number of elements in the array (maximum 10): 5
Enter the elements of the array:
1
2
3
4
5
The sorted array in descending order is:
5 4 3 2 1
The second largest element is: 4
The second smallest element is: 2
The average number 3.00 is found in the array.

Process exited after 18.09 seconds with return value 0
Press any key to continue . . .

```
#include <stdio.h>

int main()
{
    int array[] = {10, 15, 90, 200, 110};
    int n = sizeof(array) / sizeof(int);
    int max_diff = 0, i, j;

    // find the maximum difference between any two elements in the array
    for (i = 0; i < n-1; i++) {
        for (j = i+1; j < n; j++) {
            int diff = array[j] - array[i];
            if (diff > max_diff) {
                max_diff = diff;
            }
        }
    }

    printf("Maximum difference is %d\n", max_diff);

    return 0;
}
```

Maximum difference is 190

Process exited after 0.0459 seconds with return value 0
Press any key to continue . . .

```
int arr[] = {1, 2, 4, 5, 4, 2, 7, 5};
int n = sizeof(arr) / sizeof(int);
int i, j, k;

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

// remove duplicate elements
for (i = 0; i < n-1; i++) {
    for (j = i+1; j < n; j++) {
        if (arr[i] == arr[j]) {
            for (k = j; k < n; k++) {
                arr[k] = arr[k+1];
            }
            n--;
            j++;
        }
    }
}

printf("Resultant Array after removing duplicates: ");
for (i = 0; i < n; i++) {
    printf("%d ", arr[i]);
}
printf("\n");

return 0;
```

Input Array: 1 2 4 5 4 2 7 5
Resultant Array after removing duplicates: 1 2 4 5 7

Process exited after 0.08314 seconds with return value 0
Press any key to continue . . .

```
// create separate arrays for even and odd elements
int even_arr[even_count], odd_arr[odd_count];
j = 0;
k = 0;

for (i = 0; i < n; i++) {
    if (arr[i] % 2 == 0) {
        even_arr[j] = arr[i];
        j++;
    } else {
        odd_arr[k] = arr[i];
        k++;
    }
}

// print the even elements array
printf("Even elements array: ");
for (i = 0; i < even_count; i++) {
    printf("%d ", even_arr[i]);
}
printf("\n");

// print the odd elements array
printf("Odd elements array: ");
for (i = 0; i < odd_count; i++) {
    printf("%d ", odd_arr[i]);
}
printf("\n");

return 0;
```

Even elements array: 2 4 6 8
Odd elements array: 1 3 5 7 9

Process exited after 0.09489 seconds with return value 0
Press any key to continue . . .



