

```
1  /* Find the kth smallest element in an array using quicksort. */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  #define MAX 100
7
8  int partition(int *arr, int low, int high)
9  {
10     int pivot = arr[low], i = low + 1, j = high, temp;
11     do
12     {
13         while (arr[i] <= pivot)
14         {
15             i++;
16         }
17         while (arr[j] > pivot)
18         {
19             j--;
20         }
21         if (i < j)
22         {
23             temp = arr[i];
24             arr[i] = arr[j];
25             arr[j] = temp;
26         }
27     } while (i <= j);
28     temp = arr[j];
29     arr[j] = arr[low];
30     arr[low] = temp;
31     return j;
32 }
```

```

v int quicksortksmall(int *arr, int low, int high, int k)
{
    int indexpos;
v   if (low < high)
    {
v       indexpos = partition(arr, low, high);
        if (k - 1 == indexpos)
        {
            return arr[k];
        }
v       else if (k - 1 < indexpos)
        {
            quicksortksmall(arr, low, indexpos - 1, k);
        }
v       else
        {
            quicksortksmall(arr, indexpos + 1, high, k);
        }
    }
}

v int main()
{
    int ch;
v   while (1)
    {
        printf("\nEnter your choice :--\n1 - Find kth smallest element using quick sort\n2 - Exit\nChoice : ");
        scanf("%d", &ch);
        switch (ch)
        {
v         case 1:
            {
                int n, k;
                printf("\nEnter size of array : ");
                scanf("%d", &n);
                int arr[n];
                printf("Enter %d numbers : ", n);
v               for (int _ = 0; _ < n; _++)
                {
                    scanf("%d", &arr[_]);
                }
                printf("Enter k : ");
                scanf("%d", &k);
                printf("%d th smallest element : %d", k, quicksortksmall(arr, 0, n - 1, k));
                break;
            }
v         case 2:
            exit(0);
        default:
            printf("INVALID INPUT - TRY AGAIN.\n");
        }
    }
    return 0;
}

```

Enter your choice :--

1 - Find kth smallest element using quick sort

2 - Exit

Choice : 1

Enter size of array : 10

Enter 10 numbers : 10 2 5 1 0 3 10 3 7 9

Enter k : 2

2 th smallest element : 1

Enter your choice :--

1 - Find kth smallest element using quick sort

2 - Exit

Choice : 2

PS C:\Users\shuvr\OneDrive\Documents\CODING\College C codes>

SEM-2 > DSA-ASS-11 > 11_2_count_sort.c > maximum(int *, int)

```
1  /* Given an array of non-negative integers, implement the Counting Sort algorithm to sort the array in ascending order. */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  void display(int *arr, int n)
7  {
8      for (int _ = 0; _ < n; _++)
9      {
10         printf("%d ", arr[_]);
11     }
12     printf("\n");
13 }
14
15 int maximum(int *arr, int n)
16 {
17     int max = 0;
18     for (int i = 0; i < n; i++)
19     {
20         if (arr[i] > max)
21         {
22             max = arr[i];
23         }
24     }
25     return max;
26 }
```

```

28 void countsort(int *arr, int n)
29 {
30     int max = maximum(arr, n);
31     int count[max + 1];
32     for (int i = 0; i <= max; i++)
33     {
34         count[i] = 0;
35     }
36     for (int i = 0; i < n; i++)
37     {
38         count[arr[i]]++;
39     }
40     int icount = 0, imain = 0;
41     while (icontains <= max)
42     {
43         if (count[icontains] > 0)
44         {
45             arr[imain++] = icount;
46             count[icontains]--;
47         }
48         else
49         {
50             icount++;
51         }
52     }
53 }
54
55 int main()
56 {
57     int ch;
58     while (1)
59     {
60         printf("\nEnter your choice :--\n1 - Count sort given array\n2 - Exit\nChoice : ");
61         scanf("%d", &ch);
62         switch (ch)
63         {
64             case 1:
65             {

```

```

64     case 1:
65     {
66         int n;
67         printf("\nEnter size of array : ");
68         scanf("%d", &n);
69         int arr[n];
70         printf("Enter %d numbers : ", n);
71         for (int _ = 0; _ < n; _++)
72         {
73             scanf("%d", &arr[_]);
74         }
75         countsort(arr, n);
76         printf("Sorted Array : ");
77         display(arr, n);
78         break;
79     }
80     case 2:
81         exit(0);
82     default:
83         printf("INVALID INPUT - TRY AGAIN.\n");
84     }
85 }
86 return 0;
87 }

```

TERMINAL COMMENTS

Enter your choice :--

1 - Count sort given array

2 - Exit

Choice : 1

Enter size of array : 10

Enter 10 numbers : 10 2 5 1 0 3 10 3 7 9

Sorted Array : 0 1 2 3 3 5 7 9 10 10

Enter your choice :--

1 - Count sort given array

2 - Exit

Choice : 2

PS C:\Users\shuvr\OneDrive\Documents\CODING\College C codes>

SEM-2 > DSA-ASS-11 > C 11_3_radix_sort.c > main()

```
1  /* Given an array of non-negative integers, implement the Radix Sort algorithm to sort the array in ascending order. */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  void display(int *arr, int n)
7  {
8      for (int _ = 0; _ < n; _++)
9      {
10         printf("%d ", arr[_]);
11     }
12     printf("\n");
13 }
14
15 int maximum(int *arr, int n)
16 {
17     int max = 0;
18     for (int i = 0; i < n; i++)
19     {
20         if (arr[i] > max)
21         {
22             max = arr[i];
23         }
24     }
25     return max;
26 }
27
```

```

27
28 void countsort(int *arr, int n, int exp)
29 {
30     int output[n];
31     int i, count[10] = {0};
32     for (i = 0; i < n; i++)
33     {
34         count[(arr[i] / exp) % 10]++;
35     }
36     for (i = 1; i < 10; i++)
37     {
38         count[i] += count[i - 1];
39     }
40     for (i = n - 1; i >= 0; i--)
41     {
42         output[count[(arr[i] / exp) % 10] - 1] = arr[i];
43         count[(arr[i] / exp) % 10]--;
44     }
45     for (i = 0; i < n; i++)
46     {
47         arr[i] = output[i];
48     }
49 }
50
51 void radixsort(int *arr, int n)
52 {
53     int m = maximum(arr, n);
54     for (int exp = 1; m / exp > 0; exp *= 10)
55         countsort(arr, n, exp);
56 }
57
58 int main()
59 {
60     int ch;
61     while (1)
62     {
63         printf("\nEnter your choice :--\n1 - Radix sort given array\n2 - Exit\nChoice : ");
64         scanf("%d", &ch);
65         switch (ch)
66         {

```



```

66     {
67         case 1:
68         {
69             int n;
70             printf("\nEnter size of array : ");
71             scanf("%d", &n);
72             int arr[n];
73             printf("Enter %d numbers : ", n);
74             for (int _ = 0; _ < n; _++)
75             {
76                 scanf("%d", &arr[_]);
77             }
78             radixsort(arr, n);
79             printf("Sorted Array : ");
80             display(arr, n);
81             break;
82         }
83         case 2:
84             exit(0);
85         default:
86             printf("INVALID INPUT - TRY AGAIN.\n");
87     }
88 }
89 return 0;
90 }

```

TERMINAL COMMENTS

Enter your choice :--

1 - Radix sort given array

2 - Exit

Choice : 1

Enter size of array : 10

Enter 10 numbers : 10 2 5 1 0 3 10 3 7 9

Sorted Array : 0 1 2 3 3 5 7 9 10 10

Enter your choice :--

1 - Radix sort given array

2 - Exit

Choice : 2

PS C:\Users\shuvr\OneDrive\Documents\CODING\College C codes>