Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Sorting using qsort() in C/5a008e23cbb2fe34b7774fe0

TUTORIAL

Sorting using qsort() in C

Also in C language we already have a sort function named qsort() for sorting facility in stdlib.h file. The prototype is: -

```
void qsort(void *base, size_t no_of_members, size_t
size_of_one_member, int(*compare)(const void *, const void
*));
```

Here you only need to pass the compare function to it, and it will sort all type of variables and in all possible directions. The main point is that compare function should take void pointers as arguments, and before returning, it have to typecast them to required type. For example, you can write the sort program as below also: -

C

```c
#include<stdio.h>
#include<stdlib.h>

int int_ascending(const void *one, const void
*two)
{
  int first = *(int *)one;        // Typecasting
the void * to int variable
  int second = *(int*)two;        // Typecasting
the void * to int variable
  if ( first < second )    return -1;
  else if ( first == second ) return 0;
  else  return 1;
}

int int_descending(const void *one, const void
*two)
{
```

```
15    int first = *(int *)one;        // Typecasting
      the void * to int variable
16    int second = *(int*)two;        // Typecasting
      the void * to int variable
17    if ( first < second )    return 1;
18    else if ( first == second ) return 0;
19    else  return -1;
20  }
21
22  int int_abs_ascending(const void *one, const void
      *two)
23  {
24    int first = *(int *)one;        // Typecasting
      the void * to int variable
25    int second = *(int*)two;        // Typecasting
      the void * to int variable
26    if ( abs(first) < abs(second) )    return -1;
27    else if ( abs(first) == abs(second) ) return 0;
28    else  return 1;
29  }
30
31  int int_abs_descending(const void *one, const void
      *two)
32  {
33    int first = *(int *)one;        // Typecasting
      the void * to int variable
34    int second = *(int*)two;        // Typecasting
      the void * to int variable
35    if ( abs(first) < abs(second) )    return 1;
36    else if ( abs(first) == abs(second) ) return 0;
37    else  return -1;
38  }
39
40  int main()
41  {
42    int i, a[]={3,-6,5,-4,8,1,-9,2,2,43};
43
44    printf("Before sorting\n");
45    for(i=0;i<10;i++)
46      printf("%d ",a[i]);
47
48    qsort(a, 10 , sizeof(int), int_ascending);
     //call to qsort with function pointer
49
50    printf("\nAfter sorting Ascending\n");
```

```
  for(i=0;i<10;i++)
    printf("%d ",a[i]);

  qsort(a, 10 , sizeof(int), int_descending);
  //call to qsort with function pointer

  printf("\nAfter sorting Descending\n");
  for(i=0;i<10;i++)
    printf("%d ",a[i]);

  qsort(a, 10 , sizeof(int), int_abs_ascending);
 //call to qsort with function pointer

  printf("\nAfter sorting Ascending for absolute values\n");
  for(i=0;i<10;i++)
    printf("%d ",a[i]);

  qsort(a, 10 , sizeof(int), int_abs_descending);
  //call to qsort with function pointer

  printf("\nAfter sorting Descending for absolute values\n");
  for(i=0;i<10;i++)
    printf("%d ",a[i]);

  return 0;
}
```

Output:

```
Before sorting
3 -6 5 -4 8 1 -9 2 2 43
After sorting Ascending
-9 -6 -4 1 2 2 3 5 8 43
After sorting Descending
43 8 5 3 2 2 1 -4 -6 -9
After sorting Ascending for absolute values
1 2 2 3 -4 5 -6 8 -9 43
After sorting Descending for absolute values
43 -9 8 -6 5 -4 3 2 2 1
```

So, in general void pointers and pointers to functions are very useful utilities in language, use them very carefully.