Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Sorting using Function Pointers in C/5a008d9acbb2fe34b7774fd8

TUTORIAL

Sorting using Function Pointers in C

Topics

1.4 Four different sorts used with same sort function

This is useful because functions encapsulate behavior. For instance, every time you need a particular behavior such as drawing a line, instead of writing out a bunch of code, all you need to do is call the function. But sometimes you would like to choose different behaviors at different times in essentially the same piece of code. For example, If you were to write a sort routine, you might want to allow the function's caller to choose the order in which the data is sorted; some programmers might need to sort the data in ascending order, others might prefer descending order while still others may want something similar to but not quite like one of those choices. One way to let your user specify what to do is to provide a flag as an argument to the function, but this is inflexible; the sort function allows only a fixed set of comparison types (e.g., ascending and descending). A much nicer way of allowing the user to choose how to sort the data is simply to let the user pass in a function to the sort function. This function might take two pieces of data and perform a comparison on them. For example the following program is used to sort n integers: -

```c
#include<stdio.h>

int sort(int a[10])
{
  int i,j,t;
  for(i=0;i<10;i++)
```

```
  {
    for(j=i;j<10;j++)
    {
      if(a[i]>a[j])
      {
        t=a[i];
        a[i]=a[j];
        a[j]=t;
      }
    }
  }
}

int main()
{
  int i,a[]={3,6,5,4,8,1,9,2,22,43};
  printf("Before sorting\n");
  for(i=0;i<10;i++)
    printf("%d ",a[i]);

  sort(a);

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

  return 0;
}

```

Output:

```
Before sorting
3 6 5 4 8 1 9 2 22 43
After sorting
1 2 3 4 5 6 8 9 22 43
```

Now, if we have to change the way of sorting to descending, then we have to write the sort function again with only one difference of

sign in if statement. As the remaining logic is same, only thing that differ is the comparison condition. In this case, we can use function pointer to reuse the rest of the function, and to define the comparing condition only in different function. For example, following program has four different sorts used with same sort function: -

Four different sorts used with same sort function

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

int int_ascending(int first, int second)
{
  if ( first < second )    return -1;
  else if ( first == second ) return 0;
  else  return 1;
}

int int_descending(int first, int second)
{
  if ( first < second )    return 1;
  else if ( first == second ) return 0;
  else  return -1;
}

int int_abs_ascending(int first, int second)
{
  if ( abs(first) < abs(second) )    return -1;
  else if ( abs(first) == abs(second) ) return 0;
  else  return 1;
}

int int_abs_descending(int first, int second)
{
  if ( abs(first) < abs(second) )    return 1;
  else if ( abs(first) == abs(second) ) return 0;
  else  return -1;
}

```

```
int sort(int a[10],int (*compare)(int,int))
{
  int i,j,t;
  for(i=0;i<10;i++)
  {
    for(j=i;j<10;j++)
    {
      if(compare(a[i],a[j]) > 0)
      {
        t=a[i];
        a[i]=a[j];
        a[j]=t;
      }
    }
  }
}

int main()
{
  int i,a[]={3,-6,5,-4,8,1,-9,2,2,43};

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

  sort(a,int_ascending);

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

  sort(a,int_descending);

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

  sort(a,int_abs_ascending);

```

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

  sort(a,int_abs_descending);

  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
```

In the above function, same sort function will be reused by just making the compare function differ. This can be achieved only through function pointers.