**Sorting Programs of C**

Program 1 :

Bubble sort is also known as sinking sort. This algorithm compares each pair of adjacent items and swaps them if they are in the wrong order, and this same process goes on until no swaps are needed. In the following program we are implementing bubble sort in C language. In this program user would be asked to enter the number of elements along with the element values and then the program would sort them in ascending order by using bubble sorting algorithm logic.

#### Implementing bubble sort algorithm in a C program

/\* Implementing Bubble sort in a C Program

\*/

#include<stdio.h>

int main(){

int count, temp, i, j, number[30];

printf("How many numbers are u going to enter?: ");

scanf("%d",&count);

printf("Enter %d numbers: ",count);

for(i=0;i<count;i++)

scanf("%d",&number[i]);

/\* This is the main logic of bubble sort algorithm

\*/

for(i=count-2;i>=0;i--){

for(j=0;j<=i;j++){

if(number[j]>number[j+1]){

temp=number[j];

number[j]=number[j+1];

number[j+1]=temp;

}

}

}

printf("Sorted elements: ");

for(i=0;i<count;i++)

printf(" %d",number[i]);

return 0;

}

Program 2 :

Insertion sort algorithm picks elements one by one and places it to the right position where it belongs in the sorted list of elements. In the following C program we have implemented the same logic.

Before going through the program, lets see the steps of insertion sort with the help of an example.  
Input elements: 89  17  8  12  0  
Step 1: 89  17  8  12  0 (the bold elements are sorted list and non-bold unsorted list)  
Step 2: 17  89  8  12  0 (each element will be removed from unsorted list and placed at the right position in the sorted list)  
Step 3: 8 17 89 12 0  
Step 4: 8 12 17 89 0  
Step 5: 0 8 12 17 89

#### C Program – Insertion Sort implementation

#include<stdio.h>

int main(){

/\* Here i & j for loop counters, temp for swapping,

\* count for total number of elements, number[] to

\* store the input numbers in array. You can increase

\* or decrease the size of number array as per requirement

\*/

int i, j, count, temp, number[25];

printf("How many numbers u are going to enter?: ");

scanf("%d",&count);

printf("Enter %d elements: ", count);

// This loop would store the input numbers in array

for(i=0;i<count;i++)

scanf("%d",&number[i]);

// Implementation of insertion sort algorithm

for(i=1;i<count;i++){

temp=number[i];

j=i-1;

while((temp<number[j])&&(j>=0)){

number[j+1]=number[j];

j=j-1;

}

number[j+1]=temp;

}

printf("Order of Sorted elements: ");

for(i=0;i<count;i++)

printf(" %d",number[i]);

return 0;

}

Program 3:

In Selection sort, the smallest element is exchanged with the first element of the unsorted list of elements (the exchanged element takes the place where smallest element is initially placed). Then the second smallest element is exchanged with the second element of the unsorted list of elements and so on until all the elements are sorted. In the following C program we have implemented the same logic.

Before going through the program, lets see the steps of selection sort with the help of an example:  
Entered elements: 22 0 -90 89 17  
Step 1: -90 0 22 89 17 (22 and -90 exchanged position)  
Step 2: -90 0 22 89 17 (0 is at right place, no exchange needed)  
Step 3: -90 0 17 89 22 (22 and 17 exchanged position)  
Step 4: -90 0 17 22 89 (89 and 22 exchanged position)

#### C Program – Selection sort

#include<stdio.h>

int main(){

/\* Here i & j for loop counters, temp for swapping,

\* count for total number of elements, number[] to

\* store the input numbers in array. You can increase

\* or decrease the size of number array as per requirement

\*/

int i, j, count, temp, number[25];

printf("How many numbers u are going to enter?: ");

scanf("%d",&count);

printf("Enter %d elements: ", count);

// Loop to get the elements stored in array

for(i=0;i<count;i++)

scanf("%d",&number[i]);

// Logic of selection sort algorithm

for(i=0;i<count;i++){

for(j=i+1;j<count;j++){

if(number[i]>number[j]){

temp=number[i];

number[i]=number[j];

number[j]=temp;

}

}

}

printf("Sorted elements: ");

for(i=0;i<count;i++)

printf(" %d",number[i]);

return 0;

}

Program 4 :

Quicksort is a divide and conquer algorithm. The steps are: 1) Pick an element from the array, this element is called as pivot element. 2) Divide the unsorted array of elements in two arrays with values less than the pivot come in the first sub array, while all elements with values greater than the pivot come in the second sub-array (equal values can go either way). This step is called the partition operation. 3) Recursively repeat the step 2(until the sub-arrays are sorted) to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values. The same logic we have implemented in the following C program.

#### C Program – Quicksort algorithm implementation

#include<stdio.h>

void quicksort(int number[25],int first,int last){

int i, j, pivot, temp;

if(first<last){

pivot=first;

i=first;

j=last;

while(i<j){

while(number[i]<=number[pivot]&&i<last)

i++;

while(number[j]>number[pivot])

j--;

if(i<j){

temp=number[i];

number[i]=number[j];

number[j]=temp;

}

}

temp=number[pivot];

number[pivot]=number[j];

number[j]=temp;

quicksort(number,first,j-1);

quicksort(number,j+1,last);

}

}

int main(){

int i, count, number[25];

printf("How many elements are u going to enter?: ");

scanf("%d",&count);

printf("Enter %d elements: ", count);

for(i=0;i<count;i++)

scanf("%d",&number[i]);

quicksort(number,0,count-1);

printf("Order of Sorted elements: ");

for(i=0;i<count;i++)

printf(" %d",number[i]);

return 0;

}