```
/*
                        Experiment No: 02

            Author:Harsheet Sujit Chordiya
            Roll No.: 37
            Sem & Sec: CSE 3B
            Source file: expt02.c
            Date Compiled: Dec 2021

=============================================================================
        Aim: To study and implement basic comparative
sorting methods – Selection Sort,
             Insertion Sort, Shell Sort and Counting Sort.
        Problem Statement: Write a generalized function
that takes a parameter to indicate the mode
                           (say 1 for decreasing order, 2
for increasing order, 3 for increasing order
                           with the Nth element out of
order, 4 for a randomly generated element values),
                           to create a list of elements.
The parameter indicating the number of elements in
                           a statically allocated array
(the maximum size is large enough to run possible
                           iterations to test the time
complexity, say 1000000) will be multiple of 10.
                           Also write appropriate functions
to create a copy of the list and to display
                           the list contents.
                           Using above functions, write a
menu-driven C program to order the list in ascending
                           sequence using - the Selection
Sort, the Insertion Sort, the Counting Sort
                           and the Shell Sort..
=============================================================================
                                  THEORY
=============================================================================
Time Complexity is defined as the number of times a
particular instruction set is executed rather
than the total time is taken. It is because the total time
took also depends on some external factors
 like the compiler used, processor's speed, etc.
1.Selection Sort:
   Selection sort is a sorting algorithm that is
independent of the original order of elements in the array.
   In Pass 1, selecting the element with the smallest value
calls for scanning all n elements; thus,
   n–1 comparisons are required in the first pass. Then,
the smallest value is swapped with the element in
   the first position. In Pass 2, selecting the second
smallest value requires scanning the remaining n – 1
   elements and so on. Therefore,
   (n – 1) + (n – 2) + ... + 2 + 1= n(n – 1) / 2 = O(n2)
comparisons.
2.Insertion Sort:
```

```
   For insertion sort, the best case occurs when the array
is already sorted. In this case, the running time
   of the algorithm has a linear running time (i.e., O(n)).
This is because, during each iteration, the first
   element from the unsorted set is compared only with the
last element of the sorted set of the array.
   Similarly, the worst case of the insertion sort
algorithm occurs when the array is sorted in the
   reverse order. In the worst case, the first element of
the unsorted set has to be compared with
   almost every element in the sorted set. Furthermore,
every iteration of the inner loop will have
   to shift the elements of the sorted set of the array
before inserting the next element. Therefore, in
   the worst case, insertion sort has a quadratic running
time (i.e., O(n2)).
   Even in the average case, the insertion sort algorithm
will have to make at least (K–1)/2 comparisons.
   Thus, the average case also has a quadratic running time.
3. Counting Sort:
   Counting sort algorithm is a non comparison based
sorting algorithm i.e the arrangement of elements in the
array does not affect the flow of algorithm. No matter if
the elements in the array are already sorted, reverse
sorted or randomly sorted, the algorithm works the same for
all these cases and thus the time complexity for all such
cases is same i.e O(n+k).
4. Shell Sort:
   Time complexity of above implementation of shellsort is
O(n2).
================================================================================
                                ALGOROTHMS
================================================================================
1.Insertion sort:
   Step 1: Repeat Steps 2 to 5 forK=1toN–1
   Step 2: SET TEMP = ARR[K]
   Step 3: SETJ=K-1
   Step 4: Repeat while TEMP <= ARR[J]
   SET ARR[J + 1] = ARR[J]
      SETJ=J-1
         [END OF INNER LOOP]
   Step 5: SET ARR[J + 1] = TEMP
      [END OF LOOP]
   Step 6: EXIT

2. Selection sort:
   Step 1: [INITIALIZE] SET SMALL = ARR[K]
   Step 2: [INITIALIZE] SET POS=K
   Step 3: Repeat forJ= K+1 to N
      IF SMALL > ARR[J]
      SET SMALL = ARR[J]
      SET POS=J
      [END OF IF]
      [END OF LOOP]
```

```
    Step 4: RETURN POS

3.Counting Sort:
   Step 1:Initialize Array with 0
   Step 2:   max = get maximum element from array.
      define count array of size [max+1]

   Step 3:   for i := 0 to max
   do
      count[i] = 0
Step 4:
   for i := 1 to size
   do
   increase count of each number
      for i := 1 to max
   do
   count[i] = count[i] + count[i+1]
   Step 5:
   for i := size to 1 decrease by 1
   do
   store the number in the output array
      decrease count[i]
   Step 6: Return the output array
         End

4. Shell sort:
   Step 1: SET FLAG=1, GAP_SIZE=N
   Step 2: Repeat Steps 3 to 6 while FLAG=1OR GAP_SIZE>1
   Step 3: SET FLAG =
   Step 4: SET GAP_SIZE = (GAP_SIZE + 1) / 2
   Step 5: Repeat Step 6 forI= to I < (N - GAP_SIZE)
   Step 6: IF Arr[I + GAP_SIZE] > Arr[I]
      SWAP Arr[I + GAP_SIZE], Arr[I]
      SET FLAG =0
   Step 7: END
================================================================================
                                        CODE
================================================================================
*/
//Header file Inclusion
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
//Pre-Processor Directives
#define INF 999999
// User Defined Function declarations
void ascending_list(int *,int);
void descending_list(int *,int);
void part_sort_list(int *,int);
void random_list(int *,int);
int counting_list(int *,int,int,int);
void copy(int *,int *,int);
void selection(int *,int);
void insertion(int *,int);
```

```
void shell(int *,int);
void counting(int *,int,int);
void display(int *,int);
void swap(int* ,int*);
//The Driver Function
void main()
{
  int arr[INF],arr2[INF],len,key,choice,max,min,k;
  time_t t1,t2;
  do{
     do{
        printf("Enter the Array length (in multiple of 10) ?
\n ");
        scanf("%d",&len);
     }while(len%10 !=0);
      do{
           printf("Select list : 1.Ascending list
2.Descending list 3.Partial sorted list 4.Random list\n");
           scanf("%d",&choice);
           switch(choice){
               case 1:
                 ascending_list(arr,len);
                 break;
               case 2:
                  descending_list(arr,len);
                  break;
               case 3:
                  part_sort_list(arr,len);
                  break;
               case 4:
                  random_list(arr,len);
                  break;
            }
        }while(choice<1 || choice>4);
         display(arr,len);

            do{
            copy(arr,arr2,len);
             printf("Select sort : 1.Selection sort
2.Insertion sort 3.Counting sort 4. Shell sort\n");
             scanf("%d",&choice);
             switch(choice){
                case 1:
                 t1=clock();
                 selection(arr2,len);
                 t2=clock();
                 break;
               case 2:
                t1=clock();
                insertion(arr2,len);
                t2=clock();
                break;
             case 3:
               t1=clock();
```

```
                shell(arr2,len);
                t2=clock();
                break;
             }
          printf("Array is Sorted\n");
          display(arr2,len);
          printf("time taken for sorting is %f\n",(double)(t2-
t1)/CLOCKS_PER_SEC);
          printf("Want to choose another sort? 1.for yes.\n");
          scanf("%d",&key);
          }while(key==1);
          printf("Want to choose another list ? 1. for yes.-
\n");
          scanf("%d",&key);
      }
       while(key==1);
          printf("Sorting completed\n");
}
// User-Defined Function Defination
//Ascending List Function
void ascending_list(int *arr, int len) {
   int i;
    for(i=0;i<len; i++)
     arr[i]=i+1;
 }
// Descending List Function
void descending_list(int *arr, int len) {
   int i;
   for(i=0;i<len; i++)
      arr[i]=(len-i);
}
// Partial sorted lase Function //
void part_sort_list(int * arr, int len) {
   int i;
   for(i=0; i<len;i++){
   if(i%5==0)
         arr[i]=12345+(10*i);
   else
      arr[i]=i+1;
}
}
//Random list
void random_list (int *arr, int len) {
   int i, offset;
   offset=98765;
   srand(offset);
   for(i=0;i<len;i++)
      arr[i] =rand()/5;
}
// Selection Sort
void selection (int *arr, int len) {
int i,j;
  for(i=0;i<len-2;i++) {
     for(j= i+1 ; j<len-1;j++){
```

```
     if (arr[j]< arr[i]){
        swap (&arr[i], &arr[j]);
            }
 235     }
       }
    }
    //Insertion Sort
    void insertion (int *arr, int len) {
 240 int i, j, key;
    for(i=1;i<len;i++) {
        key=arr[i]; j=i-1; while (j>=0 && arr[j]>key) {
        arr[j+1]=arr[j];
         j=j-1;
 245     arr[j+1]=key;
        }
    }
    }
    //Shell Sort
 250 void shell (int *arr, int len){
    int gap, key, i, j;
      for (gap=(len/2); gap>0; gap=(gap/2)) {
         for (i=gap; i<len;i++) {
         key=arr[i];
 255        for(j=i;j>=gap && arr[j-gap]>key;j=j-gap) {
              arr[j]=arr[j-gap];
              arr[j]=key;
            }
          }
 260      }
         }
    //Counting Sort
    void counting (int *arr2, int len, int k){
         int kount[INF], sorted [INF],
 265      knt=0; while (knt<=k){
               kount[knt]=0; knt++;
          for (knt=0; knt<len; knt++){
          kount[arr2[knt]]=kount[arr2[knt]]+1;
         for (knt=1;knt<=k; knt++)
 270       kount [knt] =kount [knt]+kount[knt-1];
             for (knt=len-1;knt>=0;knt--) {
                 sorted[kount[arr2[knt]]-1]=arr2[knt];
                    kount [arr2[knt]] =kount[arr2[knt]]-1;
             }
 275 copy (sorted, arr2, len);
          }
            }
    }
    //Display Function
 280 void display (int *arr, int len) {
       int i;
       if (len<=50) {
       for (i=0;i<len; i++)
       printf("%d\t", arr[i]); printf("\n");
 285 }
```

```
}
// Swap Function
 void swap (int *a, int *b) {
   int t;
   t= (*a);
   (*a)= (*b);
   (*b)=t;
 }
 // Copy function
void copy(int *arr,int *arr2,int len){
   int i;
   for(i=0;i<len;i++)
   arr2[i]=arr[i];
 }
/*
================================================================================
                                    OUTPUT TRAIL
================================================================================

        harsheet196@harsheet196:~/Desktop/csp252$ ./test.out
        Enter the Array length (in multiple of 10) ?
         10
        Select list : 1.Ascending list 2.Descending list
3.Partial sorted list 4.Random list
        4
        366633133   387979969   215805542   196996127
357749196   310754849   87355757    239012596
351033111     144422172
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        1
        Array is Sorted
        87355757    196996127   215805542   239012596
310754849   351033111   357749196   366633133   387979969
144422172
        time taken for sorting is 0.000007
        Want to choose another sort? 1.for yes.
        1
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        2
        Array is Sorted
        87355757    144422172   196996127   215805542
239012596   310754849   351033111   357749196   366633133
387979969
        time taken for sorting is 0.000006
        Want to choose another sort? 1.for yes.
        1
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        3
        Array is Sorted
        366633133   87355757    87355757    144422172
87355757    144422172   87355757    144422172   351033111
```

```
144422172
        time taken for sorting is 0.000007
        Want to choose another sort? 1.for yes.
1       1
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        4
        Array is Sorted
        366633133   387979969   215805542   196996127
357749196   310754849   87355757    239012596   351033111
144422172
5       time taken for sorting is 0.000007
        Want to choose another sort? 1.for yes.
        0
        Want to choose another list ? 1. for yes.
        1
0       Enter the Array length (in multiple of 10) ?
         10
        Select list : 1.Ascending list 2.Descending list
3.Partial sorted list 4.Random list
        2
        10  9   8   7   6   5   4   3   2   1
5       Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        1
        Array is Sorted
        2   3   4   5   6   7   8   9   10  1
        time taken for sorting is 0.000007
0       Want to choose another sort? 1.for yes.
        1
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        2
        Array is Sorted
5       1   2   3   4   5   6   7   8   9   10
        time taken for sorting is 0.000007
        Want to choose another sort? 1.for yes.
        3
        Want to choose another list ? 1. for yes.
0       1
        Enter the Array length (in multiple of 10) ?
         10
        Select list : 1.Ascending list 2.Descending list
3.Partial sorted list 4.Random list
        3
5       12345   2   3   4   5   12395   7   8   9   10
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        1
        Array is Sorted
        2   3   4   5   7   8   9   12345   12395   10
0       time taken for sorting is 0.000006
        Want to choose another sort? 1.for yes.
        1
```

```
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        2
        Array is Sorted
        2   3   4   5   7   8   9   10  12345   12395
        time taken for sorting is 0.000006
        Want to choose another sort? 1.for yes.
        1
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        3
        Array is Sorted
        12345   2   3   4   5   12395   7   8   9   10
        time taken for sorting is 0.000005
        Want to choose another sort? 1.for yes.
        1
        Select sort : 1.Selection sort 2.Insertion sort
3.Counting sort 4. Shell sort
        4
        Array is Sorted
        12345   2   3   4   5   12395   7   8   9   10
        time taken for sorting is 0.000005
        Want to choose another sort? 1.for yes.
        0
        Want to choose another list ? 1. for yes.
        0
        Sorting completed
*/
```