

Data Structure and Algorithm Lab Experiment

QUICK AND MERGE SORTS

Name: Om Ashish Mishra

Registration No.: 16BCE0789

Slot: G2

MERGING SORT

Pseudo code:

- First we declare the header files.
- Then we declare two global character arrays one for the original array and the other for making changes.
- Then we declare the merging1 function to merge the sorting array.
- Then we declare the sort1 function to sort the arrays.
- Then we call the main function to store the array elements, number of elements, then we print the elements of the unsorted array and then we print after the sorted elements of the array.

The code:

```
#include <stdio.h>
#include<stdlib.h>
#include<string.h>
char a[100][100], b[100][100];
void merging1(int low, int mid, int high)
{
    int l1, l2, i;
    for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++)
    {
        if(strcmp(a[l1],a[l2])<=0)
            strcpy(b[i],a[l1++]);
```

```
    else
        strcpy(b[i],a[l2++]);
}
while(l1 <= mid)
    strcpy(b[i++],a[l1++]);
while(l2 <= high)
    strcpy(b[i++],a[l2++]);
for(i = low; i <= high; i++)
    strcpy(a[i],b[i]);
}

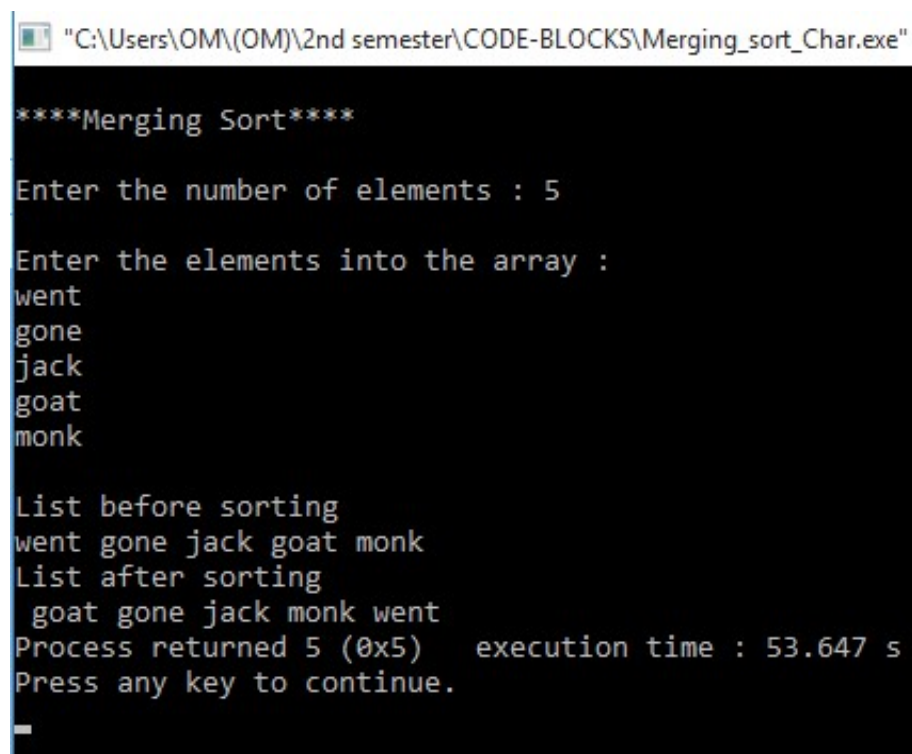
void sort1(int low, int high)
{
    int mid;
    if(low < high)
    {
        mid = (low + high) / 2;
        sort1(low, mid);
        sort1(mid+1, high);
        merging1(low, mid, high);
    }
    else
```

```
    {  
        return;  
    }  
}
```

```
int main()  
{  
    int i;  
    int n;  
    printf("\n****Merging Sort****\n");  
    printf("\nEnter the number of elements : ");  
    scanf("%d",&n);  
    printf("\nEnter the elements into the array : \n");  
    for(i=0;i<n;i++)  
    {  
        scanf("%s",&a[i]);  
    }  
    printf("\nList before sorting\n");  
    for(i = 0; i <= n; i++)  
        printf("%s ", a[i]);  
    sort1(0, n);  
    printf("\nList after sorting\n");
```

```
for(i = 0; i <= n; i++)  
    printf("%s ", a[i]);  
}
```

The Output:



```
"C:\Users\OM\OM\2nd semester\CODE-BLOCKS\Merging_sort_Char.exe"  
  
****Merging Sort****  
  
Enter the number of elements : 5  
  
Enter the elements into the array :  
went  
gone  
jack  
goat  
monk  
  
List before sorting  
went gone jack goat monk  
List after sorting  
goat gone jack monk went  
Process returned 5 (0x5)   execution time : 53.647 s  
Press any key to continue.  
_
```

QUICK SORT

Pseudo code:

- First we declare the header files.
- Then we declare a global character arrays one for the original array.
- Then we declare the display function to display the unsorted and sorted array.
- Then we declare the swap function to swap the desired elements of the array.
- Then we declare the partition function for partitioning the array's element.
- Then we declare the quicksort function to for positioning of the pivot point/element.
- Then we call the main function to store the array elements, number of elements, then we print the elements of the unsorted array and then we print after the sorted elements of the array.

The code:

```
#include <stdio.h>
#include<stdlib.h>
#include<string.h>
#include <stdbool.h>
char a[100][100];
int n;
```

```
void display(int n)
{
    int i;
    for(i = 0;i<n;i++)
    {
        printf("%s ",a[i]);
    }
    printf("\n");
}
```

```
void swap(int num1, int num2)
{
    char temp[100][100];
    strcpy(temp,a[num1]);
    strcpy(a[num1],a[num2]);
    strcpy(a[num2],temp);
}
```

```
int partition(int left, int right, int pivot)
{
    int leftPointer = left -1;
    int rightPointer = right;
```

```
while(true)
{
    while(strcmp(a[++leftPointer],pivot)<0)
    {
        //do nothing
    }
    while(rightPointer > 0 && strcmp(a[--rightPointer],pivot)>0)
    {
        //do nothing
    }
    if(leftPointer >= rightPointer)
    {
        break;
    }
    else
    {
        swap(leftPointer,rightPointer);
    }
}
swap(leftPointer,right);
display(n);
return leftPointer;
```



```
}
```

```
void quickSort(int left, int right)
```

```
{
```

```
    if(right-left <= 0)
```

```
    {
```

```
        return;
```

```
    }
```

```
    else
```

```
    {
```

```
        char pivot[100][100];
```

```
        strcpy(pivot,a[right]);
```

```
        int partitionPoint = partition(left, right, pivot);
```

```
        quickSort(left,partitionPoint-1);
```

```
        quickSort(partitionPoint+1,right);
```

```
    }
```

```
}
```

```
void main()
```

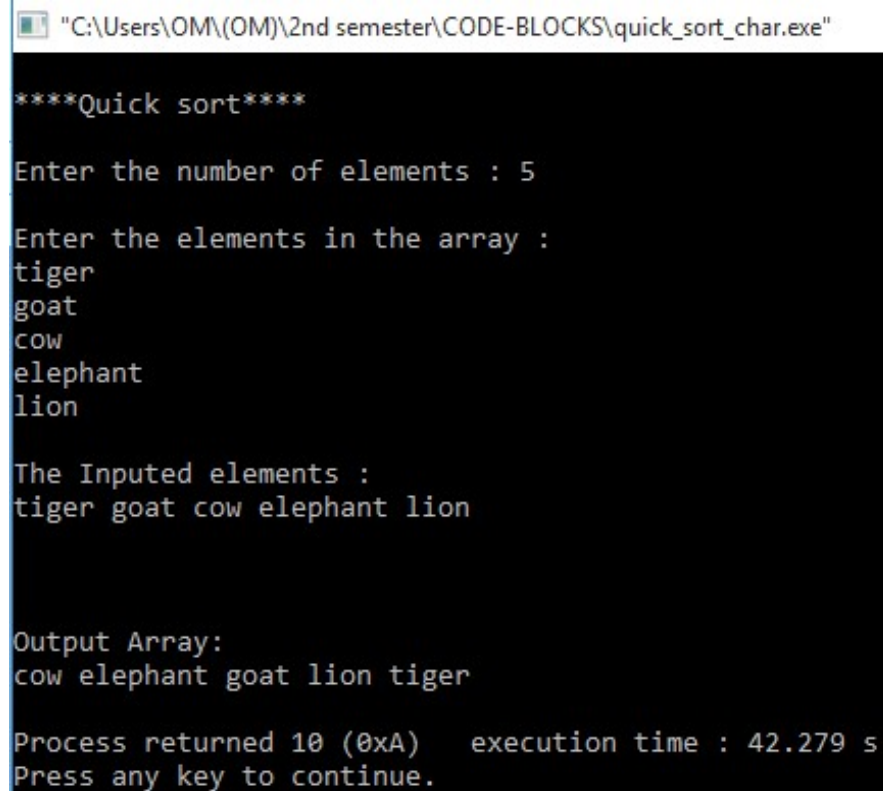
```
{
```

```
    int n,i;
```

```
    printf("\n****Quick sort****\n");
```

```
printf("\nEnter the number of elements : ");
scanf("%d",&n);
printf("\nEnter the elements in the array : \n");
for(i=0;i<n;i++)
    scanf("%s",&a[i]);
printf("\nThe Inputed elements : \n");
display(n);
quickSort(0,n-1);
printf("Output Array: \n");
display(n);
}
```

The Output:



```
"C:\Users\OM\OM\2nd semester\CODE-BLOCKS\quick_sort_char.exe"

****Quick sort****

Enter the number of elements : 5

Enter the elements in the array :
tiger
goat
cow
elephant
lion

The Inputed elements :
tiger goat cow elephant lion

Output Array:
cow elephant goat lion tiger

Process returned 10 (0xA)   execution time : 42.279 s
Press any key to continue.
```

Thank You