

Institute of Computer Technology
B. Tech Computer Science and Engineering
Subject: DS (2CSE302)

PRACTICAL-22

AIM: - Implement quick sort and radix sort.

(This practical is in continuation of previous practical scenario, so append code of below scenario along with previous practical)

1. Swati is working on different sorting methods to sort the data. She wants to prepare sorting calculator which provides the facilities to sort all kind of sorting methods for same data. Kindly refer given scenario for calculator and implement it in C:

How many numbers you want to sort?

5

Enter the Elements for Sorting:

6734

2367

8901

345

5421

List of sorting methods:

- 1. Bubble Sort**
- 2. Insertion Sort**
- 3. Selection Sort**
- 4. Merge Sort**
- 5. Quick Sort**
- 6. Radix Sort**
- 7. Exit**

Which choice do you want apply?

6

Pass-1 8901 5421 6734 345 2367

Pass-2 8901 5421 6734 345 2367

Pass-3 345 2367 5421 6734 8901

Pass-4 345 2367 5421 6734 8901

SOLUTION

```
#include <stdio.h>
#include <stdlib.h>
int p = 0, q = 0, rm = 0;
```

```
void PrintArr(int arr[], int n)
```

```
{
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void BubbleSort(int arr[], int m)
{
    int i, j, k, temp;
    for (i = 0; i < m - 1; i++)
    {
        for (j = 0; j < m - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
        printf("Pass %d: ", i + 1);
        for (k = 0; k < m; k++)
        {
            printf("%d ", arr[k]);
        }
        printf("\n");
    }
}

void InsertionSort(int arr[], int m)
{
    int i, j, k, temp;
    for (i = 1; i < m; i++)
    {
        temp = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > temp)
        {
            arr[j + 1] = arr[j];

            j -= 1;
        }
    }
}
```

```
        arr[j + 1] = temp;
        printf("Pass %d: ", i);
        for (k = 0; k < m; k++)
        {
            printf("%d ", arr[k]);
        }
        printf("\n");
    }
}
```

```
void SelectionSort(int arr[], int m)
{
    int i, j, min, k, temp;
    for (i = 0; i < m - 1; i++)
    {
        min = i;
        for (j = i + 1; j < m; j++)
        {
            if (arr[j] < arr[min])
            {
                min = j;
            }
        }
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
        printf("Pass %d: ", i + 1);
        for (k = 0; k < m; k++)
        {
            printf("%d ", arr[k]);
        }
        printf("\n");
    }
}
```

```
void merge(int arr[], int l, int mid, int r)
{
    int i, j, k;
    int n1 = mid - l + 1;
    int n2 = r - mid;
    int left[n1], right[n2];
    for (i = 0; i < n1; i++)
    {
```

```
        left[i] = arr[l + i];
    }
    for (j = 0; j < n2; j++)
    {
        right[j] = arr[mid + 1 + j];
    }
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (left[i] < right[j])
        {
            arr[k] = left[i];
            i++;
        }
        else
        {
            arr[k] = right[j];
            j++;
        }
        k++;
    }
    while (i < n1)
    {
        arr[k] = left[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        arr[k] = right[j];
        j++;
        k++;
    }
}

void MergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        int mid = (l + r) / 2;
        MergeSort(arr, l, mid);
```

```
        MergeSort(arr, mid + 1, r);
        merge(arr, l, mid, r);
        printf("Pass %d: ", ++q);
        for (rm = 0; rm < p; rm++)
        {
            printf("%d ", arr[rm]);
        }
        printf("\n");
    }
}
```

```
int partition(int arr[], int low, int high)
{
    int temp;
    int pivot = arr[low];
    int i = low + 1;
    int j = high;
    do
    {
        while (arr[i] <= pivot)
        {
            i++;
        }
        while (arr[j] > pivot)
        {
            j--;
        }

        if (i < j)
        {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    } while (i < j);

    temp = arr[low];
    arr[low] = arr[j];
    arr[j] = temp;
    return j;
}
```

```
void QuickSort(int arr[], int low, int high)
{

```

```

int partitionIndex; //index of pivot

if (low < high)
{
    printf("Pass %d: ", ++q);
    for (rm = 0; rm < p; rm++)
    {
        printf("%d ", arr[rm]);
    }
    printf("\n");
    partitionIndex = partition(arr, low, high);
    QuickSort(arr, low, partitionIndex - 1); //sort left subarray
    QuickSort(arr, partitionIndex + 1, high); //sort right subarray
}
}

int MaxElement(int arr[], int n)
{
    int max = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > max)
        {
            max = arr[i];
        }
    return max;
}

void RadixSort(int arr[], int n)
{
    int bucket[10][10], bucket_cnt[10];
    int i, j, k, r, NOP = 0, divisor = 1, Max, pass;
    Max = MaxElement(arr, n);
    while (Max > 0)
    {
        NOP++;
        Max /= 10;
    }
    for (pass = 0; pass < NOP; pass++)
    {
        for (i = 0; i < 10; i++)
        {
            bucket_cnt[i] = 0;
        }
        for (i = 0; i < n; i++)

```

```

        {
            r = (arr[i] / divisor) % 10;
            bucket[r][bucket_cnt[r]] = arr[i];
            bucket_cnt[r] += 1;
        }
        i = 0;
        for (k = 0; k < 10; k++)
        {
            for (j = 0; j < bucket_cnt[k]; j++)
            {
                arr[i] = bucket[k][j];
                i++;
            }
        }
        divisor *= 10;
        printf("Pass %d : ", pass + 1);
        for (i = 0; i < n; i++)
        {
            printf("%d ", arr[i]);
        }
        printf("\n");
    }
}

int main()
{
    int n, i, choice;
    printf("\nEnter number of elements you want to sort: ");
    scanf("%d", &n);
    p = n;
    int yash[n], prajapati[n], y[n], p[n], quickarr[n], radixarr[n];
    printf("\nEnter the Elements for Sorting: ");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &yash[i]);
        prajapati[i] = yash[i];
        y[i] = yash[i];
        p[i] = yash[i];
        quickarr[i] = yash[i];
        radixarr[i] = yash[i];
    }
    there:
        printf("\nList of sorting methods: \n1. Bubble Sort\n2. Insertion Sort\n3. Selection Sort\n4. Merge Sort\n5. Quick Sort\n6. Radix Sort\n7. Exit");

```

```
printf("\n\nWhich choice do you want to apply? ");
scanf("%d", &choice);
switch (choice)
{
case 1:
    printf("\nBefore Bubble Sort : ");
    PrintArr(yash, n);
    printf("\n");
    BubbleSort(yash, n);
    printf("\nAfter Bubble Sort : ");
    PrintArr(yash, n);
    goto there;
    break;
case 2:
    printf("\nBefore Insertion Sort : ");
    PrintArr(prajapati, n);
    printf("\n");
    InsertionSort(prajapati, n);
    printf("\nAfter Insertion Sort : ");
    PrintArr(prajapati, n);
    goto there;
    break;
case 3:
    printf("\nBefore Selection Sort : ");
    PrintArr(y, n);
    printf("\n");
    SelectionSort(y, n);
    printf("\nAfter Selection Sort : ");
    PrintArr(y, n);
    goto there;
    break;
case 4:
    printf("\nBefore Merge Sort : ");
    PrintArr(p, n);
    printf("\n");
    MergeSort(p, 0, n - 1);
    printf("\nAfter Merge Sort : ");
    PrintArr(p, n);
    goto there;
    break;
case 5:
    printf("\nBefore Quick Sort : ");
    PrintArr(quickarr, n);
    printf("\n");
```



```

        QuickSort(quickarr, 0, n - 1);
        printf("\nAfter Quick Sort : ");
        PrintArr(quickarr, n);
        goto there;
        break;
case 6:
        printf("\nBefore Radix Sort : ");
        PrintArr(radixarr, n);
        printf("\n");
        RadixSort(radixarr, n);
        printf("\nAfter Radix Sort : ");
        PrintArr(radixarr, n);
        goto there;
        break;
case 7:
        exit(0);
        break;
default:
        printf("\nInvalid choice. Enter Again!");
        goto there;
        break;
}
printf("\n");
return 0;
}

```

OUTPUT (Quick Sort)

```

CentOS 8 64-bit - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
CentOS 8 64-bit
Activities Applications Terminal
Nov 26 23:54
yash@localhost:~/Desktop/DS/practicals/Prac22
File Edit View Search Terminal Help
[yash@localhost Prac22]$ gedit p22.c
[yash@localhost Prac22]$ gcc p22.c
[yash@localhost Prac22]$ ./a.out

Enter number of elements you want to sort: 5
Enter the Elements for Sorting: 6734 2367 8901 345 5421

List of sorting methods:
1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Merge Sort
5. Quick Sort
6. Radix Sort
7. Exit

Which choice do you want to apply? 5

Before Quick Sort : 6734 2367 8901 345 5421

Pass 1: 6734 2367 8901 345 5421
Pass 2: 345 2367 5421 6734 8901
Pass 3: 345 2367 5421 6734 8901

After Quick Sort : 345 2367 5421 6734 8901

List of sorting methods:
1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Merge Sort
5. Quick Sort
6. Radix Sort
7. Exit

```

OUTPUT (Radix Sort)

