

# Practice-1

## 1. Write a program in C to perform linear and binary search

### Linear search:

Aim:

To search for particular element in the array provided by user.

Theory:

The theory of the program is to traverse each element of the array one by one until the desired number is not found. In this technique we don't have prerequisites such that the array should be sorted unlike in binary search.

Program/Code:

```
#include <stdio.h>

int linear_search(int arr[],int n,int x)
{
    for(int i=0;i<n;i++)
    {
        if(arr[i]==x)
        {
            printf("The number is at location= %d",i+1);
            return 0;
        }
    }
    printf("The number was not found");
    return 0;
}

void main()
{
    int n,x;
    printf("Enter the number of elements in array=");
```

```

scanf("%d",&n);

printf("Enter the elements for the array=");

int arr[n];

for (int i=0;i<n;i++)

{

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

}

printf("Enter the number that you want to search=");

scanf("%d",&x);

linear_search(arr,n,x);

}

```

Output:

```

PS C:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1> cd "c:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA L
ab\Practice-1\" ; if ($?) { gcc linear_search.c -o linear_search } ; if ($?) { .\linear_search }
Enter the number of elements in array=5
Enter the elements for the array=1
2
3
4
5
Enter the number that you want to search=3
The number is at location= 3
PS C:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1> cd "c:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA L
ab\Practice-1\" ; if ($?) { gcc linear_search.c -o linear_search } ; if ($?) { .\linear_search }
Enter the number of elements in array=3
Enter the elements for the array=1
5
6
Enter the number that you want to search=2
The number was not found

```

Time complexity:

$O(n)$

## Binary Search:

Aim:

To search for particular element in the sorted array provided by user.

Theory:

In this type of search we will check for the element in the sorted array by dividing the array into two part by taking the middle element and compare it with adjacent element and change the searching array size accordingly and repeat the same process until the element is found or the array size comes down to one.

Program:

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

```
int binary_search(int arr[],int n,int x)
```

```
{  
    int l=0;  
    int u=n-1;  
    while(l<=u)  
    {  
        int avg=(l+u)/2;  
        if(arr[avg]==x)  
        {  
            printf("The number is at location= %d",avg+1);  
            return 0;  
        }  
        else if(arr[avg]<x)  
        {  
            l=avg+1;  
        }  
        else  
        {  
            u=avg-1;  
        }  
    }  
    printf("The number was not found in the array");  
}
```

```
void main()
```

```
{  
    int n,x;
```

```

printf("Enter the number of elements in array=");

scanf("%d",&n);

printf("Enter the elements for the array=");

int arr[n];

for (int i=0;i<n;i++)

{

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

}

printf("Enter the number that you want to search=");

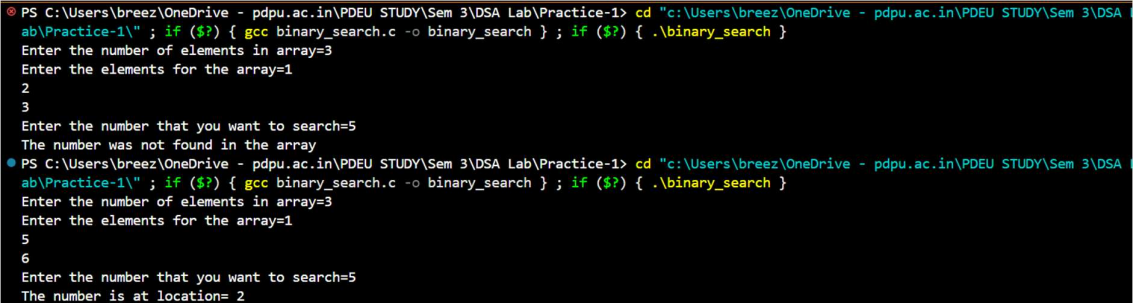
scanf("%d",&x);

binary_search(arr,n,x);

}

```

Output:



```

PS C:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1> cd "c:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1\" ; if ($?) { gcc binary_search.c -o binary_search } ; if ($?) { .\binary_search }
Enter the number of elements in array=3
Enter the elements for the array=1
2
3
Enter the number that you want to search=5
The number was not found in the array
PS C:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1> cd "c:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1\" ; if ($?) { gcc binary_search.c -o binary_search } ; if ($?) { .\binary_search }
Enter the number of elements in array=3
Enter the elements for the array=1
5
6
Enter the number that you want to search=5
The number is at location= 2

```

Time Complexity:

$O(\log n)$

## 2. Write a program in C to perform bubble sort, insertion sort and selection sort .Take the array size and array elements from user.

### Bubble sort

Aim:

To sort a given array from user in ascending order.

Theory:

In this type of sorting we use to compare the element adjacent to it and swap them if the number is larger than the previous element and we will repeat the process until the array is sorted.

Program/Code:

Program/Code:

```
#include <stdio.h>
int bubble_sort(int arr[],int n)
{
    for (int i=0;i<n;i++)
    {
        for(int i=0;i<n;i++)
        {
            int count=0;
            int temp;
            if(arr[i]>arr[i+1])
            {
                temp=arr[i];
                arr[i]=arr[i+1];
                arr[i+1]=temp;
            }
            else
            {
                count=count+1;
                if(count==n-1)
                {
                    goto lable;
                }
            }
        }
    }
}
```

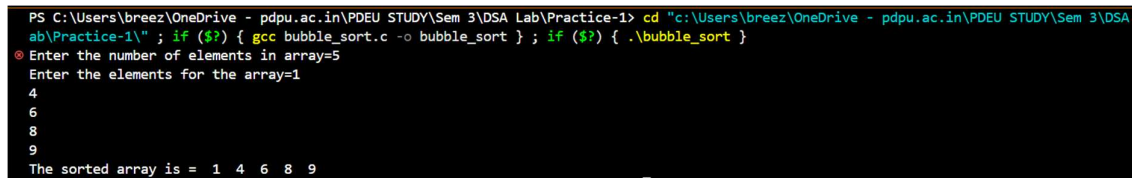
```

    }
    lable:
    printf("The sorted array is = ");
    for (int i=0;i<n;i++)
    {
        printf(" %d ",arr[i]);
    }
}

void main()
{
    int n;
    printf("Enter the number of elements in array=");
    scanf("%d",&n);
    printf("Enter the elements for the array=");
    int arr[n];
    for (int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    bubble_sort(arr,n);
}

```

Output:



```

PS C:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1> cd "c:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA
ab\Practice-1\" ; if ($?) { gcc bubble_sort.c -o bubble_sort } ; if ($?) { .\bubble_sort }
Enter the number of elements in array=5
Enter the elements for the array=1
4
6
8
9
The sorted array is = 1 4 6 8 9

```

Time Complexity:

$O(n^2)$

## Insertion sort

Aim:

To sort a given array from user in ascending order.

Theory:

In this technique we will imagine that we have divide the array into two parts first part only contains first element of the array and the rest of the array is thought to be the

second part of the array now we will compare the first element of the second part of the array we compare it with the first part of the array and insert it in first array in ascending order by shifting the places of elements in second part and then we repeat the process till we have a sorted array.

Program/Code:

```
#include <stdio.h>

int insertion_sort(int arr[],int n)
{
    for(int i=1;i<n;i++)
    {
        int key=arr[i];
        int j=i-1;
        while(j>=0 && arr[j]>key)
        {
            arr[j+1]=arr[j];
            j=j-1;
        }
        arr[j+1]=key;
    }
    printf("The sorted array is = ");
    for(int i=0;i<n;i++)
    {
        printf(" %d ",arr[i]);
    }
}

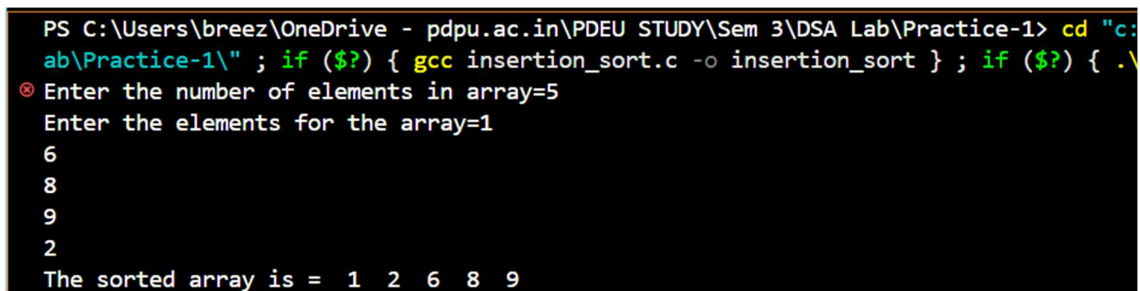
void main()
{
    int n;
    printf("Enter the number of elements in array=");
    scanf("%d",&n);
    int arr[n];
```

```

printf("Enter the elements for the array=");
for(int i=0;i<n;i++)
{
    scanf("%d",&arr[i]);
}
insertion_sort(arr,n);
}

```

Output:



```

PS C:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1> cd "c:\ab\Practice-1\" ; if ($?) { gcc insertion_sort.c -o insertion_sort } ; if ($?) { .\'
ⓧ Enter the number of elements in array=5
Enter the elements for the array=1
6
8
9
2
The sorted array is = 1 2 6 8 9

```

Time Complexity:

$O(n^2)$

## Selection sort

Aim:

To sort a given array from user in ascending order.

Theory:

In selection sort we first find the smallest element in the array and swap it with the first element and then we leave that index and then do the above process excluding the element that were being swapped earlier , we do this till we get a sorted array.

Program/Code:

```

#include <stdio.h>

int selection_sort(int arr[],int n)
{
    for(int i=0;i<n-1;i++)
    {
        int min=i;

```



```

        for(int j=i+1;j<n;j++)
        {
            if(arr[j] < arr[min])
            {
                min=j;
            }
        }
        int temp=arr[min];
        arr[min]=arr[i];
        arr[i]=temp;
    }
    printf("The sorted array is = ");
    for(int i=0;i<n;i++)
    {
        printf(" %d ",arr[i]);
    }
}

```

```

void main()
{
    int n;
    printf("Enter the number of elements in array=");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the elements for the array=");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    selection_sort(arr,n);
}

```

Output:

```
PS C:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1> cd "c:\Users\breez\ab\Practice-1\" ; if ($?) { gcc selection_sort.c -o selection_sort } ; if ($?) { .\selection_so
Enter the number of elements in array=5
Enter the elements for the array=1
4
2
3
5
The sorted array is = 1 2 3 4 5
```

Time Complexity:

$O(n^2)$

**3. Write a program in C that obtains the minimum and maximum element from the array. Modify this program to give the second largest and second smallest element of the array.**

Aim:

To find the maximum, second maximum, minimum and second minimum in the array.

Theory:

Firstly we will bubble sort the array and then we take the last element of the array for maximum, take the second last element of the array for the second maximum, take the first element of the array for minimum and then take the second element for second minimum element of the array.

Program/Code:

```
#include <stdio.h>

int bubble_sort(int arr[],int n)
{
    for (int i=0;i<n;i++)
    {
        for(int i=0;i<n;i++)
        {
            int count=0;
            int temp;
            if(arr[i]>arr[i+1])
            {
                temp=arr[i];
                arr[i]=arr[i+1];
                arr[i+1]=temp;
            }
            else
            {
                count=count+1;
            }
        }
    }
}
```

```

        if(count==n-1)
        {
            goto lable;
        }
    }
}

lable:
printf("The sorted array is = ");
for (int i=0;i<n;i++)
{
    printf(" %d ",arr[i]);
}

printf("\nThe largest element in the array is= %d \n",arr[n-1]);
printf("The second largest element in the array is= %d \n",arr[n-2]);
printf("The smallest element in the array is= %d \n",arr[0]);
printf("The second smallest element in the array is= %d\n",arr[1]);
}

```

```

void main()
{
    int n;

    printf("Enter the number of elements in array=");

    scanf("%d",&n);

    printf("Enter the elements for the array=");

    int arr[n];

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

```

```
}  
  
bubble_sort(arr,n);  
  
}
```

Output:

```
PS C:\Users\breez\OneDrive - pdpu.ac.in\PDEU STUDY\Sem 3\DSA Lab\Practice-1> cd "c:\Users\breez\ab\Practice-1\" ; if ($?) { gcc Practice-1_3.c -o Practice-1_3 } ; if ($?) { .\Practice-1_3 }  
⊗ Enter the number of elements in array=5  
Enter the elements for the array=1  
4  
8  
9  
11  
The sorted array is = 1 4 8 9 11  
The largest element in the array is= 11  
The second largest element in the array is= 9  
The smallest element in the array is= 1  
The second smallest element in the array is= 4
```

Time Complexity:

$O(n^2)$

For code please visit:

<https://github.com/PanavPatel06/DSA-Lab>