**EXPERIMENT 1**

**AIM :** Basic Array operations using switch menu and bubble sort.

**THEORY :**

An array is a collection of items stored at contiguous memory

locations. The idea is to store multiple items of the same type together. This

makes it easier to calculate the position of each element by simply adding

an offset to a base value, i.e., the memory location of the first element of

the array (generally denoted by the name of the array).

Applications of Array Data Structure:

Below are some applications of arrays.

**Storing and accessing data:** Arrays are used to store and retrieve data in a

specific order. For example, an array can be used to store the scores of a

group of students, or the temperatures recorded by a weather station.

**Sorting**: Arrays can be used to sort data in ascending or descending order.

Sorting algorithms such as bubble sort, merge sort, and quicksort rely

heavily on arrays.

**Searching:** Arrays can be searched for specific elements using algorithms

such as linear search and binary search.

**Matrices:** Arrays are used to represent matrices in mathematical

computations such as matrix multiplication, linear algebra, and image

processing.

**Stacks and queues:** Arrays are used as the underlying data structure for

implementing stacks and queues, which are commonly used in algorithms

and data structures.

Graphs: Arrays can be used to represent graphs in computer science. Each

element in the array represents a node in the graph, and the relationships

between the nodes are represented by the values stored in the array.

Dynamic programming: Dynamic programming algorithms often use arrays

to store intermediate results of subproblems in order to solve a larger

problem.

**ALGORITHM :**

**Step 1:** Initialization

Declare a constant maxSize as 100 for the maximum size of the

array.

Declare an integer array array of size maxSize.

Declare an integer variable size and initialize it to 0.

**Step 2:** Menu Display Function

Define a function displayMenu() to display the menu options.

Output the menu options using cout.

**Step 3:** Bubble Sort Function

Define a function bubbleSort(int arr[], int size).

Implement the Bubble Sort algorithm to sort the array in

ascending order.

**Step 4:** Main Function

Define the main() function.

Declare variables, including choice to store the user's menu

choice.

Use a do-while loop to repeatedly display the menu and execute

user-selected options.

Inside the loop:

Display the menu using displayMenu().

Read the user's choice using cin.

Use a switch statement to perform actions based on the user's

choice:

Case 1: Add an item to the array.

Case 2: Delete an item from the array.

Case 3: Display the elements of the array.

Case 4: Sort the array using Bubble Sort.

Case 5: Exit the program.

Default: Display an error message for an invalid choice.

**Step 5:** Add Item to Array (Case 1)

Check if the size is less than maxSize.

If true, prompt the user to enter a value and add it to the array at

index size.

Increment size.

If false, display an error message indicating that the array is full.

**Step 6:** Delete Item from Array (Case 2)

Check if the size is greater than 0.

If true, prompt the user to enter the index to delete.

If the index is valid, shift elements to remove the item at the

specified index.

Decrement size.

If false, display an error message indicating that the array is

empty.

**Step 7:** Display Array (Case 3)

Output the elements of the array up to the current size.

Step 8: Bubble Sort (Case 4)

Call the bubbleSort function to sort the array.

Output a message indicating that the array has been sorted.

Step 9: Exit (Case 5)

Display an exit message.

Step 10: Loop Continuation

Continue the loop until the user chooses to exit (Case 5).

**EXAMPLE :**

**CODE :**

#include<iostream>

using namespace std;

void input\_arr(int *\**arr,int size)

{

for(int i=0;i<size;i++){

cout<<"enter element "<<i<<"of arr: ";

cin>>arr[i];

cout<<endl;

}

}

void display\_arr(int *\**arr,int size){

for(int i=0;i<size;i++){

cout<<arr[i]<<" ";

}

cout<<endl;

}

void find\_element(int *\**arr,int size){

int no;

int check=1;

cout<<"Enter number to be found"<<endl;

cin>>no;

for(int i=0;i<size;i++){

if(arr[i]==no){

cout<<"The number is found at "<<i<<" index of array "<<endl;

check=0;

break;

}

}

if(check==1){

cout<<"Element is not present in array"<<endl;

}

}

void sort\_arr(int *\**arr, int size){

int n = size;

for(int i=0;i<size-1;i++){

for(int j=0;j<size-1;j++){

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

int temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

n--;

}

}

void delete\_arr(int *\**arr,int *&* size){

int index;

cout<<"Enter index which you want to delete"<<endl;

cin>>index;

for(int i=index+1;i<size;i++){

arr[i-1]=arr[i];

}

size=size-1;

}

void insert\_arr(int *\**arr,int *&* size){

int index=0;

cout<<"Enter index at which you want to insert"<<endl;

cin>>index;

size=size+1;

for(int i=size-1;i>index;i--){

arr[i]=arr[i-1];

}

cout<<"Enter the no to be added"<<endl;

cin>>arr[index];

}

int main(){

int size;

cout<<"enter size of arr to be created"<<endl;

cin>>size;

int \*arr=new int[size];

input\_arr(arr,size);

display\_arr(arr,size);

find\_element(arr,size);

sort\_arr(arr,size);

cout<<"Results after Bubble Sort is"<<endl;

display\_arr(arr,size);

delete\_arr(arr,size);

display\_arr(arr,size);

insert\_arr(arr,size);

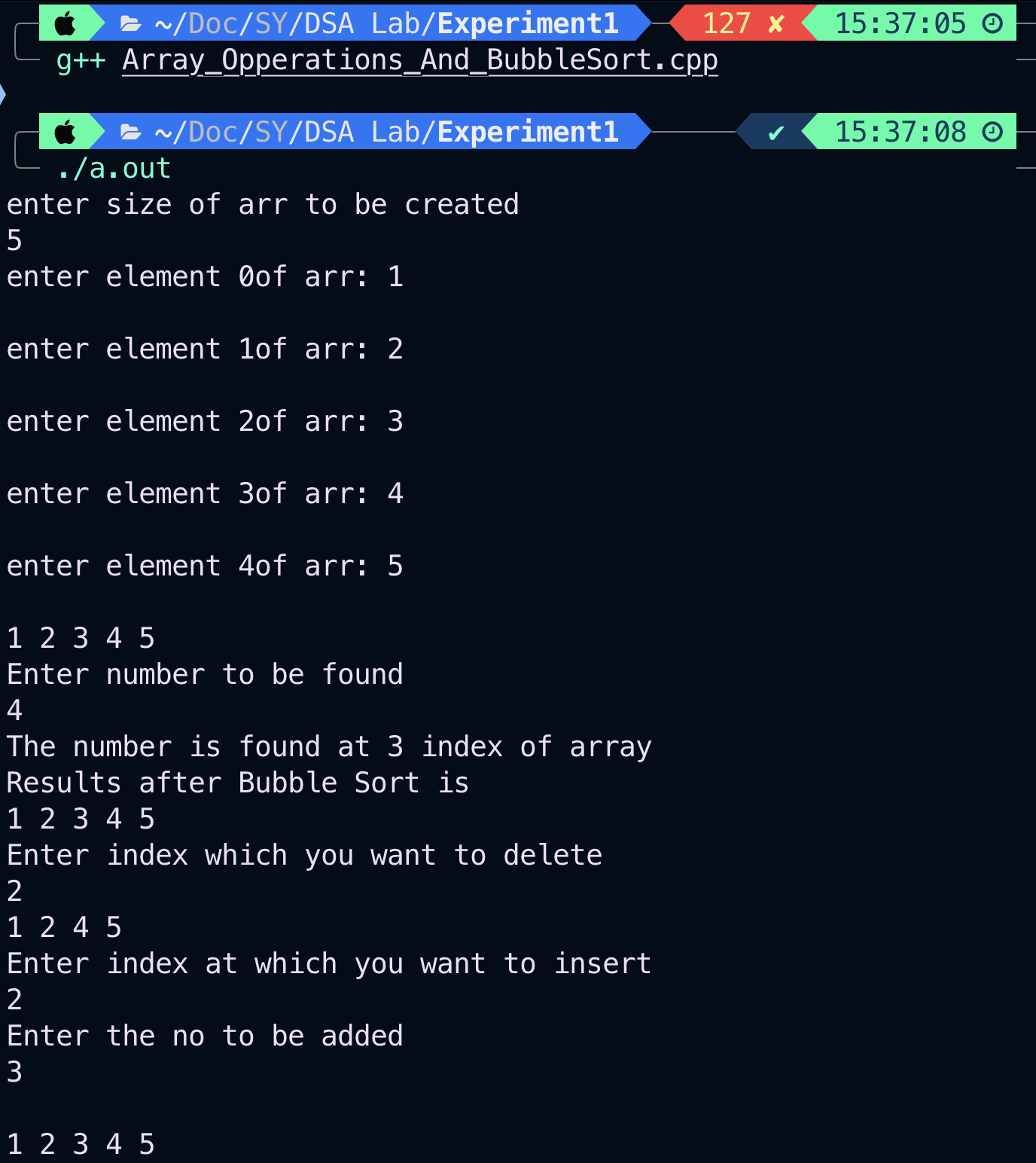
cout<<endl;

display\_arr(arr,size);

return 0;

}

**Output + Example:-**

****

**CONCLUSION :**

Through this experiment, we learned the implementation of a

menu-driven program in C++ for array manipulation, including adding,

deleting, displaying, and sorting elements. The program also demonstrated

the use of functions and a user-friendly menu for interactive control,

providing insights into basic algorithmic