**Lab Experiment No. 1**

**Objective**

To implement and analyze various operations on arrays such as **insertion, deletion, searching, traversing, and updation** using C++.

**Theory**

An **array** is a collection of elements stored in contiguous memory locations. It allows **constant-time access** to elements using an index but has fixed size once declared.

**Operations on Arrays**

1. **Insertion**: Adding an element at a specific position.
2. **Deletion**: Removing an element from a specific position.
3. **Searching**: Finding an element in the array.
4. **Traversing**: Visiting and displaying all elements.
5. **Updation**: Modifying an element at a given index.

**Algorithm & Implementation**

**1. Insertion**

**Algorithm:**

1. Check if the array is full.
2. Shift elements to the right from the insertion position.
3. Insert the new element.
4. Increase array size.

**2. Deletion**

**Algorithm:**

1. Check if the array is empty.
2. Shift elements to the left from the deletion position.
3. Reduce the array size.

void remove(int arr[], int &n, int pos) {

**3. Searching**

**Algorithm:**

1. Traverse the array.
2. If the element is found, return the index.
3. If not found, return -1.

**4. Traversing**

**Algorithm:**

1. Start from the first index.
2. Visit and display each element.

**5. Updation**

**Algorithm:**

1. Check if the index is valid.
2. Replace the element at the given index.

**Code**

**Complexity Analysis**

|  |  |  |
| --- | --- | --- |
| Operation | Best Case | Worst Case |
| Insertion | O(1) | O(n) |
| Deletion | O(1) | O(n) |
| Searching | O(1) | O(n) |
| Traversing | O(n) | O(n) |
| Updation | O(1) | O(1) |

**Sample Output**

Original Array: 10 20 30 40 50

After Insertion (at pos 2): 10 20 99 30 40 50

After Deletion (at pos 3): 10 20 99 40 50

Searching 40: Found at index 3

After Updation (pos 2 -> 77): 10 20 77 40 50

**Conclusion**

We successfully implemented and analyzed various **array operations**. The choice of operations depends on time complexity and the specific application needs.

## Flowcharts:

1. Flowchart is a diagrammatic representation of the sequence of logical steps of a program.
2. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

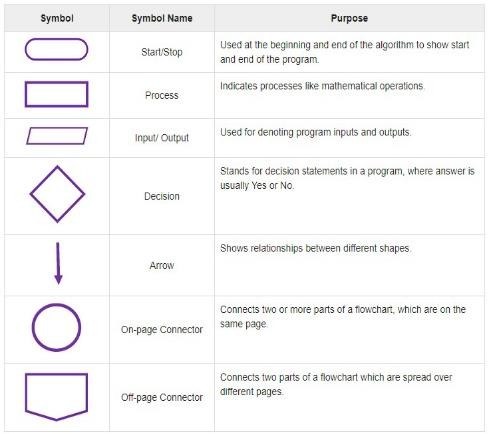


Fig 1.1: Diagram for flowchart symbol, name, and purpose

**2. Algorithms**

1. It is a finite set of instructions that, if followed accomplish a particular task.
2. It is basically used to describe a problem-solving method suitable for implementation as a computer program.
3. Algorithm is independent of the machine and language used for implementation
4. Characteristics Input
5. Zero or more quantities are supplied externally o Output
6. At least one quantity is produced o Definiteness
7. Each instruction is clear &amp; unambiguous o Finiteness
8. It terminates after finite steps o Effectiveness
9. Each instruction is simple to be carried out manually.

## Assigned Tasks:

**T-1.1**: Design an algorithm and flowchart for adding three numbers.

Start

Read a,b,c

S=a+b+c

Fig 1.2: Flowchart for adding three numbers

Output S

Stop

**T-1.2**: Design an algorithm and flowchart to find larger of two numbers.

Start

Read x, y

Is x>y

Output x

Output y

Stop

Stop

Fig 1.3: Flowchart to find largest of two numbers

## Conclusion:

We have successfully gotten the idea of how logic for problem-solving is built.

**Lab Experiment No. 2:** Basic Concepts of C Language.

**Aim:** To understand the basic concepts of C language, syntax, structure, and problem-solving techniques.

## Background

* + - 1. **About C**

C is a programming language developed by Dennis Ritchie at AT&T‟s BELL Laboratory of the USA in 1972. Because of its reliability, C is very popular. C is highly portable &amp; it is well suited for structured programming. C program consists of a collection of functions. The milestones in C’s development as a language are listed below:

* + - * 1. BCPL (Basic Combined Programming Language) - a user-friendly OS providing powerful development tools developed from BCPL in 1967. The assembler was tedious, long, and error-prone.
        2. A new language “B” was attempted in 1970 by Ken Thompson at Bell Labs.
        3. UNIX was developed in 1970 using “B” - DEC PDP-7 Assembly Language.
        4. A new language “C”, a successor to “B” was developed in c. 1971.
        5. By 1973, UNIX OS was almost totally written in “C”.

## Structure of the C Program

The C program starts with the main function followed by the opening braces which indicate the start of the function. Then follows the variable and constant declarations which are followed by the statements that include input and output statements.

DOCUMENTATION SECTION LINK SECTION

DEFINITION SECTION

GLOBAL DECLARATION SECTION

Main() Function section

{

Declaration part Executable part

}

SUBPROGRAM SECTION

User-defined functions.

## Assigned Tasks

**T-2.1**: Write a C program to display “This is my first C Program”.

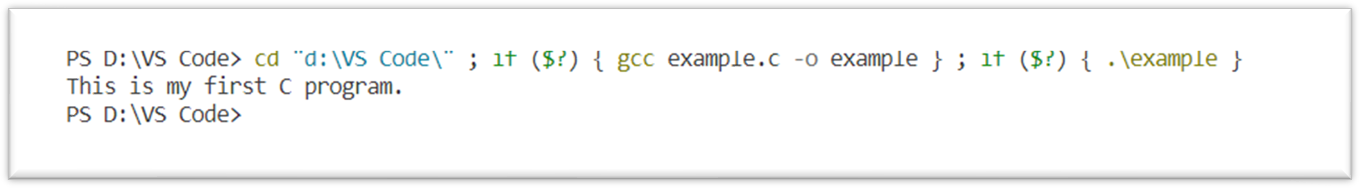
## Code:

#include <stdio.h> void main(){

printf("This is my first C Program.\n"); // print the message.

}

## Output:



**Fig. 2.1:** Output of the program to print “First c program.”

**T-2.2**: Write a C program to add two numbers (2 and 6) and display their sum.

## Code:

#include <stdio.h> void main(){

int a,b,c;

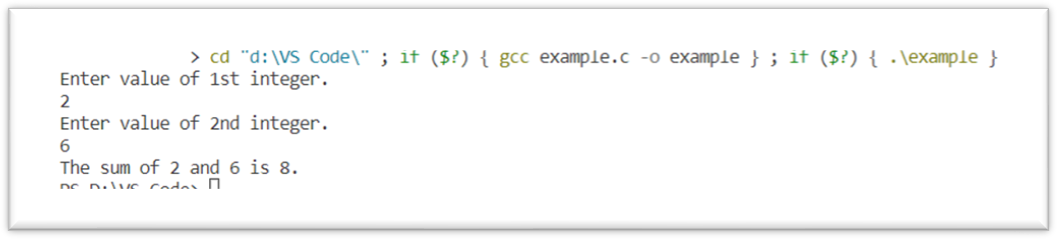
printf("Enter value of 1st integer.\n"); // print message. scanf("%d",&a); // take input and save in a. printf("Enter value of 2nd integer.\n");

scanf("%d",&b); // take input and save in b. c=a+b; // compute sum.

printf("The sum of %d and %d is %d.\n",a,b,c); // print sum.

}

## Output:



***Fig. 2.2:*** *Output of program to print addition.*

# Conclusion:

Sdkjklfdsgfjskfsdfkfsdlkfsdkljsfdklfdlkfdjfjld.