|  |
| --- |
| **Part A**  **Name:- Krutarth Patel**  **Roll No:- C093**  **Subject:- Data Structures and Algorithms**  **Program: MBA TECH (CE), B Tech (CE), Sem III/ BTI (CE), Sem VII** |
| **Aim:**  Implementation of various array operations like traversal, insertion and deletion using any real life application. |
| **Prerequisite:** C++ Programming |
| **Outcome:** To implement arrays and its operations. |
| **Theory:** An array is defined as the collection of similar type of data items stored at contiguous memory locations. The array is the simplest data structure where each data element can be randomly accessed by using its index number. Arrays are the derived data type which can store the primitive type of data such as int, char, double, float, etc.  **Declaration of C++ Array**  data\_type array\_name[array\_size];  int marks[5];  int marks[5]={20,30,40,50,60};  **OPERATIONS ON ARRAY**   1. Traversing an array 2. Inserting an element in an array 3. Searching an element in an array 4. Deleting an element from an array   **Traversing an array**  **Algorithm for array traversal**  Step 1: [INITIALIZATION] SET I= lower\_bound  Step 2: Repeat Steps 3 to 4 while I<= upper\_bound  Step 3: Apply Process to A[I]  Step 4: SET I=I+1  [END OF LOOP]  Step 5: EXIT  **Inserting an Element in an Array**   * Algorithm to insert at the end   Step 1: Set upper\_bound = upper\_bound+1  Step 2: Set A[upper\_bound] = VAL  Step 3: EXIT   * Algorithm to Insert an Element in the Middle of an Array   Step 1: [INITIALIZATION] SET I=N  Step 2: Repeat Steps 3 and 4 while I>= POS  Step 3: SET A[I + 1] = A[I]  Step 4: SET I=I–1  [END OF LOOP]  Step 5: SET N=N+1  Step 6: SET A[POS] = VAL  Step 7: EXIT  **Deleting an Element from an Array**   * **Algorithm** * Step 1: [INITIALIZATION] SET I= POS * Step 2: Repeat Steps 3 and 4 while I<=N–1 * Step 3: SET A[I] = A[I+1] * Step 4: SET I=I+1 * [END OF LOOP] * Step 5: SET N=N–1 * Step 6: EXIT |
| **Procedure:**   1. Open CodeBlock editor or visual studio editor and write the code in C++. 2. Complile and run the code |
| **Task:**   1. Insert an element in an array 2. Delete an element in an array 3. Find the smallest element in an array 4. Find highest and lowest element in an array. 5. Find the second largest element in an array. |
|  |

|  |
| --- |
| **Part B** |
| **Code: #include <iostream>**  **using namespace std;**  **int main() {**  **unsigned int numElements;**  **cout << "Enter the number of elements you want to store in array: ";**  **cin >> numElements;**  **int arr[numElements];**  **// Inserting elements into the array**  **for (int i = 0; i < numElements; i++) {**  **cout << "Enter the " << i + 1 << " element: ";**  **cin >> arr[i];**  **}**  **// Display elements**  **cout << "The elements are: ";**  **for (int i = 0; i < numElements; i++) {**  **cout << arr[i] << " ";**  **}**  **cout << endl;**  **// Searching for an element**  **int searchNumber = 0;**  **cout << "Enter the number you want to search: ";**  **cin >> searchNumber;**  **bool flag = false;**  **for (int i = 0; i < numElements; i++) {**  **if (arr[i] == searchNumber) {**  **cout << "The number you have searched for is at index number: " << i << endl;**  **flag = true;**  **break; // Stop after first occurrence**  **}**  **}**  **if (!flag) {**  **cout << "The number you have searched for is not in the array." << endl;**  **}**  **// Finding min and max values**  **int maxvalue = arr[0];**  **int minvalue = arr[0];**  **for (int i = 1; i < numElements; i++) {**  **if (arr[i] > maxvalue) {**  **maxvalue = arr[i];**  **}**  **if (arr[i] < minvalue) {**  **minvalue = arr[i];**  **}**  **}**  **cout << "The max value is: " << maxvalue << " and the min value is: " << minvalue << endl;**  **// Finding the second largest number (without <climits>)**  **int first = -1000000000; // Substitute for INT\_MIN**  **int second = -1000000000;**  **for (int i = 0; i < numElements; i++) {**  **if (arr[i] > first) {**  **second = first;**  **first = arr[i];**  **} else if (arr[i] > second && arr[i] < first) {**  **second = arr[i];**  **}**  **}**  **if (second == -1000000000) {**  **cout << "There is no second highest number (all elements might be equal)." << endl;**  **} else {**  **cout << "The second highest number is: " << second << endl;**  **}**  **// Deleting an element by index**  **int deletingIndex = 0;**  **cout << "Enter the deleting index: ";**  **cin >> deletingIndex;**  **if (deletingIndex >= 0 && deletingIndex < numElements) {**  **for (int i = deletingIndex; i < numElements - 1; i++) {**  **arr[i] = arr[i + 1];**  **}**  **numElements--;**  **cout << "Array after deletion: ";**  **for (int i = 0; i < numElements; i++) {**  **cout << arr[i] << " ";**  **}**  **cout << endl;**  **} else {**  **cout << "Invalid index to delete." << endl;**  **}**  **return 0;**  **}** |
| **Output:** |
| **Observation & Learning:**  Write your Observations & Learning after performing task  After performing the task, I observed how arrays store elements in contiguous memory locations and allow fast access using indices. I learned how to input, display, search, find max/min, and delete elements from an array, as well as how to find the second largest value using basic logic. This improved my understanding of array operations and memory management in C++. |
| **Answer Following Question**   1. How is the array stored in memory?   Arrays are stored **contiguously** in memory. Each element is placed next to the previous one, and the address of any element is calculated using the formula:  Address=Base address+(Index×Element size)   1. What are the advantages and disadvantages of an array?   **Advantages:**   * Fast access using index (constant time). * Easy to iterate and manipulate. * Memory locality improves performance.   **Disadvantages:**   * Fixed size (can't grow/shrink). * Insertion/deletion is slow (requires shifting). * Wastes memory if not fully used. |
| **Conclusion:**  We successfully implemented C++ program on arrays and its operations. |