**Practical No. 5**

**Aim:** Write a program to delete an element from a given array whose **value** is given or whose **position** is given.

**Theory**

An **array** stores elements in contiguous memory locations and has a fixed allocated size. To **delete** an element from an array we do a *logical deletion* by shifting all elements after the deleted slot one position to the left and decreasing the effective size (n) by 1. There are two common deletion modes:

* **Delete by position (index)**: the element at the given index is removed; elements to the right are shifted left.
* **Delete by value**: locate the first occurrence of the value, then shift elements after it left. If the value is not found, inform the user.

Because arrays are fixed-size, deletion does not free memory - it just reduces the number of considered elements.

**Algorithm (step-by-step)**

1. Read number of elements n and the array elements.
2. Display menu: 1) Delete by position 2) Delete by value.
3. If user chooses **position**:
   * Read pos (0-based or 1-based as per prompt - here we use 0-based).
   * If pos invalid (pos < 0 or pos ≥ n), show error.
   * Else, for i = pos to n-2, set arr[i] = arr[i+1].
   * Decrement n.
4. If user chooses **value**:
   * Read value.
   * Search from i = 0 to n-1 for first arr[i] == value.
   * If not found, show message.
   * Else shift elements from that index to left as in step 3 and decrement n.
5. Print the updated array.
6. End.

**Program (C)**

#include <stdio.h> // for printf and scanf

int main() {

int arr[100]; // array to hold up to 100 elements (adjustable)

int n; // current number of elements in array

int i, j; // loop indices

int choice; // user choice: delete by position or by value

// Read array size from user

printf("Enter number of elements in array: ");

scanf("%d", &n); // read number of elements

// Read array elements from user

printf("Enter %d elements:\n", n);

for (i = 0; i < n; i++) { // loop to read each element

scanf("%d", &arr[i]); // store element in arr[i]

}

// Ask the user which deletion mode to use

printf("\nChoose deletion mode:\n");

printf("1. Delete by position (0-based index)\n"); // option 1

printf("2. Delete by value\n"); // option 2

printf("Enter your choice (1 or 2): ");

scanf("%d", &choice); // read user's choice

if (choice == 1) { // deletion by position

int pos; // position to delete

printf("Enter position (0-based index): ");

scanf("%d", &pos); // read position

if (pos < 0 || pos >= n) { // validate position

printf("Invalid position!\n"); // error message for invalid index

} else {

// shift elements left starting from pos

for (i = pos; i < n - 1; i++) {

arr[i] = arr[i + 1]; // move next element into current slot

}

n = n - 1; // reduce effective size by 1

printf("Element at position %d deleted.\n", pos); // confirmation

}

}

else if (choice == 2) { // deletion by value

int value; // value to delete

int found = 0; // flag to indicate if value was found

printf("Enter value to delete: ");

scanf("%d", &value); // read value from user

// search for the first occurrence of value

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

if (arr[i] == value) { // value found at index i

found = 1; // set found flag

break; // exit loop with i pointing to found index

}

}

if (!found) { // if value not found in array

printf("Value %d not found in the array.\n", value); // inform user

} else {

// shift elements left starting from index i (where value was found)

for (j = i; j < n - 1; j++) {

arr[j] = arr[j + 1]; // move next element left

}

n = n - 1; // reduce size

printf("First occurrence of value %d deleted.\n", value); // confirmation

}

}

else { // invalid menu choice

printf("Invalid choice. Please run program again and enter 1 or 2.\n");

}

// Print updated array (if any elements remain)

if (n > 0) {

printf("Updated array: ");

for (i = 0; i < n; i++) { // print all valid elements

printf("%d ", arr[i]);

}

printf("\n"); // newline after printing array

} else {

printf("Array is now empty.\n"); // message if no elements remain

}

return 0; // normal program termination

}

**Output**

**Sample run 1 - delete by position (valid):**

Enter number of elements in array: 5

Enter 5 elements:

10 20 30 40 50

Choose deletion mode:

1. Delete by position (0-based index)

2. Delete by value

Enter your choice (1 or 2): 1

Enter position (0-based index): 2

Element at position 2 deleted.

Updated array: 10 20 40 50

**Sample run 2 - delete by value (found):**

Enter number of elements in array: 5

Enter 5 elements:

5 10 15 20 25

Choose deletion mode:

1. Delete by position (0-based index)

2. Delete by value

Enter your choice (1 or 2): 2

Enter value to delete: 15

First occurrence of value 15 deleted.

Updated array: 5 10 20 25

**Sample run 3 - delete by value (not found):**

Enter number of elements in array: 4

Enter 4 elements:

1 2 3 4

Choose deletion mode:

1. Delete by position (0-based index)

2. Delete by value

Enter your choice (1 or 2): 2

Enter value to delete: 10

Value 10 not found in the array.

Updated array: 1 2 3 4

**Sample run 4 - invalid position:**

Enter number of elements in array: 3

Enter 3 elements:

7 8 9

Choose deletion mode:

1. Delete by position (0-based index)

2. Delete by value

Enter your choice (1 or 2): 1

Enter position (0-based index): 5

Invalid position!

Updated array: 7 8 9

**Conclusion**

* Implemented a clear, beginner-friendly C program to delete an array element either by **position** or by **value**.
* Learned how to perform logical deletion in arrays by shifting elements and updating the effective size.
* Covered input validation (invalid position, value not found) and printed appropriate messages.
* Noted limitation: array memory remains allocated - deletion is logical; to reuse freed space or avoid repeated shifting, other data structures like linked lists or circular queues may be preferable depending on use-case.