**Aim:** To write a C program that deletes an element from an array, either by specifying its value or its position.

**Apparatus / Software Required:**

1. Computer System
2. GCC Compiler / Turbo C
3. Text Editor or IDE (Code::Blocks, Dev-C++, VS Code, etc.)

**Theory:**  
Arrays are stored in **contiguous memory**. Deletion in an array cannot simply remove an element; instead:

1. **If deletion by position**: Shift all elements after that position one step to the left.
2. **If deletion by value**: First locate the value, then shift the remaining elements to fill the gap.  
   The array size is reduced by one after deletion.

**Steps:**

1. Input the size of the array.
2. Input array elements.
3. Input choice for deletion (by position or by value).
4. If deletion by position:
   * Check if position is valid.
   * Shift all elements after that position one step left.
5. If deletion by value:
   * Find the value in the array.
   * Shift all elements after that index one step left.
6. Decrease size by 1.
7. Display the updated array.

**Advantages:**

1. Simple and straightforward implementation.
2. Can remove unwanted elements efficiently.

**Limitations:**

1. Deletion requires shifting of elements (O(n) time).
2. Fixed-size array → no shrinking of allocated memory.

**Algorithm:**

1. Start
2. Input size of array (n)
3. Input n elements into array
4. Input choice: delete by value or by position
5. If delete by position:
   * Check if position valid
   * Shift elements left from that position onward
6. If delete by value:
   * Search element in array
   * If found, shift elements left from that index onward
   * Else print "Value not found"
7. Print final array
8. Stop

**Program:**

#include <stdio.h> // Standard input-output header

int main() {

int arr[100], n, i, pos, value, choice, found = 0;

// Step 1: Input size of array

printf("Enter the number of elements: ");

scanf("%d", &n);

// Step 2: Input array elements

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

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

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

}

// Step 3: Ask for choice

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

printf("1. Delete by position\n");

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

scanf("%d", &choice);

if(choice == 1) {

// Delete by position

printf("Enter position (1 to %d): ", n);

scanf("%d", &pos);

if(pos < 1 || pos > n) {

printf("Invalid position!\n");

} else {

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

arr[i] = arr[i + 1]; // Shift left

}

n--; // Reduce size

printf("Array after deletion:\n");

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

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

}

printf("\n");

}

}

else if(choice == 2) {

// Delete by value

printf("Enter value to delete: ");

scanf("%d", &value);

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

if(arr[i] == value) {

found = 1;

break;

}

}

if(found) {

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

arr[i] = arr[i + 1]; // Shift left

}

n--; // Reduce size

printf("Array after deletion:\n");

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

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

}

printf("\n");

} else {

printf("Value not found in array!\n");

}

}

else {

printf("Invalid choice!\n");

}

return 0; // End of program

}

**Sample Output 1 (Delete by position):**

Enter the number of elements: 5

Enter 5 elements:

10

20

30

40

50

Choose deletion method:

1. Delete by position

2. Delete by value

1

Enter position (1 to 5): 3

Array after deletion:

10 20 40 50

**Sample Output 2 (Delete by value):**

Enter the number of elements: 6

Enter 6 elements:

5

10

15

20

25

30

Choose deletion method:

1. Delete by position

2. Delete by value

2

Enter value to delete: 20

Array after deletion:

5 10 15 25 30

**Sample Output 3 (Value not found):**

Enter the number of elements: 4

Enter 4 elements:

2

4

6

8

Choose deletion method:

1. Delete by position

2. Delete by value

2

Enter value to delete: 5

Value not found in array!

**Conclusion:**  
The program successfully demonstrates deletion of an element in an array, either by specifying its **position** or its **value**. Although deletion requires shifting elements (O(n) complexity), it works correctly for small to medium arrays.