**Write a program for insertion and deletion operations in an array.**

**Insertion**

#include<stdio.h>

void main()

{

int a[10],i,n;

printf("enter the limit\n");

scanf("%d",&n);

printf("enter the elements\n");

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

{

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

}

printf("Array is: \n");

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

{

printf("%d\t",a[i]);

}

}

**Deletion**

#include <stdio.h>

void main ()

{

int arr[50];

int pos, i, num; // declare int type variable

printf (" \n Enter the number of elements in an array: \n ");

scanf (" %d", &num);

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

for (i = 0; i<num; i++ )

{

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

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

}

// enter the position of the element to be deleted

printf( " Define the position of the array element where you want to delete: \n ");

scanf (" %d", &pos);

// check whether the deletion is possible or not

if (pos>= num+1)

{

printf (" \n Deletion is not possible in the array.");

}

else

{

// Delete element by shifting elements

for (i = pos - 1; i<num -1; i++)

{

arr[i] = arr[i+1];

}

printf (" \n The resultant array is: \n");

for (i = 0; i<num - 1; i++)

{

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

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

}

}

}

**Write a program to search for an element in an array using**

**Linear Search**

#include<stdio.h>

void main(){

int arr[10],size,i,Element;

//ask the user for the size of the the array

printf("Enter size of the array: ");

scanf("%d",&size);

// ask user to provide values of array

printf("Enter any %d integer values: ",size);

//save user values in arr[] array

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

{

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

}

printf("Enter the element to be Search: ");

scanf("%d",&Element);

// loop each element of the array and check if element is equal to the element

//we need to find

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

{

if(Element == arr[i])

{

// show element found message with it's position

printf("Element is found at %d index", i+1);

break;

}

}

//check if we have reached the end of the list and not found element yet

if(i == size)

{

printf("Given element is not found in the array!!!");

}

}

4. Write a program to merge two arrays.

#include<stdio.h>

void main()

{

int n1,n2,n3,a1[20],a2[20],a3[20],i,j;

printf("entersize of 1st array\n");

scanf("%d",&n1);

printf("enter size of 2nd array\n");

scanf("%d",&n2);

n3=n1+n2;

printf("\nenter elements of 1 st array\n");

for(i=0;i<n1;i++)

{

scanf("%d",&a1[i]);

a3[i]=a1[i];

}

printf("\nenter elements of 2nd array\n");

for(i=0;i<n2;i++)

{

scanf("%d",&a2[i]);

a3[k]=a2[i];

k++;

}

printf("\nmerged array \n");

for(i=0;i<n3;i++)

{

printf("\n%d\t",a3[i]);

}

}

**Binary Search**

#include<stdio.h>

void main()

{

int c, first, last, middle, n, search, array[100];

printf("Enter number of elements\n");

scanf("%d", & n);

printf("Enter %d integers\n", n);

for (c = 0; c < n; c++) scanf("%d", & array[c]);

printf("Enter value to find\n");

scanf("%d", & search);

first = 0;

last = n - 1;

middle = (first + last) / 2;

while (first <= last)

{

if (array[middle] < search) first = middle + 1;

else if (array[middle] == search) {

printf("%d found at location %d.\n", search, middle + 1);

break;

}

Else

last = middle - 1;

middle = (first + last) / 2;

}

if (first > last) printf("Not found! %d is not present in the list.\n", search);

}

**Algorithm**

1. **Input the Number of Elements**:
   * Prompt the user to enter the number of elements (n).
2. **Input the Array Elements**:
   * Read n integers and store them in an array.
3. **Input the Value to Search**:
   * Ask the user to input the value (search) to find in the array.
4. **Initialize Variables**:
   * Set first to 0 (start of the array).
   * Set last to n - 1 (end of the array).
   * Calculate middle = (first + last) / 2.
5. **Binary Search Loop**:
   * Repeat the following steps while first <= last:
     + **Check if Middle Element Matches**:
       - If array[middle] == search, print the location of the element and exit.
     + **Adjust the Search Range**:
       - If array[middle] < search, set first = middle + 1 (search the right half).
       - Otherwise, set last = middle - 1 (search the left half).
     + Recalculate middle = (first + last) / 2.
6. **Element Not Found**:
   * If first > last, print a message saying the value was not found.
7. **End**:
   * Stop the program.

### ****Code Explanation****

#### ****Initial Setup****

1. **Variables**:
   * first = 0: The starting index of the array.
   * last = n - 1: The ending index of the array (since arrays are zero-indexed, n-1 is the last element).
   * middle = (first + last) / 2: The middle index of the current search range.

#### ****Binary Search Loop****

The while loop runs as long as first <= last. This ensures that the search space is valid. Here’s what happens inside the loop:

### ****1. Compare the Middle Element****

* If the middle element (array[middle]) is **less than** the search value:
  + The target value must be in the right half of the array.
  + Update first = middle + 1 to narrow the search range to the right half.

### ****2. Match Found****

* If the middle element (array[middle]) is **equal to** the search value:
  + Print the message:

printf("%d found at location %d.\n", search, middle + 1);

* + - search: The value being searched for.
    - middle + 1: The position of the element (array positions are often displayed starting from 1 for human readability).
  + Use break to exit the loop since the value has been found.

### ****3. Target in the Left Half****

* If the middle element (array[middle]) is **greater than** the search value:
  + The target value must be in the left half of the array.
  + Update last = middle - 1 to narrow the search range to the left half.

### ****4. Update the Middle Index****

* After updating first or last, recalculate the middle index:

middle = (first + last) / 2;

### ****Condition After Loop****

* If first > last, it means the search value is **not present** in the array.
* Print:

printf("Not found! %d is not present in the list.\n", search);