**ASSIGNMENT – 1.1**

**Ques.1** What is recursion? Explain recursion with example.

**Ans.** Recursion is the process of repeating items in a self-similar way.

* In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.
* **Syntax.**

void recursion() {

recursion(); /\* function calls itself \*/

}

int main() {

recursion();

}

* The C programming language supports recursion, i.e., a function to call itself.
* But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.
* Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.
* **Example.**

#include <stdio.h>

int factorial(int i) {

if(i <= 1) {

return 1;

}

return i \* factorial(i - 1);

}

int main() {

int i = 5;

printf("Factorial of %d is %d\n", i, factorial(i));

return 0;

}

**Output.**

Factorial of 5 is 120

**Ques.2** Explain difference between recursion and iteration.

**Ans.**

| **PROPERTY** | **RECURSION** | **ITERATION** |
| --- | --- | --- |
| **Definition** | Function calls itself. | A set of instructions repeatedly executed. |
| **Application** | For functions. | For loops. |
| **Termination** | Through base case, where there will be no function call. | When the termination condition for the iterator ceases to be satisfied. |
| **Usage** | Used when code size needs to be small, and time complexity is not an issue. | Used when time complexity needs to be balanced against an expanded code size. |
| **Code Size** | Smaller code size | Larger Code Size. |
| **Time Complexity** | Very high(generally exponential) time complexity. | Relatively lower time complexity(generally polynomial-logarithmic). |

**Ques.3** Write a program to find sum of first n natural numbers using recursion.

**Ans.** #include <stdio.h>

int addNumbers(int n);

int main() {

int num;

printf("Enter a positive integer: ");

scanf("%d", &num);

printf("Sum = %d", addNumbers(num));

return 0;

}

int addNumbers(int n) {

if (n != 0)

return n + addNumbers(n - 1);

else

return n;

}

**Output.**



**Ques.4** Write a program to calculate factorial of a number using recursion.

**Ans.** #include<stdio.h>

long int multiplyNumbers(int n);

int main() {

int n;

printf("Enter a positive integer: ");

scanf("%d",&n);

printf("Factorial of %d = %ld", n, multiplyNumbers(n));

return 0;

}

long int multiplyNumbers(int n) {

if (n>=1)

return n\*multiplyNumbers(n-1);

else

return 1;

}

**Output.**



**Ques.5** Write a program to calculate GCD of two numbers using recursion.

**Ans.** #include <stdio.h>

int hcf(int n1, int n2);

int main() {

int n1, n2;

printf("Enter two positive integers: ");

scanf("%d %d", &n1, &n2);

printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1, n2));

return 0;

}

int hcf(int n1, int n2) {

if (n2 != 0)

return hcf(n2, n1 % n2);

else

return n1;

}

**Output.**



**Ques.6** Write a program for Tower of Hanoi problem using recursion.

**Ans.** #include <stdio.h>

void towers(int, char, char, char);

int main()

{

int num;

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

scanf("%d", &num);

printf("The sequence of moves involved in the Tower of Hanoi are :\n");

towers(num, 'A', 'C', 'B');

return 0;

}

void towers(int num, char frompeg, char topeg, char auxpeg)

{

if (num == 1)

{

printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);

return;

}

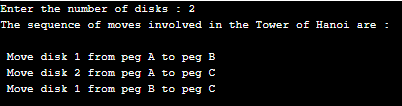
towers(num - 1, frompeg, auxpeg, topeg);

printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);

towers(num - 1, auxpeg, topeg, frompeg);

}

**Output.**

****

**ASSIGNMENT – 1.2**

**Ques.1**What is data structure? Why it is important?

**Ans.** Data Structure can be defined as the collection of data objects which provides a way of storing and managing data in the computer so that it can be used.

* Various Data Structures types are arrays, Linked List, Stack, Queue, etc. Data Structures are widely used in almost every aspect of Computer Science for simple as well as complex computations.
* Data structures are used in all areas of computer science such as Artificial Intelligence, graphics, Operating system etc.
* Data Structures are the key part of many computer algorithms as they allow the programmers to do data management in an efficient way.
* A right selection of data structure can enhance the efficiency of computer program or algorithm in a better way.

**Ques.2** What is data?  Explain how data is used for different purpose by taking example.

### Ans. Data – a collection of facts (numbers, words, measurements, observations, etc) that has been translated into a form that computers can process.

### Human-readable (also known as unstructured data) refers to information that only humans can interpret and study, such as an image or the meaning of a block of text.

### Machine-readable (or structured data) refers to information that computer programs can process. A program is a set of instructions for manipulating data.

### Example:

### Data in the news.

### Personal Data.

### Transactional Data.

### Web Data.

### Sensor Data.

**Ques.3** Differentiate between structured and unstructured data with proper example.

### Ans.

### Structured vs. Unstructured Data

**Ques.4** Write a program to sort the given numbers using user defined function.

**Ans.** #include<stdio.h>

void asc\_sort(int a[100], int n);

void main()

{

int a[100], i, n;

printf("Enter n:\n");

scanf("%d", &n);

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

{

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

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

}

asc\_sort(a,n);

printf("Array in ascending order is:\n");

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

{

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

}

}

void asc\_sort(int a[10], int n)

{

int i, j, temp;

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

{

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

{

if(a[i]>a[j])

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

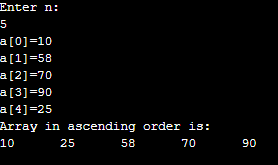
}

}

}

}

**Output.**



**Ques. 5** Write a program to search a number given by user using user defined function.

**Ans.** #include<stdio.h>

int linear\_search(int\*, int, int);

int main()

{

int array[100], search, c, n, position;

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

scanf("%d",&n);

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

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

scanf("%d",&array[c]);

printf("Enter the number to search\n");

scanf("%d",&search);

position = linear\_search(array, n, search);

if ( position == -1 )

printf("%d is not present in array.\n", search);

else

printf("%d is present at location %d.\n", search, position+1);

return 0;

}

int linear\_search(int \*pointer, int n, int find)

{

int c;

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

{

if ( \*(pointer+c) == find )

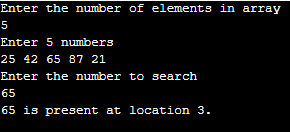
return c;

}

return -1;

}

**Output.**

****

**Ques. 6** Write a program to print Fibonacci sequence using  user defined function.

**Ans.** #include<stdio.h>

void fibonacciSeries(int range)

{

int a=0, b=1, c;

while (a<=range)

{

printf("%d\t", a);

c = a+b;

a = b;

b = c;

}

}

int main()

{

int range;

printf("Enter range: ");

scanf("%d", &range);

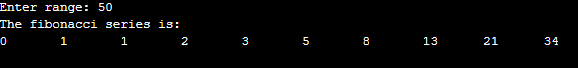
printf("The fibonacci series is: \n");

fibonacciSeries(range);

return 0;

}

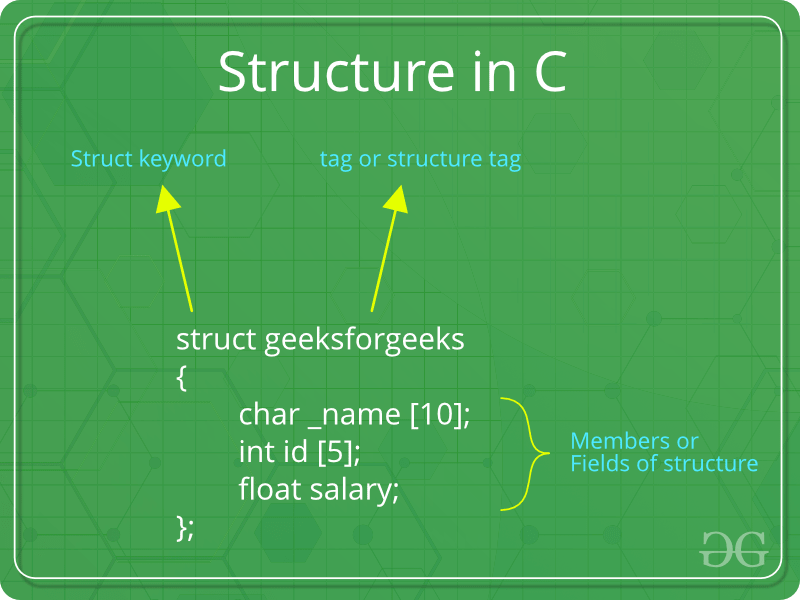
**Output.**

****

**ASSIGNMENT – 1.3**

**Ques. 1** What is structure? How to initialize structure? Explain with help of examples.

**Ans.** A structure is a user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.



‘struct’ keyword is used to create a structure. Following is an example.

|  |
| --- |
| struct address  {     char name[50];     char street[100];     char city[50];     char state[20];     int pin;  };  Structure members **cannot be** initialized with declaration. For example the following C program fails in compilation. |

struct Point

{

   int x = 0;  // COMPILER ERROR:  cannot initialize members here

   int y = 0;  // COMPILER ERROR:  cannot initialize members here

};

struct Point

{

   int x, y;

};

int main()

{

   // A valid initialization. member x gets value 0 and y

   // gets value 1.  The order of declaration is followed.

   struct Point p1 = {0, 1};

}

**Ques. 2** Store Information(enrollment no., name, branch, and semester) of 10 students using Structure and display  
that information.

**Ans.**

**Ques.3** Write a program to calculate distance between two points  (ex. x and y) given by user using Structure.

**Ans.** #include <stdio.h>

#include <stdlib.h>

#include <math.h>

struct Point {

int x, y;

};

double getDistance(struct Point a, struct Point b)

{

double distance;

distance = sqrt((a.x - b.x) \* (a.x - b.x) + (a.y-b.y) \*(a.y-b.y));

return distance;

}

int main()

{

struct Point a, b;

printf("Enter coordinate of point a: ");

scanf("%d %d", &a.x, &a.y);

printf("Enter coordinate of point b: ");

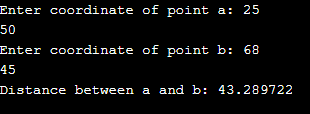
scanf("%d %d", &b.x, &b.y);

printf("Distance between a and b: %lf\n", getDistance(a, b));

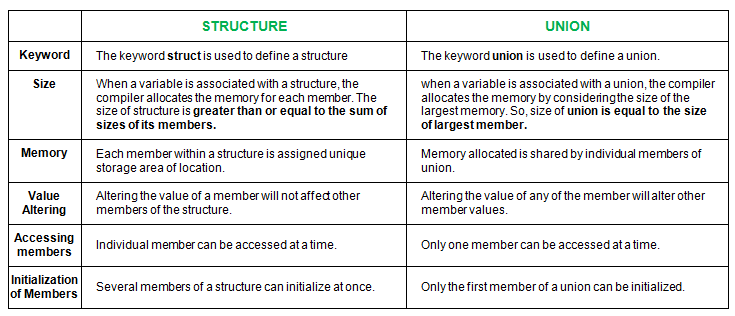
return 0;

}

**Output.**



**Ques .4** Differentiate between structure and union by taking proper example.

**Ans.** 

**ASSIGNMENT – 1.4**

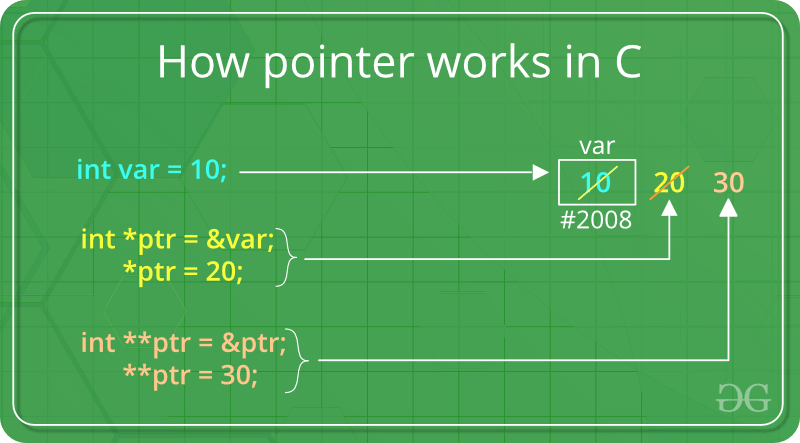
**Ques. 1** What is pointer? Explain with help of example.

**Ans.** Pointers are symbolic representation of addresses. They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures. It’s general declaration in C has the format:

**Syntax:**

datatype \*var\_name;

int \*ptr;



#include <stdio.h>

void display()

{

int var = 20;

int \*ptr;

ptr = &var;

printf("Value at ptr = %p \n",ptr);

printf("Value at var = %d \n",var);

printf("Value at \*ptr = %d \n", \*ptr);

}

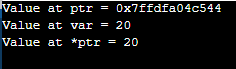
int main()

{

display();

}

**Output.**

****

**Ques.2** Write a program to swap two variables using a pointer.

**Ans.** #include <stdio.h>

void swap(int \*x,int \*y)

{

int t;

t = \*x;

\*x = \*y;

\*y = t;

}

int main()

{

int num1,num2;

printf("Enter value of num1: ");

scanf("%d",&num1);

printf("Enter value of num2: ");

scanf("%d",&num2);

printf("Before Swapping: num1 is: %d, num2 is: %d\n",num1,num2);

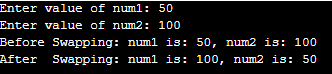
swap(&num1,&num2);

printf("After Swapping: num1 is: %d, num2 is: %d\n",num1,num2);

return 0;

}

**Output.**



**Ques.3** Write a program to sort an array using a pointer.

**Ans.** #include <stdio.h>

void sort(int n, int\* ptr)

{

int i, j, t;

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

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

if (\*(ptr + j) < \*(ptr + i)) {

t = \*(ptr + i);

\*(ptr + i) = \*(ptr + j);

\*(ptr + j) = t;

}

}

}

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

printf("%d ", \*(ptr + i));

}

int main()

{

int n = 5;

int arr[] = { 0, 23, 14, 12, 9 };

sort(n, arr);

return 0;

}

**Output.**

****

**Ques.4** Brief about 'pointer to pointer' and 'array of pointers' concept.

**Ans.** A pointer to a pointer is a form of multiple indirection, or a chain of pointers. Normally, a pointer contains the address of a variable.

* When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below.
* 
* A variable that is a pointer to a pointer must be declared as such. This is done by placing an additional asterisk in front of its name.
* For example, the following declaration declares a pointer to a pointer of type int −
* int \*\*var;

**array of pointers.**

There may be a situation when we want to maintain an array, which can store pointers to an int or char or any other data type available. Following is the declaration of an array of pointers to an integer −

int \*ptr[MAX];

It declares **ptr** as an array of MAX integer pointers. Thus, each element in ptr, holds a pointer to an int value. The following example uses three integers, which are stored in an array of pointers, as follows –

#include <stdio.h>

const int MAX = 3;

int main () {

int var[] = {10, 100, 200};

int i, \*ptr[MAX];

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

ptr[i] = &var[i];

}

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

printf("Value of var[%d] = %d\n", i, \*ptr[i] );

}

return 0;

}

**Output.**

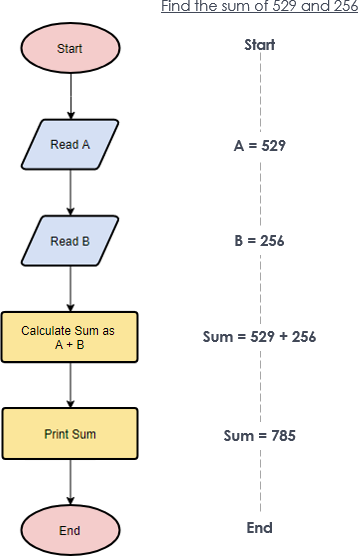
****

**ASSIGNMENT – 1.5**

**Ques.1** What is flow chart? Explain with help of example.

**Ans.** A flowchart is simply a graphical representation of steps. It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes.

Typically, a flowchart shows the steps as boxes of various kinds, and their order by connecting them with arrows.

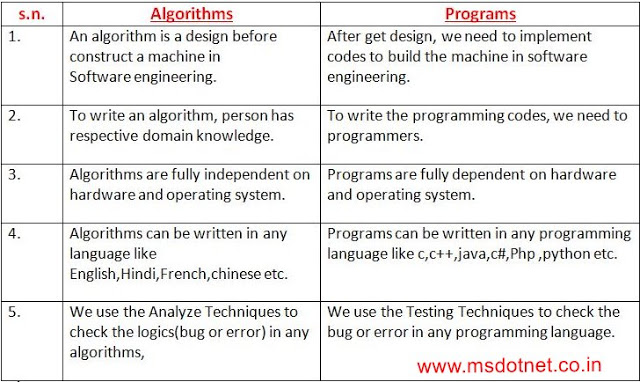


**Ques.2** Define term algorithm. Differentiate between algorithm and program.

**Ans.** It is a set of instructions that tells you what to do step by step.

An algorithm is something conceptual and can be described using language, flowcharts or pseudocode.

An algorithm can be implemented in different programming languages​​.



**Ques.3** What do you mean by time complexity? Why it is important?

**Ans. The time complexity of an algorithm is the total amount of time required by an algorithm to complete its execution.**

In simple words, every piece of code we write, takes time to execute. The time taken by any piece of code to run is known as the time complexity of that code. The lesser the time complexity, the faster the execution.

The algorithm that performs the task in the smallest number of operations is considered the most efficient one in terms of the time complexity.

**Ques.4** Find out time complexity to calculate average of three numbers.

**Ans.** The time complexity to calculate average of three numbers is O(3).

**Ques.5** Find out time complexity for swapping of two numbers.

**Ans.** The time complexity for swapping of two numbers is O(2

**ASSIGNMENT – 1.6**

**Ques.1** Calculate time complexity of factorial of a number.

**Ans.** The time complexity of factorial of a number is O(n).

**Ques.2** Write a program to search an element from given array. Also find its time complexity.

**Ans.** #include <stdio.h>

int search(int arr[], int n, int x)

{

int i;

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

if (arr[i] == x)

return i;

return -1;

}

int main(void)

{

int arr[] = { 2, 3, 4, 10, 40 };

int x = 10;

int n = sizeof(arr) / sizeof(arr[0]);

int result = search(arr, n, x);

(result == -1)

? printf("Element is not present in array")

: printf("Element is present at index %d", result);

return 0;

}

**Output.**

****

The complexity to search an element from an array is O(n).

**Ques.3** Find time complexity to sort an array in ascending or descending order.

**Ans.** The time complexity to sort an array in ascending or descending order is O(n).

**Ques.4** Find time complexity to multiply two matrix.

**Ans.** The time complexityto multiply two matrix is O().

**Ques.5** Find time complexity for the addition of 3D matrix.

**Ans.** The time complexityto multiply two matrix is O().

**ASSIGNMENT – 1.7**

**Ques.1** Explain best case, average case and worst case time complexity.

**Ans. Best Case**

* In the best case analysis, we calculate lower bound on running time of an algorithm.
* We must know the case that causes minimum number of operations to be executed. In the linear search problem, the best case occurs when x is present at the first location.
* The number of operations in the best case is constant (not dependent on n).
* So time complexity in the best case would be Θ(1)

**Average Case**

* In average case analysis, we take all possible inputs and calculate computing time for all of the inputs. Sum all the calculated values and divide the sum by total number of inputs.
* We must know (or predict) distribution of cases.
* For the linear search problem, let us assume that all cases are [uniformly distributed](http://en.wikipedia.org/wiki/Uniform_distribution_%28discrete%29) (including the case of x not being present in array).
* So we sum all the cases and divide the sum by (n+1). Following is the value of average case time complexity.

**Worst Case**

* In the worst case analysis, we calculate upper bound on running time of an algorithm.
* We must know the case that causes maximum number of operations to be executed.
* For Linear Search, the worst case happens when the element to be searched (x in the above code) is not present in the array.
* When x is not present, the search() functions compares it with all the elements of arr[] one by one. Therefore, the worst case time complexity of linear search would be Θ(n).

**Ques.2** Find all the three cases of time complexity for linear search.

**Ans.** Best case time complexity for linear search is O(1).

Average case time complexity for linear search is O(n).

Worst case time complexity for linear search is O(n).

**Ques.3**

**Ans.**

**ASSIGNMENT – 1.8**

**Ques.1** Explain best case, average case and worst case time complexity sorting.

**Ans.**

Algorithm Best Case Average Case Worst Case

|  |  |  |  |
| --- | --- | --- | --- |
| Bubble Sort | Ω(N) | Θ(N2) | O(N2) |
| Selection Sort | Ω(N2) | Θ(N2) | O(N2) |
| Insertion Sort | Ω(N) | Θ(N2) | O(N2) |
| Merge Sort | Ω(N log N) | Θ(N log N) | O(N log N) |
| Heap Sort | Ω(N log N) | Θ(N log N) | O(N log N) |
| Quick Sort | Ω(N log N) | Θ(N log N) | O(N2) |
| Radix Sort | Ω(N k) | Θ(N k) | O(N k) |
| Count Sort | Ω(N + k) | Θ(N + k) | O(N + k) |

**Ques.2** Find all the three cases of time complexity for factorial.

**Ans.** Time complexity for factorial for best case is O(1).

Time complexity for factorial for average case is O(n).

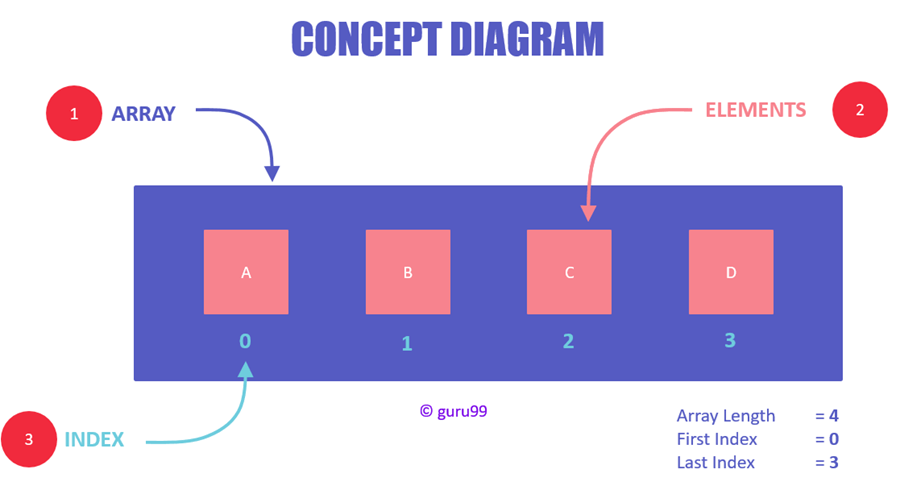
Time complexity for factorial for worst case is O(n).

**Ques.3** Discuss the array data structure with example.

**Ans.** An array is a data structure for storing more than one data item that has a similar data type. The items of an array are allocated at adjacent memory locations.

* These memory locations are called **elements** of that array.
* The total number of elements in an array is called **length**.
* The details of an array are accessed about its position. This reference is called **index**or**subscript**.

## Concept of Array

[](https://www.guru99.com/images/1/102319_0559_ArrayinData1.png)

Concept Diagram of Arrays

* Elements are stored at contiguous memory locations.
* An index is always less than the total number of array items.
* In terms of syntax, any variable that is declared as an array can store multiple values.
* Almost all languages have the same comprehension of arrays but have different ways of declaring and initializing them.
* However, three parts will always remain common in all the initializations, i.e., array name, elements, and the data type of elements.

Here, are some reasons for using arrays:

* Arrays are best for storing multiple values in a single variable
* Arrays are better at processing many values easily and quickly
* Sorting and searching the values is easier in arrays

## Basic Operations

Following are the basic operations supported by an array.

* **Traverse** − print all the array elements one by one.
* **Insertion** − Adds an element at the given index.
* **Deletion** − Deletes an element at the given index.
* **Search** − Searches an element using the given index or by the value.
* **Update** − Updates an element at the given index.

Example.  
#include <stdio.h>

int main() {

int LA[] = {1,3,5,7,8};

int item = 10, k = 3, n = 5;

int i = 0, j = n;

printf("The original array elements are :\n");

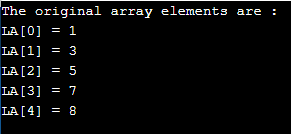
for(i = 0; i<n; i++) {

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

}

}

Output.



**ASSIGNMENT – 1.9**

**Ques.1** Discuss time complexity to access element from first, middle and last index from given array.

#### Ans. Worst Case - O(N)

If we want to insert an element to index 0, then we need to shift all the elements to right.

For example, if we have 5 elements in the array and need to insert an element in arr[0], we need to shift all those 5 elements one position to the right.

In general, if we have **n** elements we need to shift all **n** elements.

So, worst case time complexity will be **O(n)**. where n = number of elements in the array.

#### Best Case - O(1)

O(1) indicates that the insertion operation is not depended on the size of the array. Whatever the array size it will always take constant time.

**Average Case – O(N)**

**Ques.2** Calculate time complexity to add "n" elements in array.

**Ans.** The time complexity to add n elements in array requires O(n) time.

**Ques.3**  Implement or write an algorithm to insert element at given index in an array.

**Ans.** #include <stdio.h>

int main()

{

int arr[100] = { 0 };

int i, x, pos, n = 10;

for (i = 0; i < 10; i++)

arr[i] = i + 1;

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

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

printf("\n");

x = 50;

pos = 5;

n++;

for (i = n-1; i >= pos; i--)

arr[i] = arr[i - 1];

arr[pos - 1] = x;

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

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

printf("\n");

return 0;

}

**Output.**

****

**ASSIGNMENT – 1.10**

**Ques.1** Write an algorithm or a program to remove element from specific index.

**Ans.** #include <stdio.h>

int main()

{

int array[100], position, c, n;

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

scanf("%d", &n);

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

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

scanf("%d", &array[c]);

printf("Enter the location where you wish to delete element\n");

scanf("%d", &position);

if (position >= n+1)

printf("Deletion not possible.\n");

else

{

for (c = position - 1; c < n - 1; c++)

array[c] = array[c+1];

printf("Resultant array:\n");

for (c = 0; c < n - 1; c++)

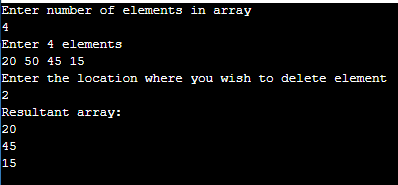
printf("%d\n", array[c]);

}

return 0;

}

**Output.**



**Ques.2** Find out time complexity for all the features listed above.

**Ans.** Want a list that can store multiple values: constant

* list should be of user specific data (int, char, float): constant
* Empty list has no value: constant
* Able to add element: constant
* Able to insert element at specific index: linear

i.e. O(number of elements in the array - the given desired index)

* Able to remove element from specific index: linear

i.e. O(number of elements in the array - 1 - the given desired index)

* Can access/modify values at specific index: constant
* Count number of elements: constant

**Ques.3** Calculate time complexity to print all the values from list.

**Ans.** If there n elements present in the array then we don’t have any other option than to iterate atleast n times.

* Hence, below is the require code:

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

cout<<array[i]<<” “;

* From the code, we can clearly see that the time complexity for printing all the values of the list is linear.