# Pointer

## What is Address Of(&) Operator and Value Of(*) Operator in C.

**Address of Operator (&)**
The & is a unary operator in C which returns the memory address of the passed operand. This is also known as **address of** operator.

**Value of Operator (*)**
The * is a unary operator which returns the value of object pointer by a pointer variable. It is known as **value of** operator. It is also used for declaring pointer variable.
For Example

```
int A = 100;
int *ptr = &A;
```

In the first statement, we first declare an integer variable and initialize it with value 100. In next statement, we are declaring a pointer to a variable of type int and initializing it with address of  A.

## What is NULL pointer in C

**NULL pointer in C** is a pointer which is pointing to nothing. It is used to initialize a pointer at the time of declaration if we don't have any explicit value to initialize. It is a good practice to initialize a pointer with NULL to ensure that it is not pointing to a random memory location.

```
int *ptr = NULL;
```

Pointer ptr is initialized with NULL. Pointer ptr is not pointing to any valid memory location.

### What happens when we try to access NULL pointer in C.

As we know that, **NULL pointer** in C is a pointer which is pointing to nothing. If we try to access the memory location pointed by a null pointer, program can crash.

## What are the advantages of using Pointers in C

- We can dynamically allocate or deallocate space in memory at run time by using pointers.
- Using pointers we can return multiple values from a function.
- We can pass arrays to a function as call by Reference.
- Pointers are used to efficiently access array elements, as array elements are stored in adjacent memory locations. If we have a pointer pointing to a particular element of array, then we can get the address of next element by simply incrementing the pointer.
- Pointers are used to efficiently implement dynamic Data Structures like Queues, Stacks, Linked Lists, Tress etc.
- The use of pointers results into faster execution of program.

# Pointer Arithmetic in C

We can perform arithmetic operations on the pointers like addition, subtraction, etc. However, as we know that pointer contains the address, the result of an arithmetic operation performed on the pointer will also be a pointer if the other operand is of type integer. In pointer-from-pointer subtraction, the result will be an integer value. Following arithmetic operations are possible on the pointer in C language:

- Increment
- Decrement
- Addition
- Subtraction

---

# Incrementing Pointer in C

If we increment a pointer by 1, the pointer will start pointing to the immediate next location. This is somewhat different from the general arithmetic since the value of the pointer will get increased by the size of the data type to which the pointer is pointing.

The Rule to increment the pointer is given below:

1. new_address= current_address + i * size_of(data type)

## Decrementing Pointer in C

Like increment, we can decrement a pointer variable. If we decrement a pointer, it will start pointing to the previous location. The formula of decrementing the pointer is given below:

new_address= current_address - i * size_of(data type)

## C Pointer Addition

We can add a value to the pointer variable. The formula of adding value to pointer is given below:

new_address= current_address + (number * size_of(data type))

### 32-bit

For 32-bit int variable, it will add 2 * number.

#### 64-bit

For 64-bit int variable, it will add 4 * number.

# C Pointer Subtraction

Like pointer addition, we can subtract a value from the pointer variable. Subtracting any number from a pointer will give an address. The formula of subtracting value from the pointer variable is given below:

new_address= current_address - (number * size_of(data type))

## 32-bit

For 32-bit int variable, it will subtract 2 * number.

## 64-bit

For 64-bit int variable, it will subtract 4 * number.