## Pointers :

Before discussing about pointers let me tell you when we define and initialize a variable at that time we come to know about these things :

- Memory block i.e. variable get some space in RAM and we can think of that as a block.
- Name of memory block or Variable's name
- Content of that block i.e. value in that variable
- Address of memory block i.e. unique address which allows us to access that variable.

We can print address of any variable by using printf function as :

printf("%d",&variable_name);

## Pointer :

- Pointer is a variable that contains address of another variable. It means it is a variable but this variable contains address or memory address of any other variable.
- It can be of type int, char, array, function, or any other pointer.
- Its size depends on architecture.
- Pointers in C Language can be declared using *(asterisk symbol).

```
#include<stdio.h>


int main()
{
       int x=5;
       int *a =&x;


       printf("%d\n",&x );
       printf("%d",a );
       return 0;
}
```

After checking above example you will understand the basics of pointers.

So, pointers are nothing just a variable which stores the address of other variables and by using pointers we can access other variables too and can even manipulate them.

Now let’s see about some of the operators which we use with Pointers :

- Address of Operator (&) :

· It is a unary operator.

· Operand must be the name of the variable.

· & operator gives address no. of variable.

· & is also known as “Referencing Operator”.

- Indirection Operator :

· * is indirection operator.

· It is also known as “Dereferencing Operator”.

· It is a unary operator.

· It takes address as an argument.

· * returns the content/container whose address is it’s argument.

```
#include<stdio.h>

int main()
{
       int a=5;

       printf("%d\n",&a );
       printf("%d",a );
       return 0;
}
```

In above example you will see that we printed variable ‘a’ address and its value. So in second printf statement you can see we used two unary operators i.e. * and "&" operator. As we know unary operator associativity is from right to left so first of all & of operator will be resolved and then the address of variable ‘a’ will be the argument of * operator. That’s how we can print values or can use these Address of Operator and Indirection Operator.

```
#include<stdio.h>


int main()
{
       int x,*a;
       return 0
}
```

When * (indirection operator) is written before any pointer variable like (*a) then that whole variable resembles like original variable i.e. the variable to whom the pointer is pointing.

Or,

We can say that *a pointer becomes the variable whose address is in a pointer.

## Null Pointer :

- A pointer that is not assigned any value but NULL is known as NULL pointer.
- In computer programming NULL pointer is a pointer that does not point to any object, variable or function.
- We can use it to initialize a pointer variable when that pointer variable isn’t assigned any valid memory address yet.
- int *ptr = NULL;

```
#include<stdio.h>
int main()
{
       printf("Pointer Basics\n");
       int a =5;
       int *p=NULL;



       printf("%d\n", p);


       return 0;
}
```

Uses of Pointers:

- Dynamic Memory Allocation
- Arrays, Functions and Structures
- Return multiple values from a function
- Pointer reduces the code and improves the performance

```c
#include <stdio.h>

int main()
{
    printf("Lets learn about pointers\n");
    int a=76;
    int *ptra = &a;
    int *ptr2 = NULL;
    printf("The address of pointer to a is %p\n", &ptra);
    printf("The address of a is %p\n", &a);
    printf("The address of a is %p\n", ptra);
    printf("The address of some garbage is %p\n", ptr2);
    printf("The value of a is %d\n", *ptra);
    printf("The value of a is %d\n", a);
    return 0;
}
```