Practical - 01
Introduction to Pointers and structures using C

Introduction:

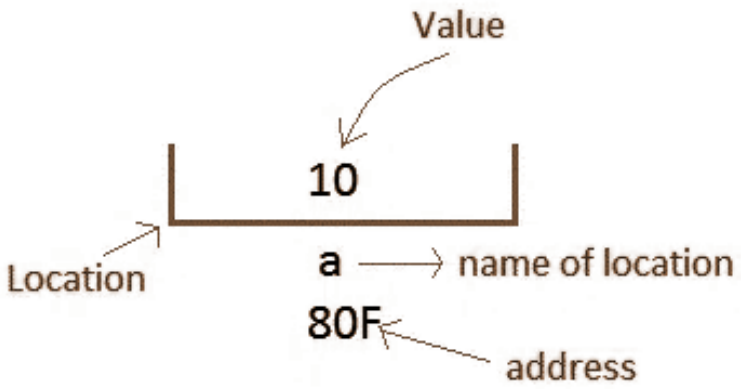
A Pointer in C language is a variable that holds a memory address. This memory address is the address of another variable(mostly) of the same data type. In simple words, if one variable stores the address of the second variable then the first variable can be said to point towards the second variable.

We can use variables to hold the address of a memory location and such variables are known as pointer variables. Normal variables are used to store data items of a particular data type (char, int, float etc). Before using a variable in a program, we declare it at the beginning. Similarly we need to declare a pointer variable too in a special way – to let the compiler know we have declared a variable as a pointer (not as a normal variable). To do this we have the * operator – known as indirection operator in C.

Whenever a variable is declared in a program, the system allocates a location i.e an address to that variable in the memory, to hold the assigned value.

Let us assume that the system has allocated memory location 80F for a variable a.

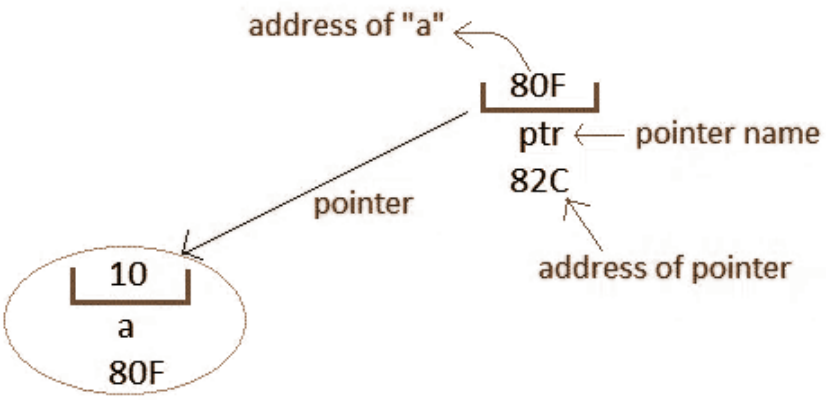
int a = 10;



We can access the value 10 either by using the variable name a or by using its address 80F.

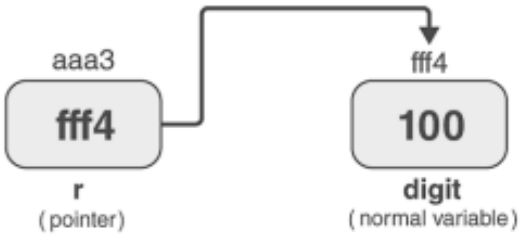
The question is how we can access a variable using its address? Since the memory addresses are also just numbers, they can also be assigned to some other variable. The variables which are used to hold memory addresses are called Pointer variables.

A pointer variable is therefore nothing but a variable which holds an address of some other variable. And the value of a pointer variable gets stored in another memory location.



Pointer variable declaration :
The syntax to declare a pointer variable is

(data type) *(variable name);
Ex:- int *ptr ;



It is possible to assign address of single variable or that of an array or the address of a structure etc to a pointer variable.

There are two pointer operators in C, they are:
& operator - The & operator returns the memory address of its operand.
*** operator -** The * operators is the complement of &. This operator returns the value located at the given address.

Reference And Dereference Operators:

Before diving deeper into the concept of pointer let's understand some basics that will help us later on. While using pointers you will definitely use '&' and '*' operators. Now is the time to understand their meaning and use.
First, let's understand the Reference operator often called the 'Address of' operator. Using (ampersand) operator with a variable returns us a memory location also known as the address of the given variable.

Example,

int *ptr;
int a;
ptr = &a;
The address of variable 'a' is stored in variable ptr.

Now, let's understand the De-Referencing or 'Value at' operator which is denoted by an asterisk (*). It helps in retrieving the value from the memory location which is stored in the pointer variable

Example,

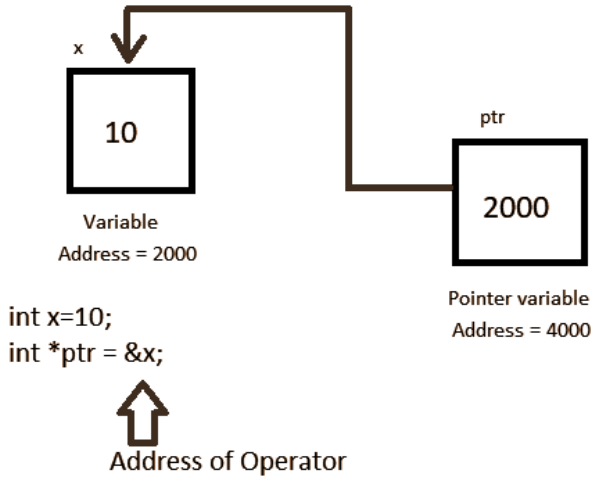
int *ptr;
int a;
****ptr = &a;***
printf("Value of a = %dn", *ptr);

Sample Program:

```
#include<stdio.h>
int main(void)
{
    int a = 10;
    // declare a pointer
    int *ptr;
    // assign value to pointer
    ptr = &a;
    printf("Value at ptr is: %d \n", *ptr);
    printf("Address pointed by ptr is: %p \n", ptr);
    return 0;
}
```

Sample Output:

```
Value at ptr is: 10
Address pointed by ptr is: 0x7fff99c0e6c4
```



Advantages:

- Pointers are more efficient in handling Arrays in C and Structures in C.
- Pointers allow references to function and thereby help in passing of function as arguments to other functions.
- Pointers also provide means by which a function in C can change its calling arguments.
- It reduces the length of the program and its execution time as well.
- It allows the C language to support Dynamic Memory management.

Note:

While declaring a pointer variable, if it is not assigned to anything then it contains garbage value. And that can lead to unexpected errors in your program. Hence, it is recommended to assign a *NULL* value to it.

When we assign *NULL* to a pointer, it means that it does not point to any valid address. NULL denotes the value 'zero'.

int *ptr = 0;

int *ptr = NULL;

- A pointer variable stores the address of a variable. We use * to declare and identify a pointer.
- We can find the address of any variable using the & (ampersand) operator.
- The declaration int *a doesn't mean that a is going to contain an integer value. It means that a is going to contain the address of a variable of int type.
- We can dereference a pointer variable using a * operator. Here, the * can be read as 'value at'.

Exercise:

1. Define a structure which has members that include name, idno and marks. Write a C program to read the information about N students and print the name and idno of the students having marks less then M.
2. Write a C program that takes a Student structure array [for example struct Student st[10] defines an array called student that consists of 10 elements. Each element is defined to be of the type struct Student]. The program should store marks of 5 subjects and calculate total and percentage. Display each subject's marks, total and percentage for all students.
3. Write a C program to create a number variable. Create a pointer variable for this number variable. Create another pointer-to-pointer variable. Display the address and value of all the variables including pointer variables.
4. Write a C program to create an array of 15 elements. create a pointer which points to an array. Now print the base address of the array. Then display the array elements using pointer arithmetic.
5. Write a program to create a value array and another pointer array. Both array sizes should be 5. Now store some values in the value array. Then store the address of these five elements in the pointer array. Then print the address and value of each of the pointer arrays.
6. Write a C program to swap 2 numbers using pass by value and pass by reference to a function.