**Declaring, Initializing and using a pointer variable**

**Declaration of Pointer variable**

General syntax of pointer declaration is,

**datatype \*pointer\_name;**

Data type of a pointer must be same as the data type of the variable to which the pointer variable is pointing. void type pointer works with all data types, but is not often used.

Here are a few examples:

int \*ip // pointer to integer variable  
float \*fp; // pointer to float variable  
double \*dp; // pointer to double variable  
char \*cp; // pointer to char variable

**Initialization of Pointer variable**

Pointer Initialization is the process of assigning address of a variable to a pointer variable.

Pointer variable can only contain address of a variable of the same data type.

In C language address operator **&** is used to determine the address of a variable. The **&** (immediately preceding a variable name) returns the address of the variable associated with it.

Ex :

#include<stdio.h>   
void main()  
{  
 int a = 10;  
 int \*ptr; //pointer declaration  
 ptr = &a; //pointer initialization - address of **a** will be stored in pointer **ptr**  
}

Pointer variable always point to variables of same datatype. Let's have an example to showcase this:

Ex :

#include<stdio.h>  
void main()  
{  
 float a;  
 int \*ptr;  
 ptr = &a; // ERROR, type mismatch  
}

If you are not sure about **which variable's address to assign** to a pointer variable while declaration, it is **recommended to assign a NULL** value to your pointer variable. A pointer which is assigned a NULL value is called a NULL pointer.

Ex :

#include <stdio.h>  
int main()   
{  
 int \*ptr = NULL;  
 return 0;  
}

**Dereferencing pointer**

Getting the value from an address stored by pointer

Int n =10, x;

Int \*ptr;

ptr = &n;

x = \*ptr; //same as x= \*&n because (ptr same as &n)

Note : use \* on pointer for dereferencing

EX :

#include <stdio.h>

int main()

{

int a, \*p; // declaring the variable and pointer

a = 10;

p = &a; // initializing the pointer

printf("%d\n", \*p); //this will print the value of 'a' - **dereferencing**

printf("%d\n", \*&a); //this will also print the value of 'a'

printf("%u\n", &a); //this will print the address of 'a'

printf("%u\n", p); //this will also print the address of 'a'

printf("%u\n", &p); //this will print the address of 'p'

return 0;

}

**Note** :

Points to remember while using pointers:

1. While declaring/initializing the pointer variable, \* indicates that the variable is a pointer.
2. The address of any variable is given by preceding the variable name with Ampersand &.
3. The pointer variable stores the address of a variable. **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 storing integer value.**
4. To access the value of a certain address stored by a pointer variable, \* is used. Here, the \* can be read as **'value at'**.