Pointers

→ Pointer Variable is a special variable which stores the address of other variable.

EX :- int a, *p;

p = &a;

Here 'a' is simple variable & p is a pointer variable

→ Pointer directly deal with address of other variable so it is faster.

→ To use pointer variable, first we have to declare & after declaration we can use it

int *p; → Declaration of pointer variable

int a; → Declaration of simple variable

a = 9;

p = &a;

printf("%d", *p); → will give '9' output

*p = 100; → it change the value of a to 100

printf("%d", a); → will print '100'

call by value & call by reference

→ Functions can be invoked in two ways: call by value or call by Reference. These two ways are generally differentiated by the type of values passed to them as parameters.

→ The parameters passed to function are called actual parameters whereas the parameters received by function are called formal parameters.

→ call by value:- In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory location. so any-changes made inside functions are not reflected in actual parameters of ~~called~~ caller.

→ call by Reference: Both the actual and formal parameters refer to same locations, so any changes made inside the function are actually reflected in actual parameters of caller.

// c prog. to illustrate call by value

```
#include<stdio.h>
#include<conio.h>
// function prototype
void swap(int, int);
// main function
void main()
{ int a=10, b=20;
// pass by values
swap(a, b);
printf("a=%d, b=%d", a, b);
getch();
}
// swap functions that swaps two values.
void swap(int x, int y)
{ int t;
t = x;
x = y;
y = t;
Printf("x=%d y=%d", x, y);
}
```

o/p

x = 20 y = 10

a = 10 b = 20

→ Thus actual values of 'a' & 'b' remain unchanged even after exchanging the values of 'x' & 'y'

// c prog. to illustrate call by Reference

```
#include<stdio.h>
#include<conio.h>
// function prototype
void swap(int *, int *);
// main function
void main()
{ int a=10, b=20;
// Pass reference
swap(&a, &b);
printf("a=%d b=%d", a, b);
getch();
}
// function to swap two variables by reference
void swap(int *x, int *y)
{ int t;
t = *x;
*x = *y;
*y = t;
Printf("x=%d y=%d", x, y);
}
```

o/p: x = 20 y = 10

a = 20 b = 10

→ Thus actual values of 'a' and 'b' get changed after exchanging values of x and y

⇒ Pointers and array

```
int *p, a[3] = {8, 9, 11};
```

p = a; → It assign base address of first element of array variable a

p = a is equal to p = &a[0], both are same

example :-

```
void main()
{  int a[5] = {1, 2, 3, 4, 5};
   int *p;
   p = &a[2];
   Printf("*p = %d\n", *p); → Print third element '3'
   Printf("*(p+1) = %d\n", *(p+1)); → Print '4'
   Printf("*(p-1) = %d", *(p-1)); → Print '2'
   getch();
}
```

⇒ pointers and Strings :

We can declare dynamic string variable using pointer

ex: char *p; → p is a dynamic character string variable which can store any string of dynamic size

```
char *p = "hello computer";
gets(p); → will print hello computer.
```