Pointer in C

Pointer Variable: A variable that can hold or store the address of another variable is called as pointer variable.

Syntax to declare pointer variable:

datatype *pointervarname;

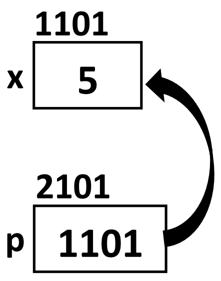
Example: int *p;
float *q;

In above example p is pointer variable which can be stored the address of int type variable only. Q is also pointer variable which can be stored the address of float type of variable.

Address of Operator (&): The ampersand symbol is called as Address of operator. If we can used address of operator with variable then it will gives the address of variable.

Example:

```
int x=5;
int *p;
p=&x;
printf("\n x=%d, Address of x=%u, Size of x=%d", x, &x, sizeof(x));
printf("\n x=%d, Address of x=%u, Size of x=%d", *p, p, sizeof(*p));
```



Output:

Pointer to Pointer: A variable that can hold or store the address of pointer variable is called as pointer to pointer variable.

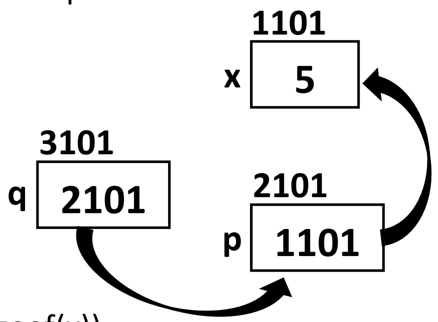
Syntax:

Datatype **pointername;

Example:

```
int **q;
q=&p;

printf("\n x=%d, Address of x=%u, Size of x=%d", x, &x, sizeof(x));
printf("\n x=%d, Address of x=%u, Size of x=%d", *p, p, sizeof(*p));
printf("\n x=%d, Address of x=%u, Size of x=%d", **q, *q, sizeof(**q));
```



Pointer Arithmetic: When we perform arithmetic operations on variable then it can be perform operation on value. But if we can perform arithmetic operations on pointer variable then it can be perform operation on address by memory size.

Example:
```
int x=5;
int *p;
p=&x; // Here p hold the address of x. i.e. p=1101
x++; // x=6
p++; // p=1105
```
Output:

Parameter Passing Methods:

There are two types of Parameter Passing Methods as

1) **Call by value:** When we can passed value at the time of function call is called as call by value parameter passing method.
   Example: Except prg131 all other program using function are the example of call by value.
2) **Call by Address or Reference:** When we can passed address instead of value at the time of function call is called as call by address or reference parameter passing method.

```
/*131) Swap or interchange the content of 2 number*/
#include<stdio.h>
int main()
{
        int x,y;
        void swap(int,int);
        printf("\n Enter 2 Number:");
        scanf("%d%d",&x,&y);
        printf("\n Before swap x=%d, y=%d",x,y);
        swap(x,y);
        printf("\n After swap x=%d, y=%d",x,y);
}
void swap(int a, int b)
{
        int t;
        t=a;
        a=b;
        b=t;
}
/*
The above program will be executed successfully, But the changes we are made on formal
parameter
a and b will not be changed or affected on actutal parameter x and y. Because formal parameter
scope
and life is same as local variable.
Function allowed to return only one value to calling block. So if we want to achieve more than one
output from function then we can use Pointer variable.
*/

/*132) write a function Swap() that will swap or  interchange the content of 2 number */
#include<stdio.h>
int main()
{
        int x,y;
        void swap(int*,int*);
        printf("\n Enter 2 Number:");
        scanf("%d%d",&x,&y);
        printf("\n Before swap x=%d, y=%d",x,y);
        swap(&x,&y);
        printf("\n After swap x=%d, y=%d",x,y);
}
```

```
void swap(int *a, int *b)
{
	int t;
	t=*a;
	*a=*b;
	*b=t;
}
/*133) write a function processNumber() that will calculate count of digit, sum of digit &
average of digit of given number*/
#include<stdio.h>
int main()
{
	int n, cnt, sum;
	float avg;
	int processNumber(int,int*,float*);
	printf("\n Enter Number:");
	scanf("%d",&n);
	cnt=processNumber(n,&sum,&avg);
	printf("\n Count:%d",cnt);
	printf("\n Sum:%d",sum);
	printf("\n Average:%0.2f",avg);
}
int processNumber(int x, int *s, float *a)
{
	int count=0,sum=0;
	float avg;
	while(x!=0)
	{
		count++;
		sum=sum+x%10;
		x=x/10;
	}
	avg=(float)sum/count;
	*s=sum;
	*a=avg;
	return count;
}
```