Functions in C

Parameters in a Function

Actual parameters

- The arguments or parameters used inside the parentheses of a function call.
- The parameters passed to a function.
- These parameters are sending the values.

Formal parameter

- an identifier that represents a corresponding actual argument in a function definition
- The parameters received by a function

- add (m, n);

  Actual Parameter

-

```
int add (int x, int y)  //Formal parameter
 {
     return (a+b);
  }
```

Call by value

- There are two methods to pass the data into the function in C language, i.e., *call by value* and *call by reference*.
- In call by value method, the value of the actual parameters is copied into the formal parameters.
- We can not modify the value of the actual parameter by the formal parameter.
- Different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter.

```
#include<stdio.h>
void change(int x, int y);
int main() {
  int x=10, y =20;
  printf("Before function call x=%d and y =%d \n", x, y);
 change(x, y);          //passing value in function
 printf("After function call x=%d and y =%d \n", x, y );
return 0;
}
void change(int x, int y) {
printf("Before adding value inside function  x=%d and y =%d \n", x, y );
x= 30;                        // These vars are local to the function
y= 40
printf("After changing value inside function  x=%d and y =%d \n", x, y );
}
```

Call by reference

- In call by reference, the memory allocation is similar for both formal parameters and actual parameters.
- In call by reference, the address of the variable is passed into the function call as the actual parameter instead of values.
- Any changes made to the formal parameters will get reflected to the actual parameters,
- All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address.

```
 int x=10, y =20;
*ptr2);
 change(&x, &y);
  printf("x = %d, y = %d", x, y);
```

```
 change(int *ptr1, int

{
      *ptr1 = 30;
          *ptr2 = 40;
       }
```

```
#include<stdio.h>
void change(int *x, int *y);
int main() {
  int x=10, y =20;
  printf("Before function call x=%d and y =%d \n", x, y);
 change(&x, &y);   //passing by reference
 printf("After function call x=%d and y =%d \n", x, y );
return 0;
}
void change(int *x, int *y) {
printf("Before adding value inside function  x=%d and y =%d \n", x, y );
x= 30;    // These vars are local to the function
y= 40
printf("After changing value inside function  x=%d and y =%d \n", *x, *y );
}
```

Program to print the addresses of the variables