Passing Arguments to a Function
- To pass arguments to a function, list them in parentheses following function name.
- The number of arguments and the type of each argument must match the parameters in the function header and prototype.
- For example, if a function is defined to take two type int arguments, one must pass it exactly two int arguments—no more, no less—and no other type.
- If one tries to pass a function an incorrect number and/or type argument, the compiler will detect it, based on the information in the function prototype.
- If the function takes multiple arguments, the arguments listed in the function call are assigned to the function parameters in order.
- The first argument to the first parameter, the second argument to the second parameter, and so on.
- Multiple arguments are assigned to function parameters in order.
- Each argument can be any valid C expression: a constant, a variable, a mathematical or logical expression, or even another function (one with a return value).

Parameters passing methods
1. Call by Reference
   a. In this method the address of an argument is copied into the parameter.
   b. Inside the subroutine the address is used to access the actual argument used in the call.
   c. This means that the change made to the parameter affect the variable used to call the subroutine.
   d. The address of x and y are passed as the parameter to the function swap().
2. Call by Value –
   a. This method copies value of the argument into the formal parameter of the subroutine.
   b. Therefore changes made to the parameters of the subroutines have no effect on the variable used to call it.
   c. The value of \( t \) is passed as the parameter to the function sqr().
   d. The following Program demonstrates call by value.

```c
#include<stdio.h>

void interchange(int *num1,int *num2)
{
    int temp;
    temp = *num1;
    *num1 = *num2;
    *num2 = temp;
}

int main()
{
    int num1=50,num2=70;
    interchange(&num1,&num2);
    printf("\nNumber 1 : %d",num1);
    printf("\nNumber 2 : %d",num2);
    return(0);
}
```

**Output:**
Number 1 : 70
Number 2 : 50
Recursive Functions

- The term recursion refers to a situation in which a function calls itself either directly or indirectly.
- Indirect recursion occurs when one function calls another function that then calls the first function.
- C allows recursive functions, and they can be useful in some situations.

Program: Recursion can be used to calculate the factorial of a number.

```c
#include <stdio.h>
int factorial(unsigned int i)
{
    if(i <= 1)
    {
        return 1;
    }
    return i * factorial(i - 1);
}
int main()
{
    int i = 15;
    printf("Factorial of %d is %d\n", i, factorial(i));
    return 0;
}
```

Program: Recursion can be used to calculate the Fibonacci number. 1 1 2 3 5 8 13 21..

```c
#include <stdio.h>
int fibonacci(int i)
{
    if(i == 0)
    {
        return 0;
    }
    if(i == 1)
    {
        return 1;
    }
    return fibonacci(i-1) + fibonacci(i-2);
}
int main()
{
    int i;
    for (i = 0; i < 10; i++)
    {
        printf("%d\t%dn", i, fibonacci(i));
    }
    return 0;
}
```