Chapter 5 - Functions and Recursions

Sometimes our program gets bigger in size, and its not possible for a programmer to track which piece of code is doing what.

The function is a way to break our code into chunks so that it is possible for a programmer to reuse them.

What is a function?

A function is a block of code that performs a particular task. A function can be reused by the programmer in a given program any number of times.

Example and syntax of a function:

```
#include<stdio.h>
void  display();                    // Function prototype
int main(){
int a;
display();                           // Function call
return(0);
}

void display(){                              // Function definition
printf("Hi I am display");
}
```

Copy

Function prototype:

Function prototype is a way to tell the compiler about the function we are going to define in the program.

Here void indicates that the function returns nothing.

Function call:

Function call is a way to tell the compiler to execute the function body at the time the call is made.

Note that the program execution starts from the main function in the sequence the instructions are written.

Function definition:

This part contains the exact set of instructions that are executed during the function call. When a function is called from main(), the main function falls asleep and gets temporarily suspended. During this time, the control goes to the function being called when the function body is done executing main() resumes.

Quick Quiz: Write a program with three functions,

1. Good morning function which prints "Good Morning."
2. Good afternoon function which prints "Good Afternoon."
3. Good night function, which prints "Good night."

main() should call all of these in order 1 - 2 - 3.

Important Points:

- Execution of a c program starts from main()
- A c program can have more than one function
- Every function gets called directly or indirectly from main()
- There are two types of functions in c. Let's talk about them.

Types of Functions:

- Library functions: Commonly required functions grouped together in a library file on disk.
- User-defined functions: These are the functions declared and defined by the user.

Why use functions?

- To avoid rewriting the same logic again and again
- To keep track of what we are doing in a program

- To test and check logic independently

Passing values to functions:

We can pass values to a function and can get a value in return from a function

int sum(int a, int b)

The above prototype means that sum is a function which takes values a(of type int) and b(of type int) and returns a value of type int

Function definition of sum can be:

```
int   sum(int a, int b){

int c;                                    => a and b are
parameters

c=a+b;

return c;

}
```

Copy

Now we can call sum(2,3) [here 2 and 3 are arguments]; from main to get 5 in return.

```
int d=sum(2,3);   => d becomes 5
```

Copy

Note:

1. Parameters are the values or variable placeholders in the function definition. Ex: a & b

2. Arguments are the actual values passed to the function to make a call. Ex: 2 & 3

3. A function can return only one value at a time.

4. If the passed variable is changed inside the function, the function call doesn't change the value in the calling function.

```
int change(int a){

a=77;                          => Misnomer

return 0;

}
```

Copy

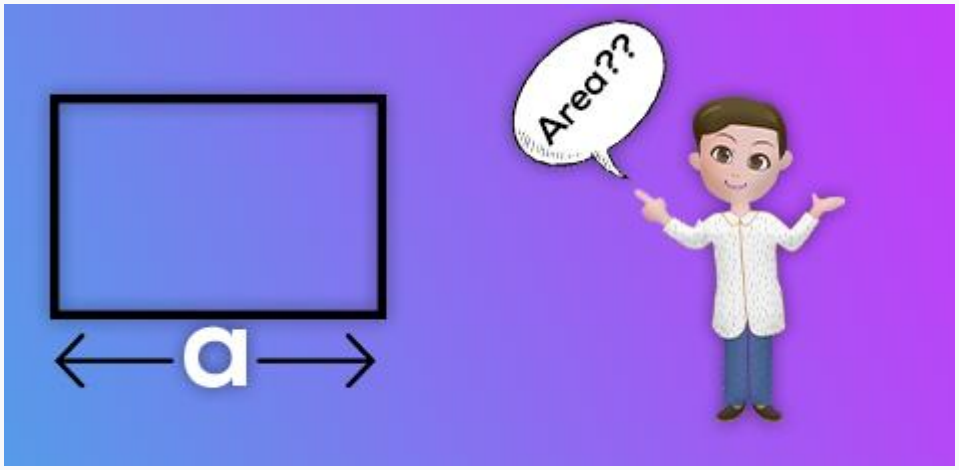
change is a function which changes a to 77. No, if we call it from main like this.

```
int b=22;

change(b);                              => The value of b remains 22

printf("b is %d",b);                 => prints "b is 22"
```

Copy

This happens because a copy of b is paused to the change function.

Quick Quiz: Use the library function to calculate the area of a square with side a.



Recursion

A function defined in C can call itself. This is called recursion.

A function calling itself is also called a recursive function.

Example of Recursion:

A very good example of recursion is factorial

```
factorial(n) = 1x 2 x 3..........x n

factorial(n)= 1 x 2 x 3..........n-1 x n

factorial(n)= factorial of (n-1) x n
```

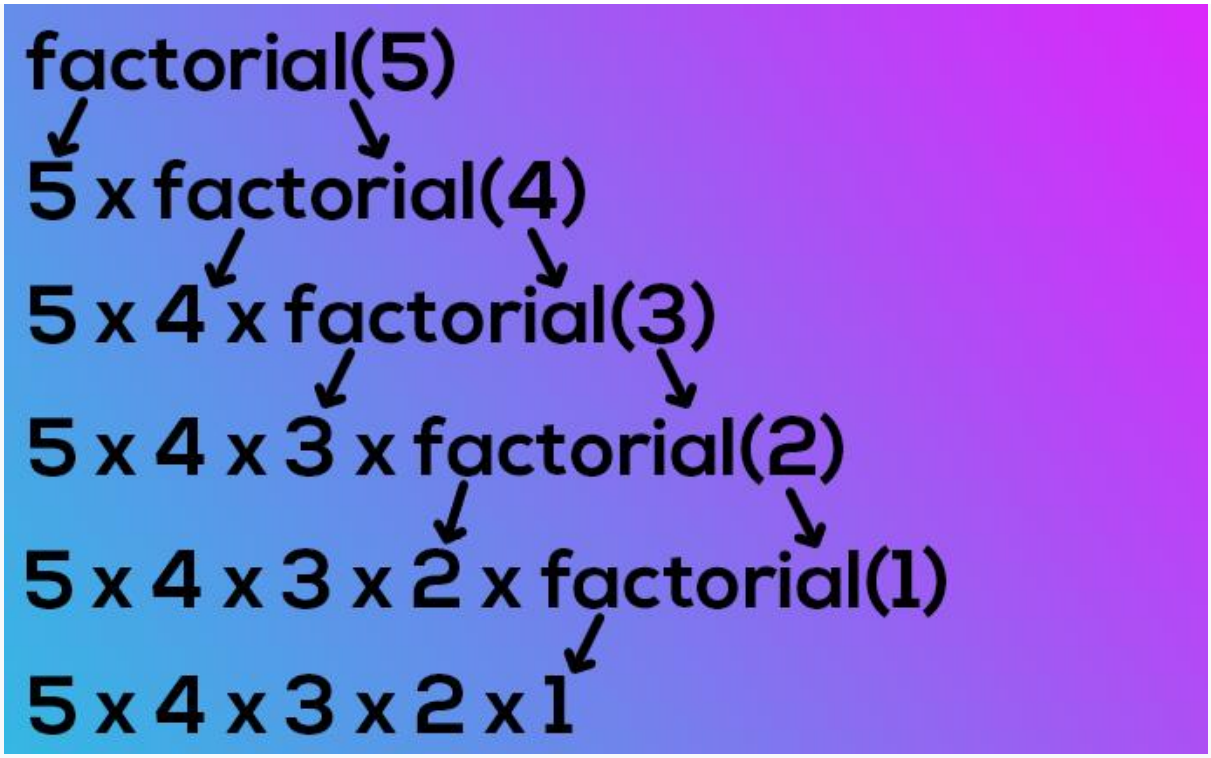
Copy

Since we can write factorial of a number in terms of itself, we can program it using recursion.

```
// Function to calculate factorial using recursion

int factorial(int x){
int f;
if(x==0||x==1)
return 1;
else
f=x * factorial(x-1);
return f;
}
```

Copy

How does it work?



Important Notes:

1. Recursion is sometimes the most direct way to code an algorithm

2. The condition which doesn't call the function any further in a recursive function is called the base condition.

3. Sometimes, due to a mistake made by the programmer, a recursive function can keep running without returning, resulting in a memory error.

Chapter 5- Practice Set

1. Write a program using functions to find the average of three numbers.
2. Write a function to convert Celcius temperature into Fahrenheit.
3. Write a function to calculate the force of attraction on a body of mass m exerted by earth. ($g=9.8m/S^2$)
4. Write a program using recursion to calculate the n^{th} element of the Fibonacci series.
5. What will the following line produce in a C program: printf("%d %d %d\n",a,++a,a++);
6. Write a recursive function to calculate the sum of first n natural numbers.
7. Write a program using functions to print the following pattern(first n lines):

```
*


***


*****
```

KEEP LEARNING & KEEP PRACTICING :)

Nitin-787