URBAN EDGE

Chapter 5 - Functions and Recursion

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

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 which 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

URBAN EDGE

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. Lets talk about them

Types of functions

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

Why use functions?

1. To avoid rewriting the same logic again and again.
2. To keep track of what we are doing in a program
3. To test and check logic independently.

URBAN EDGE

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;
    c = a+b;
    return c;
}
```

⇒ a and b are parameters

Now we can call sum(2, 3); from main to get 5 in return.

↳ Here 2 & 3 are arguments

```
int d = sum(2, 3);
```

⇒ d becomes 5

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 doesnt change the value in the calling function.

```
int change (int a) {
    a = 77;
    return 0;
}
```

⇒ Misnomer

change is a function which changes a to 77. Now if we call it from main like this

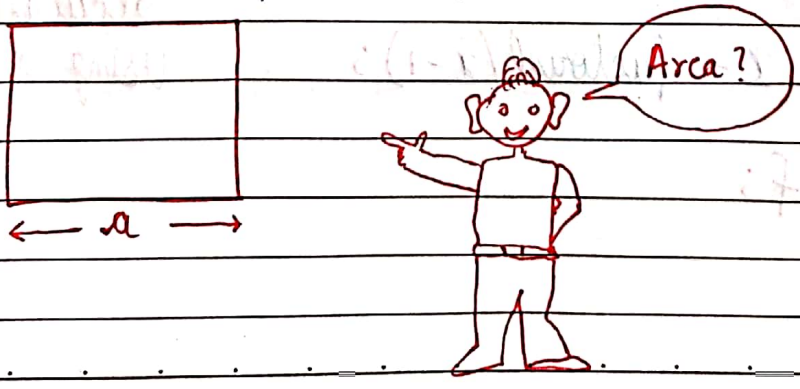
```
int b = 22
change (b);
printf(" b is %d", b);
```

⇒ The value of b remains 22

⇒ prints "b is 22"

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

Quick Quiz → Use the library functions to calculate the area of a square with side a.



URBAN EDGE

Recursion

A function defined in C can call itself. This is called recursion.
A function calling itself is also called 'recursive' function.

Example of Recursion

A very good example of recursion is factorial

$$factorial(n) = 1 \times 2 \times 3 \cdots \times n$$

$$factorial(n) = \underbrace{1 \times 2 \times 3 \cdots n-1}_{} \times n$$

$$factorial(n) = factorial(n-1) \times n$$

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

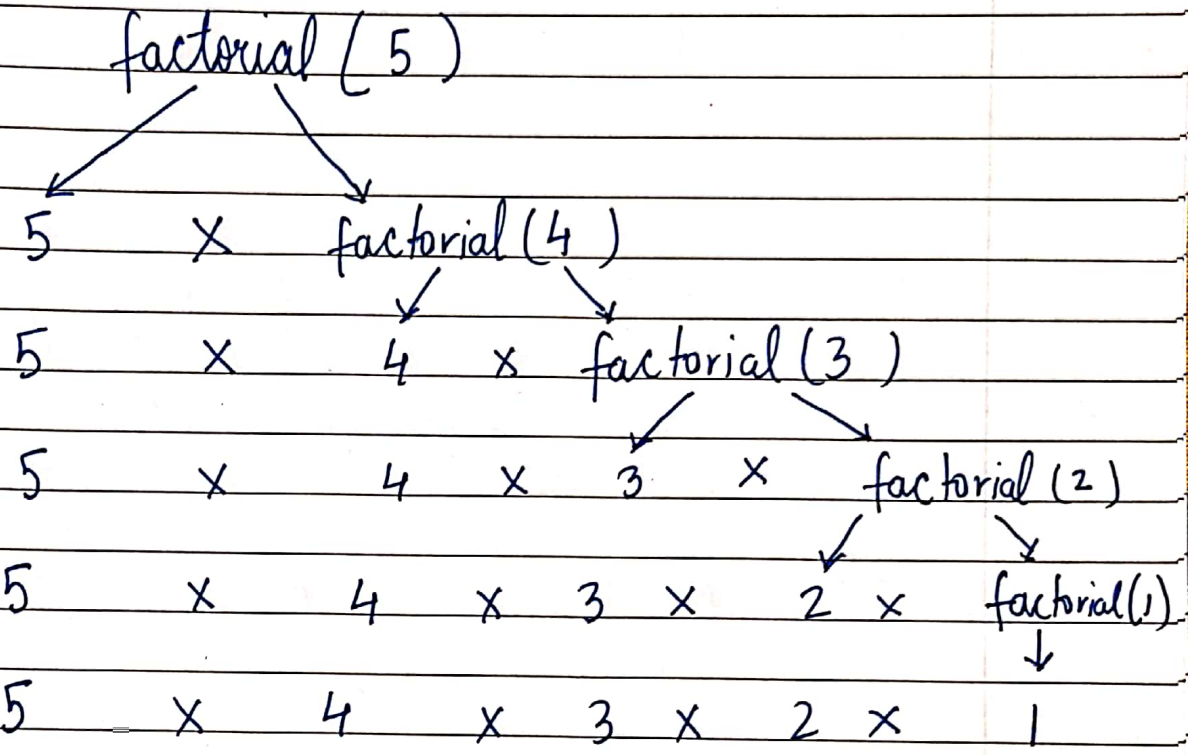
```
int factorial (int x) {
  int f;
  if (x == 0 || x == 1)
      return 1;
  else
    f = x * factorial(x-1);

  return f;

  }
```

⇒ A program to calculate factorial using recursion

How does it work?



Important Notes:

1. Recursion is sometimes the most direct way to code an algorithm
2. The condition which doesnt call the function any further in a recursive function is called as 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.