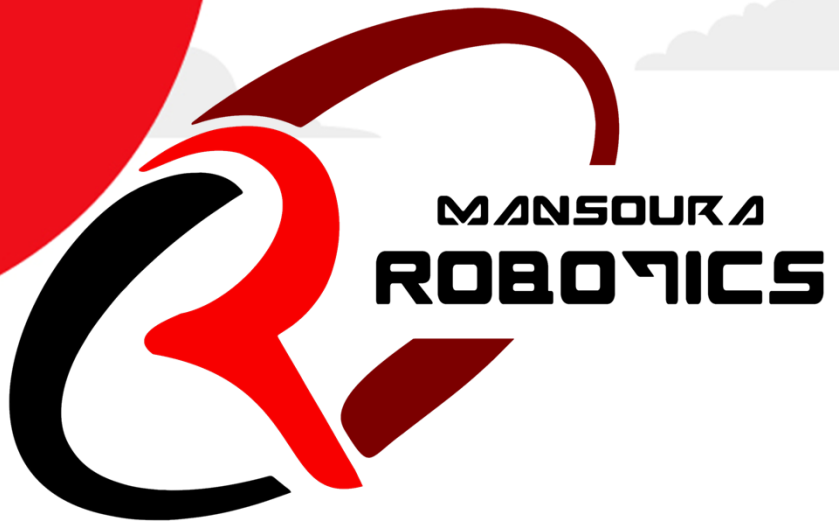
C Functions

Lecture 4



01

Introduction to Functions in C

02

Global Vs Local Variables

03

Divide C project into many Files

04

Recursion in C

01

Introduction to Functions in C

Functions Analogy

To understand the idea of developing the function, let's imagine the following analogy.

Someone is going to open a shop , first thing he will do is to make a banner announcing his shop. This banner is called a ***prototype***.

Then, he will develop his shop and prepare all services that will be provided. This is called the ***implementation***.

Last thing, customers will call the shop and ask for a service, he will answer them and provide them with the required service. This is called the ***call***.

MANSOURA ROBOTICS

What is a function

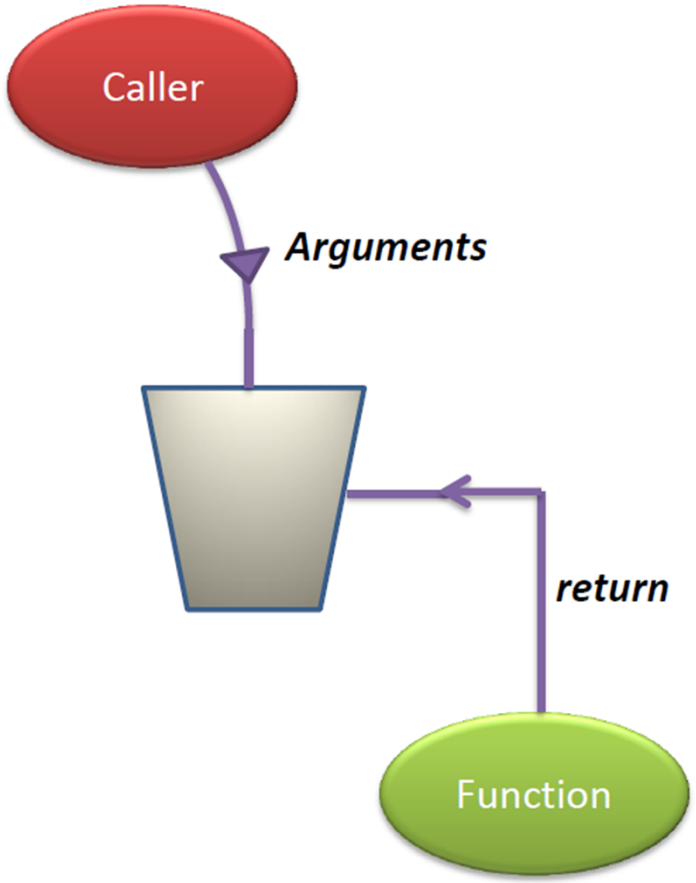
A ***function*** is a block of code that will be defined one time and can be executed many times.

To execute the function you will need to call it. The function provides you with the advantage that it is defined one time and can be executed many times so it takes the same size in the memory whatever how many times it would be called.

When you call a function, you can send to it some inputs and it could return back an output.

One example of the functions is the ***printf*** function that takes a string as an input argument and print it on the screen.

Any C project composed of one function or more, the most popular function is the ***main*** function which the first function to be called (The entry point)



MANSOURA ROBOTICS

Function Syntax

Function composed of two parts

1- The prototype

Used to declare the function (The banner)


```
return_type   Function_Name   (Input_Type Input_Name , ..... );
```

2- The function body (The implementation)

Used to define the function behavior

```
return_type   Function_Name   (Input_Type Input_Name , .....)
{
        Function statements
}
```

3- Function call

Used to execute the function

```
Output = Function_Name(Inputs)
```

MANSOURA ROBOTICS

Example

```
int Add (int x, int y);

void main(void)
{
    int var = Add(3,4);
    printf("Result is %d",var);
}


int Add (int x , int y)
{
    int z = x+y;

    return z;
}
```

1- ***Prototype*** of function named Add and takes 2 arguments one called x of type ***int*** and the other one called y of type ***int*** too. The function will return an ***int*** value

3- ***Function call***, the main function calls the Add function and send to it the two arguments 3 and 4, then it will receive the result in the variable var.

2- ***Function implementation***, it will add x and y and save the value in the variable z. Then it will return the value of z.

MANSOURA ROBOTICS

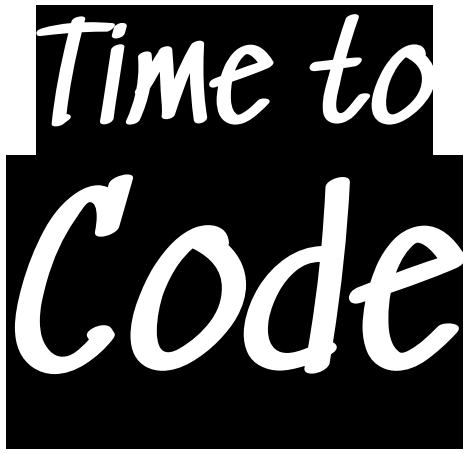
LAB1

Write C code that will ask the user to enter 2 numbers, then the main function will call a function named Get_Max that takes the 2 values entered by the user then return the maximum of them.

The main function then will print the returned value

Expected Output

```
Please Enter Value 1: 10
Please Enter Value 2: 20
The Maximum value is 20
```



MANSOURA ROBOTICS

Void Keyword

The ***void*** keyword is used to any function to give the meaning of No thing.

For example if we need to define a function that takes no arguments, we would write between its ( ) the keyword ***void***.

If we need to define a function that doesn't return any outputs, we will write instead of the return type the keyword ***void***.

The function can neither takes input arguments nor return outputs.

```
void print_My_Name(void)
{
    printf("Ahmed");
}
```

Example for a function that takes void and return void

MANSOURA ROBOTICS

Void Keyword

To Call a function that takes void, then write its and and empty ( )

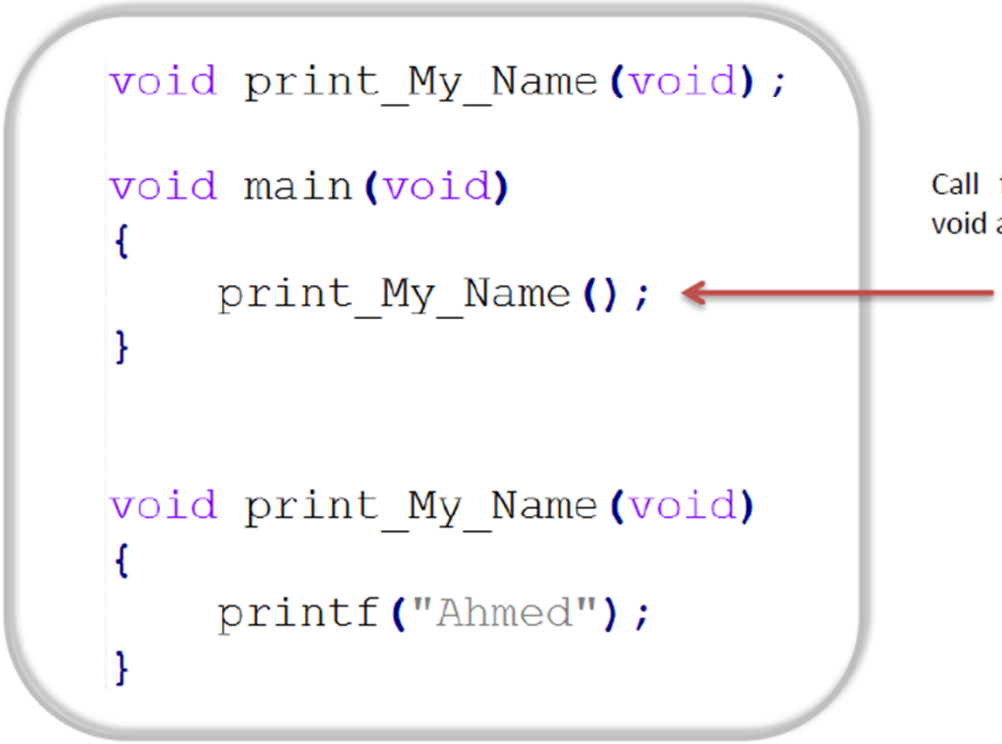
To Call a function that returns void, then don't receive its output in a variable

```
void print_My_Name(void);

void main(void)
{
    print_My_Name();
}


void print_My_Name(void)
{
    printf("Ahmed");
}
```

Call for a function that takes void and return void



Question 1

What will be the output of the following code … ?

```c
#include <stdio.h>

void func(void);

void main(void)
{
    int x = func();
}

void func (void)
{
    printf("I'm a void function ");
}
```

Question 1 Answer

What will be the output of the following code … ?

```c
#include <stdio.h>

void func(void);

void main(void)
{
    int x = func();
}

void func (void)
{
    printf("I'm a void function ");
}
```

Compilation error, trying to receive a value from a function that returns void

Question 2

What will be the output of the following code ... ?

```c
#include <stdio.h>

void func(void);

void main(void)
{
    func(10);
}

void func (void)
{
    printf("I'm a void function ");
}
```

Question 2 Answer

What will be the output of the following code ... ?

```c
#include <stdio.h>

void func(void);

void main(void)
{
    func(10);
}

void func (void)
{
    printf("I'm a void function ");
}
```

Compilation error, trying to send a value for function that takes void

02

Global Vs Local Variables

Global Vs Local variables

Local variable is the variable that it is defined inside any function, this variable can be accessed only on the function that defines it.

```
void func (void)
{
    int x = 10;
    /*
        Some Code
                    */
}
```

x is a local variable can be accessed only the function named func

Global variable is the variable that is defined outside any function, this variable can be accessed in any function in the code

x is a Global variable can be accessed in any function in the program

```
int x;

void main(void)
{
    /* Some code */
}

void func1 (void)
{
    /*
        Some Code
                    */
}
```

MANSOURA ROBOTICS

Lab2

Write C code implement a function to swap 2 global variables.

Expected Output

```
X before swap = 10
Y before swap = 20


X after swap = 20
Y after swap = 10
```

Time to Code

MANSOURA ROBOTICS

03

Divide C project into many files

Divide C project into many files

Sometimes it would be recommended to split your project into some C files for issue of modularity and organization. All the c files will compile together to generate one output file.

The general rule is, To use call function_x (that is defined in file_1) in file_2, file_2 must know the prototype of function_x

file_1.c

```
void func_x(void);

void main(void)
{
    func_x();
}
```

file_2.c

```
void func_x(void)
{
    /*
        Some Code
                    */
}
```

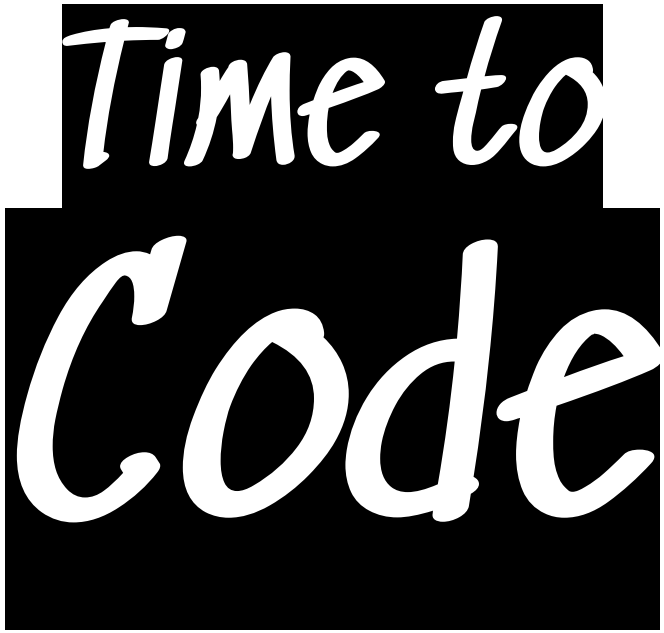
To compile the two files on gcc together, just write them in the command as follows:

```
gcc file1.c file2.c -o out.exe
```

MANSOURA ROBOTICS

Lab3

Write C code that contains two functions; PrintMyName and main. PrintMyName is defined in file2.c and its role is to print your name. main is define in file1.c and its role is to call PrintMyName function



MANSOURA ROBOTICS

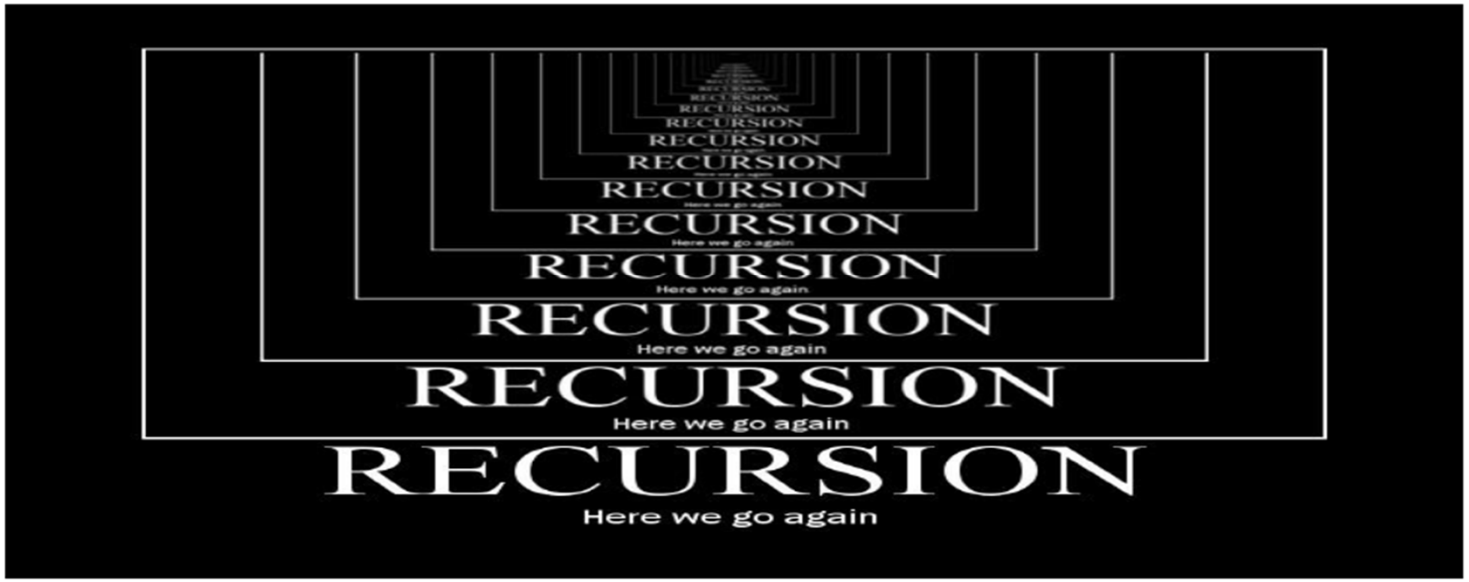
04

Recursion in C

Recursion in C

A function that calls itself is known as a recursive function. And, this technique is known as recursion.

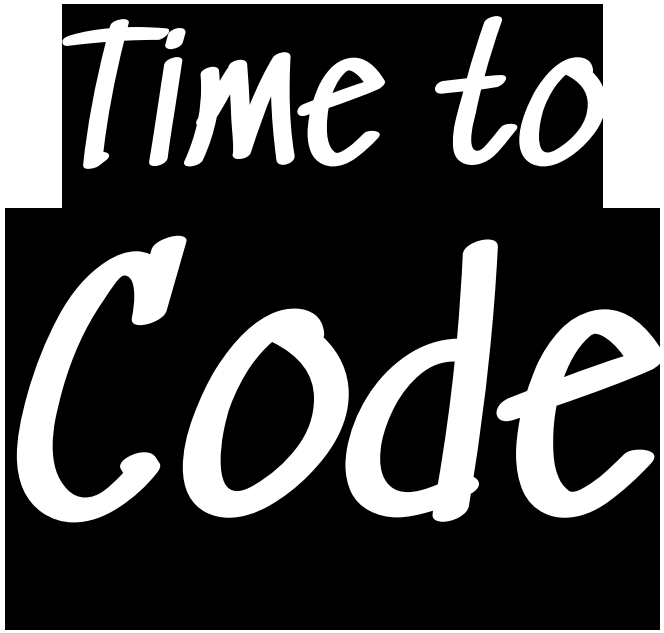
Recursion in not recommended in Embedded Systems development because it use more memory and are generally slow.



MANSOURA ROBOTICS

Lab4

Write C code that calculates the factorial of a number entered by the user using recursive function



MANSOURA ROBOTICS

Thank you!

Do you have any questions?

Assignment 1

Write a C program the implement 2 function:

1- Function to get the maximum of 4 values
2- Function to get the minimum of 4 values

The program will ask the user first to enter the 4 values, then print the maximum number and minimum number of them.

MANSOURA ROBOTICS

Assignment 2

Write a C program to act as simple calculator, first it will ask the user to enter the operation ID, depending on the operation, the program will ask the user either to enter 1 operand or 2 operands and the program will execute the operation and print the result. Each operation should be implemented in a stand a long function.

ID	Operation	Number of Operands
1	Add	2
2	Subtract	2
3	Multiply	2
4	Divide	2
5	And	2
6	Or	2
7	Not	1
8	Xor	2
9	reminder	2
10	Increment	1
11	Decrement	1

MANSOURA ROBOTICS