Google Developer Student Clubs

Functions in C

Agenda

- Function definition.
- Function Types.
- Function Syntax.
- Function Recursion.
- Local and Global variables.
- Block scope.
- Divide project to many C files

What is function ?

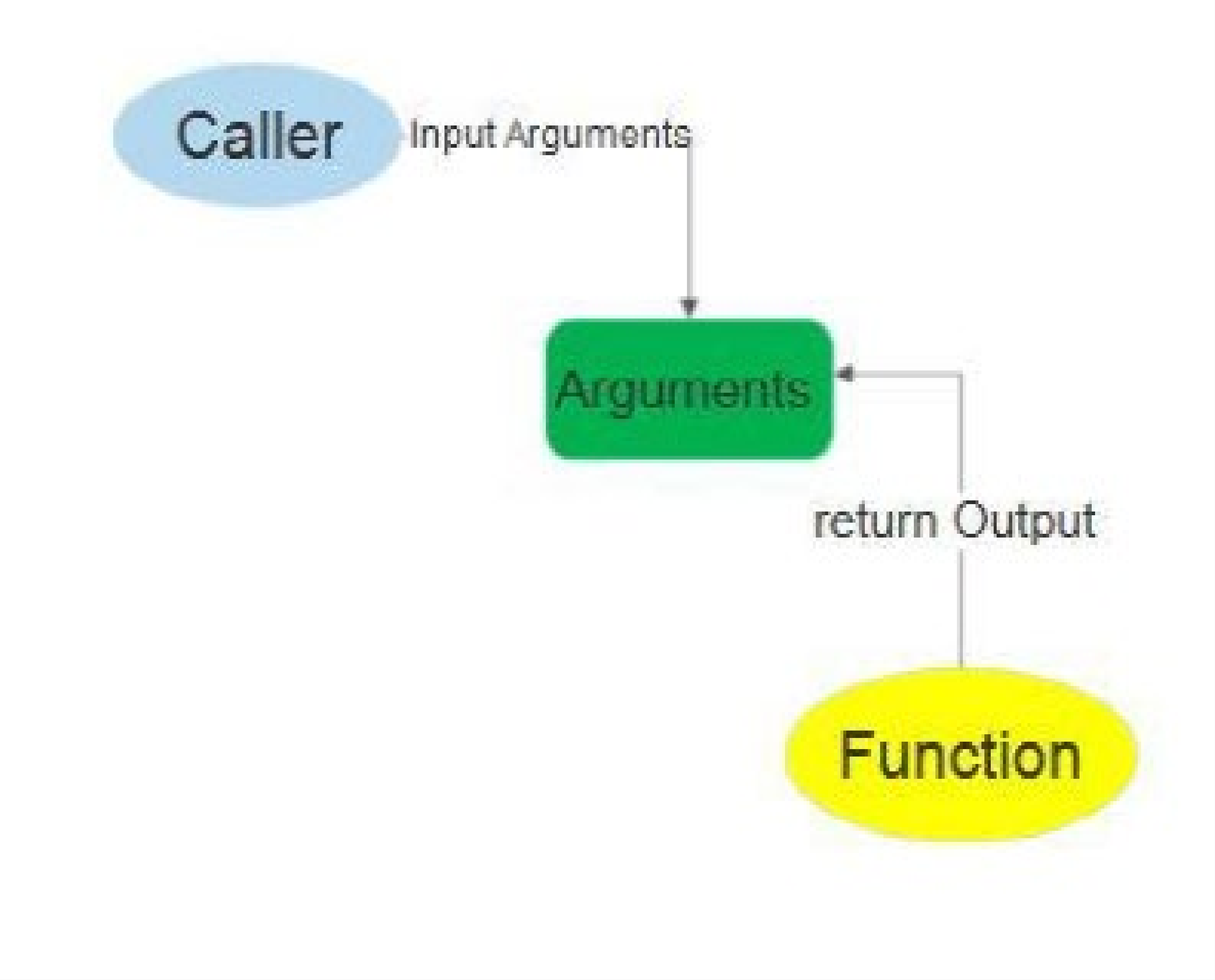
- A function is a block of code that will be defined one time and can be executed many times.
- To execute the function you will need to call it.

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.

Google Developer Student Clubs

What is function ?

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



Function Types

There are two types of functions in C programming:

1- User defined functions :

function is defined and given its name by the user to perform a specific task .

2- Standard Library function :

a built in functions that has a certain name and take certain arguments previously defined in the standard library.

Function Syntax

1- The prototype : Used to declare the function .

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

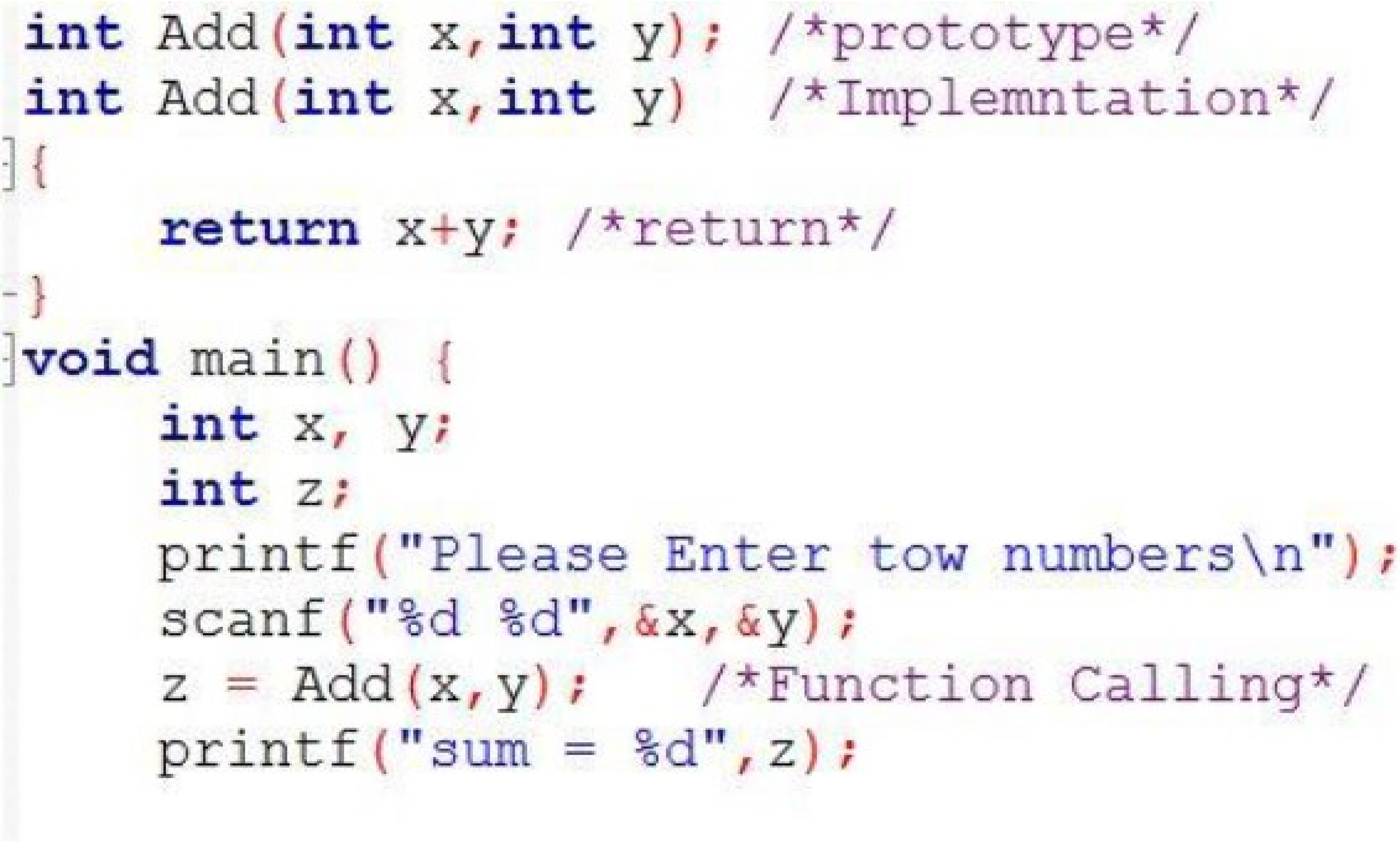
2- The function body (The implementation) : Used to define the function behavior

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

3- Function call Used to execute the function.

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

Example



```
int Add(int x,int y); /*prototype*/
int Add(int x,int y)  /*Implemntation*/
{
    return x+y; /*return*/
}
void main() {
    int x, y;
    int z;
    printf("Please Enter tow numbers\n");
    scanf("%d %d",&x,&y);
    z = Add(x,y);   /*Function Calling*/
    printf("sum = %d",z);
```

Exercise

Write a C function to calculate the factors of a number .

Ex : factors of 25 : 1,5,25

Write a C function to calculate the power of a number .

Void keyword

The void keyword is used to any function to give the meaning of Nothing.
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
.Example :

```
 void  printWelcome(void)
{
 printf(“wlcome Ahmed”);
}
```

Notes

Subtitle

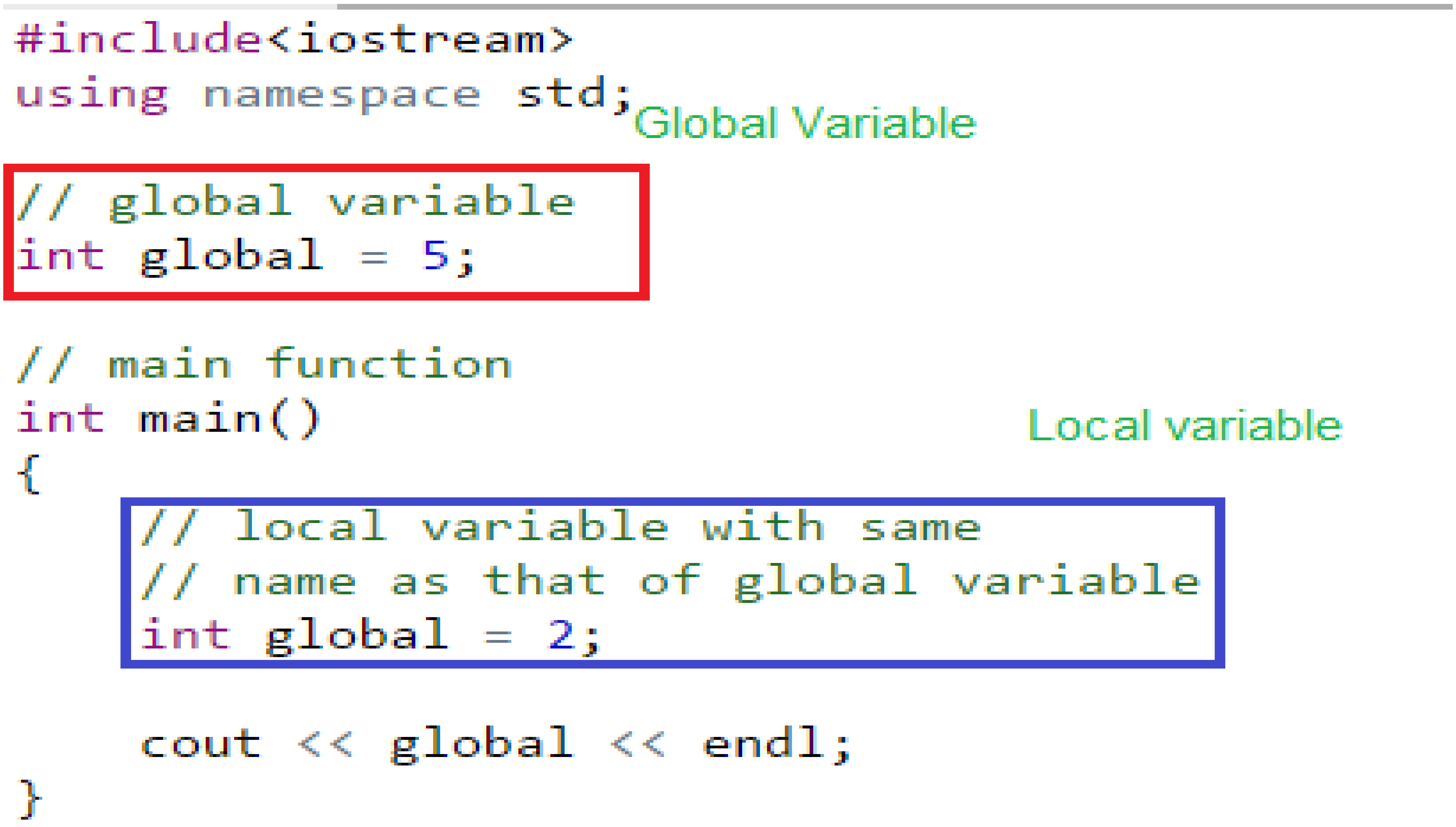
1. Trying to receive an output from a function that returns void will give Compilation error.

2. Trying to send an input from a function that takes void will give Compilation error.

3. Trying to send fewer or more arguments than Declared in a function prototype/ implementation gives Compilation error.

Local Vs Global variable

Local variable is a variable that will be identified or seen only within a function scope or block scope.
Local variables are saved in stack section in RAM,unless it has the modifier static. it is saved in data segment.
Global variable is a variable outside of any function , that is why it is seen from all the function in the same file , it has a file scope or project scope.
Global variables are put in data segment sectione in RAM .
Uninitialized global variables are initialized to zero by default.

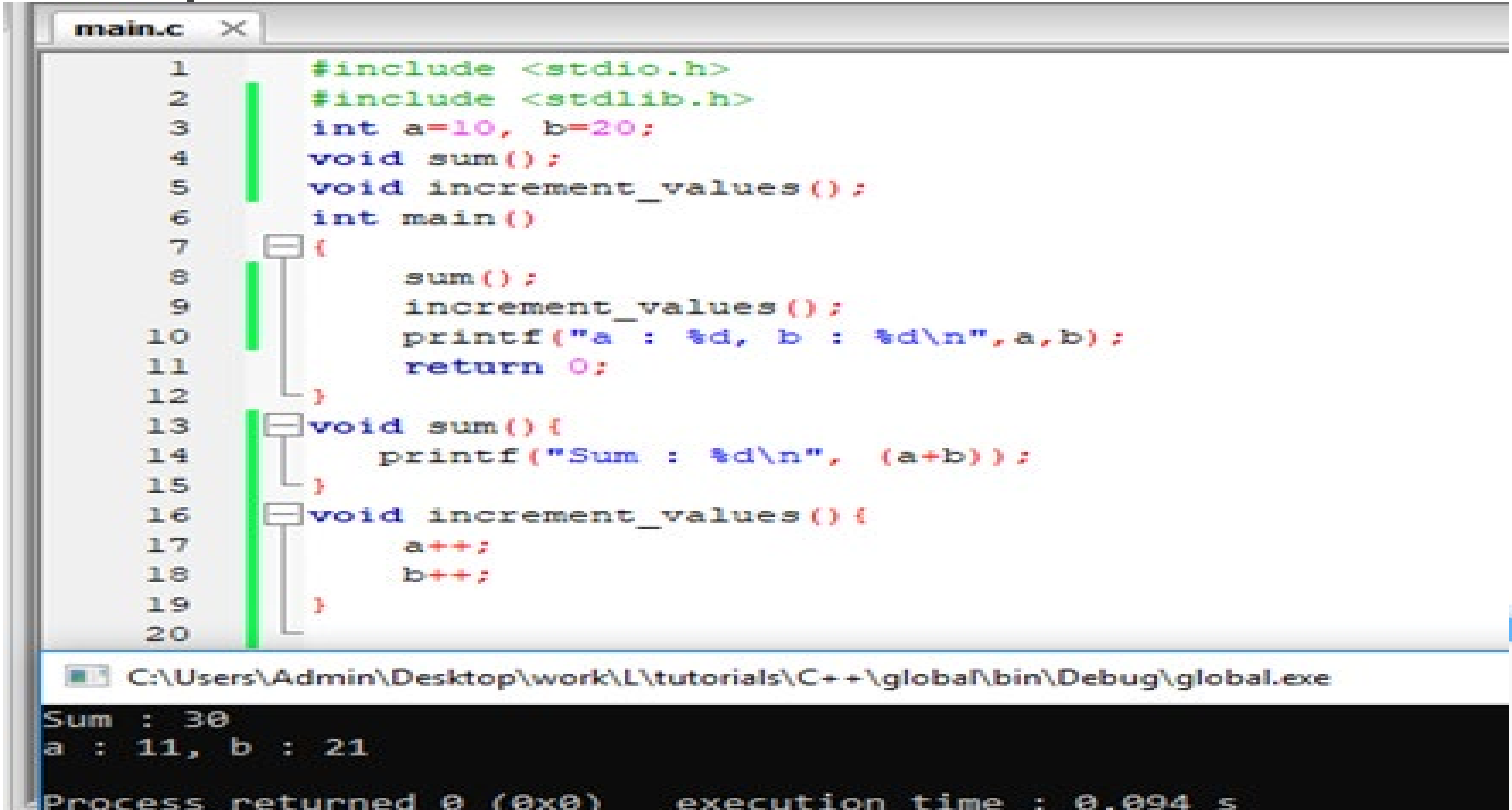


```
#include<iostream>
using namespace std;

// global variable
int global = 5;

// main function
int main()
{
    // local variable with same
    // name as that of global variable
    int global = 2;

    cout << global << endl;
}
```

Global Variable

Local variable

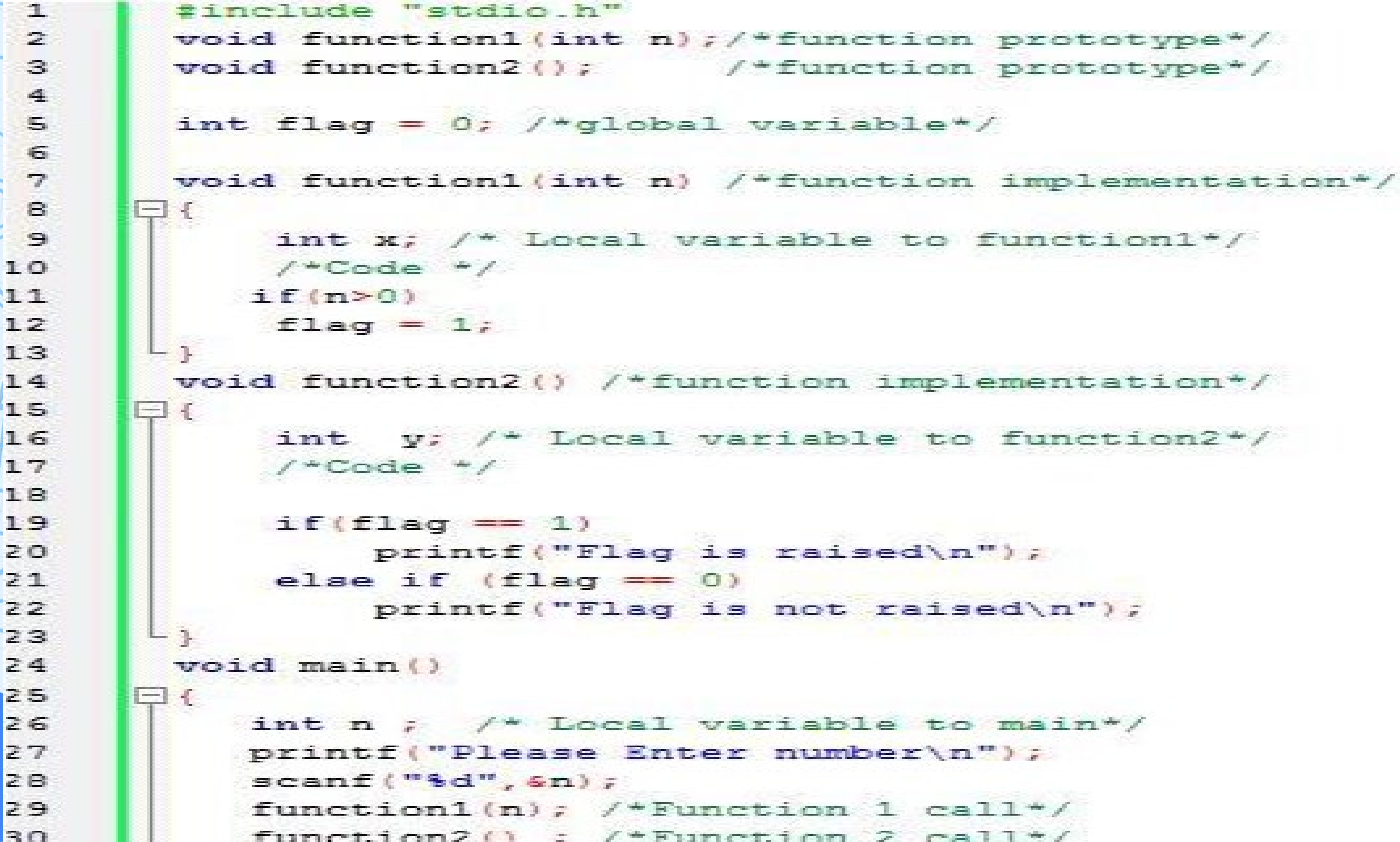
Example



```c
#include <stdio.h>
#include <stdlib.h>
int a=10, b=20;
void sum();
void increment_values();
int main()
{
    sum();
    increment_values();
    printf("a : %d, b : %d\n",a,b);
    return 0;
}
void sum(){
   printf("Sum : %d\n", (a+b));
}
void increment_values(){
    a++;
    b++;
}
```

C:\Users\Admin\Desktop\work\L\tutorials\C++\global\bin\Debug\global.exe

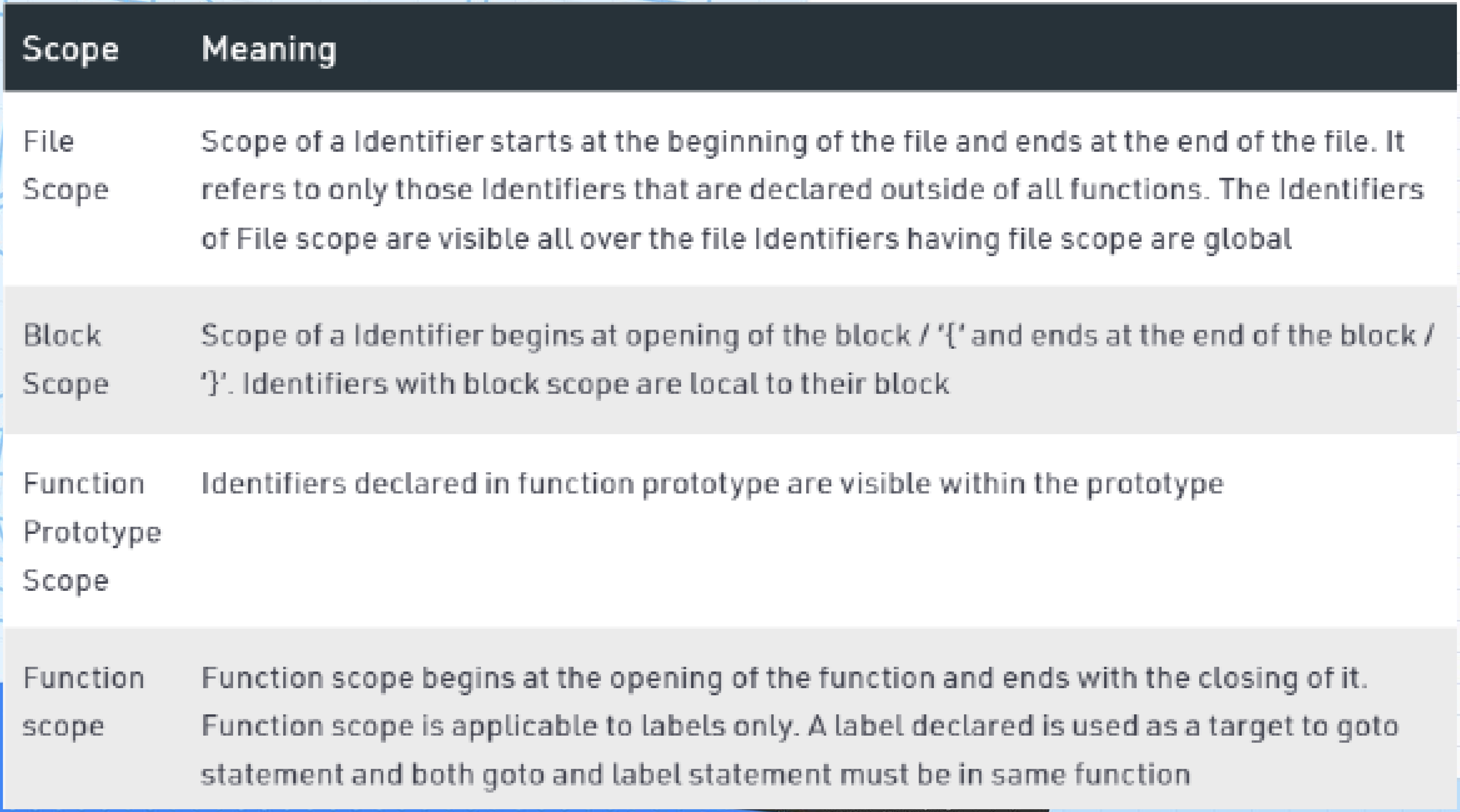
```
Sum : 30
a : 11, b : 21

Process returned 0 (0x0)   execution time : 0.094 s
```



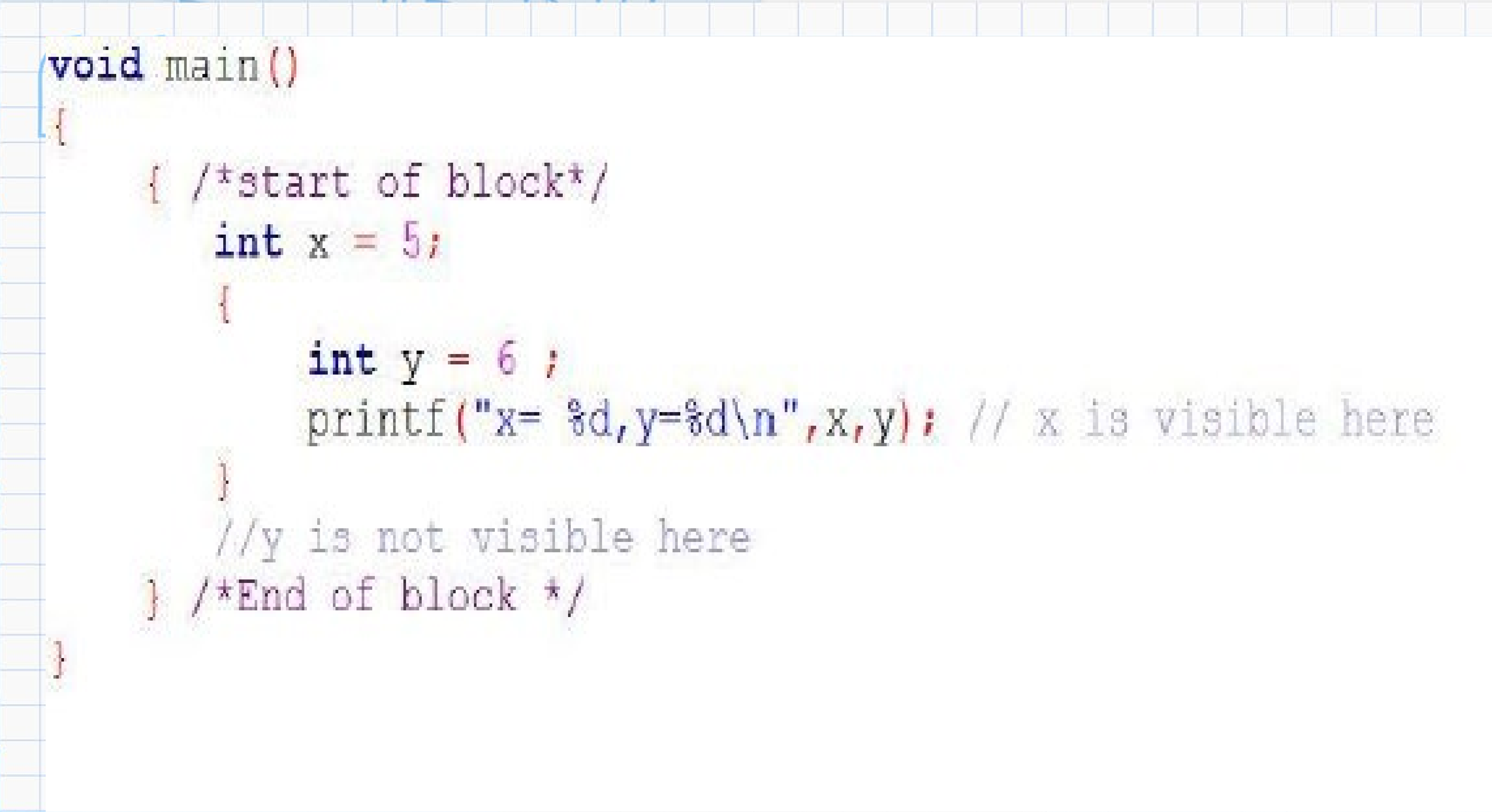
```c
#include "stdio.h"
void function1(int n);/*function prototype*/
void function2();     /*function prototype*/

int flag = 0; /*global variable*/

void function1(int n) /*function implementation*/
{
    int x; /* Local variable to function1*/
    /*Code */
   if(n>0)
    flag = 1;
}
void function2() /*function implementation*/
{
    int  y; /* Local variable to function2*/
    /*Code */

    if(flag == 1)
        printf("Flag is raised\n");
    else if (flag == 0)
        printf("Flag is not raised\n");
}
void main()
{
   int n ;  /* Local variable to main*/
   printf("Please Enter number\n");
   scanf("%d",&n);
   function1(n); /*Function 1 call*/
   function2() ; /*Function 2 call*/
```

Block scope

- Blocks combine multiple statements into a single unit .
- Variables declared inside the block have a local scope.
- Blocks can be nested.



Scope	Meaning
File Scope	Scope of a Identifier starts at the beginning of the file and ends at the end of the file. It refers to only those Identifiers that are declared outside of all functions. The Identifiers of File scope are visible all over the file Identifiers having file scope are global
Block Scope	Scope of a Identifier begins at opening of the block / ‘{’ and ends at the end of the block / ‘}’. Identifiers with block scope are local to their block
Function Prototype Scope	Identifiers declared in function prototype are visible within the prototype
Function scope	Function scope begins at the opening of the function and ends with the closing of it. Function scope is applicable to labels only. A label declared is used as a target to goto statement and both goto and label statement must be in same function



```
void main()
{
    { /*start of block*/
      int x = 5;
      {
          int y = 6 ;
          printf("x= %d,y=%d\n",x,y); // x is visible here
      }
      //y is not visible here
    } /*End of block */
}
```

Divide project to many C files

- We divide a large C project to modules based on functionality .
- Each module is composed of a .c file and a .h file.
- A .c file contains the the following :

1- Implementation of functions .

2- Global variable definition.

- A .h file contains the following:

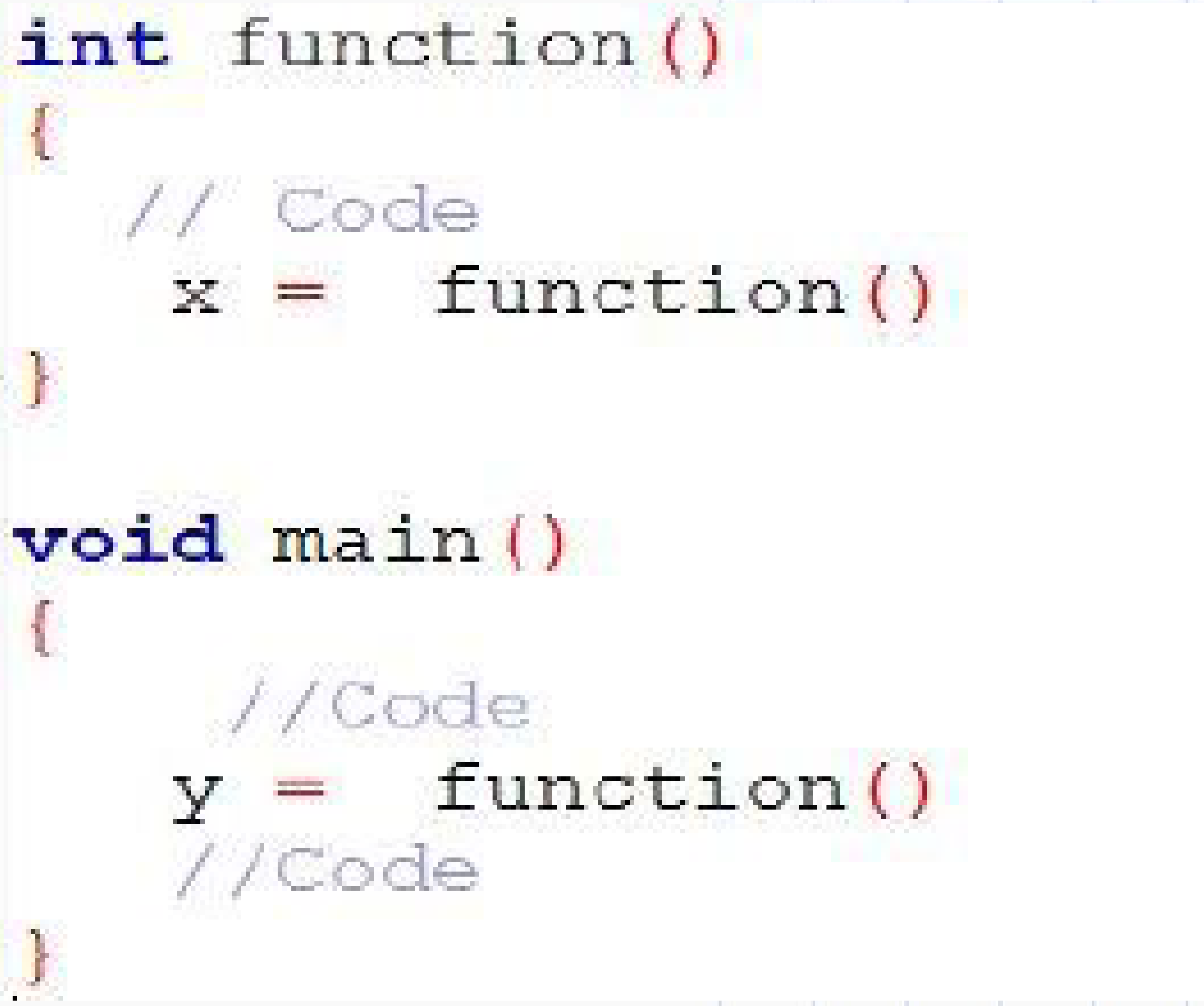
1- constants (#define) .

2- functions prototype.

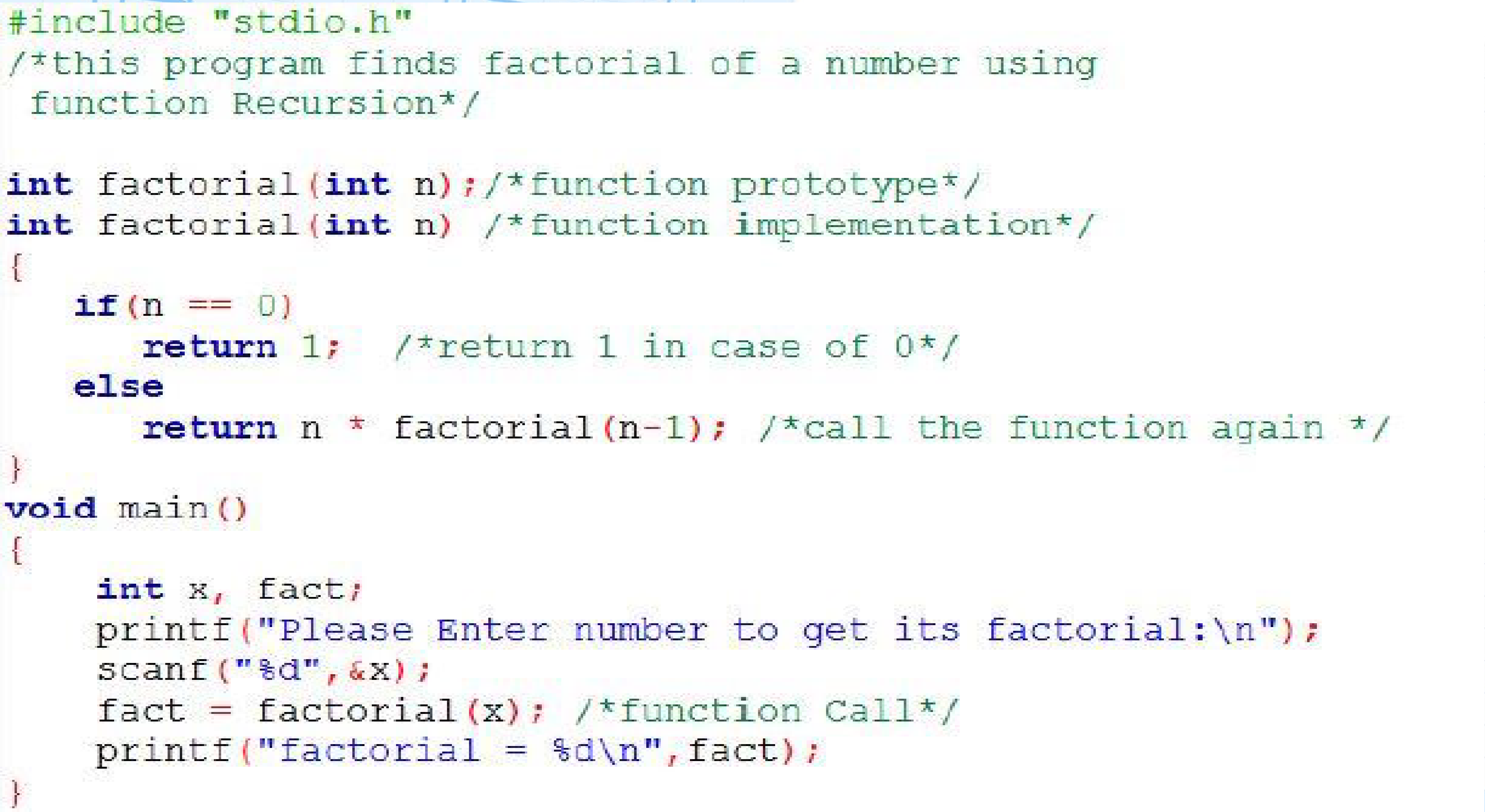
Recursion

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 uses more memory and is generally slow.



```
int function()
{
  // Code
   x =  function()
}

void main()
{
    //Code
   y =  function()
   //Code
}
```



```c
#include "stdio.h"
/*this program finds factorial of a number using
 function Recursion*/

int factorial(int n);/*function prototype*/
int factorial(int n) /*function implementation*/
{
   if(n == 0)
      return 1;  /*return 1 in case of 0*/
   else
      return n * factorial(n-1); /*call the function again */
}
void main()
{
    int x, fact;
    printf("Please Enter number to get its factorial:\n");
    scanf("%d",&x);
    fact = factorial(x); /*function Call*/
    printf("factorial = %d\n",fact);
}
```

Assignment

Write a C function to print fibonacci series given the number of terms.

Design a program composed of the following:
Math Library filles:MathLib.c and MathLib.h
Math Library functions: add,sub,multi,and dev
And make a programe by using this library.

"Thank You!"

```
function filterStudies({ studies, filterByOrg = false, filte
                 studies.filter(study => {
                                      organizat
```