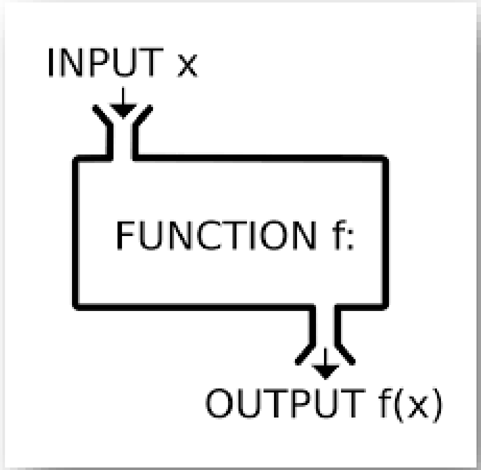
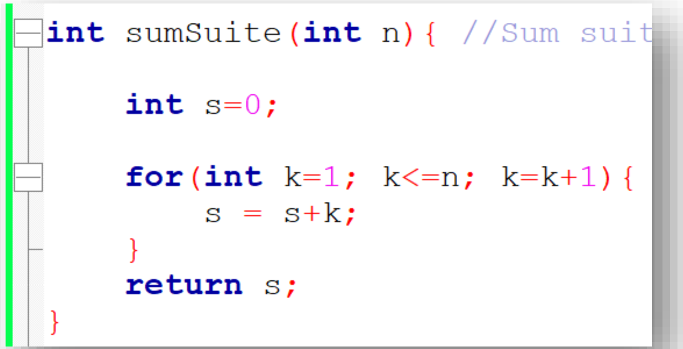
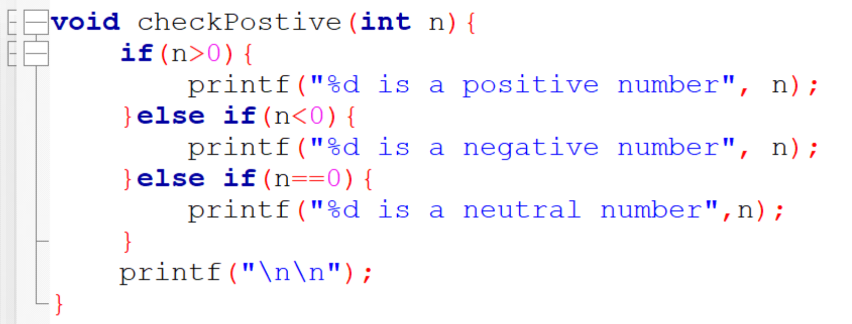
DATA STRUCTURE AND PROGRAMMING

Sub-Program (Function)





```
int sumSuite(int n){  //Sum suit

    int s=0;

    for(int k=1; k<=n; k=k+1){
        s = s+k;
    }
    return s;
}
```



```
void checkPostive(int n){
    if(n>0){
        printf("%d is a positive number", n);
    }else if(n<0){
        printf("%d is a negative number", n);
    }else if(n==0){
        printf("%d is a neutral number",n);
    }
    printf("\n\n");
}
```

Lecture overview

❑ Overall lectures

1. Introduction to algorithm
2. Basic data types and statements
3. Control structures and Loop
4. Array
5. *Sub-program*
6. Data structure

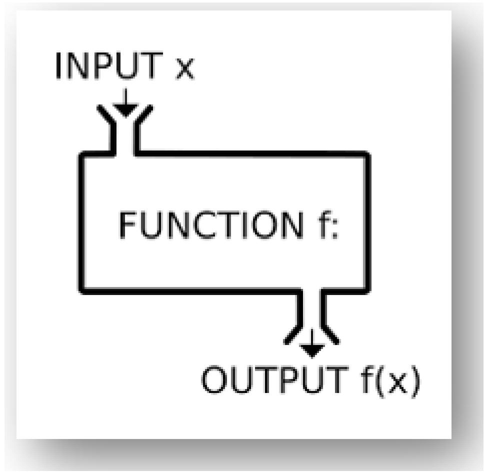
Objective

- Introduction to subprogram / function
- Advantages of using subprogram
- How to create your own function
- How to use your own function

Sub-program

❑ Introduction

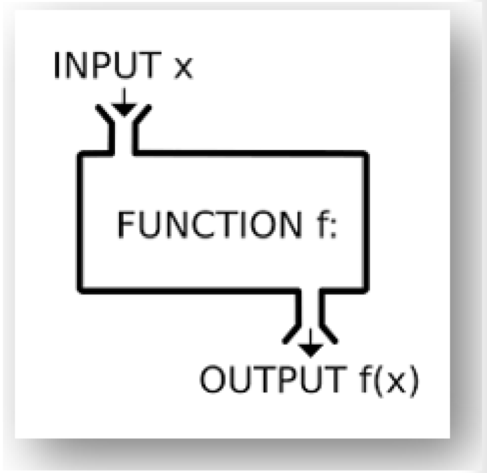
- **Sub-program** is a part of feature/functionality in a program.
- It is a block of codes to perform particular task.
- It is basically a set of statements that takes **inputs**, perform some computation, and produces output.

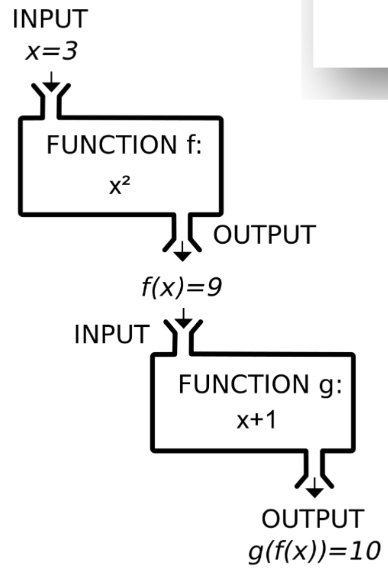


Sub-program

❑ Introduction

- When we start writing a larger program, it becomes
  - Difficult to have a global vision on its functionalities
  - Difficult to determine the errors
- **Solution**: Divide the problem to sub-problem
  - Solve each sub-problem
  - Put those sub-problems in sub-programs





Advantages

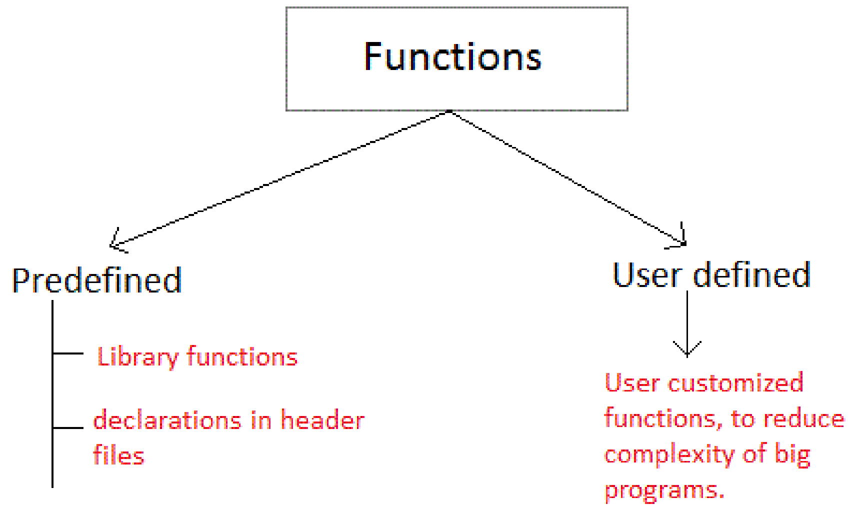
❑ Advantages of using sub-program

- Clear and readable code
- Reusability
  - A sub-program can be used many time with the same instruction without rewriting it by just calling its name once defined.
- Easy to test and find error
- Helpful for team work
  - E.g: Each team member is assigned to work on a specific feature which can be implemented as a sub-program

Predefined subprogram

❑ Predefined sub-program (existing/built-in sub-program)

- There are predefined sub-programs
  - E.g: `strcmp(),` `strlen(),` `strrev(),` `strcat()`
  - …… `etc.`


Functions
Predefined
User defined
Library functions
declarations in header files
User customized functions, to reduce complexity of big programs.

- Those predefined sub-program **are not enough**
  - so we need to define our own sub-program to solve our problem based on our needs

Sub-program

❑ Type of sub-program

- **Sub-program** is a small program that can be executed in the other program
- There are two types of sub-programs:
  1. Function : used to calculate something and **has return value**
  2. Procedure : a set of commands executed in order and it **has no return value**

Sub-program

❑ Function: DEFINTION

- A function is a set of instructions grouped under a name, that is called a sub-program and will return a value.
- A function is a sub-program that consists of:

  - ✓ Has a name
  - ✓ Have none, one or many parameters (arguments/inputs to functions)
  - ✓ **Return a value** in a certain type (void, int, float, ...)
  - ✓ Can have variables inside functions (*local variable*)
  - ✓ Composed of instructions/codes (body of function)

Sub-program

Function: SYNTAX

1. Name of function

3. Type of value returning from the function

```
function functionName(parameters: types) : type
    local variables
    block of instruction
    return value
end function
```

Body of subprogram

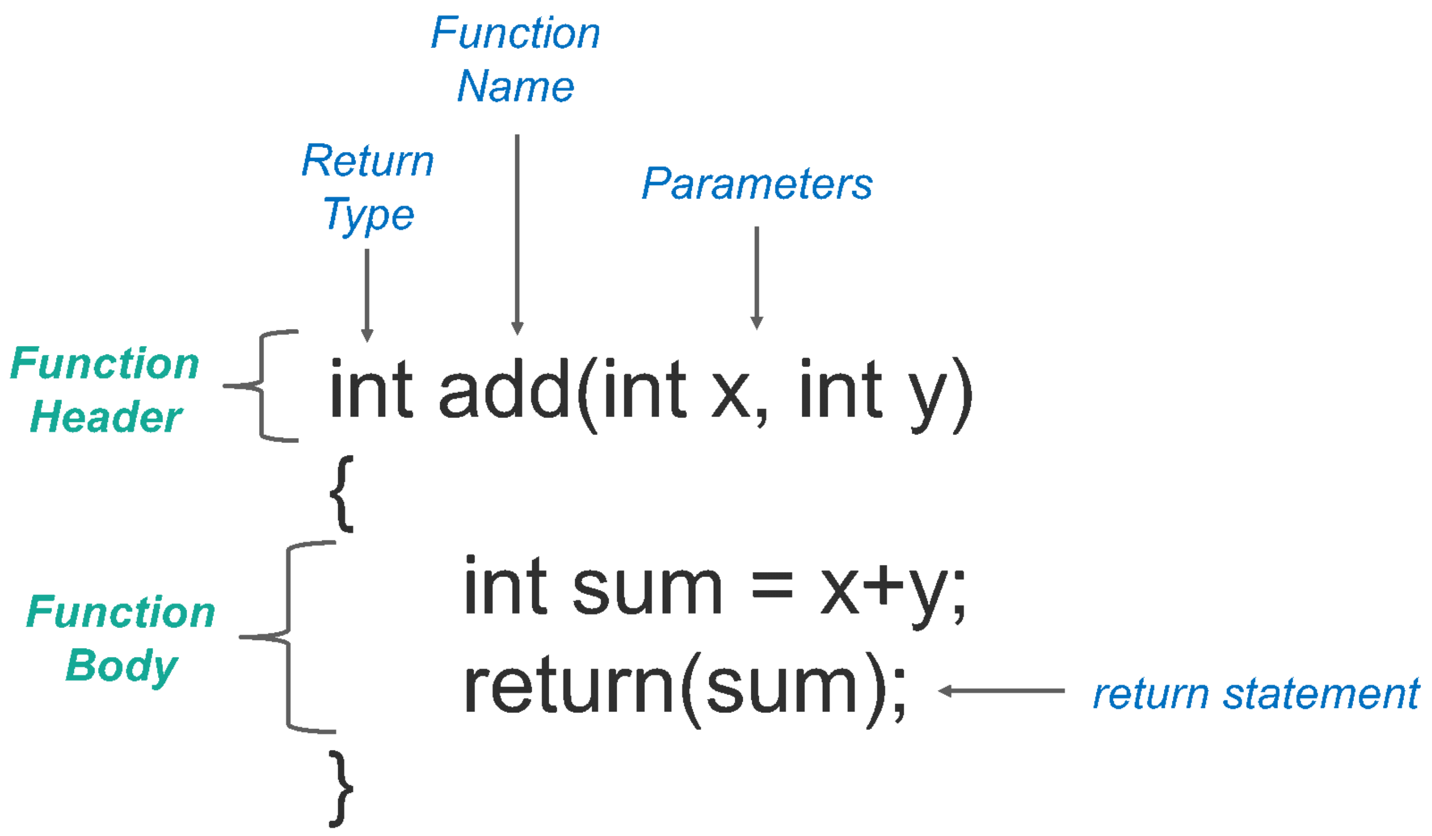
Must have the same type

2. Parameters of function
Each parameter has a **name** and its **type.**
A parameter is a variable passed to use in a function

- Parameters: declaration parameters of function
    - Order of parameter is important
    - A function can have no parameter, one or many parameters (if more than one, separate them by comma)
- Type of return value must be the same with type of function
- ***return*** is an instruction for sending a value from function to where this function is called.

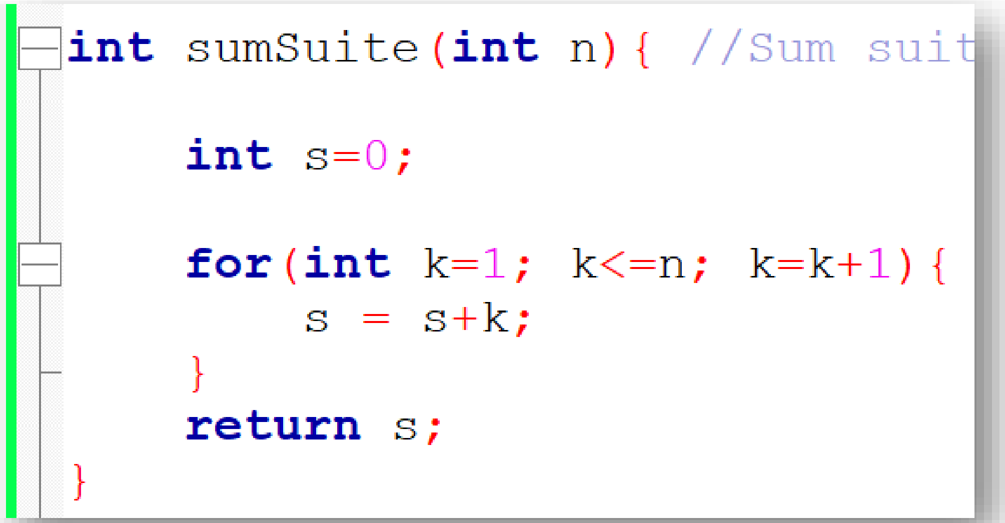
Function

❑ Syntax in C programming



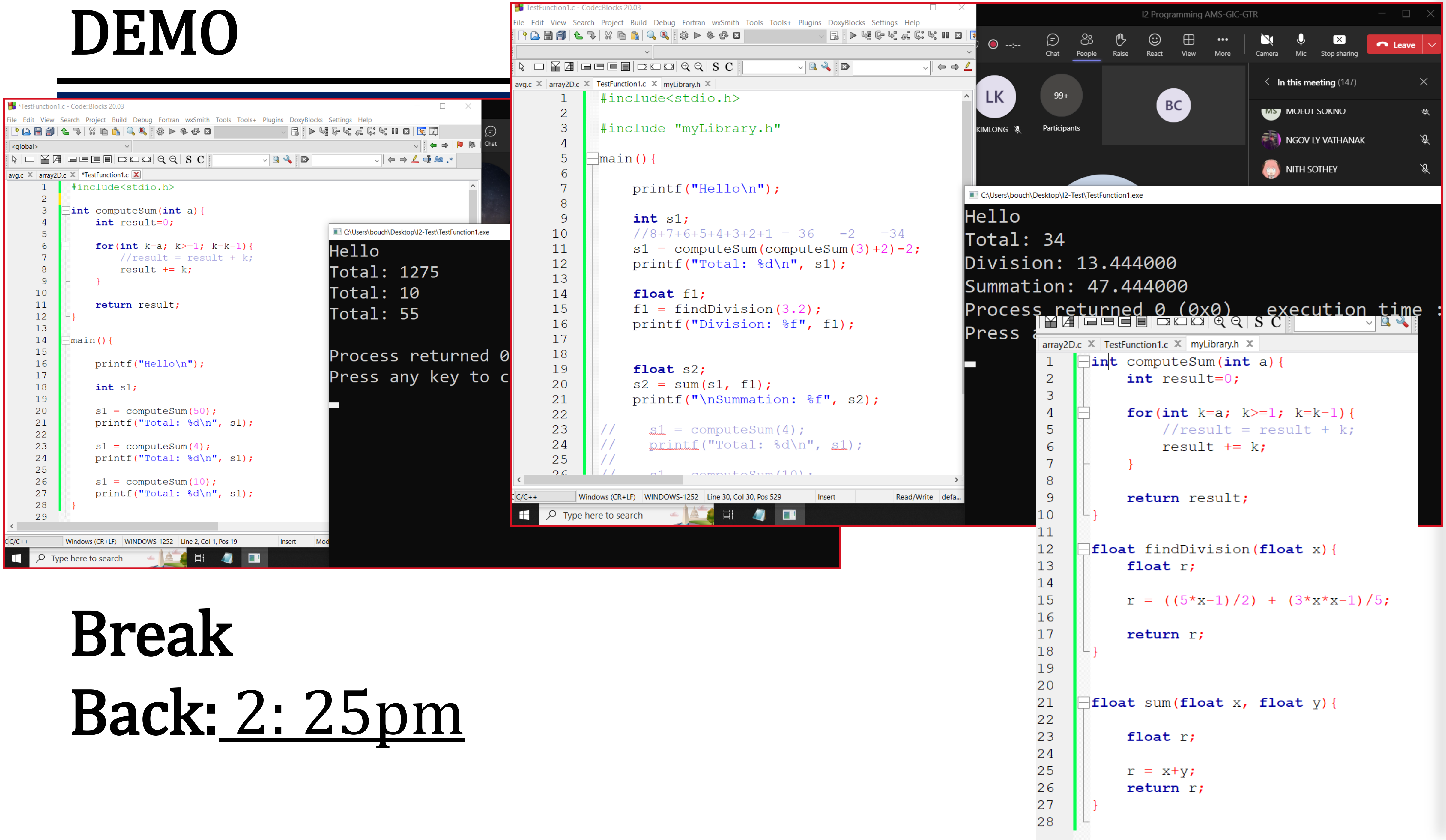
```
int add(int x, int y)
{
      int sum = x+y;
      return(sum);
}
```

Function with return type



```
int sumSuite(int n){ //Sum suit

    int s=0;

    for(int k=1; k<=n; k=k+1){
        s = s+k;
    }
    return s;
}
```

DEMO



```
#include<stdio.h>

int computeSum(int a){
    int result=0;

    for(int k=a; k>=1; k=k-1){
        //result = result + k;
        result += k;
    }

    return result;
}

main(){

    printf("Hello\n");

    int s1;

    s1 = computeSum(50);
    printf("Total: %d\n", s1);

    s1 = computeSum(4);
    printf("Total: %d\n", s1);

    s1 = computeSum(10);
    printf("Total: %d\n", s1);
}
```

```
Hello
Total: 1275
Total: 10
Total: 55

Process returned 0
Press any key to c
```

```
#include<stdio.h>

#include "myLibrary.h"

main(){

    printf("Hello\n");

    int s1;
    //8+7+6+5+4+3+2+1 = 36   -2   =34
    s1 = computeSum(computeSum(3)+2)-2;
    printf("Total: %d\n", s1);

    float f1;
    f1 = findDivision(3.2);
    printf("Division: %f", f1);


    float s2;
    s2 = sum(s1, f1);
    printf("\nSummation: %f", s2);

//    s1 = computeSum(4);
//    printf("Total: %d\n", s1);
//
```

```
Hello
Total: 34
Division: 13.444000
Summation: 47.444000
Process returned 0 (0x0)   execution time :
Press
```

```
int computeSum(int a){
    int result=0;

    for(int k=a; k>=1; k=k-1){
        //result = result + k;
        result += k;
    }

    return result;
}

float findDivision(float x){
    float r;

    r = ((5*x-1)/2) + (3*x*x-1)/5;

    return r;
}


float sum(float x, float y){

    float r;

    r = x+y;
    return r;
}
```

Break

Back: 2: 25pm

Sub-program

❑ Procedure: SYNTAX

```
Procedure procedureName(parameters: types)
    … local variables …
    … block of instruction …
end procedure
```

Example 1 –not using subprogram

- ❑ A program to get max btw two numbers **without using subprogram**

```
Var a1, b1, a2, b2, c1, c2 : Integer
Var maxa, maxb, maxc : Integer
Begin
    read(a1, b1)
    read(a2, b2)
    read(c1, c2)

    if (a1>a2) then
        maxa ← a1
    else
        maxa ← a2
    end if
    if (b1>b2) then
        maxb ← b1
    else
        maxb ← b2
    end if
    if (c1>c2) then
        maxc ← c1
    else
        maxc ← c2
    end if
    Write(maxa, maxb, maxc)
End
```

Main program without sub-program

Example 1 –using sub program

Name of sub-program

Sub-program

```
function max(x: integer, y: integer): integer
    var res : integer
    if (x>y) then
        res ← x
    else
        res ← y
    end if
    return res
end function
```

```
Var a1, b1, a2, b2, c1, c2 : integer
Var maxa, maxb, maxc : Integer
Begin
    read(a1,b1)
    read(a2,b2)
    read(c1,c2)
    maxa ← max(a1, a2)
    maxb ← max(b1, b2)
    maxc ← max(c1, c2)
    write(maxa, maxb, maxc)
End
```

Main program

Sub-program

❑ Example 2

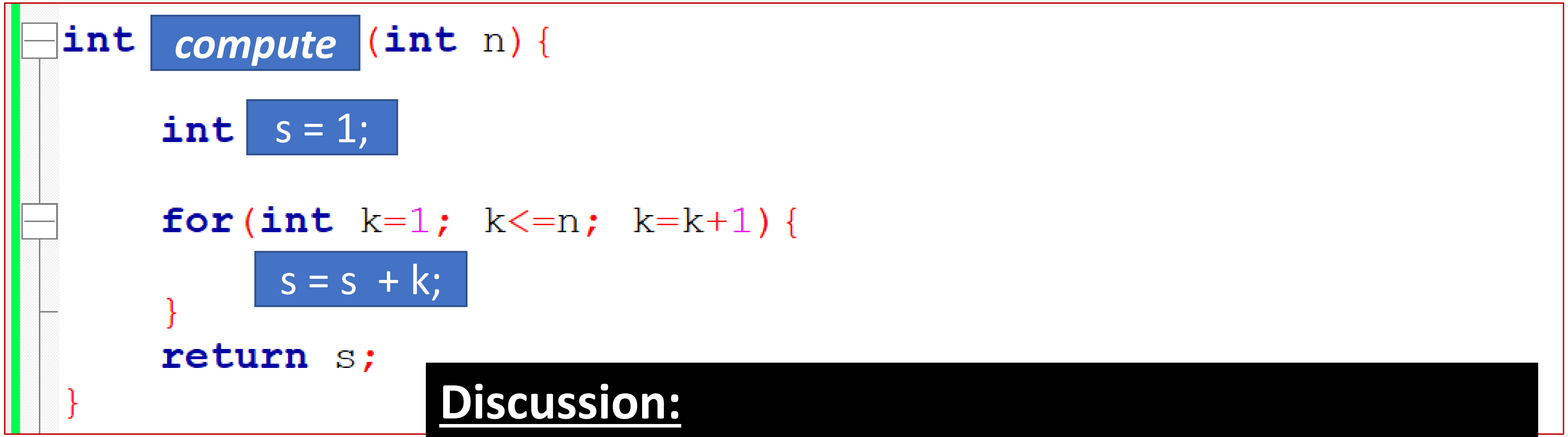
Create a function to calculate addition of two integer

```
function add(a:integer, b:integer) : integer
    Var r: integer
    r ← (a + b)*2
    return r
end function
```

Use the created function in a main program

```
var x, y, z: integer
begin
    z ← add(3,5)
    write(z)   16

    read(x,y)           2,3
    z ←  add(x,y)      ?? 10
    z ←  z - add(x,y) ?? 0
    z ←  add(1, add(2,3)) ?? 22
    z ←  add(1, z-1)) ??? 44
end
```

Example on how to create a function in C programming

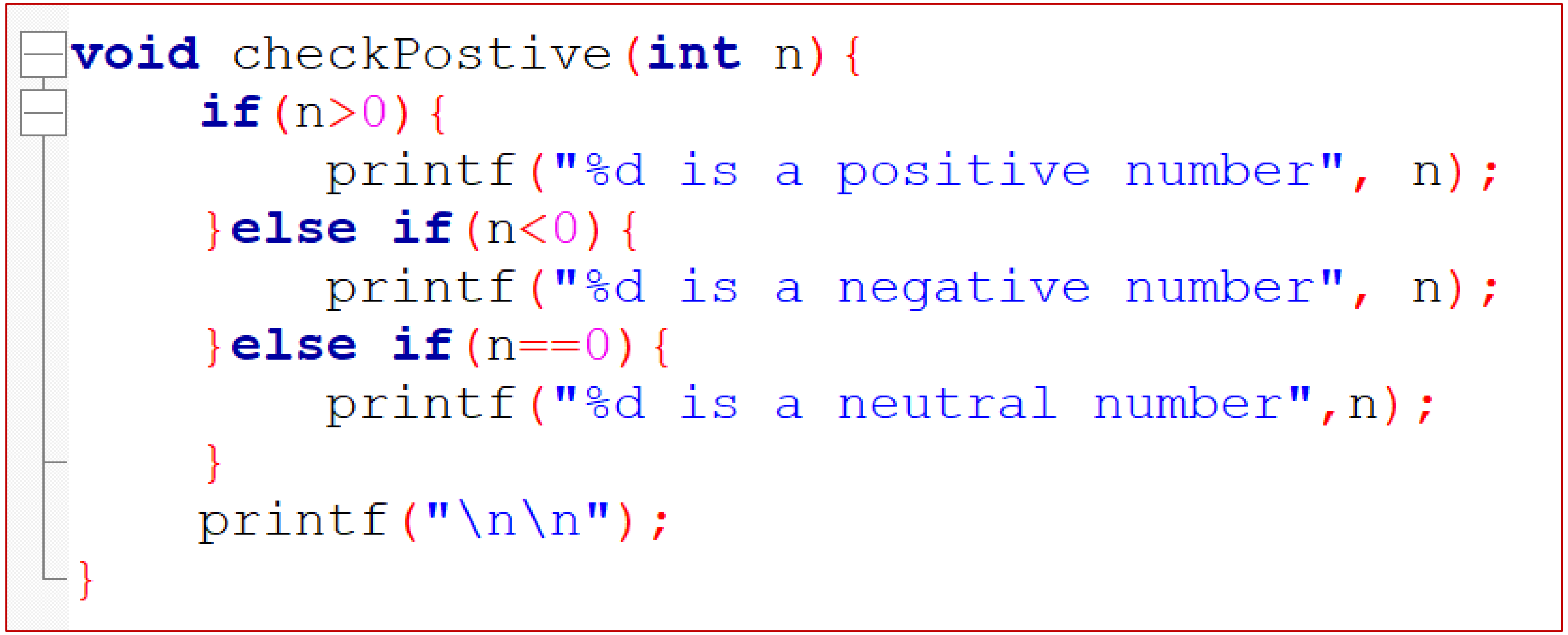


```
int compute (int n){

    int s = 1;

    for(int k=1; k<=n; k=k+1){
        s = s  + k;
    }
    return s;
}
```

Discussion:

a. What does this function do?
b. What is the name of this function?
c. How many parameters does it have? What is the parameter type?
d. Does this function return any value?
   - If yes, what type does it return?

Example in C programming



```c
void checkPostive(int n){
    if(n>0){
        printf("%d is a positive number", n);
    }else if(n<0){
        printf("%d is a negative number", n);
    }else if(n==0){
        printf("%d is a neutral number",n);
    }
    printf("\n\n");
}
```

a. What does this function do?
b. What is the name of this function?
c. How many parameters does it have? What is the parameter type?
d. Does this function return any value? If yes, what type does it return? How to know?

Predefined subprogram

❑ Predefined sub-program (existing/built-in sub-program)

- There are predefined sub-programs
  - E.g: `strcmp(),    strlen(), ......`
- Those predefined sub-program **are not enough**
  - so we need to define our own sub-program to solve our problem based on our needs

C Programming

Function syntax

C program

Defining a Function

The general form of a function definition in C programming language is as follows

```
return_type function_name( parameter list ) {
   body of the function
}
```

Components of function:

- ✓ Has a name
- ✓ Have none, one or many parameters (arguments)
- ✓ **Return a value** in a certain type
- ✓ Can have variables inside functions

(that variables are called *local variable*)

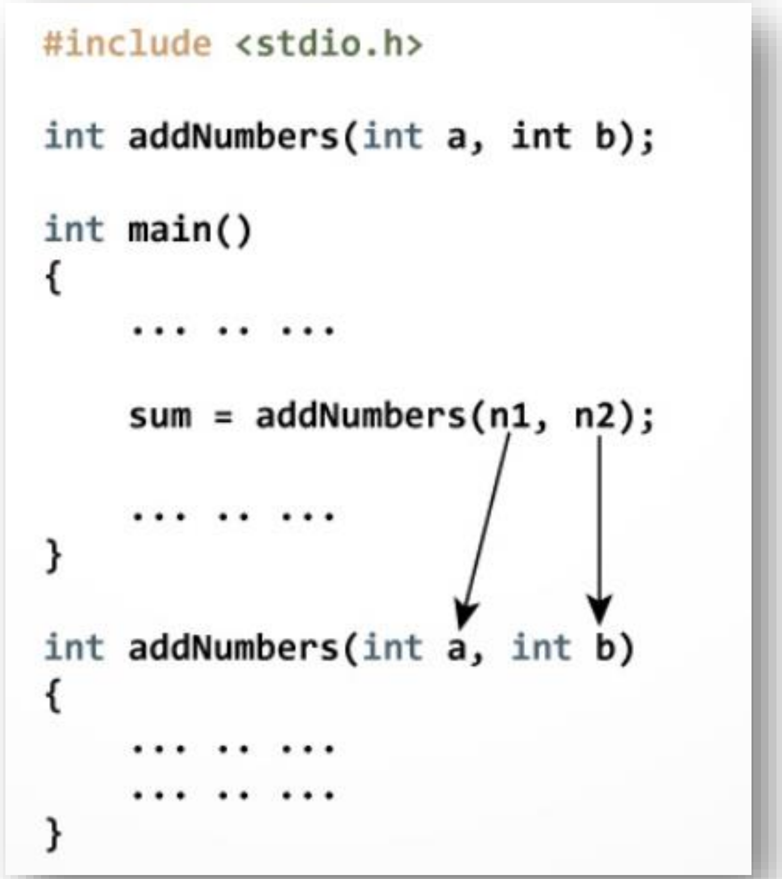
- ✓ Composed of instructions/codes (body of function)

```
/* function returning the max between two numbers */
int max(int num1, int num2) {

   /* local variable declaration */
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

An example of function in C programming

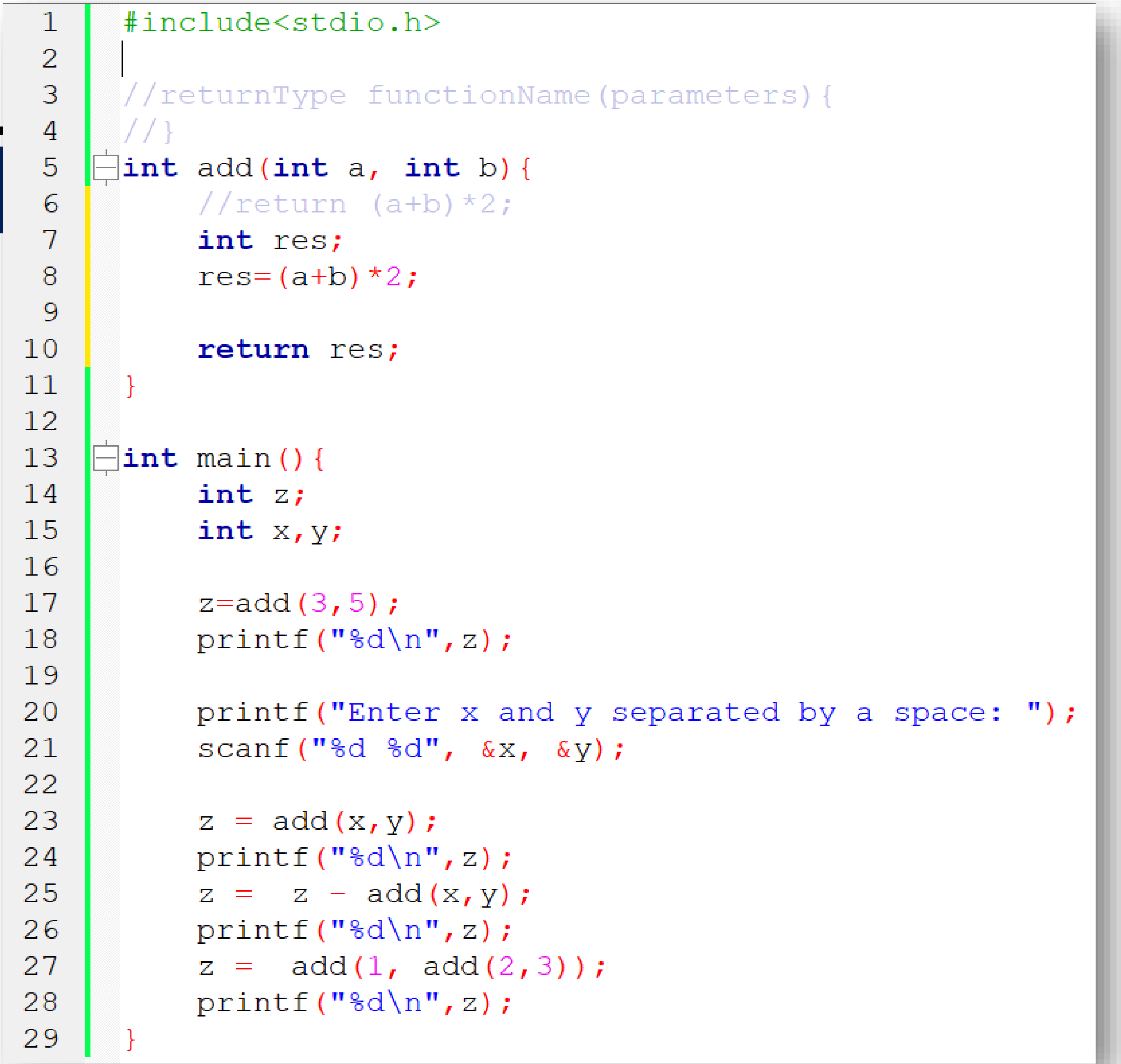
C program

❑ Examples



```c
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    ... .. ...
}
```

Create function as prototype

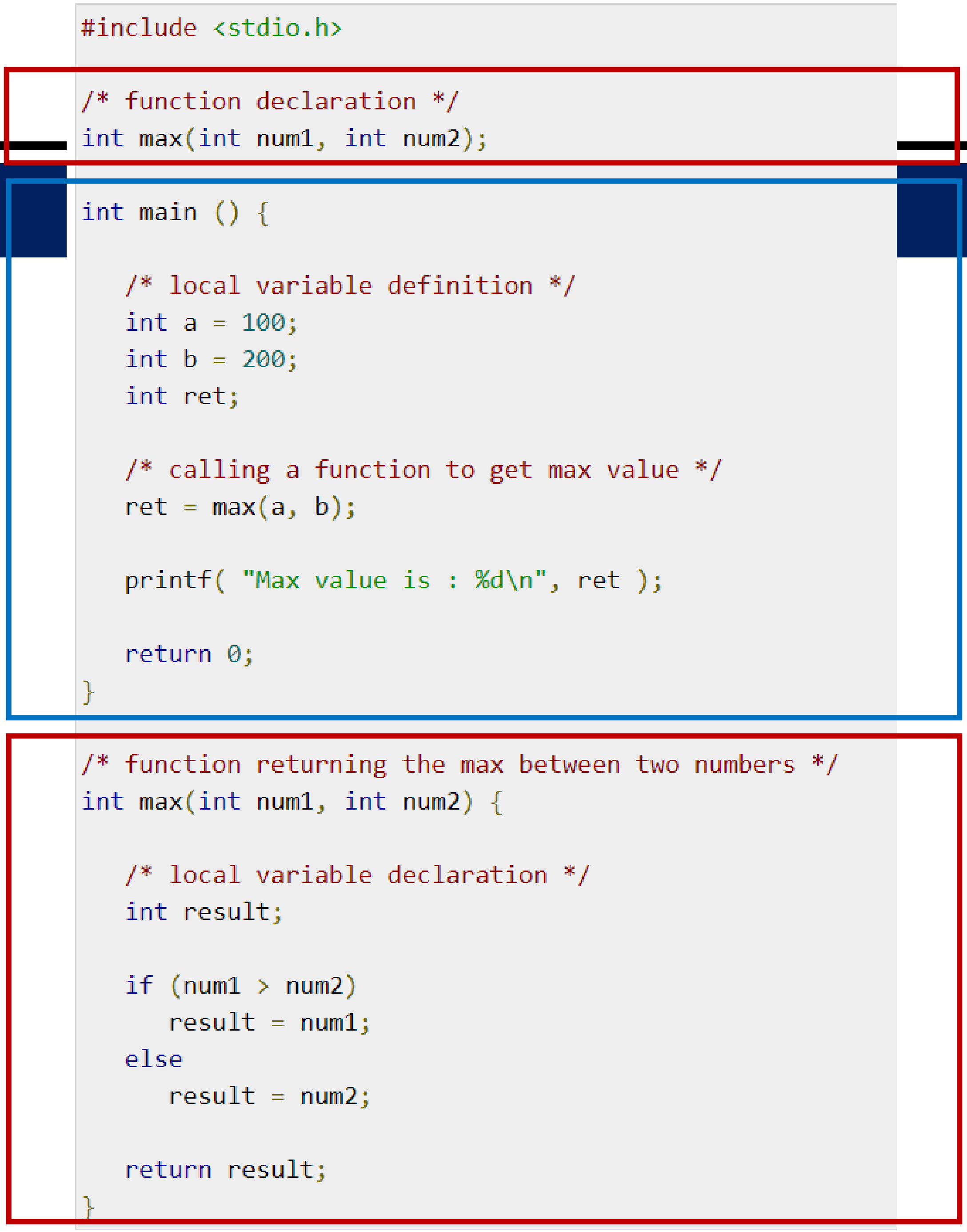


```c
#include<stdio.h>

//returnType functionName(parameters){
//}
int add(int a, int b){
    //return (a+b)*2;
    int res;
    res=(a+b)*2;

    return res;
}

int main(){
    int z;
    int x,y;

    z=add(3,5);
    printf("%d\n",z);

    printf("Enter x and y separated by a space: ");
    scanf("%d %d", &x, &y);

    z = add(x,y);
    printf("%d\n",z);
    z =  z - add(x,y);
    printf("%d\n",z);
    z =  add(1, add(2,3));
    printf("%d\n",z);
}
```

Create a function and call it in main

C program

❑ Examples



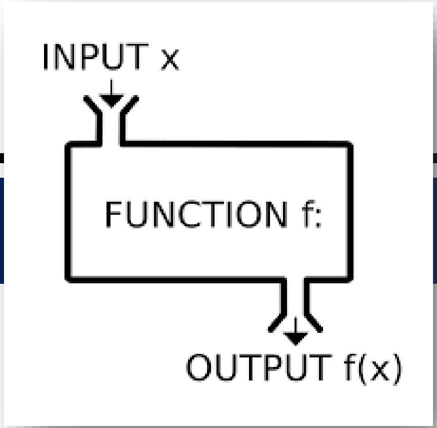
```c
#include <stdio.h>

/* function declaration */
int max(int num1, int num2);

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;
   int ret;

   /* calling a function to get max value */
   ret = max(a, b);

   printf( "Max value is : %d\n", ret );

   return 0;
}

/* function returning the max between two numbers */
int max(int num1, int num2) {

   /* local variable declaration */
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

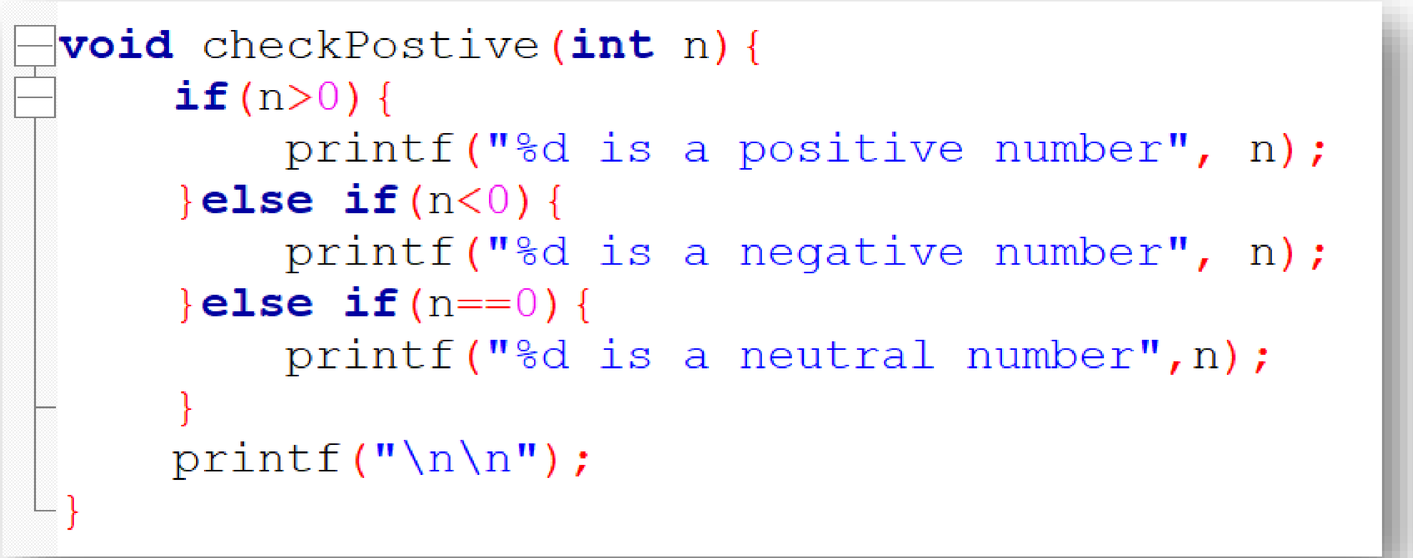
C program

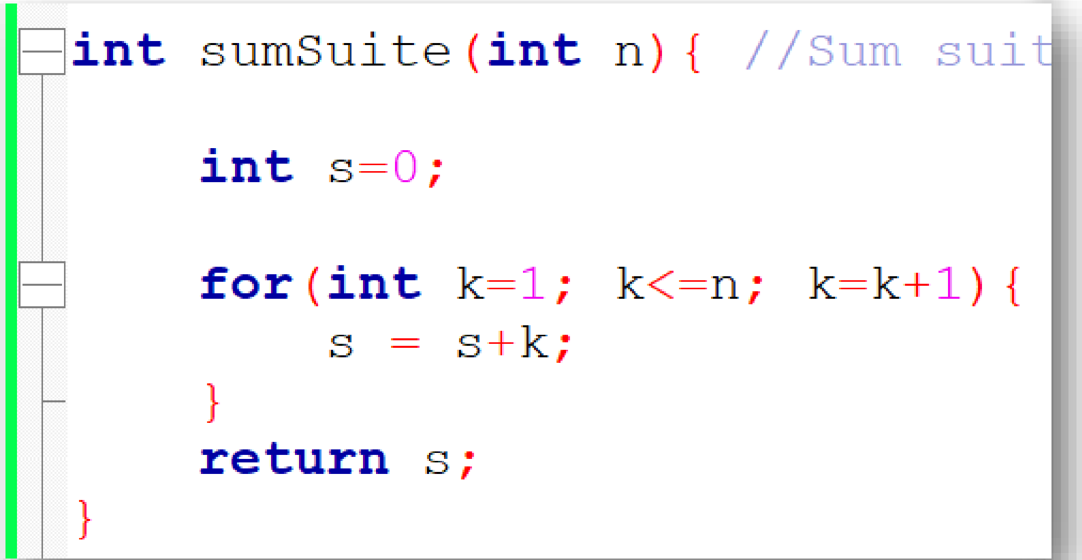
❑ Example

```c
#include <stdio.h>
int addNumbers(int a, int b);         // function prototype

int main()
{
    int n1,n2,sum;

    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);

    sum = addNumbers(n1, n2);        // function call
    printf("sum = %d",sum);

    return 0;
}

int addNumbers(int a, int b)         // function definition
{
    int result;
    result = a+b;
    return result;                  // return statement
}
```

Function *No return* Vs. *with return*




```
void checkPostive(int n){
    if(n>0){
        printf("%d is a positive number", n);
    }else if(n<0){
        printf("%d is a negative number", n);
    }else if(n==0){
        printf("%d is a neutral number",n);
    }
    printf("\n\n");
}
```


```
int sumSuite(int n){ //Sum suit

    int s=0;

    for(int k=1; k<=n; k=k+1){
        s = s+k;
    }
    return s;
}
```

checkPostive(20);

OUTPUT: ………?…….

sumSuite(6);

OUTPUT: ………?…….

Assignment

❑ Using functions to solve the following problems:

1. Write a function to calculate this formula **$y=3x^2-2x$** , where x is the parameter of the function. The function returns the value of y.
   - Let test this function in main by using the values when x=1, x=5, x=20. What are the values of y returning from your program?
2. Write a function to display whether a person is allowed to vote or not. This function have one parameter which is the age of a person.
   - **A person is allowed to vote when his/her age is greater than or equal 18.**
   - **Display the message either "You are allowed to vote" or "You are not allowed to vote"**

Deadline: 1 week