04/09/2023

* **FUNCTIONS** – Function is a block of code or sub-program which performs a specific task multiple times when we call it. OR we can say it is a block of code that is linked to a well designed task.

When functions were not in use →

```
int main( ){

 for (int i=0 ; i<2 ; i++){
    cout<<"one"<<endl;
    }
 for (int i=0 ; i<2 ; i++) {
    cout<<"two"<<endl; }
 for (int i=0 ; i<2 ; i++){
    cout <<"three" <<endl;
   }
  }
```

Output →
```
one
one
two
two
three
three
```

Here, we are using the same loop again and again. The difference is only in the string.

Issues when functions were not in use →

- lengthy and bulky code
- Buggy
- Zero Readability
- Zero Reusability

Here, function comes in the picture. Suppose, we have to reuse a specific logic or set of operations, we will use function these instead of writing a bulky code.

for instance,

```
void printName (string Name){
    for (int i=0 ; i<2 ; i++) {
        cout << name << endl;
    }
}
int main ( ){

    printName ("Babbar");
    printName ("Love");
}
```

Output – Babbar
Babbar
love
love

__/__/____

Syntax -

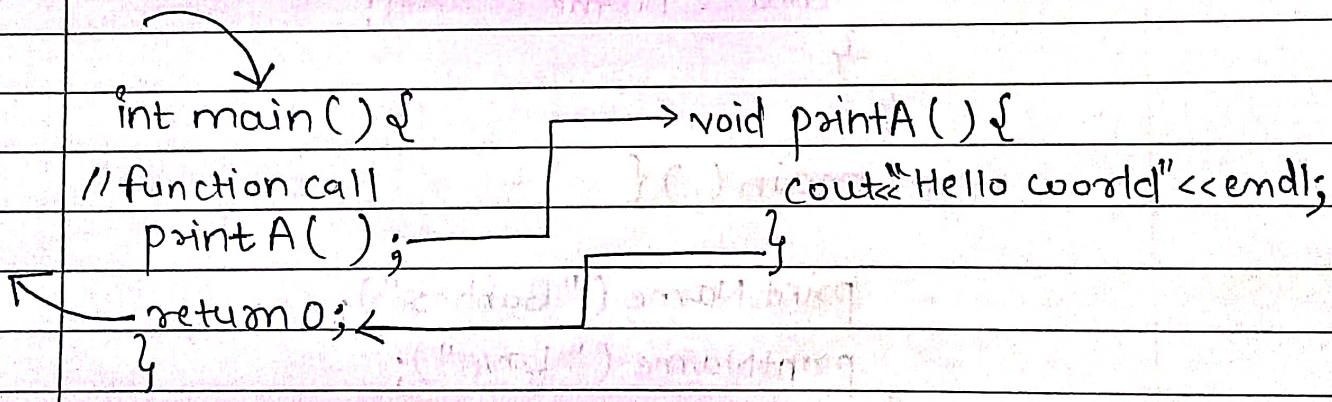
```
return_type function_name ( )
    {
        //function body
    }
```

Example -

```
int main( ){
        //function body
    }
```

The curly braces defines the scope of the function.

How things are working ?



* **FUNCTION DECLARATION** - Function declaration tells the return type, function name and input parameters.

for instance,

```
void printA ( );
int printSum (int a, int b);
```

No logic is defined during function declaration.

__/__/____

* **FUNCTION DEFINITION** – When function is declared and logic is also defined with it, it is known as function definition.

for instance,

```
int printSum(int a, int b){
    int sum = a+b;
    return sum;
}
```

* **FUNCTION CALL** – Whenever we have to use a function, we must have to call it.

for instance,

```
function_name( );
printSum(2, 3);
printmsg("Hello");
```

NOTE – We must define a function before calling it. If we want to define a function after calling it, then we must declare it above otherwise it will give error.

Example –

```
void printA( );
int main( ) {
    printA( );
}
void printA( ){
    cout << "Hello world" <<endl;
}
```

Output → Hello World

__/__/____

* **FUNCTIONS**

- which returns some value
  - int
  - char
  - string
  - bool
  - array
- which returns nothing
  - void

* **FUNCTION CALL STACK →**

- Tracks function call.
- local variables → check upon input arguments.
- Tracks which function is called by which another function.
- Return value.

Stack - Stack works on last in first Out i.e. LIFO principle. Just like the stack of plates in a marriage.

__/__/__

In function call stack →

(1) The first entry in the function call stack must be for main function.

(2) Whenever we get a function call, an entry should be added for that in function call stack.

(3) Whenever the function body ends, that entry should be removed from the function call stack.

Eg →

```
void printA(){
cout<<"inside A"<<endl;
}
void printB(){
cout<<"inside B"<<endl;
printA();
}
void printC(){
cout<<"inside C"<<endl;
printB();
}
int main(){
printC();
return 0;
}
```

Execution

Function call stack

printA()
printB()
printC()
main()

These entries will be removed when function body of these function ends.

Output →

```
inside C
inside B
inside A
```

__/__/____

* Program to print sum using functions →

```
void printSum(int a, int b){
 cout << a+b <<endl;
  }
int main( ){

    printSum(1, 3);
  }
```

Output = 4

* Program to return Sum using functions →

```
int printSum(int a, int b){
 int ans= a+b;
  return ans;
 }

 int main( ){
 int sum= printSum(3, 4);
 cout << sum <<endl;
 }
```

Output = 7

Note – max(a,b) is an in-built or pre-defined function used to find maximum value.

__/__/____

* **Using return keyword in void data type →**

```
void printmsg( ){
    cout << "msg 1" << endl;
    cout << "msg 2" << endl;
    return;
    cout << "msg 3" << endl;
}
int main( ){
    printmsg( );
}
```

Output = msg 1
msg 2

The statement after return will not be executed.

* **Program to return maximum of three →**

```
int printMax (int n1, int n2, int n3){
  int maximum= max (max(n1,n2), n3);
   return maximum;
  }
int main( ){

   int ans= printMax (2, 8, 6);
   cout<<" Maximum is : "<< ans<< endl;
}
```

Output → Maximum is: 8