FUNCTIONS

Page No.
Date 4 9 23

Functions :- A block of code that is linked to a well-designed task. Try to make it re-usable.

Need of Functions :-

→ If we don't use functions, and perform same task multiple times, then our code becomes bulky / lengthy.

→ Less readability of code.

→ Chances of bugs are high if we just copy-paste it multiple times.

→ Less reusability.

Syntax :-

```
return type  function_name ( parameters )
{
    ———
    ———
    ———
}
```

} body of function

Ex :-

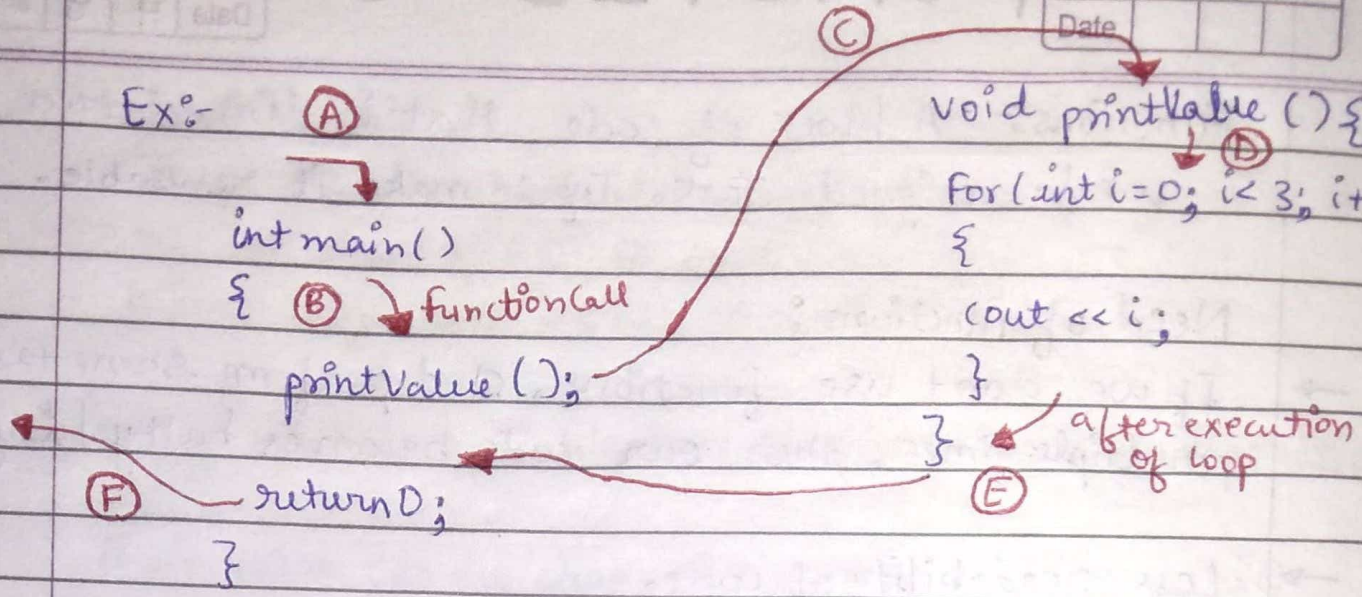
```
int main ()
{
    ———
    ———
    ———
return 0;
}
```

→ It represents the successful execution of our program executed by Operating System.

* To access any function, we have to call it with its name and required parameters (if any).

Page No.
Date

Ex:-



Flow of execution :-

(A) - Execution of main function begins.

(B) - Enters inside main and get a function call.

(C) - Reached at the function definition.

(D) - Enters inside 'printValue' function and executes the loop.

(E) - Completes the execution of 'printValue' and reached at the closing bracket.
Then flow of execution return back exactly below the function call.

(F) - Reached at return 0 and terminates the program.

Function declaration :-

* It will let to know about declaration of a function, when program gets executed by compiler.

Ex :- (i) void printMessage();

(ii) int addNumbers(int a, int b);

→ Compiler have information about a function named 'addNumbers' with integer datatype which get two integer values as input parameters.

function definition :-

* It means we are defining a complete logic of a program inside curly brackets.

Ex :-

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

} complete Logic inside function body

function call :-

* It is mandatory to make a function call to execute the specific function and access its logic written inside. Syntax :- functionName(arguments);

Ex :-

```
addNumbers(4, 6);
```

* In C++, we have to define the function before/above the function call, otherwise it produce error.

* If we want to define a function after/below the function call, then we must declare the function before/above the function call.

Page No.
Date

Ex:-

```
// function declaration
void printName();

int main() {

// function call
printName();

return 0;
}

// function definition
void printName() {
    cout << "My Name is Ditij Tanwar";
}
```

Function Call Stack :-

↳ We get information about function like about its name, local variables, function parameters, which function called which other functions, what the value is returning from a function.

* Stack is a datastructure which follows LIFO ordering i.e (Last In First Out).

* Inside function call stack, first entry is of main function. Bcz its the entry point of a program.

* Whenever a function call appears, create its entry in the stack.

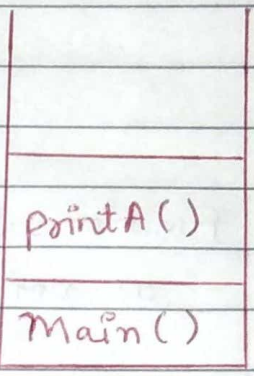
* When execution of any function is completed, remove its entry.

Page No.
Date

Ex:-

```
int main()
{
    cout<<"inside main";
    printA();
    cout<<"Back in main";
    return 0;
}

void printA(){
    cout<<"inside A";
    cout<<"going back to main";
}
```



Output:-

inside main
inside A
going back to main
Back in main

Flow of execution:-

(A) → Entry of main() is created in call stack.
(B) → Execution starts & 'inside main' is printed.
(C) → Function call appears, entry of printA() is created in stack.
(D) → Reached at function definition & start execution & printed "inside A" in output.
(E) → Next statement executed & printed "going back to main".
(F) → Execution of function completed & reached at closing bracket, so its time to remove 'printA()' from call stack. Flow of execution returned back to the parent function.
(G) → Now, code below the function call will execute & print 'Back in main'. Recieved 'return 0', means main is successfully executed so, remove its entry from call stack.

Page No.
Date

** If we want to use return keyword inside a void function, then we have to write return and do not send any data with it.

Ex:-
```
void messagePrint() {
    cout << "Msg 1" << endl;
    cout << "Msg 2" << endl;
    return;
}
```

⇒ Function which do not return any value, we can make them as void datatype.