Functions

IN 1101 PROGRAMMING FUNDAMENTALS

Functions

- A function is a self-contained block of statements that perform a coherent task of some kind.
- Some situations when we need to write a particular block of code for more than once in our program.
- C language provides an approach in which you need to declare and define a group of statements once and that can be called and used whenever required.

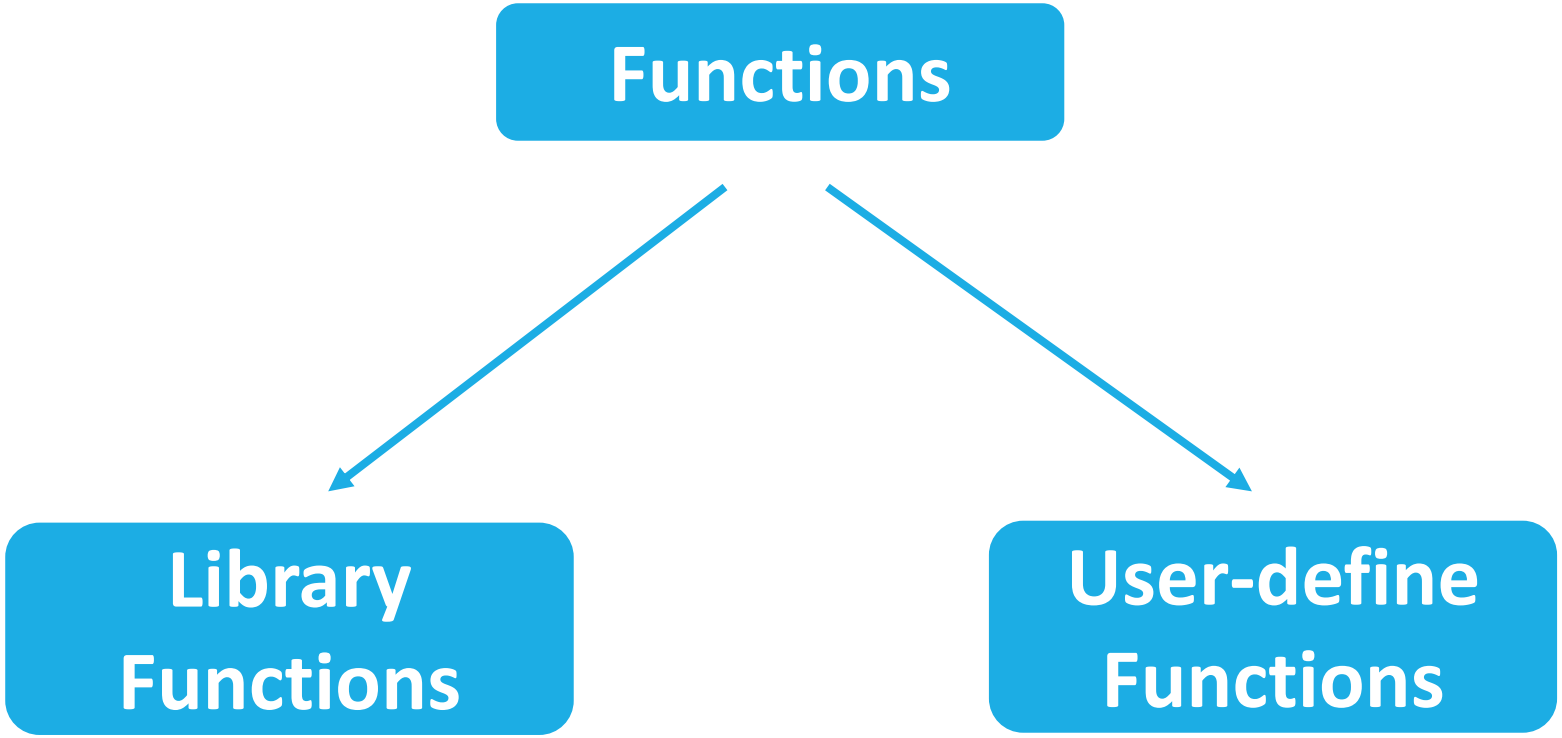
Advantages of Functions

- Code Reusability.
  - Can call it many time.
  - So we don't need to write the same code again and again.
- Modular Programming.
- Length of source code can be reduced.
  - Code optimization.
  - It makes the code optimized, we don't need to write much code.

Example

- Suppose, you have to check 3 numbers(10,7,21) whether it is odd number or even number.
- Without using a function, you need to write the logic 3 times.
- So, there is repetition of code.
- But if you use functions, you need to write the logic once and you can reuse it several times.

Types of Functions



Library Functions

- Declared in C header files.
  - Printf(), scanf(), gets(), puts()
- Need to include appropriate header files to use these functions.

User-define Functions

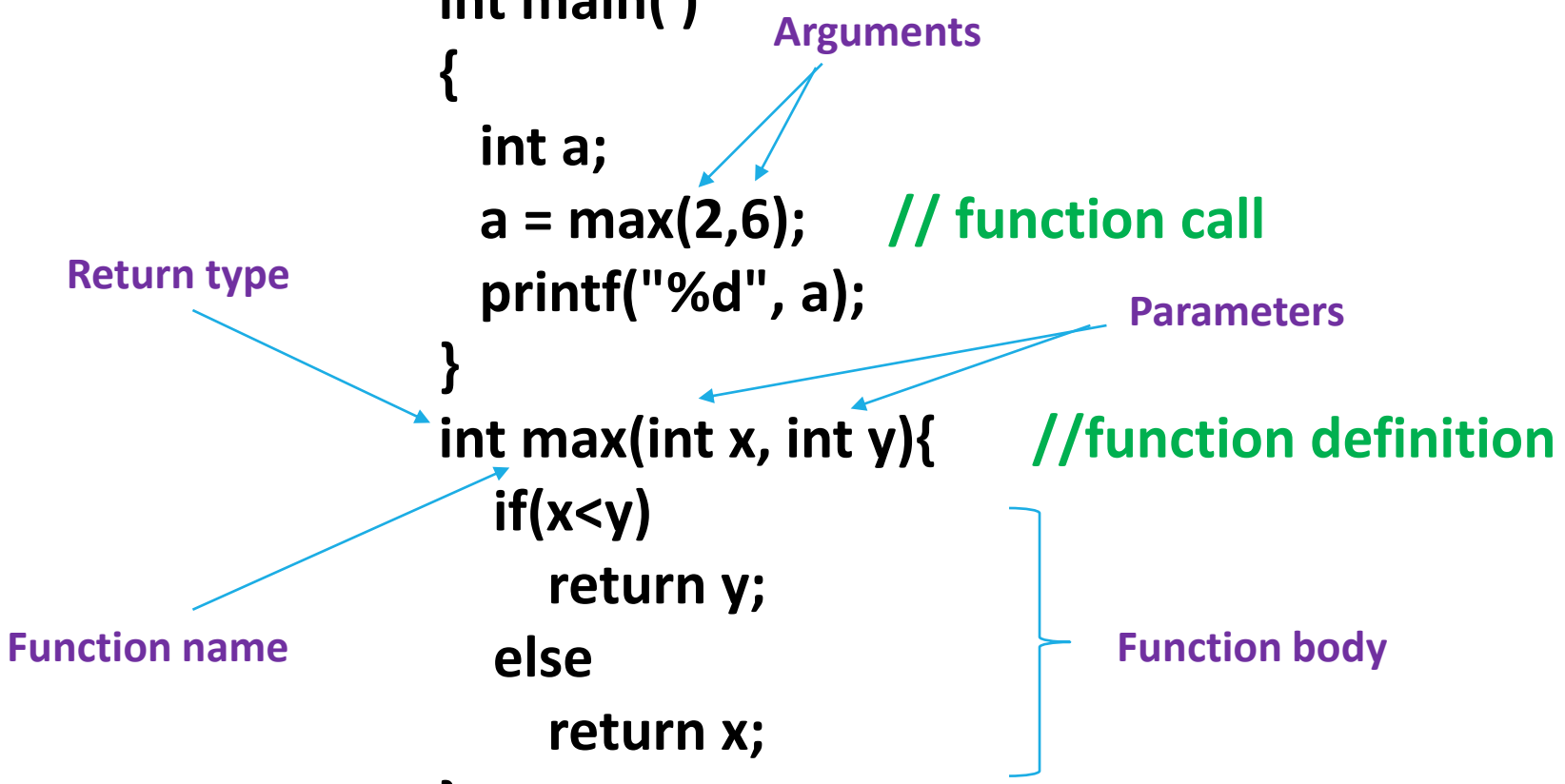
- Written by the programmer.
- Definition:

```
Return type  function_name(data_type parameter,….) {

    //code to be executed

}
```

User-define Functions

- Return type
  - The data type of the value the function returns
  - Void - Some functions perform the desired operations without returning a value.
- Function name
  - Name of the function.
  - The function name and the parameter list together constitute the function signature.
- Parameters/Function arguments
  - This value is referred to as actual parameter or argument.
  - Arguments are written within parenthesis at the time of function call.
  - Function may/not contain no parameters.
- Function body
  - The function body contains a collection of statements that define what the function does.

```
# include <stdio.h>

int max(int x, int y);     // function prototype
int main( )
{
   int a;
   a = max(2,6);      // function call
   printf("%d", a);
}
int max(int x, int y){      //function definition
    if(x<y)
        return y;
    else
        return x;
}
```



Activity

Write a program to get the summation of three numbers.

Different Function Calls

- Function may/may not accept arguments.
- Function may/may not return any value.
- There are four aspects that can be identified in function calls.
  - function without arguments and without return value
  - function without arguments and with return value
  - function with arguments and without return value
  - function with arguments and with return value

Methods of Calling Function

- Call by value
- Call by reference

Call by Value

- Copies the value of actual parameters into formal parameters.
- Value being passed to the function is locally stored by the function parameter in stack memory location.
- If you change the value of function parameter, it is changed for the current function only. It will not change the value of variable inside the caller method such as main().
- During execution whatever changes are made in the formal parameters are not reflected back in the actual parameters.

```
# include <stdio.h>
int change(int x);
int main( )
{
   int a = 5;
   change(a);
   printf("Value after function call is %d\n", a);
}
int change(int x){
    printf("Value inside the function before change is %d\n",x);
    x = x+10;
    printf("Value inside the function after change is %d\n",x);
}
```

Call by Reference

- Reference of the original variable is passed.
- Function does not create its own copy, it refers to the original value by reference.
- Functions works with the original data and changes are made in the original data.

```
# include <stdio.h>
int change(int *x);
int main( )
{
   int a = 5;
   change(&a);
   printf("Value after function call is %d\n", a);
}
int change(int *x){
    printf("Value inside the function before change is %d\n",*x);
    *x = *x+10;
    printf("Value inside the function after change is %d\n",*x);
}
```

Difference Between Call By Value and Call By Reference

Call by Value	Call by Reference
▪ A copy of the value is passed to the function.	▪ An address of value is passed to the function.
▪ Changes made inside the function is not reflected on other functions.	▪ Changes made inside the function is reflected outside the function also.
▪ Actual and formal arguments will be created in different memory location.	▪ Actual and formal arguments will be created in same memory location.

Recursion

- Function call itself.
- We will discuss recursion in a separate lesson.

Exercise

1. Write a program to calculate the area of a circle by using functions.

Exercise

2. Write a program to calculate the average of five numbers.

Exercise

Let’s make a calculator to do operations for two numbers.

1. Create a group of 5 members.
2. Choose 5 operations to be performed for two input numbers.(e.g. Addition, Substraction, etc.).
3. Divide the 5 operations among 5 members and write them using functions.
4. Write the complete program to perform calculation operations.

Questions?