Functions in C

Course Code: CSE-121

Course Title: Structural Programming Language

Khandaker Jannatul Ritu, Lecturer, BAIUST

❑ Outlines

1. Introduction to Function and it's types
2. Function Declaration, Call, Return type, definition
3. Why functions?
4. Conditions of Return type and arguments
5. Types of function arguments/parameter
6. Passing Parameters to functions
7. Use of Static variable in function
8. Exercise problems of functions: solve this functions
9. Use of Global variable in function
10. Online Judge Problems

❑ C Functions

✓ A function in C is a set of statements that when called perform some specific task.

✓ The programming statements of a function are enclosed within { } braces, having certain meanings and performing certain operations.

```
#include <stdio.h>
int cse(){
    int n;
    printf("Enter a Number: ");
    scanf("%d", &n);
    printf("The Number is: %d", n);
    return 0;
}
int main(){
   cse();
}
```

❑ Types of Functions:

1. Library Function / built-in function

For Example: pow(), sqrt(), strcmp(), strcpy()

<u>Library/Built-in functions:</u>

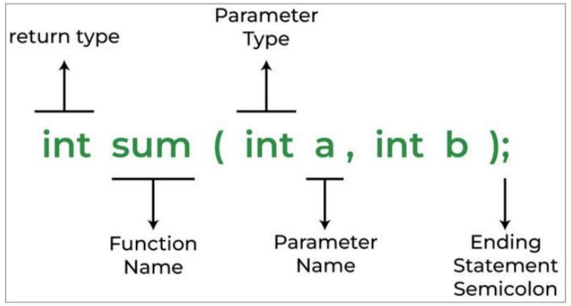
```
#include <math.h>
#include <stdio.h>
int main(){
 double Number;
 Number = 49;
 double squareRoot = sqrt(Number);
 printf("The Square root of %.2lf = %.2lf", Number, squareRoot);
}
```

2. User Defined Function

```
#include <stdio.h>
int sum(int a, int b) {
        return a + b;
}
int main(){
 int a = 30, b = 40;
 int res = sum(a, b);
 printf("Sum is: %d", res);
}
```

❑ Function Declaration

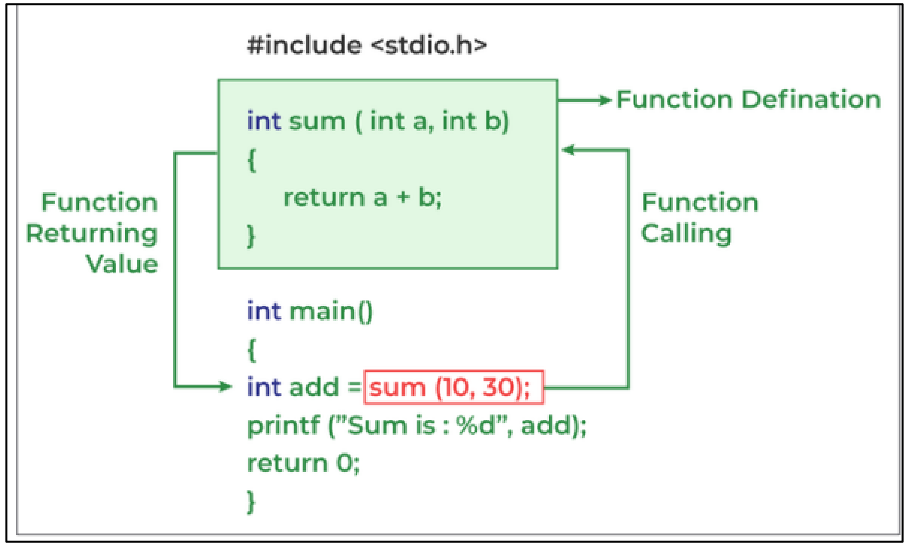
Functions are the block of code that is executed every time they are called during an execution of a program.



```
int sum ( int a , int b );
```

❑ Function Call

A function call is a statement that instructs the compiler to execute the function. We use the function name and parameters in the function call.



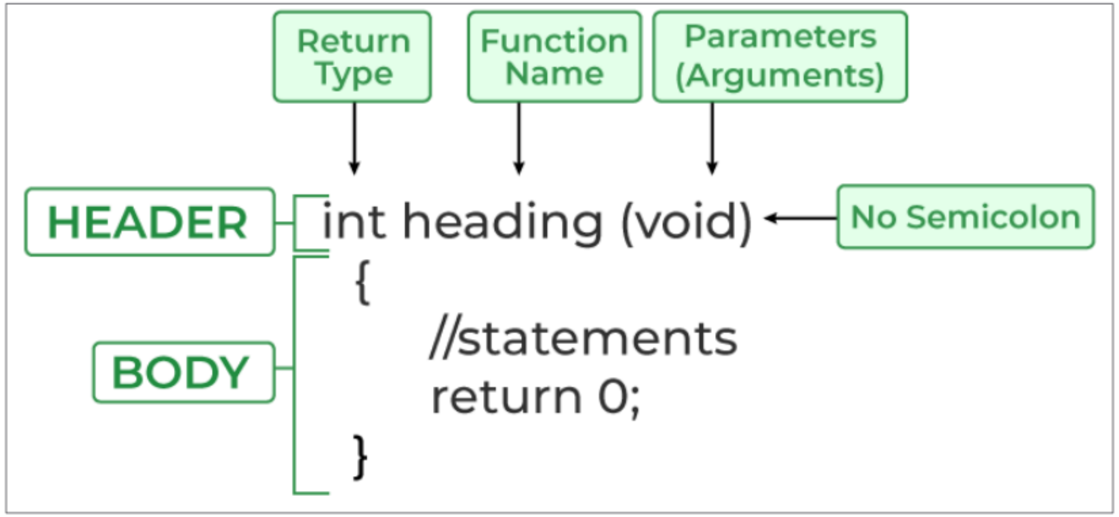
```
#include <stdio.h>

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

int main()
{
int add = sum (10, 30);
printf ("Sum is : %d", add);
return 0;
}
```

❑ Function return type

Function return type tells what type of value is returned after all function is executed. When we don’t want to return a value, we can use the void data type.

❑ Function definition

The function definition consists of actual statements which are executed when the function is called (i.e. when the program control comes to the function).



```
int heading (void)
{
    //statements
    return 0;
}
```

❑ Why function?

➢ Before using Function

```c
#include <stdio.h>
int main(){
    int n;
    printf("Enter a Number: ");
    scanf("%d", &n);
    printf("The Number is: %d\n", n);

    printf("Enter a Number: ");
    scanf("%d", &n);
    printf("The Number is: %d\n", n);

    printf("Enter a Number: ");
    scanf("%d", &n);
    printf("The Number is: %d\n", n);
}
/*
Enter a Number: 45
The Number is: 45
Enter a Number: 3
The Number is: 3
Enter a Number: 44
The Number is: 44
*/
```

➢ After using Function

```c
#include <stdio.h>
int cse(){
    int n;
    printf("Enter a Number: ");
    scanf("%d", &n);
    printf("The Number is: %d\n", n);
}

int main(){
   cse();
   cse();
   cse();
}

/*
Enter a Number: 45
The Number is: 45
Enter a Number: 3
The Number is: 3
Enter a Number: 44
The Number is: 44
*/
```

❑ Why function?

➢ Before using Function

```c
#include <stdio.h>
int main(){
    int n;
    printf("Enter Array Size: ");
    scanf("%d", &n);
    int myArray[n];
    printf("Enter Array Elements: \n");
    for(int i=0; i<n; i++){
        scanf("%d", &myArray[i]);
    }
    printf("Output: \n");
    for(int i=0; i<n; i++){
        printf("%d ", myArray[i]);
    }
    printf("\n--------------------\n");
    printf("Enter Array Size: ");
    scanf("%d", &n);
    int myArray1[n];
    printf("Enter Array Elements: \n");
    for(int i=0; i<n; i++){
        scanf("%d", &myArray1[i]);
    }
    printf("Output: \n");
    for(int i=0; i<n; i++){
        printf("%d ", myArray1[i]);
    }
}
```

➢ After using Function

```c
#include <stdio.h>
int cse(){
    int n;
    printf("Enter Array Size: ");
    scanf("%d", &n);
    int myArray[n];
    printf("Enter Array Elements: \n");
    for(int i=0; i<n; i++){
        scanf("%d", &myArray[i]);
    }
    printf("Output: \n");
    for(int i=0; i<n; i++){
        printf("%d ", myArray[i]);
    }
    printf("\n--------------------\n");
    return 0;
}

int main(){
   cse();
   cse();
   cse();
}
```

❑ Conditions Of Return Type And Arguments

✓ Types of Function According to Arguments and Return Value

Functions can be differentiated into 4 types according to the arguments passed and value returns these are:

1.Function with no arguments and no return value
2.Function with no arguments and with return value
3.Function with arguments and no return value
4.Function with arguments and return value

1. Function with no argument and no return value	2. Function with no arguments but returns a value

`#include <stdio.h>`
`void myfunction(){`
`     printf("Hello World");`
`}`
`int main(){`
`    myfunction();`
`}`
	`#include <stdio.h>`
	`int myfunction(){`
	`    return 10*10;`
	`}`
	`int main(){`
	`    int answer = myfunction(); ///method-1`
	`    printf("%d \n", answer);`
	`    printf("%d", myfunction()); ///method-2`
	`}`

3. Function with arguments but no return value

➢ Passing Array To A Function

```
#include <stdio.h>
void check(int pupil[], int sz){
     for(int i=0; i<sz; i+/){
         printf("%d ", pupil[i]);
     }
}
int main(){
    int N; scanf("%d", &N);
    int student[N];
    for(int i=0; i<N; i+/){
        scanf("%d", &student[i]);
    }
    check(student, N);
}
```

➢ Passing Characters To A Function

```
#include <stdio.h>
void myfunction(char* str){
     printf("Output: %s", str);
}
int main(){
    int str[100];
    scanf("%s", str);
    myfunction(str);
}
```

4. Function with arguments and return value

➢ C program to Find average of 3 values

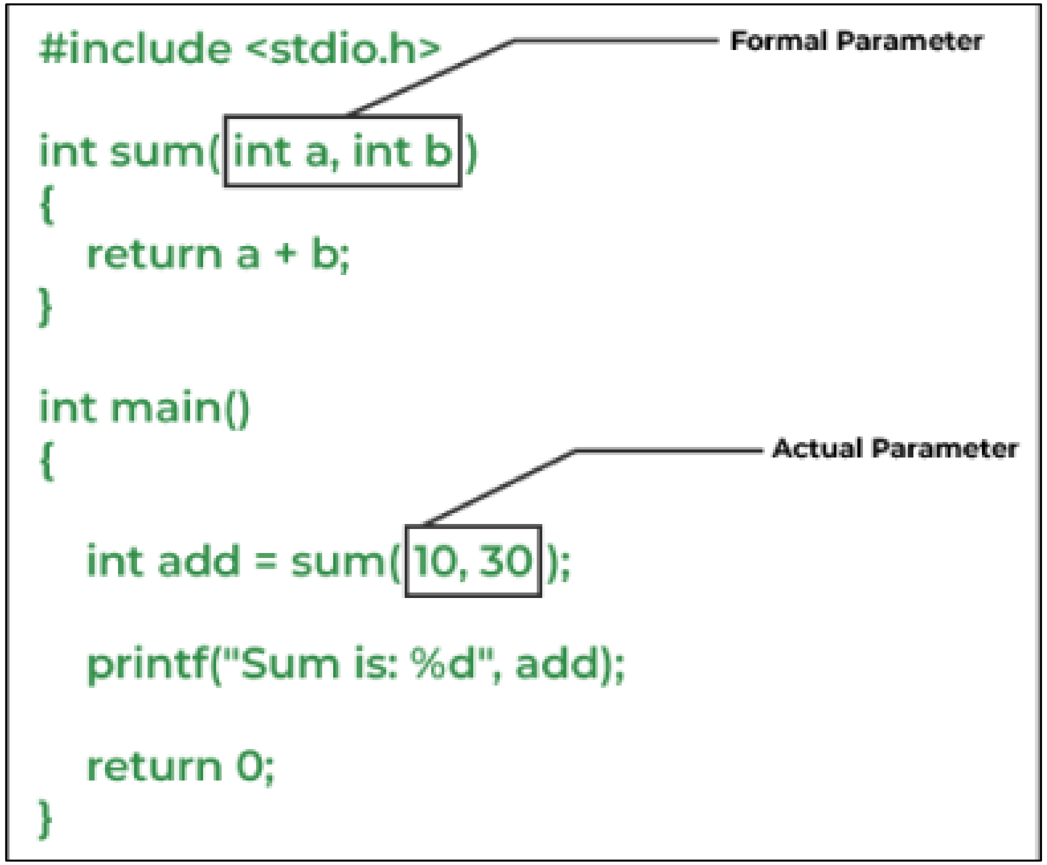
```
#include <stdio.h>
int average(int x, int y, int z){
    int add = x+y+z;
    return add/3;
}
int main(){
    int answer = average(2,4,6);  ///method-1
    printf("%d \n", answer);
    printf("%d", average(10,4,6));  ///method-2
}
```

➢ C program to find the x y without library functions

```
#include <stdio.h>
int power(int x, int y){
    int ans = 1;
    for(int i=1; i</y; i+/){
       ans = ans * x;
    }
    return ans;
}
int main(){
    printf("power :%d\n", power(2,4));
    printf("power :%d\n", power(5,2));
    printf("power :%d\n", power(2,5));
}
```

❑ Function Arguments/ Parameter

1. Formal parameter/ Formal arguments
2. Actual parameter/ Arguments



```
#include <stdio.h>

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

int main()
{

   int add = sum(10, 30);

   printf("Sum is: %d", add);

   return 0;
}
```

Q-1. What is the actual and formal parameter?

Formal parameter: The variables declared in the function prototype is known as Formal arguments or parameters.

Actual parameter: The values that are passed in the function are known as actual arguments or parameters.

Q-2. Can we return multiple values from a C Function?

No, it is generally not possible to return multiple values from a function. But we can either use pointers to static or heap memory locations to return multiple values or we can put data in the structure and then return the structure.

Q-3. What is the difference between function arguments and parameters?

Function parameters are the values declared in a function declaration. Whereas, function arguments are the values that are passed in the function during the function call.

Example:

```
int func(int x,int y);
func(10,20);
```

Here, int x and int y are parameters while,
10 and 20 are the arguments passed to the function.

Passing Parameters to Functions

We can pass arguments to the C function in two ways:

1.Pass by Value/ Call by value

2.Pass by Reference/ Call by reference

1. Pass By Value

In call by value method of parameter passing, the values of actual parameters are copied to the function’s formal parameters.

There are two copies of parameters stored in different memory locations.

One is the original copy and the other is the function copy. Any changes made inside functions are not reflected in the actual parameters of the caller.

C program to illustrate call by value

Example-1:

```
#include <stdio.h>
void func(int a, int b){
     a += b;
     printf("In func, a = %d b = %d\n", a, b);
}
int main(void){
    int x = 5, y = 7;
    func(x, y);
    printf("In main, x = %d y = %d\n", x, y);
}
```

Example-2: Swap Two Variables

```
#include <stdio.h>
void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
}
int main(){
    int x = 10, y = 20;
    printf("before swap are: %d, %d\n", x, y);
    swap(x, y);
    printf("after swap are: %d, %d", x, y);
}
```

2. Pass by Reference

In call by reference method of parameter passing, the address of the actual parameters is passed to the function as the formal parameters. In C, we use pointers to achieve call-by-reference.

- Both the actual and formal parameters refer to the same locations.
- Any changes made inside the function are actually reflected in the actual parameters of the caller.

Example-1: Swap Two Variables Using Pointers and Functions

```
#include <stdio.h>
void swap(int* a, int* b){
       int temp = *a;
       *a = *b;
       *b = temp;
}
int main(){
       int x = 10, y = 20;
       printf("x and y before swap are: %d, %d\n", x, y);
       swap(&x, &y);
       printf("x and y after swap are: %d, %d", x, y);
}
```

H.W.: Differentiate between the Call by Value and Call by Reference in C

- **Static Variable:** Static variables have the property of preserving their value even after they are out of their scope! Hence, a static variable preserves its previous value in its previous scope and is not initialized again in the new scope.

```
static data_type var_name = var_value;
```

Use of static variable in function

```
int fun(){
    static int count = 0;
    count++;
    return count;
}
int main(){
    printf("%d ", fun());
    printf("%d ", fun());
    return 0;
}
```

```
int incr(int i){
   static int count = 0;
   count = count + i;
   return (count);
}
main(){
   int i,j;
   for (i = 0; i <=4; i++)
      j = incr(i);
}
```

```
int a, b, c = 0;
void prtFun(void);
main( ){
   static int a = 1;
   prtFun( );
   a += 1;
   prtFun( );
   printf(" \n %d %d ", a, b);
}
void prtFun(void){
   static int a = 2;
   int b = 1;
   a += ++b;
   printf(" \n %d %d ", a, b);
}
```

```
#include <stdio.h>
int fun(){
    int count = 0;
    count++;
    return count;
}
int main(){
    printf("%d ", fun());
    printf("%d ", fun());
    return 0;
}
```

```
void foo(void){
      static int staticVar;
      staticVar++;
      printf("foo: %d\n",staticVar);
}
void bar(void){
      static int staticVar;
      staticVar++;
      printf("bar: %d\n",staticVar);
}
int main(){
      foo(), bar(), foo();
}
```

❑ Exercise problems of functions: solve this functions

Ex-1:

```
#include <math.h>
#include <stdio.h>
int sum();
int main(){
  int num;
  num = sum();
  printf("Sum of two given values = %d", num);
  return 0;
}
  int sum(){
  int a = 50, b = 80, sum;
  sum = sqrt(a) + sqrt(b);
  return sum;
}
```

Ex-2:

```
#include <stdio.h>
void value(void);
void main(){
  value();
}
void value(void){
  float year = 1, period = 5, amount = 5000, inrate = 0.12;
  float sum;
  sum = amount;
  while(year ≤ period){
       sum = sum * (1 + inrate);
       year = year + 1;
  }
  printf(" The total amount is :%f", sum);
}
```

Ex-3:

```
#include <stdio.h>
void myfunction(){
  int x;
  printf("Enter a value: ");
  scanf("%d", &x);
  printf("The value is : %d", x);
}
int main(){
  myfunction();
}
```

Ex-4:

```
#include <stdio.h>
void myfunction(float area){
 printf("Area : %d", area);
}
int main(){
 int a,b,c;
}
```

Ex-5:

```c
#include <stdio.h>
int sum(int x, int y) {
  int c;
  c = x + y;
  return c;
}
int main() {
int a = 3, b = 2;
int c = sum(a, b);
printf("Sum of %d and %d : %d", a, b, c);
}
```

Ex-6:

```c
#include <stdio.h>
int sum(int, int);
int sum(int x, int y){
  int sum;
  sum = x + y;
  return x + y;
}
int main(){
  int x = 10, y = 11;
  int result = sum(x, y);
  printf("Sum of %d and %d = %d ", x, y, result);
}
```

Ex-7:

```c
#include <stdio.h>
#include <string.h>
void fun(char* arr){
  int i;
  unsigned int n = strlen(arr);
  printf("n = %d\n", n);
  for (i = 0; i < n/2; i++)
  printf("%c ", arr[i]);
}
int main(){
  char arr[] = "geeksquiz";
  fun(arr);
}
```

Ex-8:

```c
#include <stdio.h>
void printString(char* str){
  printf("Array of Characters: ");
  int i = 0;
  while(str[i] != '\0') {
       printf("%c \n", str[i]);
       i++;
  }
}
int main(){
  char arr[] = "String";
  printString(arr);
}
```

Ex-9:

```
void fun(char* arr){
  int i;
  unsigned int n = strlen(arr);
  printf("n = %d\n", n);
  for(i = 0; i < n; i++)
      printf("%c ", arr[i]);
  }
int main(){
  char arr[] = { 'g', 'e', 'e', 'k', 's', 'q', 'u', 'i', 'z' };
}
```

Ex-10:

```
void cse(int p){
    p = p+10;
}
int main(){
   int p = 5;
   printf("Before Passing to function: %d\n", p);
   printf("After Calling Function: %d\n", cse(p));
}
```

Ex-11:

```
int checkOddEven(int n1){
  return (n1 % 2);
}
int main(){
 int n1;
 printf("Input any number : ");
 scanf("%d", &n1);
 if(checkOddEven(n1)){
    printf("The entered number is odd.\n\n");
 }
 else{
    printf("The entered number is even.\n\n");
 }
}
```

Ex-12:

```
void flag(float sum){
   printf("sum : %f", sum);
}
void myfunction(int x, float y) {
   float sum = x + y;
   flag(sum);
}
int main(){
   int a;
   float b;
   printf("Enter two value: ");
   scanf("%d %f", &a, &b);
   myfunction(a, b);
}
```

Ex-13:

```
#include <stdio.h>
int g(int p) { printf("%d", p); return p; }
int h(int q) { printf("%d", q); return q; }
void f(int x, int y) {
g(x);
h(y);
}
int main() {
 f(g(10),h(20));
}
```

Ex-14:

```
#include < stdio.h >
int funcp(){
   static int x = 1;
   x++;
   return x;
}
int main(){
   int x,y;
   x = funcp();
   y = funcp()+x;
   printf("%d\n", (x+y));
   return 0;
}
```

Ex-15:

```
#include <stdio.h>
void foo(void){
 printf(“ict");
}
void bar(void){
 printf("Quiz");
}
int main(){
 foo(), bar();
}
```

Ex-16:

```
#include <stdio.h>
int f1() { printf (“ICT"); return 1;}
int f2() { printf ("Quiz"); return 1;}

int main(){
  int p = f1() + f2();
  return 0;
}
```

❑ Use of Global variable in function

Ex-1:

```
int value = 100;
void check(){
 printf("check() function : %d\n", value);
}

int main(){
 check();
 printf("main() function: %d\n", value);
 check();
}
```

Ex-2:

```
void check(){
 int value = 100;
 printf("check() function : %d\n", value);
}
int main(){
 int value = -5;
 check();
 printf("main() function: %d\n", value);
 check();
}
```

Ex-3:

```
int answer;
void check(){
        printf("inside check() function: %d\n", answer);
}
void check1(){
        printf("inside check1() function: %d\n", answer);
}
int main(){
        scanf("%d", &answer);
        check();
        check1();
}
```

Ex-4:

```
#include <stdio.h>
int student[100];
int sz;
void check(){
     for(int i=0; i<sz; i++){
     printf("%d ", student[i]);
     }
}
int main(){
     scanf("%d", &sz);
     for(int i=0; i<sz; i++){
     scanf("%d", &student[i]);
     }
     check();
}
```

Ex-5:

```
#include <stdio.h>
char str[100];
int sz;
void check(){
     printf("%s", str);
}
int main(){
     scanf("%s", str);
     sz = strlen(str);
     check();
}
```

Ex-6:

```
#include<stdio.h>
void function_1(){
    printf("I am function - 4\n");
    function_2();
}
void function_2(){
    function_3();
    printf("I am function - 3\n");
}
void function_3(){
    printf("I am function - 2\n");
    function_4();
}
void function_4(){
    printf("I am function - 1\n");
}
int main() {
    function_1();
}
```

Ex-7:

```
#include <stdio.h>
#define sz 10
int name[sz];
int main(){
     for(int i=0; i<sz; i++){
       printf("%d ", name[i]);
     }
     printf("\n");
     int fame[sz];
     for(int i=0; i<sz; i++){
       printf("%d ", fame[i]);
     }
}
```

Online Judge Problems:

1. https://codeforces.com/group/MWSDmqGsZm/contest/223205/problem/A
2. https://codeforces.com/group/MWSDmqGsZm/contest/223205/problem/B
3. https://codeforces.com/group/MWSDmqGsZm/contest/223205/problem/D
4. https://codeforces.com/group/MWSDmqGsZm/contest/223205/problem/G
5. https://codeforces.com/group/MWSDmqGsZm/contest/223205/problem/J
6. https://codeforces.com/group/MWSDmqGsZm/contest/223205/problem/L