**Functions**

In c, we can divide a large program into the basic building blocks known as function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the C program. In other words, we can say that the collection of functions creates a program.

Advantage

* By using functions, we can avoid rewriting same logic/code again and again in a program.
* We can call C functions any number of times in a program and from any place in a program.
* We can track a large C program easily when it is divided into multiple functions.
* Reusability is the main achievement of C functions.
* However, Function calling is always a overhead in a C program.

|  |  |  |
| --- | --- | --- |
| **Function declaration** | **Function call** | **Function definition** |
| A function must be declared globally in a c program to tell the compiler about the function name, function parameters, and return type.  **Syntax**  return\_type function\_name (argument list); | Function can be called from anywhere in the program. The parameter list must not differ in function calling and function declaration. We must pass the same number of functions as it is declared in the function declaration.  **Syntax**  function\_name (argument\_list) | It contains the actual statements which are to be executed. It is the most important aspect to which the control comes when the function is called. Here, we must notice that only one value can be returned from the function.  **Syntax**  return\_type function\_name (argument list) {  function body;  } |
|  |  |  |
|  |  |  |

Types of Functions

1. **Library Functions**: are the functions which are declared in the C header files such as scanf(), printf(), gets(), puts(), ceil(), floor() etc.
2. **User-defined functions**: are the functions which are created by the C programmer, so that he/she can use it many times. It reduces the complexity of a big program and optimizes the code.

Different aspects of function calling

A function may or may not accept any argument. It may or may not return any value. Based on these facts, There are four different aspects of 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

### Function without argument and return value

|  |  |
| --- | --- |
| 1. #include<stdio.h> 2. **void** printName(); 3. **void** main () 4. { 5. printf("Hello "); 6. printName(); 7. } 8. **void** printName() 9. { 10. printf("STCET"); 11. } | 1. #include<stdio.h> 2. **void** sum(); 3. **void** main() 4. { 5. printf("\nGoing to calculate the sum of two numbers:"); 6. sum(); 7. } 8. **void** sum() 9. { 10. **int** a,b; 11. printf("\nEnter two numbers"); 12. scanf("%d %d",&a,&b); 13. printf("The sum is %d",a+b); 14. } |

### Function without argument and with value

#include<stdio.h>

**int** sum();

**void** main()

{

**int** result;

    printf("\nGoing to calculate the sum of two numbers:");

    result = sum();

    printf("%d",result);

}

**int** sum()

{

**int** a,b;

    printf("\nEnter two numbers");

    scanf("%d %d",&a,&b);

**return** a+b;

}

### Example for Function with argument and without return value

#include<stdio.h>

**void** sum(**int**, **int**);

**void** main()

{

**int** a,b,result;

    printf("\nGoing to calculate the sum of two numbers:");

    printf("\nEnter two numbers:");

    scanf("%d %d",&a,&b);

    sum(a,b);

}

**void** sum(**int** a, **int** b)

{

    printf("\nThe sum is %d",a+b);

}

### Example for Function with argument and with return value

|  |  |
| --- | --- |
| 1. #include<stdio.h> 2. **int** sum(**int**, **int**); 3. **void** main() 4. { 5. **int** a,b,result; 6. printf("\nGoing to calculate the sum of two numbers:"); 7. printf("\nEnter two numbers:"); 8. scanf("%d %d",&a,&b); 9. result = sum(a,b); 10. printf("\nThe sum is : %d",result); 11. } 12. **int** sum(**int** a, **int** b) 13. { 14. **return** a+b; 15. } | 1. #include<stdio.h> 2. **int** even\_odd(**int**); 3. **void** main() 4. { 5. **int** n,flag=0; 6. printf("\nGoing to check whether a number is even or odd"); 7. printf("\nEnter the number: "); 8. scanf("%d",&n); 9. flag = even\_odd(n); 10. **if**(flag == 0) 11. { 12. printf("\nThe number is odd"); 13. } 14. **else** 15. { 16. printf("\nThe number is even"); 17. } 18. } 19. **int** even\_odd(**int** n) 20. { 21. **if**(n%2 == 0) 22. { 23. **return** 1; 24. } 25. **else** 26. { 27. **return** 0; 28. } |