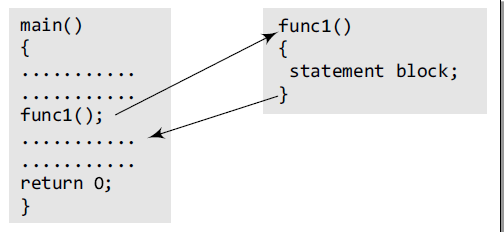
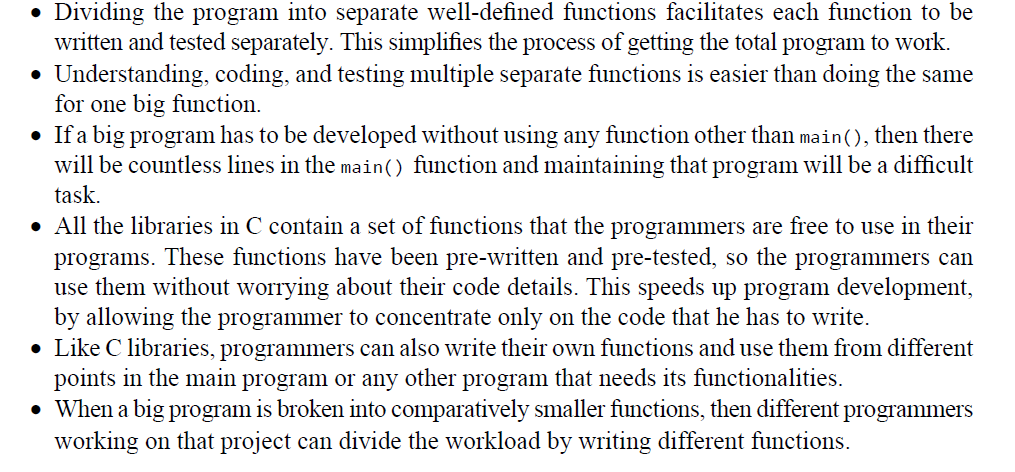
**FUNCTIONS**

* C enables its programmers to break up a program into segments commonly known as **functions**, each of which can be written more or less independently of the others. Every function in the program is supposed to perform a well-defined task. Therefore, the program code of one function is completely insulated from the other functions.
* Every function interfaces to the outside world in terms of how information is transferred to it and how results generated by the function are transmitted back from it. This interface is basically specified by the **function name**.

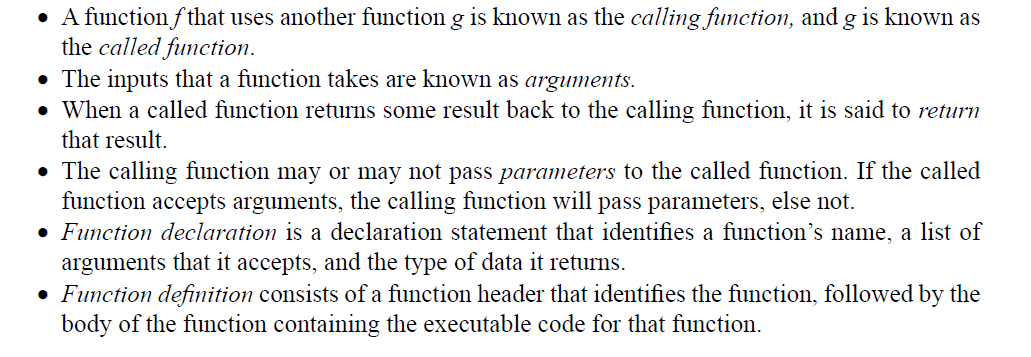


* ***main()*** calls a function named ***func1()***. Therefore, main() is known as the ***calling function*** and func1() is known as the ***called function***.
* The moment the compiler encounters a function call, the control jumps to the statements that are a part of the called function. After the called function is executed, the control is returned to the calling program.
* The main() function can call as many functions as it wants and as many times as it wants.
* Not only main(), any function can call any other function.

**Why are Functions Needed.**



Definition of Terms.



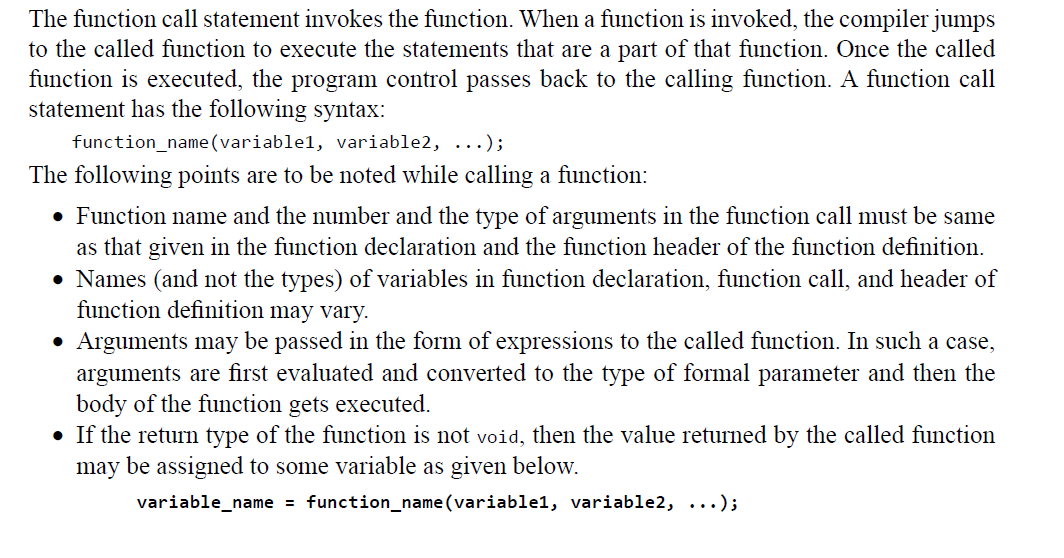
Function Declaration.

* Before using a function, the compiler must know the number of parameters and the type of parameters that the function expects to receive and the data type of value that it will return to the calling program.
* The general format for declaring a function that accepts arguments and returns a value as result can be given as:

***return\_data\_type function\_name(data\_type variable1, data\_type variable2,..);***

* Note that the number of arguments and the order of arguments in the function header must be the same as that given in the function declaration statement.

**Function call**



#include <stdio.h>

int evenodd(int); //FUNCTION DECLARATION

int main(){

    int num, flag;

    printf("\n Enter the number : ");

    scanf("%d", &num);

    flag = evenodd(num); //FUNCTION CALL

    if (flag == 1)

        printf("\n %d is EVEN", num);

    else

        printf("\n %d is ODD", num);

    return 0;

}

    int evenodd(int a) // FUNCTION HEADER

{

    // FUNCTION BODY

    if(a%2 == 0)

        return 1;

    else

        return 0;

}

Exercise.

1. Ammend your calculator code to have individual functions handling its operation. These functions need to be callable by the main() function and return their response to main.
2. The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones, usually starting with 0 and 1. The sequence starts like this: 0, 1, 1, 2, 3, 5, 8, 13, 21, and so on. Mathematically, the Fibonacci sequence can be defined recursively as follows: F(0) = 0

F(1) = 1

F(n) = F(n-1) + F(n-2) for n > 1

Write a C program to generate the Fibonacci series up to a given number n.

1. Write a C program to check whether a given number n is a palindrome or not.
2. An Armstrong number (also known as narcissistic number, plenary number, or pluperfect digital invariant) is a number that is equal to the sum of its own digits each raised to the power of the number of digits. For example, 153 is an Armstrong number because:

13+53+33 = 153

Write a C program to check whether a given number n is an Armstrong number or not.

**This assessment is to be completed and uploaded to github by midnight Friday 16th 2024. Happy Coding.**