Functions - Basics

- A function is a self-contained unit of program code to accomplish a particular task.
- Syntax rules define the structure of a function and how it can be used .
- A function in C is the same as subroutine or procedures in other programming languages.
- Some functions cause an action to take place.
    - ➔ printf()causes data to be printed on your screen.
- Some functions find a value for a program to use .
    - ➔ strlen() tells the program how long a certain string is .

Advantages

- Allows for the divide and conquer strategy.
    - ➔ It is very difficult to write an entire program as a single large main function.
        - ★ Difficult to test, debug and maintain,
- With divide and conquer, tasks can be divided into several independent subtasks.
    - ➔ Reduces the overall complexity
    - ➔ Separate functions written for each subtask.
    - ➔ We can further divide each subtask into smaller subtasks, futher reducing the complexity.
- If you have to do a certain task several times in a program, you only need to write an appropriate function once .
    - ➔ Program can then use that function wherever needed,
    - ➔ You can also use the same function in different programs (printf)
- Helps with readability
    - ➔ Program is better organised .
    - ➔ Easier to read and easier to change or fix
- The divide and conquer approach also allows the parts of a program to be developed, tested and debugged independently.
    - ➔ Reduces the overall development time.
- The function developed for one program can be used in another program.
        - ➔ Software reuse.
- Many programmers like to think of a function as a “black Box”
    - ➔ Information that goes in (its input)
    - ➔ The value or action it produces (its output)
- Using the “black Box” thinking helps you concentrate on the program’s overall design rather than the details .
    - ➔ What the function should do and how it relates to the program as a whole before worrying about writing the code.

Examples

- You have already used built-in functions such as printf() and scanf()

- You should have noticed how to invoke these functions and pass data to them.
    - ➔ Arguments between parentheses following the function name.

e.g.

- printf()
    - ➔ First argument is usually a string literal, and the succeeding arguments (of which there may be none) are a series of variables or expressions whose values are to be displayed .
- You also should have noticed how you can receive information back from a function in two ways.
    - ➔ Through one of the function arguments (scanf)
        - ★ The input is stored in an address that you supply as an argument as a return value.

Example

```
#define SIZE 50
int main(void)
{
  float list[SIZE];

  readlist(list, SIZE);
  sort(list, SIZE);
  average(list, SIZE);
  return 0;
}
```

Implementing functions

- Remember, just calling functions does not work unless we implement the function itself.
    - ➔ User defined functions
    - ➔ We would have to write the code for the three functions readlist(), sort(), and average() in our previous examples.
- Always use descriptive function names to make it clear what the program does and how it is organized.
    - ➔ If you can make the functions general enough, you can reuse them in other programs.
- In the upcoming we will understand.
    - ➔ How to define them
    - ➔ How to invoke them
    - ➔ How to pass and return data from them .

Main() Function

- As a reminder, the main() is a specially recognized name in the C system.
    - ➔ Indicates where the program is to begin execution.
    - ➔ All C programs must always have a main().

- ➔ Can pass data to it (command line arguments).
- ➔ Returning data optional (error code).