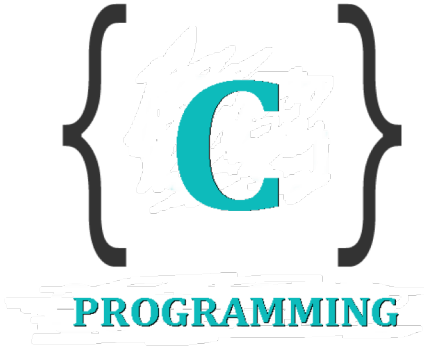


IT 105: Introduction to Programming

Functions (Introduction)

Dr. Manish Khare
Dr. Saurabh Tiwari

Lecture 13

What is a Function?

- A function is a self-contained block of statements that performs a coherent task of some kind.
- Every C program can be thought of as a collection of these functions.
- Every C program has at least one function, which is **main()**, and all the most trivial programs can define additional functions.

What is a Function?

- You can divide up your code into separate functions.
  - How you divide up your code among different functions is up to you, but logically the division is such that each function performs a specific task.
- A function **declaration** tells the compiler about a function's name, return type, and parameters.
- A function **definition** provides the actual body of the function.
- The C standard library provides numerous built-in functions that your program can call. For example, **strcat()** to concatenate two strings, **memcpy()** to copy one memory location to another location, and many more functions.

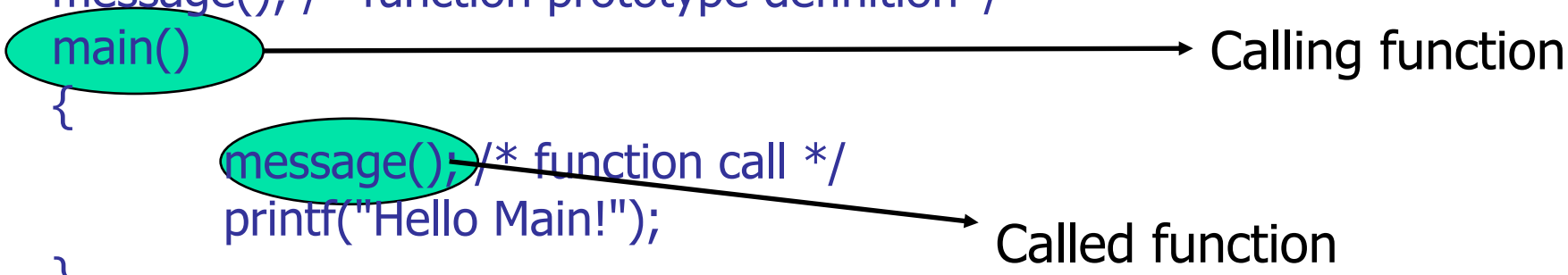
Structured Programs in C

- Structured programming is a programming technique in which a larger program is divided into smaller subprograms to make it easy to understand, easy to implement and makes the code reusable etc,. The structured programming enables code reusability. **Code reusability** is a method of writing code once and using it many times. Using structured programming technique, we write the code once and use it many times. Structured programming also makes the program easy to understand, improves the quality of the program, easy to implement and reduces time.

- In C, the structured programming can be designed using **functions** concept. Using functions concept, we can divide larger program into smaller subprograms and these subprograms are implemented individually. Every subprogram or function in C is executed individually.

What is a Function?

```
#include <stdio.h>
message(); /* function prototype definition*/
main()
{
        message(); /* function call */
        printf("Hello Main!");
}
message() /* function definition */
{
        printf("Smile! I am here\n");
}
```



Functions...

```
#include <stdio.h>
void italy();
void brazil();
void argentina();

int main()
{
        printf("I am in main\n");
        italy();
        brazil();
        argentina();
        return 0;
}
```

```
void italy()
{
        printf("I am in Italy\n");
}

void brazil()
{
        printf("I am in Brazil\n");
}

void argentina()
{
        printf("I am in
Agrentina\n");
}
```

Functions: Facts

- A C program is a collection of one or more functions
- If a C program contains only one function, it must be main()
- No limit on the number of functions in C program
  - Execution always begin with main()
- Each function in a program is called in the sequence specified y the function calls in main()
- A function can call itself [recursion]

Functions: Facts

- After each function has done its thing, control returns to main().When main( ) runs out of function calls, the program ends.
- A function gets called when the function name is followed by a semicolon.
- A function is defined when function name is followed by a pair of braces in which one or more statements may be present.
- A function can be called any number of times.
- Any function can be called from any other function. Even main( ) can be called from other functions.

Functions: Facts

- A function cannot be defined in another function. Not allowed.

```
int main()
{
        printf("I am in main\n");
        void argentina() /* user defined functions */
        {
                printf("I am in argentina\n"); /* library functions */
        }
}
```

Advantages of Functions

- Using functions we can implement modular programming.
- Functions makes the program more readable and understandable.
- Using functions the program implementation becomes easy.
- Once a function is created it can be used many times (**code re-usability**).
- Using functions larger program can be divided into smaller modules.