Functions in C

In C programming, functions play a significant role in organizing code, making it more modular, reusable, and easier to understand. Programming performance in C can be improved by dividing work into small parts, which is why it is recommended to use C functions.

1. Chapter objectives

Extra

- Acquiring knowledge on how to use functions correctly.
- Recognizing the advantages of using functions in programming.
- The ability to differentiate between different types of functions and use each one in the correct way.
- The ability to solve a problem by knowing whether this problem requires the use of functions and knowing the appropriate function to solve this problem.

2. Required tribal gains

1. **Basic Syntax**: Understand the basic syntax of a function declaration and definition. This includes the return type, function name, parameters (if any), and the function body enclosed within curly braces.
2. **Function Declaration vs. Definition**: Understand the difference between declaring a function and defining a function. Declaration tells the compiler about the existence of a function, while definition provides the actual implementation.
3. **Return Types**: Learn about different return types functions can have (void, int, float, etc.) and how to use them effectively.

3. Promotion of tribal gains

Quiz 1

[solution n°1 p. 33]

What does `main()` represent in C?

Quiz 2

[solution n°2 p. 33]

What do we mean by `main()` function

- ☐ In C programming, the main() function is an essential component of every C program.Typically, when the C program is running, the operating system calls the function to initiate the execution of the program's code.
- ☐ The main function in C **marks the beginning of any program in C**. The main function in C is the first function to be executed by the Operating System.

Quiz 3

[solution n°3 p. 33]

why we use curly braces {} in C ?

- ☐ The curly braces **denote a block of code, in which variables can be declared**.
- ☐ {and} are used to limit the scope of declarations and to act as a single statement for control structures.

4. What does it mean a function?

Definition

In C programming language, a function is a series of statements or instructions that known under one name. These statements have been grouped together to accomplish a task. A program in C can be considered as a series of functions, so that in the program we have at least one function which is "main( )" function. A function building block is a subprogram (subroutine) defined outside the main program. The benefit of using functions is that the function is defined once and can be used multiple times within the program. There is a distinction, that must be taken into account, between functions in **C library** and the functions created by the programmer [4] p.36[1] p.36 [1] p.36.

5. Steps to create a function

There are three steps to create a function which are the function declaration, function definition and function calls. These steps can be explored in the following program

Program	Output

#include <stdio.h>
void message() //function declaration
{ printf("hello world"); //function definition }
int main()
{ message(); //function call
return 0; }
	hello world

5.1. Function declaration:

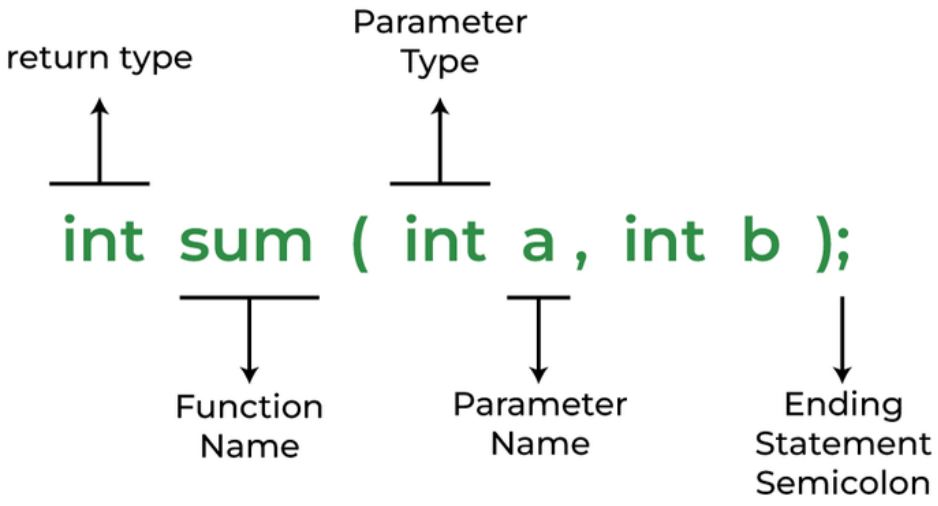
The general form to make a function declaration is

```
return_type name_of_the_function (parameter_1, parameter_2,...,....);
```

In the program above: void is the return type, message is the function name. The appropriate way to declare a function without a parameter or an argument (as we see in the previous example) is that

void name_of_the_function ( );

void is used when a function does not return any value. There are other return-types of function rather than void likewise int, float, double... ,and this depends on the type of parameters (*see the*[https://www.geeksforgeeks.org/c-functions/] p.35 Figure1).



This figure explain the function declaration step

5.2. Function definition:

The definition of a function (also called body of the function) contains the statements must be executed enclosed by a curly brackets { }.

5.3. Function call:

Declaring a function is accomplished when we write the function name followed by a semicolon ";" inside the main program. A function call is a command to the compiler to access the body of the function and start executing it.

6. Different cases of a function declaration

According to the function definition there are many ways to declare and use a function, and this due to the result of the function sub-program execution.

6.1. Function without argument (parameter) and without a return value

Display a message ? *Example*

Program	Output
#include <stdio.h> void letter() { printf("To me, you are the center of the universe. "); } int main() { letter(); return 0; }	To me, you are the center of the universe.

6.2. Function with arguments and without a return value

The sum of two numbers *Example*

Program	Output
#include <stdio.h> void sum(int a, int b) { int c; c=a+b; printf("%d",c); } int main() { sum(5,6); return 0; }	11

In this program it is worth noting that C copies the values (5,6) into the function parameters (a,b). This form (**sum(5,6);**) of parameter passing named **"call by value"**.

6.3. Function without argument and with a return value

The sum of two numbers *Example*

Program	Outputs

#include <stdio.h>
int sum()
{int a, b;
printf("enter a:\n");
scanf("%d",&a);
printf("enter b:\n");
scanf("%d",&b);
return a+b; }
int main()
{ int c;
c=sum();
printf("sum=%d",c);
return 0; }
	sum=8

There's something different about this program that we have not seen before, which is represented by this statement "**c=sum( );**" and this is because the function returns a value (**return a+b;**). Meaning that the sum( ) function is just a value and this value must be declared within a variable which is c in the main program.

6.4. Function with argument and with a return value

Average of three numbers *Example*

Program	Output
#include <stdio.h> float average(float a, float b, float c) { float d=(a+b+c)/3; return d; } int main() { float w, x=5, y=6.5, z=10; w=average(x,y,z); printf("The average of the three numbers=%f",w); return 0; }	The average of the three numbers=7.166667

The return type **float**, of the function, represents the data type that the function returns. This type of function declaration called "***function prototype***[https://www.programiz.com/c-programminghttps://www.programiz.com/c-programming p.35]". Every function prototype is a function declaration. But not every declaration is a prototype. A function declaration without specifying parameters types (or with a void) is not a prototype.

7. Recursive function

In general, a recursion is a technique used to solve a difficult problem by dividing it into connected and less difficult parts. During resolving, we move to solve a less difficult part, then to the less difficult one, and so on *until we finish*[4 p.36].

In C, recursion occurs when a function calls itself, and this function said to be recursive.

At all times, the function does not stop calling itself. In order to avoid this indefinite case, a termination condition is required, in the function definition, to stop the recursion. This condition is called **base case.**

In order for the function to call itself, it is necessary to make a statement that makes a backward change in the function argument, which is known as the **recursive case.**

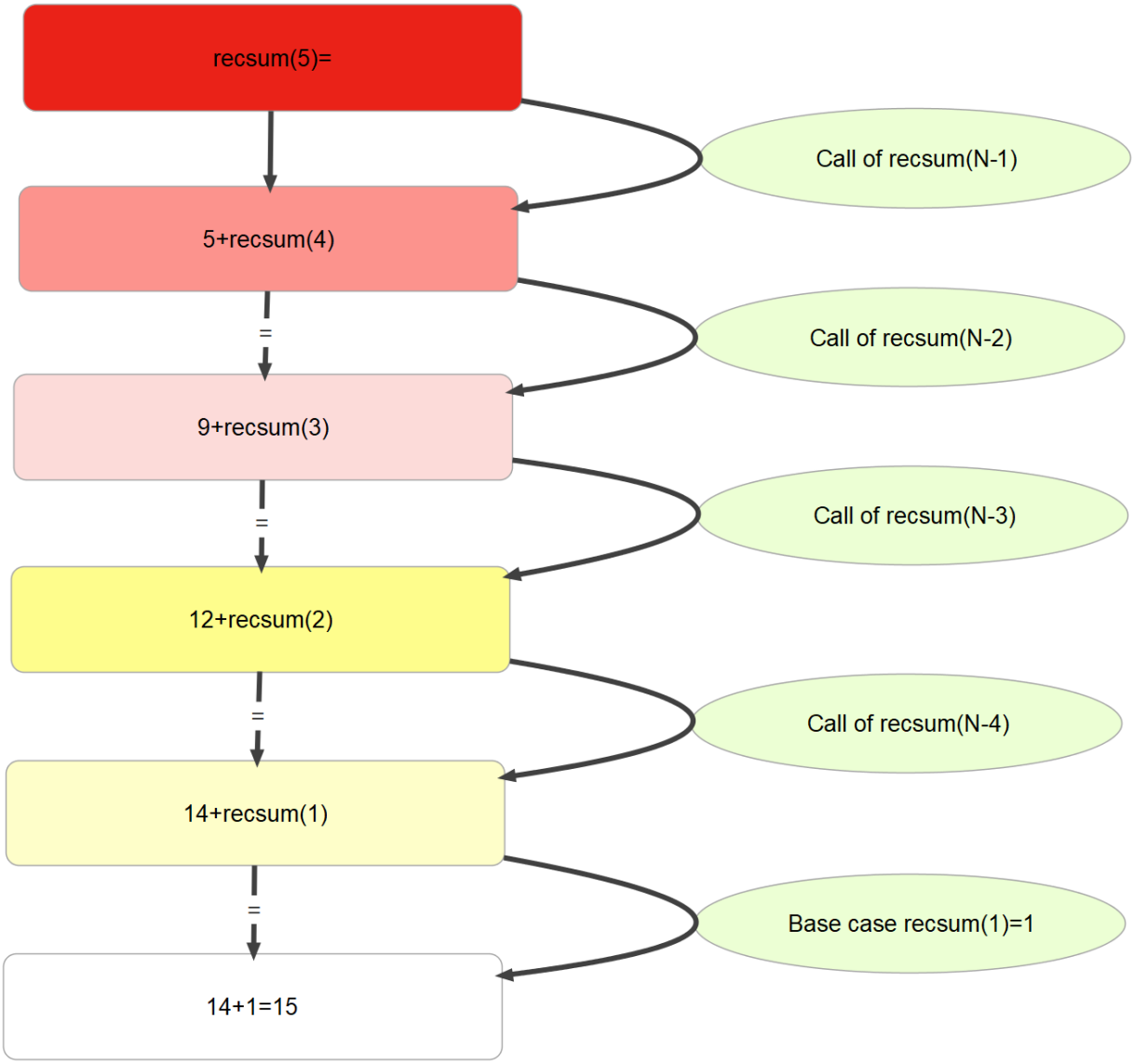
A recursive function to find the sum of numbers from 1 to N *Example*

Program	Output

#include <stdio.h>
int recsum(int N)
{ if(N==1) //base case
{ return 1; }
return N+recsum(N-1); //recursive case
}
int main()
{ int m;
	15

m=recsum(5);
printf("%d\n",m);
return 0; }	

Example explanation:



Explain how the function in the example calls itself

In the previous program, the execution was done step by step, each step representing an individual call for the function as follows:

recsum(5)=5+recsum(4)

=5+(4+recsum(3))

=5+(4+(3+recsum(2)))

=5+(4+(3+(2+recsum(1))))

=5+4+3+2+1

=15

For mor informations you can watch this video.[1]

8. Series of Exercises

You will find in the following link a series of exercises related to the subject

https://drive.google.com/file/d/1l-_dQPFSiCoh-GLve_iQU5v2s1-fsfyx/view?usp=sharing

As we conclude this exploration of C functions, let us reflect on their timeless importance and enduring relevance in the ever-evolving landscape of computer science and software engineering. From their humble beginnings in the early days of programming to their indispensable role in modern software development, functions in C continue to shape the way we write, organize, and conceptualize code, inspiring generations of programmers to push the boundaries of what is possible in the world of computing.

1. Recursion | C Programming Tutorial