**Function** - A **function** in C is a block of code that performs a **specific task**.

**Why Use Functions?**

* Reuse code (Don't Repeat Yourself).
* Divide big problems into smaller ones.
* Improve readability and debugging.
* Allow code collaboration and modularity.

**Syntax of a Function in C**

return\_type function\_name(parameter\_list) {

// body of the function

}

Example:

int add(int a, int b) {

return a + b;

}

**Types of Function:**

| Type | Example |
| --- | --- |
| With arguments, with return | int sum(int a, int b) |
| With arguments, no return | void greet(char name[]) |
| No arguments, with return | int get\_value() |
| No arguments, no return | void say\_hello() |

**Parameter passing (by value, by reference)**

1. C uses two main ways of parameter passing:

### 1. ****Pass by Value**** (Default in C)

* When you pass a variable to a function, **a copy** of the variable is passed.
* Any changes made inside the function **do not affect** the original variable.

#include <stdio.h>

void modify(int x) {

x = x + 10;

printf("Inside modify(): %d\n", x);

}

int main() {

int a = 5;

modify(a);

printf("In main(): %d\n", a);

return 0;

}

#include <stdio.h>

void modify(int x) {

x = x + 10;

printf("Inside modify(): %d\n", x);

}

int main() {

int a = 5;

modify(a);

printf("In main(): %d\n", a);

return 0;

}

### **2. **Pass by Reference** (Using Pointers)**

* You pass the **address** of the variable.
* Changes made in the function **reflect** in the original variable.

#include <stdio.h>

void modify(int \*x) {

\*x = \*x + 10;

printf("Inside modify(): %d\n", \*x);

}

int main() {

int a = 5;

modify(&a);

printf("In main(): %d\n", a);

return 0;

}

### ****Recursion in C Programming -** Recursion** is a programming technique where a function calls **itself** to solve a problem.

**Syntax of Recursion**

A recursive function must have:

1. **Base Case** – A condition that stops the recursion.
2. **Recursive Case** – Where the function calls itself.

Example: Factorial using Recursion

#include <stdio.h>

int factorial(int n) {

if (n == 0) // Base case

return 1;

else // Recursive case

return n \* factorial(n - 1);

}

int main() {

int num = 5;

printf("Factorial of %d is %d\n", num, factorial(num));

return 0;

}

**Scope** - **Scope** defines **where** a variable can be **accessed or used** in your program.

In C, the **scope of a function** is typically **global** by default. This means: **Any function defined outside of another function can be called from anywhere in the program**, **after its declaration** (if needed).

| Scope Type | Description |
| --- | --- |
| **Local** | Variable declared **inside** a function. Only accessible there. |
| **Global** | Declared **outside all functions**, accessible **anywhere** in the file. |
| **Formal parameters** | Treated as local variables in the function where they’re used. |
| **Block scope** | Declared inside {} and limited to that block only. |