**Functions in C Programming**

**1. Library Functions:**

* Predefined functions provided by C libraries (e.g., printf(), scanf(), sqrt()).
* No need to define them; they are ready to use by including appropriate header files (#include <stdio.h>, #include <math.h>).

**2. User-Defined Functions:**

* Functions created by the programmer to perform specific tasks.
* Syntax:

c

return\_type function\_name(parameters) {

// Function body

}

**3. Function Declaration:**

* Informs the compiler about the function’s name, return type, and parameters.
* Syntax:

c

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

* Example:

c

int add(int, int);

**4. Prototype Declaration:**

* Ensures that the function is declared before its actual use, enforcing type checking of arguments and return values.
* Similar to function declaration but includes the types of arguments.

**5. Types of Arguments:**

* **Actual Arguments:**  
  The values/variables passed to the function when it is called. Example:

c

add(5, 10); // 5 and 10 are actual arguments

* **Formal Arguments:**  
  Parameters listed in the function definition that receive the actual arguments. Example:

c

int add(int x, int y); // x and y are formal arguments

**6. Function Definition:**

* Contains the actual code for the function.
* Example:

c

int add(int x, int y) {

return x + y;

}

**7. Passing Arrays as Parameters:**

* Arrays can be passed as arguments to functions.
* Only the address of the array is passed, not the entire array.
* Syntax:

c

void display(int arr[], int size) {

for (int i = 0; i < size; i++) {

printf("%d ", arr[i]);

}

}

**8. Methods to Call a Function:**

* **Call by Value:**  
  Copies the value of actual arguments to the formal parameters. Changes made to the parameters do not affect the actual arguments.
  + Example:

c

void change(int x) {

x = 10;

}

* **Call by Reference:**  
  Passes the address of actual arguments. Changes made to formal arguments affect the actual arguments.
  + Example:

c

void change(int \*x) {

\*x = 10;

}