7. User-Defined Functions

- Introduction to functions
- Advantages of using functions
- Function definition and function prototype
- Function parameters and return values
- Types of user-defined functions
- Recursive functions
- Call by value and call by reference
- Passing arrays and strings to functions
- Scope rules and storage classes (basic idea)

7.1 Introduction to Function

Definitions:

- **A function is a self-contained block of code that performs a specific task or set of tasks.**

Components of a function: C provides standard streams:

- Function Name
- Return Type
- Parameters
- Functions Body

Syntax:

```
return_type function_name(parameters)
{
    //Functions body
    // statements
}
```

7.2 Advantages of Function

- **Code reusability**
- **Modularity**
- **Easier debugging**
- **Improved readability**
- **Better program maintenance**

7.3 Elements of User-Defined Function: A user-defined function consists of:

- Function prototype
- Function definition
- Function call

7.3.2 Function Prototype

- Declares function before use
- Helps compiler check correctness

Syntax:

```
int sum(int, int);
// Written before main()
```

7.3 Elements of User-Defined Function

7.3.3 Function Parameters: Used to pass data to functions

- **Types**:
  - Actual parameters
  - Formal parameters

Syntax:

```
sum(10, 20);   // actual
int sum(int a, int b) // formal
```

7.4 Storage Class

- **Defines**:
  - Scope
  - Lifetime
  - Visibility
- **Types**:
  - auto
  - register
  - static
  - extern

Summary of C Storage Classes

Storage Class	Scope	Lifetime	Storage Location	Default Initial Value
`auto`	Local (within the block/function)	Exists only while the control remains within the block	Stack Memory (RAM)	Garbage (unpredictable)
`register`	Local (within the block/function)	Exists only while the control remains within the block	CPU Register (if available)	Garbage (unpredictable)
`static`	Local (within block/function) or File (global)	Persists throughout the entire program execution	Data Segment Memory (RAM)	Zero (0)
`extern`	Global (across all program files)	Persists throughout the entire program execution	Data Segment Memory (RAM)	Zero (0)

Storage Class Example: Used to pass data to functions

- Retains value between function calls

Syntax:

```
static int count = 0;
```

7.5 Scope Rules

Scope determines where a variable is accessible.

- **Types of Scope:**
  - Local scope
  - Global scope
  - Block scope

Syntax:

```
int x = 10;  // global
```