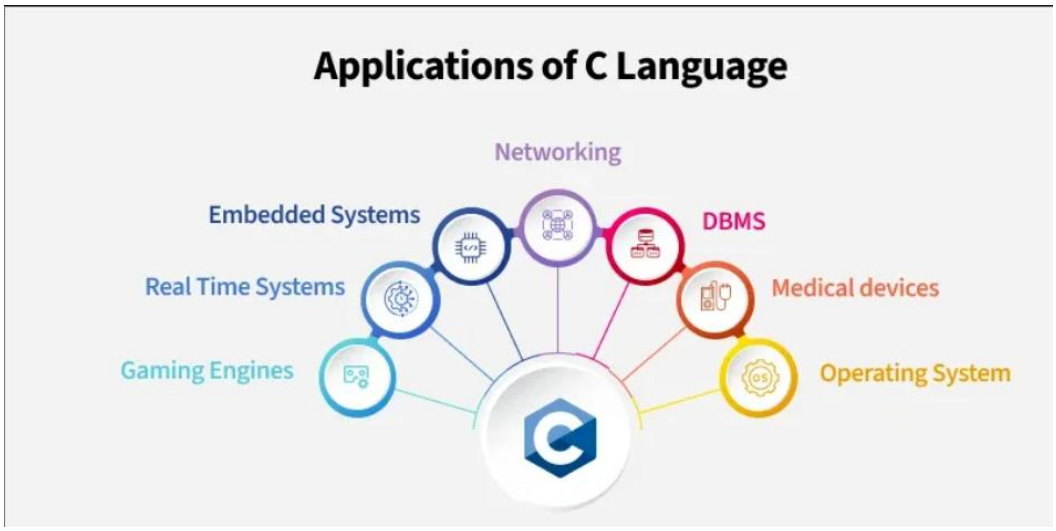
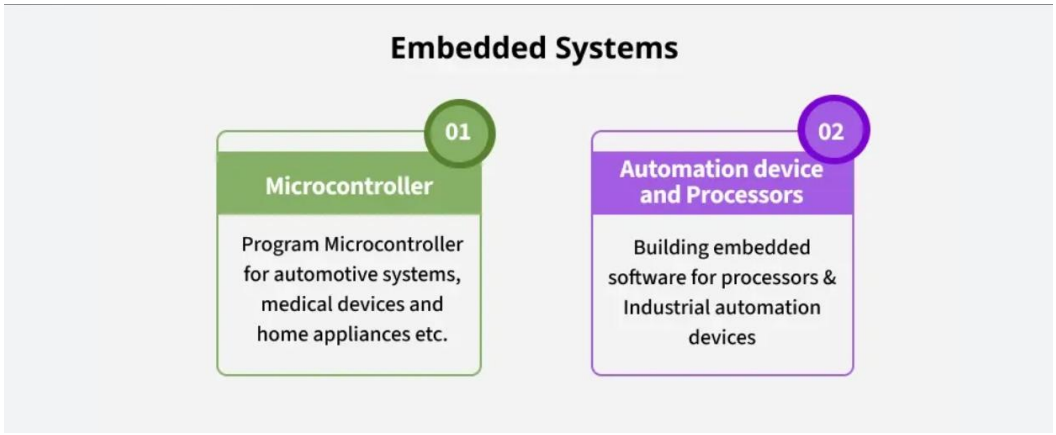
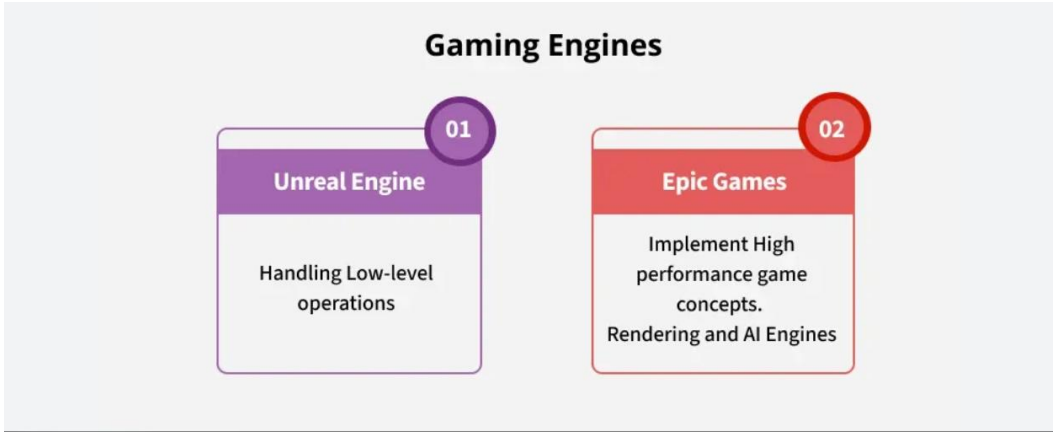
**C Programming Language Tutorial**

Last Updated : 12 Aug, 2025

C is a general-purpose mid-level programming language developed by Dennis M. Ritchie at Bell Laboratories in 1972. It was initially used for the development of UNIX operating system, but it later became popular for a wide range of applications. Today, C remains one of the top three most widely used programming languages.



## DBMS

**01 MySql**

Developed in C for high performance.

**02 Oracle**

Handle large scale data queries
Fast Data Retrieval.

## Medical devices

**01 Pacemaker**

Develop firmware for pacemaker.
Real time monitoring and control

**02 Medical Equipments**

Software for diagnostic tools and monitoring systems.

## Networking

**01 Communication Protocols**

Develop network operating Systems, communication protocols and routing protocols.

**02 Networking Hardware**

program networking devices like routers for high speed packet switching and network security systems.

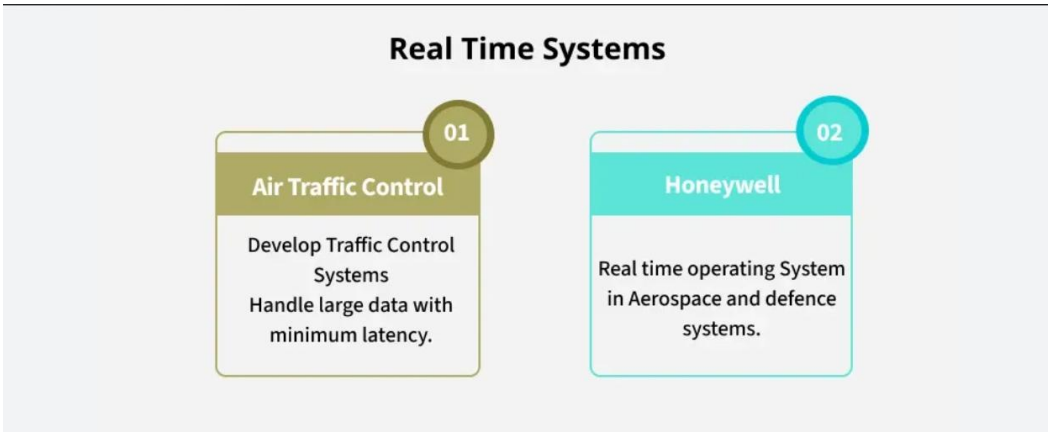
## Operating System

**01 Operating System Components**

develop the core components of operating system, such as kernel, device drivers, and system utilities

**02 Performance and Efficiency**

Low-level access to memory
Developing high-performance software for operating systems.




**Why learn C?**

C is considered as the fundamental language of computer programming. Many modern languages such as C++, Java, Python, and Go derive their syntax from C. So, if you learn C, not only you will have strong grasp of fundamental concepts but also find it easier to learn other programming languages.

Also, C is used in creating almost all of the operating systems and embedded systems. So, if you have interest in low level coding, learning C is mandatory.

**C Fundamentals**

This section of the C Tutorial includes basic concepts that build the foundation for writing C programs. It teaches you how to store and output data, perform arithmetic and other operations, control the program flow, etc.

- C Introduction
- Compilation Process
- Identifiers
- C Keywords
- C Variables
- Data Type in C
- **Quiz**: C Basics | Variables | Data Types
- Input and Output in C
- Operators
- **Quiz**: Input and Output | Operators
- Conditional Statements
- Loops
- **Quiz**: Conditional Statements and Loops

**C Functions**

**Functions** are block of code that performs a specific task. They allow programmers to write modular and reusable code.

- Functions in C
- Parameter Passing Techniques
- Main Function
- Recursion
- Inline Function
- Nested Functions
- **Quiz**: Functions

**Compound Data Types in C**

Compound data types are created from primitive data types and provides a different way to use them according to our needs. This section of C tutorial teaches you about the compound data types and how to efficiently organize and process real world data.

- C Arrays
- Pointer in C
- C Strings
- **Quiz**: Array | Pointers | Strings
- Structures
- Unions
- Enumeration (or enum)
- **Quiz**: Structure & Union

**Memory Management in C**

This section covers the manual memory management in C using pointers. The dynamic memory management in C uses functions like malloc(), calloc(), realloc(), and free() to manually manage the memory while avoiding errors such as memory leak.

- Program's Memory Layout
- Dynamic Memory Allocation
- Memory Leaks
- **Quiz**: Memory Management

**File Handling in C**

This section teaches you how to work with files in C, including creating, reading, writing, manipulating and deleting files.

- Basics of File Handling
- Read a File
- Read/Write Structure From/to a File

- EOF, getc() and feof()
- Delete a File
- **Quiz**: File Handling

**C Error Handling**

Unlike other programming languages that have automatic error handling, In C language error handling is to be manually done by the developers using error-handling methods, debugging strategies, and functions like perror(), strerror(), etc.

- Error Handling in C
- Exception Handling Using goto
- File Error Handling
- Divide by Zero Exception

**Miscellaneous C Concepts**

This section explores various essential of C language that do not fit into a single category but play a vital role in C programming and provide advanced functionality to your program.

- C Preprocessors
- Macros
- **Quiz**: Preprocessors and Macros
- C Header Files
- Date and Time
- Linkage
- Storage Classes
- **Quiz**: Storage Classes

**Advanced C**

This section teaches you high-level C programming techniques such as multi-threading, signal handling, socket programming, etc which are used in creating high-performance robust applications and systems.

- Variadic Functions
- Input-Output System Calls
- Signals
- Socket Programming
- _Generics Keyword
- Multithreading in C

**Important Links**

- Comparison of various programming languages
- Careers and Jobs in C
- Companies That Use C

**What's Next?**

Now that you have a strong foundation in C language, you can explore more advanced topics such as Data structures and Algorithms in C. Keep challenging yourself with our collection of practice problems to further strengthen you understanding of the language:

- Fundamental Practice Problems
- Functions Practice Problems
- Compound Data Types Practice Problems

You can explore advanced topics such as Data structures and Algorithms in C to solve complex problems and you can also contribute to open-source projects, or even start preparing for competitive programming contests to and gain some hands-on experience.

**Job Roles for C Programmers**

C is worth learning in 2025 as it is still used for system programming, embedded systems, operating systems, game engines, networking, and high-performance computing. C programmers can work in roles such as:

- **Embedded Systems Engineer:** Engineers who develop embedded systems, that mainly concern on hardware and software integration for devices like IoT.
- **Firmware Engineer:** Create firmware for hardware devices to control their functionality and ensure proper operation of embedded systems.
- **System Programmer:** People who work on system-level software like operating systems, device drivers.
- **Game Developer**: Designs and creates video games ensuring smooth gameplay across platforms.
- **Network Engineer**: Configure create and manage computer networks (like LANs, WANs) to ensure efficient and secure communication across the network.