1. Introduction to Programming Languages

A **programming language** is a formal language used to write instructions that a computer can understand and execute. Computers do not understand human language directly; they only understand **machine language (0s and 1s)**. Programming languages act as a bridge between humans and computers.

Need for Programming Languages

- To communicate with computers
- To solve real-world problems
- To automate tasks
- To develop software, operating systems, games, websites, etc.

Levels of Programming Languages

1. **Machine Level Language**
2. **Assembly Level Language**
3. **High-Level Language**

C language belongs to the **High-Level Programming Language** category but also supports **low-level features**, making it very powerful.

---

2. What is C Language?

C is a general-purpose, procedural, structured programming language that was developed to write system software, especially operating systems.

Definition of C Language

C is a structured, middle-level programming language that combines the features of both high-level and low-level languages.

Why C is Called a Middle-Level Language

- Supports **low-level operations** (pointers, memory access)
- Supports **high-level programming concepts** (functions, loops, structures)

---

3. History and Origin of C Language

Developer

- **Dennis Ritchie**

Year of Development

- **1972**

Place of Development

- **AT&T Bell Laboratories, USA**

Purpose

- To develop the **UNIX Operating System**

C language was developed as an improvement over existing languages to overcome their limitations.

---

4. Evolution of Programming Languages (Before C)

The evolution of C language is closely related to the development of earlier programming languages.

4.1 Machine Language (1940s)

- Uses **binary digits (0 and 1)**
- Directly understood by computer hardware

Advantages

- Fast execution
- No translator required

Disadvantages

- Very difficult to write and understand
- Error-prone
- Machine dependent

4.2 Assembly Language (1950s)

- Uses **mnemonics** like ADD, SUB, MOV

- Requires an **assembler**

Advantages

- Easier than machine language
- More readable

Disadvantages

- Machine dependent
- Complex for large programs

4.3 High-Level Languages (1960s onwards)

Examples:

- FORTRAN
- COBOL
- BASIC
- Pascal

These languages are easy to learn but lack low-level control.

---

5. Languages That Led to the Development of C

5.1 ALGOL (1960)

- Introduced **structured programming**
- Used block structures

Limitation: Not suitable for system programming

5.2 CPL (Combined Programming Language) – 1963

- Very powerful
- Designed for system programming

Limitation: Too complex and difficult to implement

5.3 BCPL (Basic Combined Programming Language) – 1967

- Developed by **Martin Richards**
- Simplified version of CPL

Limitation: Less powerful and inefficient

5.4 B Language (1970)

- Developed by **Ken Thompson**
- Derived from BCPL
- Used for early UNIX systems

Limitation:

- No data types
- Not suitable for modern hardware

---

6. Birth of C Language

Dennis Ritchie developed **C language** in **1972** by adding:

- Data types
- Structured programming features
- Improved memory handling

C was initially used to rewrite the **UNIX Operating System**, which was earlier written in Assembly Language.

---

7. Standardization of C Language

7.1 K&R C (1978)

- Written by **Brian Kernighan and Dennis Ritchie**
- Book: *"The C Programming Language"*
- Became the unofficial standard

7.2 ANSI C (C89 / C90)

- Standardized by **ANSI (American National Standards Institute)**
- Year: **1989**
- Improved portability

7.3 ISO C

- Adopted by **ISO (International Organization for Standardization)**
- Ensured global standardization

7.4 C99

- Introduced:
  - New data types
  - Inline functions
  - Variable-length arrays

7.5 C11

- Improved security
- Multithreading support

---

8. Features of C Language

8.1 Simple and Efficient

- Easy syntax
- Fast execution

8.2 Structured Programming

- Uses functions
- Breaks program into modules

8.3 Portability

- Same code can run on different machines with little modification

8.4 Middle-Level Language

- Supports both high-level and low-level features

8.5 Rich Library

- Built-in functions like printf(), scanf()

8.6 Pointer Support

- Direct memory access
- Efficient memory management

8.7 Extensible

- New functions can be added

---

9. Applications of C Language

C language is widely used in many fields:

1. **Operating Systems** (UNIX, Linux)
2. **Embedded Systems**
3. **Compilers and Interpreters**
4. **Device Drivers**
5. **Game Development**
6. **Database Systems**
7. **Networking Programs**
8. **Real-Time Systems**

---

10. Importance of C Language

- Foundation for learning other languages like:
  - C++
  - Java
  - Python
- Helps understand:
  - Memory management
  - Data structures
  - Operating systems
- Widely accepted in academics and industry

---

11. Advantages of C Language

- Fast execution
- Close to hardware
- Reusable code
- Powerful and flexible
- Large community support

---

12. Limitations of C Language

- No Object-Oriented features
- No runtime error checking
- Manual memory management
- No built-in security features

---

13. C Language vs Other Languages

Feature	C	C++	Java
Speed	Very Fast	Fast	Slower
OOP	No	Yes	Yes
Portability	High	High	Very High
Memory Control	Manual	Manual	Automatic

---

14. Conclusion

C language is one of the most **powerful and influential programming languages** in computer science history. Its simplicity, efficiency, and flexibility make it ideal for system programming. Understanding the **introduction and evolution of C language** helps students build a strong foundation in programming concepts.