# ✅ UNIT – I : C Programming (Complete Notes)

---

## 1. Overview of C Language

### Introduction

C is a **general-purpose, procedural programming language** developed by **Dennis Ritchie in 1972** at Bell Laboratories.
It is widely used for **system programming, application development, embedded systems**, and **operating systems**.

### Features of C Language

- Simple and efficient
- Portable (machine independent)
- Structured language
- Rich set of operators
- Supports pointers
- Fast execution

### Applications of C

- Operating systems (UNIX, Linux)
- Embedded systems
- Compilers and interpreters
- Game development
- Device drivers

### Advantages

- Easy to learn
- Fast execution
- Memory control using pointers

---

## 2. C Language Elements

### Basic Elements

1. Keywords
2. Identifiers
3. Constants
4. Variables

5. Operators
6. Special symbols

**Example**

int a = 10;

- int → keyword
- a → variable
- 10 → constant

---

**3. Variable Declarations and Data Types**

**Variable Declaration**

Used to reserve memory.

**Syntax:**

data_type variable_name;

**Example:**

int age;

float salary;

---

**Data Types in C**

| Type | Size | Example |
|---|---|---|
| int | 4 bytes | 10 |
| float | 4 bytes | 10.5 |
| double | 8 bytes | 99.99 |
| char | 1 byte | 'A' |

**Derived Data Types**

- Arrays
- Pointers
- Structures
- Unions

---

**4. Executable Statements**

Executable statements perform actions during program execution.

**Types**

- Assignment statements
- Input/Output statements
- Control statements

**Example:**

```
sum = a + b;

printf("%d", sum);
```

---

**5. General Form of a C Program**

**Structure**

1. Documentation section
2. Link section
3. Definition section
4. Global declaration
5. main() function
6. User-defined functions

**Example Program**

```
#include <stdio.h>


int main() {

   int a = 10, b = 20;

   printf("Sum = %d", a + b);

   return 0;

}
```

---

**6. Arithmetic Expressions**

**Operators**

| **Operator** | **Meaning** |
|---|---|
| + | Addition |

| **Operator** | **Meaning** |
|---|---|
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulus |

**Example**

```
result = (a + b) * c;
```

---

## 7. Formatting Numbers in Output

**Using printf()**

```
printf("%d", num);     // integer
printf("%f", x);       // float
printf("%.2f", x);     // 2 decimal places
```

**Example**

```
printf("Value = %.2f", 12.3456);
```

Output:

```
Value = 12.35
```

---

## 8. Selection Structures

**Control Structures**

Used to control the flow of execution.

Types:

- Selection
- Repetition
- Jump

---

## 9. Conditions

Used with relational and logical operators.

**Relational Operators**

```
> < >= <= == !=
```

**Logical Operators**

&& (AND), || (OR), ! (NOT)

---

**10. if Statement**

**Syntax**

```
if(condition)
{
  statements;
}
```

**Example**

```
if(a > b)
   printf("A is greater");
```

---

**11. if–else Statement**

```
if(condition)
{
  statements;
}
else
{
  statements;
}
```

---

**12. Compound Statements**

Multiple statements grouped using { }.

```
if(a > b)
{
  c = a;
  printf("%d", c);
}
```

---

**13. Decision Steps in Algorithms**

**Flow Diagram (Text Form)**

```
Start
  |
Check Condition
  |
True → Execute Block
False → Skip Block
  |
 End
```

---

**14. Repetition in Programs**

Repetition allows execution of statements repeatedly.

Types:

- while loop
- for loop
- do-while loop

---

**15. Counting Loops and while Statement**

**Syntax**

```
while(condition)
{
  statements;
}
```

**Example**

```
int i = 1;
while(i <= 5)
{
   printf("%d ", i);
   i++;
}
```

---

## 16. Computing Sum or Product in Loop

**Sum Example**

```
int sum = 0;
for(i = 1; i <= 5; i++)
   sum += i;
```

**Product Example**

```
int product = 1;
for(i = 1; i <= 5; i++)
   product *= i;
```

---

## 17. for Statement

**Syntax**

```
for(initialization; condition; increment)
{
  statements;
}
```

**Example**

```
for(i = 1; i <= 10; i++)
   printf("%d ", i);
```

---

## 18. Conditional Loops

Loops that run based on conditions:

- while
- do-while

---

## 19. Loop Design

Steps:

1. Initialize variable
2. Set condition
3. Execute loop body

4. Update value

---

**20. Nested Loops**

Loop inside another loop.

**Example**

```
for(i = 1; i <= 3; i++)
{
  for(j = 1; j <= 2; j++)
  {
    printf("* ");
  }
  printf("\n");
}
```

---

**21. do–while Statement**

**Syntax**

```
do {
  statements;
} while(condition);
```

**Example**

```
int i = 1;
do {
  printf("%d ", i);
  i++;
} while(i <= 5);
```

---

**Difference: while vs do-while**

| while | do-while |
|---|---|
| Condition checked first | Condition checked last |
| May not execute | Executes at least once |

---

**Important Keywords for Exam**

- Control Structures
- Conditional Statements
- Looping Statements
- Iteration
- Initialization
- Termination Condition

---

**Conclusion**

C language provides strong control structures such as **selection and repetition** to solve complex problems efficiently. Understanding **loops, conditions, and program structure** is essential for writing optimized and logical programs.

---