C Language Complete Notes

---

1. What is a Program?

A **program** is a set of instructions written to perform a specific task by the computer. It can be written in various languages like C, Python, Java, etc. C is one of the oldest and most powerful languages used for system programming.

Simple Example:

```c
#include <stdio.h>
int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Explanation: This program prints "Hello, World!" to the console using the `printf()` function.

Practice Example:

```c
#include <stdio.h>
int main() {
    int a = 5, b = 10;
    int sum = a + b;
    printf("Sum = %d", sum);
    return 0;
}
```

---

2. Tokens in C

Tokens are the smallest units of a C program. They include:

- **Keywords:** Reserved words like `int`, `return`, `if`
- **Identifiers:** Names of variables, functions, arrays (e.g., `main`, `sum`)
- **Constants:** Fixed values like `10`, `3.14`, `'A'`
- **String Literals:** Sequences of characters in double quotes (`"Hello"`)
- **Operators:** Symbols that perform operations (e.g., `+`, `-`, `*`)
- **Special Symbols:** `;`, `{}`, `()`, `[]`, `#`

Example Program Using Tokens:

```
int a = 10;          // 'int', 'a', '=', '10', ';' are tokens
printf("%d", a);     // 'printf', '(', '"%d"', ',', 'a', ')', ';'
```

Practice Example:

```
int x = 20, y = 15;
int z = x + y;
printf("%d", z);  // Output: 35
```

---

3. Elements of Programming

Elements are the building blocks of a program. They include:

- **Variables:** Named memory locations to store values
- **Constants:** Values that do not change
- **Operators:** Symbols used to perform operations
- **Data Types:** Specify the type of data (int, float, etc.)
- **Control Structures:** if, switch, loops
- **Functions:** Reusable blocks of code
- **Arrays and Structures:** Data collection formats

Example:

```
int add(int x, int y) {
    return x + y;
}

int main() {
    int result = add(5, 3);
    printf("Result = %d", result);
    return 0;
}
```

---

4. Variables

Variables are containers for storing data values.

Syntax:

```
<data_type> <variable_name> = <value>;
```

Examples:

```
int age = 25;             // integer variable
float salary = 45000.50;  // float variable
char grade = 'A';         // character variable
```

Multiple Declaration:

```
int a = 5, b = 10, c;
```

Practice Example:

```
char initial = 'R';
int roll = 23;
float percent = 85.6;
printf("%c %d %.2f", initial, roll, percent);
```

---

5. Data Types

C supports the following basic data types:

Data Type	Description	Example Value
int	Integer	10
float	Single-precision decimal	3.14
double	Double-precision decimal	3.1415926535
char	Single character	'A'

Example:

```
int a = 10;
float b = 5.5;
double pi = 3.14159;
char ch = 'C';
```

Practice Example:

```c
int x = 7;
float y = 2.5;
double area = x * y;
char grade = 'B';
printf("Area = %.2lf, Grade = %c", area, grade);
```

---

6. Operators

a. Arithmetic Operators

```
+, -, *, /, %
```

Example:

```c
int a = 10, b = 3;
printf("%% = %d\n", a % b); // prints 1
```

Practice: Calculate the square and cube of a number.

```c
int num = 4;
printf("Square = %d, Cube = %d", num * num, num * num * num);
```

b. Relational Operators

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

Example:

```c
if (a > b) printf("a is greater\n");
```

Practice: Compare two numbers for equality.

```c
int x = 7, y = 7;
if (x == y) printf("Equal\n"); else printf("Not equal\n");
```

c. Logical Operators

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

Example:

```
if (a > 0 && b > 0)
    printf("Both positive\n");
```

Practice: Check if a number lies between 10 and 20.

```
int n = 15;
if (n >= 10 && n <= 20) printf("In range\n");
```

d. Assignment Operators

```
=, +=, -=, *=, /=, %=
```

Example:

```
a += 5; // a = a + 5
```

Practice: Update values using assignment operators.

```
int val = 10;
val *= 2;
val -= 5;
printf("%d", val);
```

e. Bitwise Operators

```
&, |, ^, ~, <<, >>
```

Example:

```
int x = 5;   // 0101
int y = 3;   // 0011
printf("AND = %d", x & y);  // Output: 1
```

Practice:

```c
int a = 6; // 0110
int b = 1;
int shift = a << b;  // Left shift
printf("Shifted = %d", shift);  // Output: 12
```