C Programming Language - Complete Notes

1. Introduction to C

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

Basic Structure of a C Program - Detailed Explanation

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Let's understand each part:

1. #include <stdio.h>

- This is a **preprocessor directive**
- It tells the compiler to include the standard input-output library
- stdio.h contains declarations for printf(), scanf(), and other I/O functions
- The # symbol indicates it's processed before compilation
- Without this, you can't use printf() or scanf()

2. int main()

- **int** - return type of the function (returns an integer)

- **main** - name of the function; every C program must have exactly one main() function
- **()** - parentheses can contain parameters; empty means no parameters
- This is the **entry point** of the program - execution starts here

3. { } - Curly Braces

- These define the **body** of the function
- All code inside these braces belongs to the main() function
- Every opening brace { must have a closing brace }

4. printf("Hello, World!\n");

- **printf()** - function to print output to the screen
- **"Hello, World!"** - string to be printed (enclosed in double quotes)
- **\n** - escape sequence for newline (moves cursor to next line)
- **;** - semicolon marks the end of a statement (required!)

5. return 0;

- Returns the value 0 to the operating system
- 0 typically means the program executed successfully
- Non-zero values indicate errors
- Must match the return type of main() (int)

> **How the Program Executes:**
>
> 1. Preprocessor includes stdio.h library
> 2. Compiler compiles the code into machine language
> 3. Linker links the object code with libraries
> 4. Operating system loads the program into memory
> 5. Execution starts from main() function
> 6. printf() displays "Hello, World!" on screen
> 7. return 0 sends success code to OS
> 8. Program terminates

> 🎯 **Practice Questions:**
>
> 1. Write a program to print your name, age, and city on separate lines
> 2. Write a program that prints "Welcome to C Programming" five times

3. Create a program that displays a simple pattern using asterisks (*)

2. Data Types

Data types specify the type of data a variable can store. C has several built-in data types:

Data Type	Size	Range	Format Specifier
char	1 byte	-128 to 127	%c
unsigned char	1 byte	0 to 255	%c
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	%d or %i
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295	%u
float	4 bytes	1.2E-38 to 3.4E+38 (6 decimal places)	%f
double	8 bytes	2.3E-308 to 1.7E+308 (15 decimal places)	%lf
long int	4 or 8 bytes	-2,147,483,648 to 2,147,483,647 or larger	%ld

🎯 **Practice Questions:**

1. Write a program to find the size of different data types using sizeof() operator
2. Create a program that stores your age (int), height (float), and grade (char) and displays them
3. Write a program to calculate the area of a circle (use double for precision)

3. Variables and Constants

Variable Declaration and Initialization

```
// Declaration
int age;

// Initialization
age = 25;

// Declaration + Initialization
int age = 25;
float salary = 50000.50;
char grade = 'A';
double pi = 3.14159265359;
```

Constants

```
// Using #define (preprocessor)
#define PI 3.14159
#define MAX_SIZE 100

// Using const keyword
const int MAX_VALUE = 100;
const float GRAVITY = 9.8;
```

🎯 **Practice Questions:**

1. Write a program to swap two numbers using a third variable
2. Calculate simple interest using principal, rate, and time as variables
3. Convert temperature from Celsius to Fahrenheit

4. Input and Output

Taking Integer Input

```
int num;
printf("Enter a number: ");
scanf("%d", &num);  // & is the address-of operator
```

Taking String Input

```
// Method 1: Using scanf (reads until space)
char name[50];
scanf("%s", name);

// Method 2: Using fgets (recommended - safe)
fgets(name, 50, stdin);

// Method 3: Reading line with spaces
scanf("%[^\n]", name);  // Reads until newline

// Method 4: Reading with spaces using scanf
char name[50];
scanf(" %[^\n]s", name);  // Space before % consumes previous n
```

Complete Input/Output Example:

```
#include <stdio.h>

int main() {
    char name[50];
    int age;
    float height;

    printf("Enter your name: ");
    scanf("%[^\n]", name);  // Reads name with spaces

    printf("Enter your age: ");
```

```
    scanf("%d", &age);

    printf("Enter your height (in meters): ");
    scanf("%f", &height);

    printf("\n--- Your Details ---\n");
    printf("Name: %s\n", name);
    printf("Age: %d years\n", age);
    printf("Height: %.2f meters\n", height);

    return 0;
}
```

🎯 **Practice Questions:**

1. Write a program to take two numbers as input and display their sum, difference, product, and quotient
2. Create a program that takes length and breadth of a rectangle and calculates area and perimeter
3. Write a program to convert days into years, weeks, and remaining days

5. Operators in Detail

5.1 Arithmetic Operators

Operator	Name	Example	Result
+	Addition	5 + 3	8
-	Subtraction	5 - 3	2
*	Multiplication	5 * 3	15
/	Division	5 / 2	2 (integer division)

%	Modulus (remainder)	5 % 2	1

5.2 Relational Operators

Used to compare two values. Returns 1 (true) or 0 (false)

Operator	Name	Example	Result (if a=5, b=3)
==	Equal to	a == b	0 (false)
!=	Not equal to	a != b	1 (true)
>	Greater than	a > b	1 (true)
<	Less than	a < b	0 (false)
>=	Greater than or equal	a >= b	1 (true)
<=	Less than or equal	a <= b	0 (false)

5.3 Logical Operators

Operator	Name	Example	True When
&&	Logical AND	a && b	Both a and b are true (non-zero)
||	Logical OR	a || b	At least one is true
!	Logical NOT	!a	a is false (zero)

5.4 Bitwise Operators

Operate on individual bits of integers

Operator	Name	Example	Description
&	Bitwise AND	a & b	Sets bit to 1 if both bits are 1

|	Bitwise OR	a | b	Sets bit to 1 if any bit is 1
^	Bitwise XOR	a ^ b	Sets bit to 1 if bits are different
~	Bitwise NOT	~a	Inverts all bits
<<	Left shift	a << 2	Shifts bits left by 2 (multiplies by 4)
>>	Right shift	a >> 2	Shifts bits right by 2 (divides by 4)

Bitwise Operators Example:

```
#include <stdio.h>

int main() {
    int a = 5;   // Binary: 0101
    int b = 3;   // Binary: 0011

    printf("a & b = %d\n", a & b);   // 0001 = 1
    printf("a | b = %d\n", a | b);   // 0111 = 7
    printf("a ^ b = %d\n", a ^ b);   // 0110 = 6
    printf("~a = %d\n", ~a);         // Inverts all bits
    printf("a << 1 = %d\n", a << 1); // 1010 = 10
    printf("a >> 1 = %d\n", a >> 1); // 0010 = 2

    return 0;
}
```

5.5 Assignment Operators

Operator	Example	Equivalent To
=	a = 5	a = 5
+=	a += 3	a = a + 3

-=	a -= 3	a = a - 3
*=	a *= 3	a = a * 3
/=	a /= 3	a = a / 3
%=	a %= 3	a = a % 3
&=	a &= 3	a = a & 3
|=	a |= 3	a = a | 3

5.6 Increment and Decrement Operators

```
int a = 5;

// Post-increment: use value first, then increment
int b = a++;  // b = 5, a = 6

// Pre-increment: increment first, then use value
int c = ++a;  // a = 7, c = 7

// Post-decrement
int d = a--;  // d = 7, a = 6

// Pre-decrement
int e = --a;  // a = 5, e = 5
```

5.7 Ternary Operator (Conditional Operator)

Syntax: `condition ? value_if_true : value_if_false`

```
int a = 10, b = 20;
int max = (a > b) ? a : b;  // max = 20

// Another example
int age = 18;
```

```
char* status = (age >= 18) ? "Adult" : "Minor";

// Nested ternary
int x = 5;
char* result = (x > 0) ? "Positive" : (x < 0) ? "Negative" : "Z
```

Ternary Operator Program:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    // Check even or odd
    (num % 2 == 0) ? printf("%d is Even\n", num) : printf("

    // Find maximum of three numbers
    int a = 15, b = 25, c = 20;
    int max = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b :
    printf("Maximum: %d\n", max);

    return 0;
}
```

🎯 **Practice Questions:**

1. Write a program to check if a number is even or odd using ternary operator
2. Find the largest of three numbers using ternary operator
3. Check if a year is leap year or not using bitwise operators
4. Swap two numbers without using a third variable (use bitwise XOR)
5. Write a program to count number of set bits (1s) in a number

6. Control Structures

6.1 If-Else Statement

```
if (condition) {
    // code executes if condition is true
} else if (another_condition) {
    // code executes if another_condition is true
} else {
    // code executes if all conditions are false
}
```

Example: Grade Calculator

```
#include <stdio.h>

int main() {
    int marks;
    printf("Enter marks: ");
    scanf("%d", &marks);

    if (marks >= 90) {
        printf("Grade: A+\n");
    } else if (marks >= 80) {
        printf("Grade: A\n");
    } else if (marks >= 70) {
        printf("Grade: B\n");
    } else if (marks >= 60) {
        printf("Grade: C\n");
    } else if (marks >= 50) {
        printf("Grade: D\n");
    } else {
        printf("Grade: F\n");
    }
```

```
    return 0;
}
```

6.2 Switch Statement

```
switch (variable) {
    case value1:
        // code
        break;
    case value2:
        // code
        break;
    default:
        // code if no case matches
}
```

Example: Simple Calculator

```
#include <stdio.h>

int main() {
    char op;
    double num1, num2;

    printf("Enter operator (+, -, *, /): ");
    scanf("%c", &op);

    printf("Enter two numbers: ");
    scanf("%lf %lf", &num1, &num2);

    switch(op) {
        case '+':
            printf("%.2lf + %.2lf = %.2lf\n", num1, num2, n
            break;
        case '-':
```

```
            printf("%.2lf - %.2lf = %.2lf\n", num1, num2, n
            break;
        case '*':
            printf("%.2lf * %.2lf = %.2lf\n", num1, num2, n
            break;
        case '/':
            if(num2 != 0)
                printf("%.2lf / %.2lf = %.2lf\n", num1, num
            else
                printf("Error: Division by zero!\n");
            break;
        default:
            printf("Invalid operator!\n");
    }

    return 0;
}
```

🎯 **Practice Questions:**

1. Write a program to check if a number is positive, negative, or zero
2. Check if a person is eligible for voting (age >= 18)
3. Find the greatest of three numbers
4. Check if a triangle is valid (sum of any two sides > third side)
5. Create a menu-driven program using switch for basic arithmetic operations

7. Loops

7.1 For Loop

Used when you know the number of iterations in advance

```
for (initialization; condition; increment/decrement) {
    // code to repeat
```

```
}

// Example
for (int i = 0; i < 10; i++) {
    printf("%d ", i);
}
```

Example: Multiplication Table

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    printf("Multiplication table of %d:\n", num);
    for(int i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", num, i, num * i);
    }

    return 0;
}
```

7.2 While Loop

Used when you don't know the number of iterations; checks condition first

```
initialization;
while (condition) {
    // code to repeat
    increment/decrement;
}
```

Example: Sum of Digits

```c
#include <stdio.h>

int main() {
    int num, sum = 0, digit;
    printf("Enter a number: ");
    scanf("%d", &num);

    int temp = num;
    while(temp > 0) {
        digit = temp % 10;  // Get last digit
        sum += digit;
        temp /= 10;  // Remove last digit
    }

    printf("Sum of digits of %d = %d\n", num, sum);
    return 0;
}
```

7.3 Do-While Loop

Executes at least once; checks condition at the end

```c
initialization;
do {
    // code to repeat
    increment/decrement;
} while (condition);
```

Example: Menu-Driven Program

```c
#include <stdio.h>

int main() {
    int choice;
```

```
    do {
        printf("\n--- MENU ---\n");
        printf("1. Print Hello\n");
        printf("2. Print Goodbye\n");
        printf("3. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                printf("Hello, World!\n");
                break;
            case 2:
                printf("Goodbye!\n");
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice!\n");
        }
    } while(choice != 3);

    return 0;
}
```

🎯 **Practice Questions:**

1. **Factorial:** Find factorial of a number
2. **Fibonacci Series:** Print first n Fibonacci numbers (0, 1, 1, 2, 3, 5, 8...)
3. **Prime Number:** Check if a number is prime
4. **Armstrong Number:** Check if a number is Armstrong ($153 = 1^3 + 5^3 + 3^3$)
5. **Palindrome:** Check if a number is palindrome (121, 12321 are palindromes)
6. **Reverse Number:** Reverse the digits of a number
7. **GCD/HCF:** Find GCD of two numbers using Euclidean algorithm
8. **LCM:** Find LCM of two numbers

9. **Perfect Number:** Check if a number is perfect (6 = 1+2+3)
10. **Sum of Series:** Calculate sum 1 + 1/2 + 1/3 + ... + 1/n

Solution: Armstrong Number

```
#include <stdio.h>
#include <math.h>

int main() {
    int num, temp, remainder, sum = 0, digits = 0;

    printf("Enter a number: ");
    scanf("%d", &num);

    temp = num;

    // Count digits
    while(temp != 0) {
        digits++;
        temp /= 10;
    }

    temp = num;

    // Calculate sum of digits raised to power of number of
    while(temp != 0) {
        remainder = temp % 10;
        sum += pow(remainder, digits);
        temp /= 10;
    }

    if(sum == num)
        printf("%d is an Armstrong number\n", num);
    else
        printf("%d is not an Armstrong number\n", num);
```

```
    return 0;
}
```

Solution: Palindrome Number

```
#include <stdio.h>

int main() {
    int num, temp, reverse = 0, remainder;

    printf("Enter a number: ");
    scanf("%d", &num);

    temp = num;

    // Reverse the number
    while(temp != 0) {
        remainder = temp % 10;
        reverse = reverse * 10 + remainder;
        temp /= 10;
    }

    if(reverse == num)
        printf("%d is a palindrome\n", num);
    else
        printf("%d is not a palindrome\n", num);

    return 0;
}
```

8. Functions

Functions are reusable blocks of code that perform specific tasks

```
// Function declaration (prototype)
return_type function_name(parameter_list);

// Function definition
return_type function_name(parameter_list) {
    // function body
    return value;
}

// Function call
function_name(arguments);
```

Example: Prime Number Function

```
#include <stdio.h>

// Function to check if a number is prime
int isPrime(int n) {
    if(n <= 1)
        return 0;  // false

    for(int i = 2; i * i <= n; i++) {
        if(n % i == 0)
            return 0;  // false
    }
    return 1;  // true
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    if(isPrime(num))
        printf("%d is a prime number\n", num);
    else
        printf("%d is not a prime number\n", num);
```

```
    return 0;
}
```

🎯 **Practice Questions:**

1. Write a function to calculate factorial of a number
2. Create a function to check if a number is Armstrong or not
3. Write a function to find GCD of two numbers
4. Create a function to swap two numbers using call by reference (pointers)
5. Write a function to convert decimal to binary

9. Arrays

9.1 One-Dimensional Array

Collection of elements of the same data type stored in contiguous memory

```
// Declaration
int numbers[5];

// Declaration with initialization
int numbers[5] = {1, 2, 3, 4, 5};

// Partial initialization
int numbers[5] = {1, 2};  // Rest are 0

// Size determined automatically
int numbers[] = {1, 2, 3, 4, 5};

// Accessing elements (index starts from 0)
int first = numbers[0];
numbers[2] = 10;
```

Example: Find Largest Element

```c
#include <stdio.h>

int main() {
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    int
```