MENTORBRO LEARNING LLP,
Email: mentorbroofficial@gmail.com
Ph no : + 91 8157867616

Mentor Bro Notes

📚 Topics in C Programming

---

1. Introduction to C 🌟

- 🤔 What is C?
- 📜 History and Features of C
- 🛠 Applications of C
- ✍️ Writing Your First C Program

---

2. Structure of a C Program 🏗

- 📥 `#include`, main() Function
- 📄 Statements and Blocks
- 🖥 Compiling and Executing a C Program

---

3. Data Types and Variables 🔢

- 🔤 **Basic Data Types**: `int`, `float`, `char`, `double`, etc.
- 📝 **Declaring and Initializing Variables**
- 🚫 **Constants** (`const` and `#define`)

---

4. Operators in C ➕➖✖️

- 🤝 Arithmetic, Relational, Logical, and Bitwise Operators
- 🔄 Increment and Decrement
- 📦 Assignment and Compound Assignment Operators
- 🧮 Operator Precedence and Associativity

---

5. Control Flow Statements 🔀

- ✅ **if**, **else if**, **else**
- 🔄 **switch Statements**
- 🔁 **Loops**: `for`, `while`, and `do...while`
- ⏹️ **break** and ⏭️ **continue**

---

6. Functions ⚙️

- 🛠 **Function Declaration, Definition, and Calling**
- 📤 **Passing Arguments by Value and by Reference**
- 🔄 **Recursion**

---

7. Arrays and Strings 📋 🔤

- 🧩 **Single and Multidimensional Arrays**
- 🧵 **String Handling Using Arrays**
- 🛠️ Common String Functions: `strlen`, `strcpy`, `strcmp`, etc.

---

8. Pointers 🎯

- 📍 **Basics of Pointers**
- ➕ **Pointer Arithmetic**
- 🔗 **Pointers and Arrays**
- ⚡ **Pointers to Functions**

---

9. Structures and Unions 🏗️

- 📁 **Declaring and Using Structures**
- 🗂️ **Array of Structures**
- ❓ **Difference Between Structures and Unions**

---

10. File Handling 📂✍️

- 📖 **Opening, Reading, and Writing Files**
- 🔖 **Modes**: `r`, `w`, `a`, etc.
- 🛠 **Working with** `fscanf`, `fprintf`, `fgetc`, and `fputc`

---

11. Dynamic Memory Allocation 🧠

- 💾 **malloc**, **calloc**, **realloc**, and **free**
- ⚠️ **Memory Management and Pitfalls**

---

12. Preprocessor Directives 📜

- 🖊️ **Macros**, `#define`, and `#include`
- ⚙️ **Conditional Compilation**: `#if`, `#else`, `#endif`

---

13. Error Handling 🐛

- ❌ **Types of Errors**: Syntax, Runtime, and Logical
- 🛠 **Debugging Tips**

---

14. Advanced Topics 🚀

- 🎯 **Function Pointers**
- 🖥️ **Command-Line Arguments**
- 🔗 **Working with Linked Lists**
- 🧮 **Introduction to Data Structures in C**

📌 Important topics with explanation for beginners

1. Algorithm 🧠

An **algorithm** is a step-by-step method to solve a problem logically and systematically.
It is the foundation for coding because it defines the flow before implementation.

Example Algorithm: Finding the Sum of Two Numbers
1️⃣ Start
2️⃣ Input two numbers (A, B)
3️⃣ Calculate the sum: `Sum = A + B`
4️⃣ Display the sum
5️⃣ End

Advantages of Algorithms ✅

- 🛠 **Systematic Approach**: Ensures problem-solving clarity.
- 🧩 **Logic Building**: Helps break problems into manageable steps.
- 🔍 **Ease of Debugging**: Errors can be identified in logical steps.

---

2. Flowchart 🔄

A **flowchart** is a graphical representation of an algorithm using symbols. It visualizes the steps in a process, making logic easier to understand.

Flowchart Symbols 🛑 📊

Symbol	Meaning
🟢 **Oval**	Start/End
🟦 **Rectangle**	Process (e.g., calculations)
🔷 **Diamond**	Decision (Yes/No, True/False)
➡️ **Arrow**	Flow Direction

Flowchart Example for Adding Two Numbers

```
[🟢 Start]
     ↓
[⬇️ Input A, B]
     ↓
[📊 Sum = A + B]
     ↓
[📊 Display Sum]
     ↓
  [🟢 End]
```

---

3. Conditional Statements 🔀

Conditional statements allow a program to execute specific blocks of code based on conditions.

Types of Conditional Statements in C

1️⃣ If Statement
Executes code only if the condition is true.

```
if (x > 0) {
    printf("Positive number");
}
```

2️⃣ If-Else Statement
Executes one block for true and another for false.

```
if (x > 0) {
    printf("Positive");
} else {
    printf("Non-positive");
}
```

3️⃣ If-Else Ladder
Evaluates multiple conditions sequentially.

```
if (x > 0) {
    printf("Positive");
} else if (x < 0) {
    printf("Negative");
} else {
    printf("Zero");
}
```

4️⃣ **Nested If**
Contains an if inside another if.

```
if (x > 0) {
    if (x % 2 == 0) {
        printf("Positive Even");
    }
}
```

5️⃣ **Switch Case**
Handles multiple fixed conditions.

```
switch (choice) {
    case 1: printf("Option 1"); break;
    case 2: printf("Option 2"); break;
    default: printf("Invalid");
}
```

---

4. Loops in C 🔁

Loops allow repetition of a block of code multiple times.

Types of Loops

1️⃣ **While Loop** 🔄
Executes while the condition is true.

```
int i = 0;
while (i < 5) {
    printf("%d ", i);
    i++;
}
```

2️⃣ For Loop 🎯
Executes a fixed number of times.

```
for (int i = 0; i < 5; i++) {
    printf("%d ", i);
}
```

3️⃣ Do-While Loop 🔃
Executes at least once, checking the condition after the first run.

```
int i = 0;
do {
    printf("%d ", i);
    i++;
} while (i < 5);
```

---

5. Arrays 📋

An **array** is a collection of elements of the same type stored in contiguous memory locations.

Types of Arrays

1️⃣ 1D Array
A single row of elements.

```
int arr[5] = {1, 2, 3, 4, 5};
```

- **Access**: `arr[2]` returns `3`.

2️⃣ 2D Array
Elements stored in rows and columns (matrix).

```
int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

- **Access**: `arr[1][2]` returns `6`.

6. Variables and Data Types 💾

A **variable** is a named location in memory to store data.
Data types define the type of data a variable can hold.

Data Type	Description	Example
🔢 **int**	Stores whole numbers	`int x = 10;`
💧 **float**	Stores decimal numbers	`float y = 3.14;`
🔤 **char**	Stores a single character	`char c = 'A';`
📝 **char[]**	Stores strings	`char name[] = "John";`

7. Pointers 🎯

A **pointer** is a variable that stores the memory address of another variable.

Example

```
int a = 10;
int *ptr = &a; // Pointer storing the address of 'a'
```

- **ptr** points to **a**'s address.
- Access **a**'s value indirectly using `*ptr`.

Feature	Variable	Pointer
Stores	A value	Address of a variable
Access	Directly	Indirectly via `*`

8. Functions ⚙️

A **function** is a reusable block of code that performs a specific task.

Types of Functions

- 📚 **Built-in Functions**: Predefined, like `printf()` and `scanf()`.
- 🛠 **User-Defined Functions**: Created by programmers.

Example Function

```
int add(int a, int b) {
    return a + b;
}
```

Summary Table

Concept	Key Points
🧠 **Algorithm**	Logical steps to solve a problem
🔄 **Flowchart**	Visual representation of an algorithm
🔀 **Conditional**	Control logic (`If-Else`, `Switch`)
🔁 **Loops**	Repeat actions (`For`, `While`, `Do-While`)
📋 **Arrays**	1D and 2D collections of the same data types
💾 **Variables**	Store values in memory
🎯 **Pointers**	Store addresses of variables
⚙️ **Functions**	Reusable blocks of code

This document provides a **complete understanding of C concepts** with theory, examples, differences, and comparisons. 🚀