Date: Today

C Language Tutorial

Notes by Kamal Kishor..

Important Setup / Ingredients (The Recipe)

- ☑ Code Editor: VS Code (Visual Studio Code). → Like a notebook for our code.
- ☑ Compiler: MinGW (GCC). → The Translator! Converts Code to Machine Language.

Source Code (.c) Compiler (MinGW/GCC) Machine Code (.exe) -> Output

⚠ Don't forget to set the Environment Path variable in System Settings!

Chapter 1: Variables, Constants & Keywords

The Analogy

Memory is like a kitchen container.
The container is the **Variable**,
the content is the **Data**.

Definitions & Rules

- ✓ **Definition:**
  Variable: Name of a memory location which stores data.
- ✓ **Naming Rules:**
  - ✓ Case sensitive (a ≠ A).
  - ✓ 1st char must be Alphabet or _.
  - ✓ No commas or blank spaces.
  - ✓ No symbols other than _.

Data Types

Type	Size	Description & Examples
int	2 bytes	Integers (1, 2, -5)
char	1 byte	Characters ('a', '$')
float	4 bytes	Real values (3.14, 2.5)

Keywords: 32 reserved words like **int**, **return**, 'etc', 'if'. Cannot be used as variable names!

Program Structure: Hello World

```c
#include <stdio.h>  // Preprocessor Directive

int main() {
    printf("Hello World");  // Output
    return 0;
}
```

Header File (Necessary Library)

Execution starts here (The entry point)

Statement Terminator (Like a full stop)

Single line comment (Ignored by compiler)

Output:

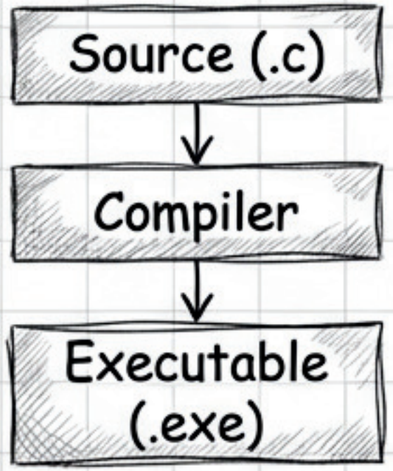
```c
printf("Age is %d", age);
```

Note: %d for int, %f for float, %c for char. \n for new line.

Input:

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

The & (ampersand) is crucial! It represents the Address.



Chapter 2: Instructions & Arithmetic

Type Declaration

✓ **Valid:**

```
int a = 22;
int a = b = c = 4;
```

✗ **Invalid:**

```
int a = b;
```

(if b is not defined yet)

Rule: Declare variable before using it!

Arithmetic Instructions

+ (Add) **-** (Subtract) ***** (Multiply) **/** (Divide)

The Modulo Operator (%)

Returns the Remainder.

Example: 5 % 2 = 1

⚠ Modulo % does **NOT** work on **float** values!

Note: Sign depends on numerator: -5 % 2 = -1

Type Conversion Logic

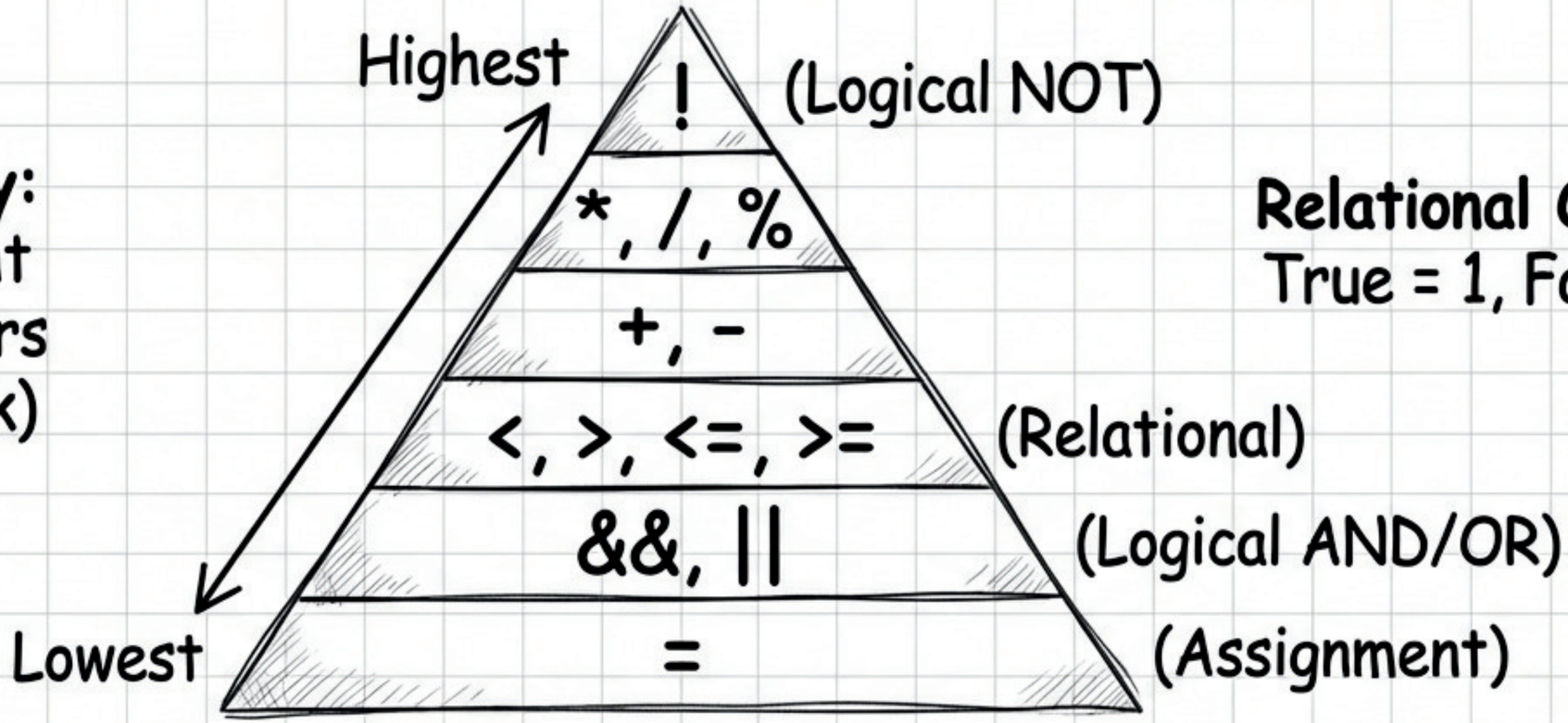
int op int ⟶ int (e.g., 2 / 3 = 0)

int op float → float (e.g., 2.0 / 3 = 0.66)

int op float → float (e.g., 2.0 / 3 = 0.66)

Explicit Cast: (int) 1.9999 becomes 1 (Decimals are dropped, not rounded!)

Operator Control & Precedence



Associativity:
Left to Right
(for operators
of same rank)

Relational Output:
True = 1, False = 0

AND (`&&`)	
T	F
T && T =	T
Any F =	F

OR (`||`)	
F	F
F || F =	F
Any T =	T

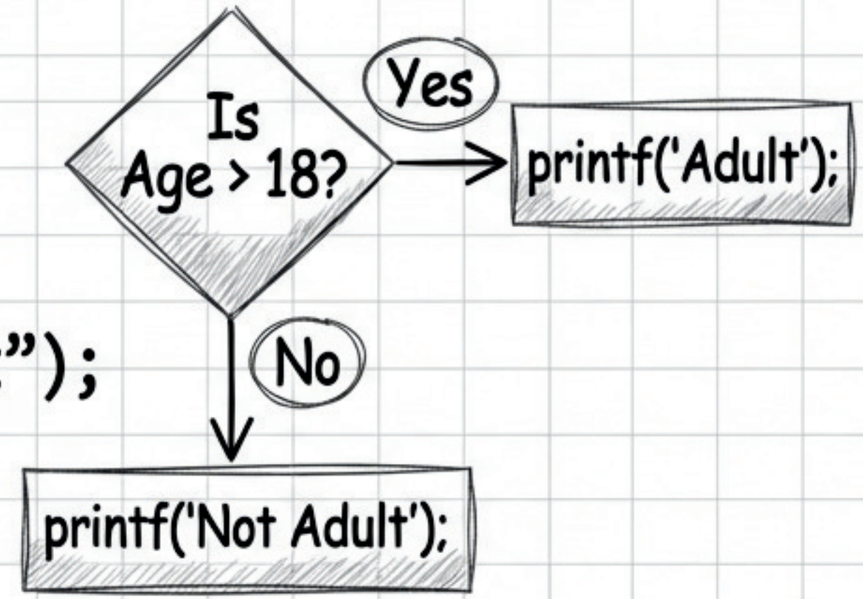
NOT (`!`)	
!	F
!T =	F
!F =	T

Short-hand Assignment:
a = a + b is same as a += b

Chapter 3: Conditional Statements

If-Else:

```
if (age > 18) {
   printf("Adult");
} else {
   printf("Not Adult");
}
```



Else-If:

Used for multiple conditions
(e.g., Grading System: A, B, C).

Ternary Operator (Magic One-Liner):

Condition ? TrueStatement : FalseStatement;

```
(age >= 18) ? printf("Vote") : printf("Cannot Vote");
```

Switch Case

```
switch(number) {
   case 1: ... break;
   default: ...
}
```

Key Properties

* Cases can be in any order.
* Nested switches allowed.

! Only int or char allowed in switch! No floats.

Chapter 4: Loop Control (Basics)

Loop: Used to repeat parts of the program.

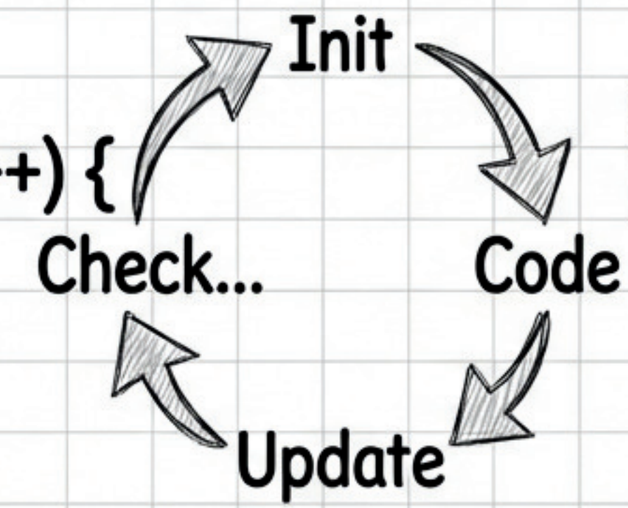
The Operators:

Increment: ++i (Pre: Change then use) vs i++ (Post: Use then change).

Decrement: --i vs i--.

For Loop:

```
for(initialisation; condition; update) {
   ...
}
for(int i=0; i<=10; i++) {
   printf("%d", i);
}
```



While Loop:

```
while(condition) {
   ...
}
```

Checks condition before entry.

Do-While Loop:

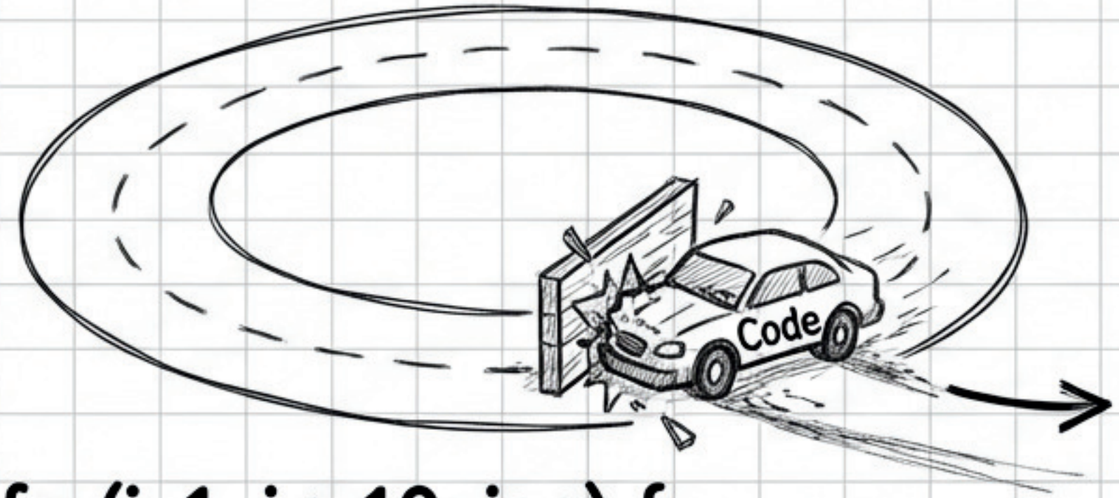
```
do {
   ...
} while(condition);
```

Executes AT LEAST ONCE even if condition is false!

Advanced Loop Control: Break & Continue

The Break Statement

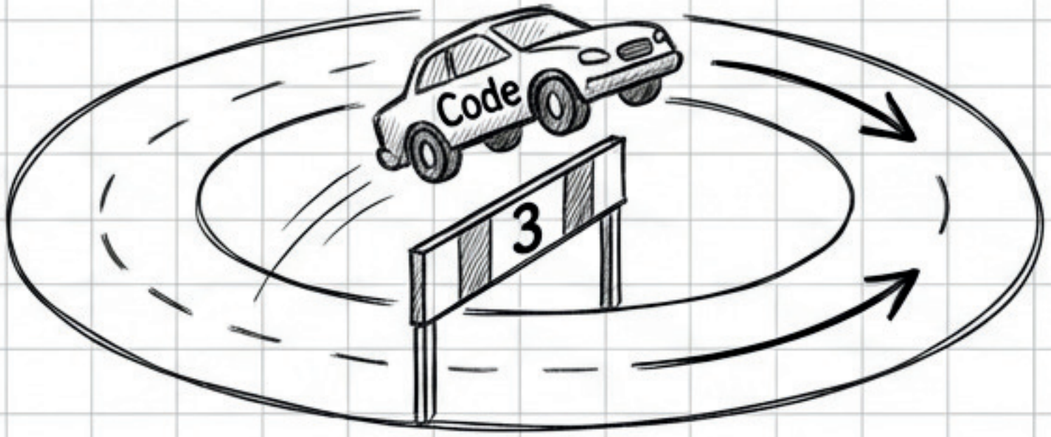
Action: Exits the loop immediately.




```
for(i=1; i<=10; i++) {
    if(i==3) break;  // Stops at 3
    printf("%d", i);
}
```

The Continue Statement

Action: Skips current iteration, goes to next.




```
for(i=1; i<=5; i++) {
    if(i==3) continue; // Skips 3
    printf("%d", i);    // 1 2 4 5
}
```

Practice Qs: 1. Print Factorial of N. 2. Print Reverse Table of N.

Chapter 5: Functions & Recursion

Functions

Definition: Block of code that performs a particular task.

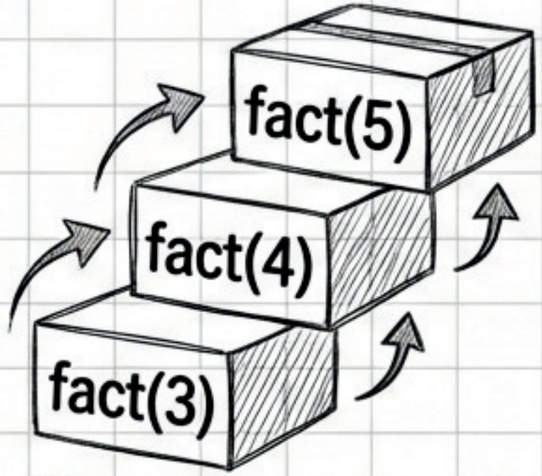
The 3 Steps:

1. Prototype: void printHello(); (Declaration)
2. Call: printHello(); (Inside main)
3. Definition: void printHello() { printf("Hello"); } (The Body)

Concept: Parameters (Formal) vs Arguments (Actual Values passed).

Recursion

Definition: When a function calls itself.



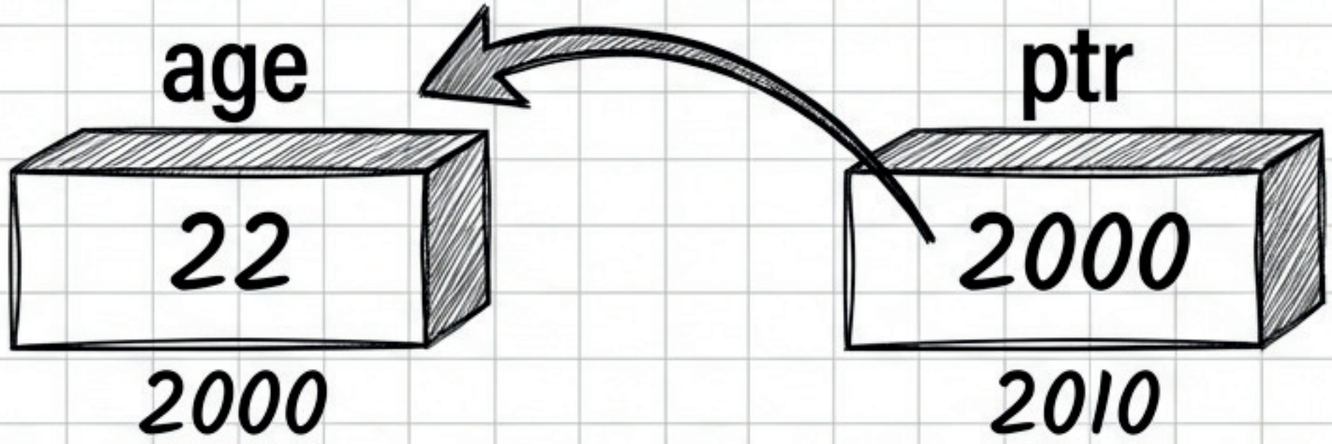
Base Case is mandatory!
Without it → Stack Overflow
Without it → Stack Overflow
(Infinite Recursion).

Note: Execution always starts at main().

Formula: Factorial(n) = n * Factorial(n-1)

Chapter 6: Pointers

A variable that stores the **memory address** of another variable.



Syntax Reference

`*` (Asterisk): "Value at Address" operator.
`&` (Ampersand): "Address of" operator.

Declaration: `int *ptr = &age;`

Format Specifier: `%p` (for printing pointers).

Advanced Concepts

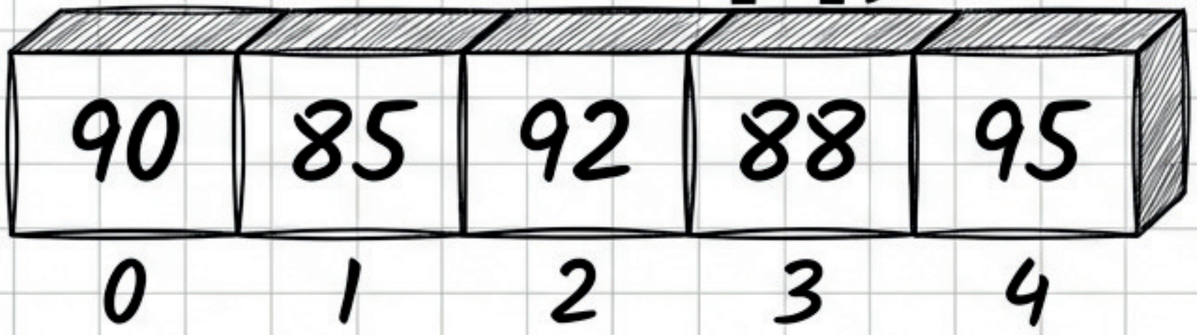
Pointer to Pointer: `int **pptr;`
(Stores address of a pointer).

Functions:
Call by Value: Passes a copy.
Call by Reference: Passes the address (changes affect original).

Chapter 7: Arrays

Definition: Collection of similar data types stored at **contiguous** memory locations.

`int marks[5];`

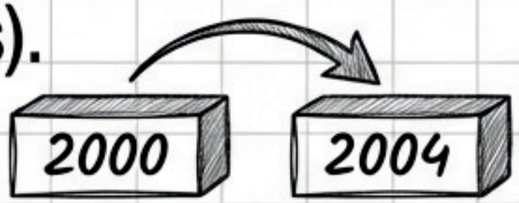


Syntax & Usage

Init: `int marks[] = {90, 85, 92};`
Input: `scanf("%d", &marks[0]);`
Traversal: Use Loops (for loop 0 to n).

Pointer Arithmetic

ptr++ : "Increments address by size of data type."
Example: int (4 bytes).
If **ptr** is 2000, **ptr++** becomes 2004.



2D Arrays (Visualized as Matrix)

int arr[2][3];

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]

Chapter 8: Strings

Definition: A character array terminated by a `\0` (Null Character).

Syntax

Declaration: char name[] = "SHRADHA";
Format Specifier: %s

String Library

strlen(str): Count length (excludes `\0`).
strcpy(new, old**):** Copies value.
strcat(first, sec**):** Concatenates (Joins).

String Library Functions (<string.h>)

strlen(str): Count length (excludes `\0`).
strcpy(new, old): Copies value.
strcat(first, sec): Concatenates (Joins).
strcmp(s1, s2): Compares (Returns 0 if equal).

Input Note

scanf´ stops at space!
Use: `**fgets(str, n, stdin)´** for multi-word input.

Chapter 9: Structures

Definition: Collection of values of **dissimilar** data types.

`struct Student`

Name: Shradha (String)

Roll No: 1664 (Int)

CGPA: 9.2 (Float)

Code Syntax

```
struct Student {
    char name[100];
    int roll;
    float cgpa;
};
```

Operations

Declaration: `struct Student s1;`
Initialization: `s1.roll = 1664;`
Access: Use the dot operator .' (e.g., `s1.name`).
Arrow Operator (->): Used with pointers to structures (e.g., `ptr->roll`).

Typedef

Creates an alias (nickname) for types to save typing effort.

Chapter 10: File I/O

RAM is **volatile** (Data lost on restart). **Files** store data **persistently.**

Key Operations

File Pointer: FILE *fptr;

Open: fptr = fopen("filename.txt", "mode");

Mode	Description
"r"	Read (Open to read)
"w"	Write (Overwrites or creates new)
"a"	Append (Adds to end)

Close: fclose(fptr); (Always close!)

Functions

Reading: fscanf, fgetc (char).
Writing: fprintf, fputc (char).
EOF: "End Of File" constant.

Summary & Next Steps

What we learned:

- ✓ Variables & Data Types
- ✓ Instructions & Operators
- ✓ Conditionals (If/Switch)
- ✓ Loops (For/While)
- ✓ Functions & Recursion
- ✓ Pointers & Memory
- ✓ Arrays & Strings
- ✓ Structures & File I/O

Practice makes progress! Try solving the 100 practice questions attached to the course. Keep Coding!

~ Kamal Kishor.....