C Language

Complete Revision Module

Beginner to Advanced · Exam Oriented

Contents

1 Basics of C

1.1 History & Structure

C was developed by **Dennis Ritchie** at Bell Labs in 1972. It is a procedural, structured language often called the "Mother of all languages."

```
// STRUCTURE OF A C PROGRAM
#include <stdio.h>          // 1. Preprocessor Directive
                            // 2. Global Declarations (optional)

int main() {                // 3. Main Function (Entry Point)
    int a = 10;             // 4. Variable Declaration
    printf("Hi");           // 5. Executable Statement
    return 0;               // 6. Return Statement
}
```

1.2 Data Types & Variables

Type	**Keyword**	**Size (Approx)**	**Format Specifier**
Integer	`int`	2 or 4 bytes	`%d`
Floating Pt	`float`	4 bytes	`%f`
Character	`char`	1 byte	`%c`
Double	`double`	8 bytes	`%lf`
Void	`void`	0 bytes	-

Table 1: Primary Data Types

1.3 Operators

- **Arithmetic:** `+, -, *, /, % (modulo)`
- **Relational:** `==, !=, >, <, >=, <=`
- **Logical:** `&& (AND), || (OR), ! (NOT)`
- **Bitwise:** `&, |, ^, <<, >>, ~`
- **Ternary:** `condition ? true_val : false_val;`

Note: Input/Output

`printf("Format string", vars);` used for output.
`scanf("Format string", &vars);` used for input. **Don't forget the &!**

Practice Q: Write a program to swap two numbers without using a third variable.

2 Control Flow

2.1 Conditional Statements

Switch Case Syntax

```
switch(expression) {
    case constant1:
        // code
        break;
    case constant2:
        // code
        break;
    default:
        // default code
}
```

2.2 Loops

- **while:** Entry controlled. Checks condition first.
- **do-while:** Exit controlled. Runs at least once.
- **for:** `for(initialization; condition; update)`

Jump Statements:

- `break`: Exits the loop/switch immediately.
- `continue`: Skips current iteration, goes to next.

3 Functions

Functions allow code modularity and reusability.

3.1 Types of Parameter Passing

Call by Value	Call by Reference
Value of variable is passed. Changes in function do **not** affect original. Memory created for new variables.	Address of variable is passed. Changes **do** affect original. Pointer holds the address.

3.2 Storage Classes

- **auto:** Default for local variables. Stack memory.
- **register:** Stored in CPU register (fast access).
- **static:** Preserves value between function calls.
- **extern:** Global visibility across files.

4 Arrays & Strings

4.1 Memory Layout (1D Array)

```
int arr[5] = {10, 20, 30, 40, 50};
Index:    [0]   [1]   [2]   [3]   [4]
Address:  1000  1004  1008  1012  1016  (Assuming 4 bytes int)
```

4.2 Strings

Strings are `char` arrays terminated by a null character `'\0'`.

- `strlen(s)`: Length of string.
- `strcpy(d, s)`: Copy s to d.
- `strcat(d, s)`: Concatenate s to d.
- `strcmp(s1, s2)`: Compare strings (returns 0 if equal).

Practice Q: Write a program to check if a string is a Palindrome.

5 Structures & Unions

Feature	Structure (struct)	Union (union)
Keyword	`struct`	`union`
Memory	Sum of size of all members	Size of largest member
Access	All members active simultaneously	Only one member active at a time
Use Case	Storing complex records	Memory saving / hardware access

```c
struct Student {
    int id;
    char name[20];
};
struct Student s1;
s1.id = 1;            // Dot operator
struct Student *ptr = &s1;
ptr->id = 1;          // Arrow operator for pointers
```

6 File Handling

Operations: `fopen`, `fclose`, `fprintf`, `fscanf`, `fgetc`, `fputc`.

6.1 File Modes

- `"r"`: Read (File must exist).
- `"w"`: Write (Creates new or truncates existing).
- `"a"`: Append (Adds to end).
- `"rb", "wb"`: Binary modes.

Practice Q: Copy contents from `source.txt` to `dest.txt`.

7 Advanced Topics

7.1 Preprocessor

- `#define PI 3.14` (Macro)
- `#ifdef`, `#ifndef`, `#endif` (Conditional Compilation)

7.2 Typedef & Enumeration

- `typedef`: Creates an alias. `typedef unsigned long ulong;`
- `enum`: Named integer constants. `enum Color {RED, GREEN, BLUE};`

7.3 Bitwise Manipulation

Setting a bit: `num | (1 << pos)`
Clearing a bit: `num & ~(1 << pos)`
Toggling a bit: `num ^(1 << pos)`

8 Data Structures Overview

- **Linked List:** Dynamic size, easy insertion/deletion. Nodes connected via pointers.
- **Stack:** LIFO (Last In First Out). Operations: Push, Pop.
- **Queue:** FIFO (First In First Out). Operations: Enqueue, Dequeue.

```
LINKED LIST NODE:
struct Node {
    int data;
    struct Node* next;
};
[ Data | Next ] -> [ Data | Next ] -> NULL
```

9 Compilation Process

1. **Preprocessing (.c -> .i):** Expands macros, includes headers.
2. **Compilation (.i -> .s):** Converts to Assembly code.
3. **Assembly (.s -> .o):** Converts to Machine code (Object file).
4. **Linking (.o -> .exe):** Links libraries, generates executable.

10 Best Practices

- **Naming:** Use meaningful variable names (camelCase or snake_case).
- **Modularity:** Break code into small functions.
- **Comments:** Explain *why*, not just *what*.
- **Memory Safety:** Always check if `malloc` returns NULL. Always `free`.
- **Indentation:** Use consistent spacing (4 spaces or 1 tab).