UNIT I · LESSON 01

C Fundamentals

Tokens · data types · operators · precedence · I/O · control flow

20
questions in Unit I
C · Python · C++ / OOP

BASICS STRUCTURE OF A C PROGRAM

```c
#include <stdio.h>   /* preprocessor   */
#define PI 3.14159    /* macro constant */
int g = 10;           /* global         */
int main(void) {      /* entry point    */
    int x = 5;        /* local          */
    printf("%d\n", x + g);
    return 0;         /* 0 = success    */
}
```

- Steps: **edit** → **preprocess** → **compile** → **assemble** → **link** → **load**
- Files: `.c` source → `.i` → `.s` → `.o` object → `a.out` / `.exe`
- C is **case-sensitive**; every statement ends with `;`

C developed by **Dennis Ritchie** at Bell Labs (1972) for UNIX; ANSI C = C89, then C99, C11.

6 KINDS TOKENS

Token	Examples
Keywords (32 in C89)	int, auto, static, goto, sizeof
Identifiers	sum, _count, x1
Constants	25, 3.5e2, 'A', 0x1F, 017
String literals	"SIPCOT" (ends with `\0`)
Operators	+ - * / % ++ && ?:
Special symbols	{ } [ ] ( ) ; , #

- Identifier: letters, digits, `_`; **cannot start with a digit**; no keywords
- `sizeof` is an **operator**, not a function
- Octal starts with `0`, hex with `0x`; `017` = 15

IMPLICIT & CAST TYPE CONVERSION

- **Integer promotion**: char/short → int in expressions
- Usual arithmetic conversion: int → unsigned → long → float → double → long double
- `int/int` stays int: `5/2*2.0 = 4.0`, but `5/2.0*2 = 5.0`
- Explicit cast: `(float)a/b` casts **a** first (cast binds tighter than `/`)
- Assigning double to int **truncates**: `int i = 3.99;` → 3

SIZE DATA TYPES

Type	Typical size	Format	Range
char	1 B	%c	-128 … 127
unsigned char	1 B	%c	0 … 255
short	2 B	%hd	-32768 … 32767
int	4 B (2 B on 16-bit)	%d	-2^{31} … $2^{31}-1$
long	4/8 B	%ld	≥ 32 bits
long long	8 B	%lld	≥ 64 bits
float	4 B	%f	6–7 digits
double	8 B	%lf	15–16 digits
long double	10/12/16 B	%Lf	≥ double

- Range of n-bit signed: **-2^{n-1} to $2^{n-1}-1$**; unsigned: **0 to 2^n-1**
- `void` = no value; derived types: arrays, pointers, structures, unions, functions
- Rule: sizeof(char)=1 ≤ short ≤ int ≤ long ≤ long long

LITERALS CONSTANTS & ESCAPE SEQUENCES

- `const int x = 5;` read-only variable; `#define X 5` text substitution (no type)
- `'A'` char constant = 65 (int in C); `"A"` string = 2 bytes
- `5u` unsigned, `5L` long, `5.0f` float, `5.0` double by default

Escape	Meaning	Escape	Meaning
\n	newline	\t	tab
\0	null	\\	backslash
\r	carriage ret.	\a	bell
\'	quote	\b	backspace

GROUPS OPERATORS

- **Arithmetic** `+ - * / %` — `%` only on integers; sign of `%` follows the **dividend**: `-7 % 3 = -1`
- **Integer division truncates**: `7/2 = 3`, `-7/2 = -3`

GROUPS OPERATORS (CONTD.)

- **Relational** `< > <= >= == !=` give 1 or 0
- **Logical** `&& || !` — **short-circuit**: right side skipped if result known
- **Bitwise** `& | ^ ~ << >>`; `x<<n` = $x \cdot 2^n$; `x>>n` = $x/2^n$ (unsigned)

GROUPS OPERATORS (CONTD.)

- **Assignment** `= += -= *= /= %= &= |= ^= <<= >>=`
- **Increment** `++x` (change then use), `x++` (use then change)
- **Conditional** `c ? a : b` — the only **ternary** operator
- **Comma** `(a=1, b=2, a+b)` → value of last expression = 3
- **sizeof**, `&` address, `*` dereference, `.` and `->` member, `(type)` cast

MUST KNOW OPERATOR PRECEDENCE & ASSOCIATIVITY (HIGH → LOW)

#	Operators	Assoc.	#	Operators	Assoc.
1	() [] -> . postfix ++ --	L→R	9	^	L→R
2	! ~ ++ -- unary + - * & (type) sizeof	**R→L**	10	|	L→R
3	* / %	L→R	11	&&	L→R
4	+ -	L→R	12	||	L→R
5	<< >>	L→R	13	?:	**R→L**
6	< <= > >=	L→R	14	= += -= ... (assignment)	**R→L**
7	== !=	L→R	15	, (comma)	L→R
8	&	L→R			

Memory hook: **"Unary → Arithmetic → Shift → Relational → Equality → Bitwise (& ^ |) → Logical (&& ||) → Ternary → Assignment → Comma"**. Only unary, ternary and assignment are right-to-left.

STDIO.H INPUT / OUTPUT

Function	Purpose
printf / scanf	formatted output / input
getchar / putchar	one character
gets / puts	string (gets unsafe — removed in C11)

STDIO.H INPUT / OUTPUT (CONTD.)

Function	Purpose
fgets(s,n,stdin)	safe line input

- `scanf("%d", &x);` needs **address**; for strings `scanf("%s", s);` (no &, stops at space)

STDIO.H INPUT / OUTPUT (CONTD.)

- `printf` returns **number of characters printed**; `scanf` returns **items read**
- `%5d` width 5, `%-5d` left-justify, `%.2f` 2 decimals, `%e` exponent, `%x` hex, `%o` octal, `%u` unsigned, `%p` pointer, `%%` percent

CONTROL DECISION MAKING

```c
if (m >= 90) grade = 'A';
else if (m >= 75) grade = 'B';
else grade = 'C';

switch (ch) {      /* int/char, no float */
  case 'a': case 'e': v++; break;
  default : c++;   /* optional, anywhere */
}
```

- Without `break`, control **falls through** to the next case
- `case` labels must be **constant integer expressions**, unique
- Dangling else: `else` pairs with the **nearest unmatched if**

ITERATION LOOPS

Loop	Test	Runs at least once?
while	entry-controlled	No
for	entry-controlled	No
do ... while	**exit-controlled**	**Yes**

- `for(;;)` and `while(1)` are infinite loops
- `break` exits innermost loop/switch; `continue` skips to next iteration (in `for`, the update still runs)
- `goto label;` unconditional jump (avoid); `return` exits function
- `exit(0)` (stdlib.h) ends the whole program

```c
for (i=1, s=0; i<=10; i++) s += i; /* 55 */
do { n /= 10; d++; } while (n);  /* digits */
```

TRAP EXAM TRAPS

- `if (x = 5)` assigns → always true; compare with `==`
- `x = 5; y = x++ + ++x;` → **undefined behaviour** (modified twice)
- `printf("%d", 10/3*3)` → **9**, not 10
- `1 && 0 || 1` → 1 (&& binds tighter)
- `char c = 255; printf("%d", c);` → **-1** on signed char
- `for (i=0; i<5; i++);` stray `;` makes the body empty
- `sizeof('A')` = 4 in C (int) but 1 in C++
- `while (i < 10)` with no update → infinite loop
- `-5 >> 1` is implementation-defined (usually -3)
- `unsigned u = -1;` → largest unsigned (4294967295)

QUICK BIT TRICKS & NUMBER SYSTEMS

Task	Expression
Check odd	`n & 1`
Set bit k	`n | (1<<k)`
Clear bit k	`n & ~(1<<k)`
Toggle bit k	`n ^ (1<<k)`
Power of 2?	`n && !(n & (n-1))`
Swap without temp	`a^=b; b^=a; a^=b;`
Lowest set bit	`n & -n`

- 2's complement of x = `~x + 1`; -1 is all 1-bits
- $(25)_{10} = (11001)_2 = (31)_8 = (19)_{16}$

PREDICT QUICK OUTPUTS

Expression	Value
7 % -3	1
-7 % 3	-1
5 > 3 > 1	0 (1 > 1 is false)
~5	-6 (~x = -x-1)
5 ^ 3	6
1 << 4	16
!!7	1
(3, 4)	4
10 == 10.0	1

HEADERS LIBRARY FUNCTIONS

Header	Common functions
stdio.h	printf, scanf, fopen, fgets, getchar
stdlib.h	malloc, free, atoi, rand, exit, abs, qsort
string.h	strlen, strcpy, strcat, strcmp, memcpy
math.h	sqrt, pow, fabs, ceil, floor, sin, log
ctype.h	isalpha, isdigit, toupper, tolower
time.h	time, clock, difftime

- `ceil(2.1)=3.0`, `floor(-2.1)=-3.0`, `abs()` int, `fabs()` double
- `math.h` programs may need `-lm` when linking (gcc)

MEMORY LAYOUT STORAGE OF A PROGRAM

Segment	Holds
Text (code)	machine instructions, read-only
Data	initialised globals & statics
BSS	uninitialised globals & statics (zeroed)
Heap	malloc / calloc blocks, grows up
Stack	locals, parameters, return addresses, grows down

Uninitialised **local** variable = garbage; uninitialised **global/static** = 0.