***

# C language -

Variables - Containers that store data.

memory (o/p)

data types → int — integer — 2/4 byte ⇒ %d
char — character — 1 byte ⇒ %c
float — float data — 4 byte ⇒ %f
double, void

1 byte = 2 Nibbles

1 Nibbles = 4 bits

ASCII
$2^8 = 256$

Compiler

—1) Pre-process → a) Remove cmt
b) Replace macros

—2) Compling → Finding errors

—3) Assembly → Convert [file to binary / hex]

—4) Linking → Verify the fun? in library.

`#include <stdio.h>` — This line includes the standard input/output library, which provide fun? like `printf()` for printing to the console.

`int main()` — main fun? of the program. Every C program must have a `main()` ~~program~~ function.

`printf("Hello \n");` — This line uses the `printf()` fun to print text "Hello" to the console.

The `\n` character adds a newlines at the end.

`return 0;` — That indicates the program has executed successfully.

#

Ex:-

```
# include <stdio.h>
int main()
{
  printf("Hello \n");
}
return 0;
}
```

output = Hello.

#

## Data types

int - stores whole numbers (10, -5, 0).
Float - stores Real numbers (with decimal point) (3.14, -2.11, 3.1)
char - stores a single characters ('A', '!', $, 9)
double - stores larger real numbers with higher precision.

#

Ex- `int age = 30;`
`float price = 9.99;`
`char initial = 'j';`

#

## Operators -

- Arithmetic Operators - ( +, −, X, %, / )
- Comparison operators - ( == (equal to), != (not equal), > (greater), < (less), >= (greater than or equal to), <= (less than or equal to) )
- Logical operators - ( && (AND), || (OR), ! (NOT) )

## Control Flow -

- if / else statement — Execute code blocks conditionally

```
if (condition)
    {
        // code to execute if cond" is true
    } else
        {
        // code to execute if cond" is false.
    }
```

## Loops -

- for loop — Repeats a block of code specific no. of times.
- while loop - Repeats a block of code as long as a cond" is true.
- do - while loop - Executes a block of code at least once, then repeats as long as a cond" is true.

**Functions** –

- Reusable blocks of code that perform a specific task.
- They can take input (arguments) & return a value.

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

**Arrays** – Ordered collections of elements of the same data type.

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

**Pointers** – Variables that store the memory addresses of other variables.

- Can be used for efficient memory manageme and accessing data directly.

coding → PC