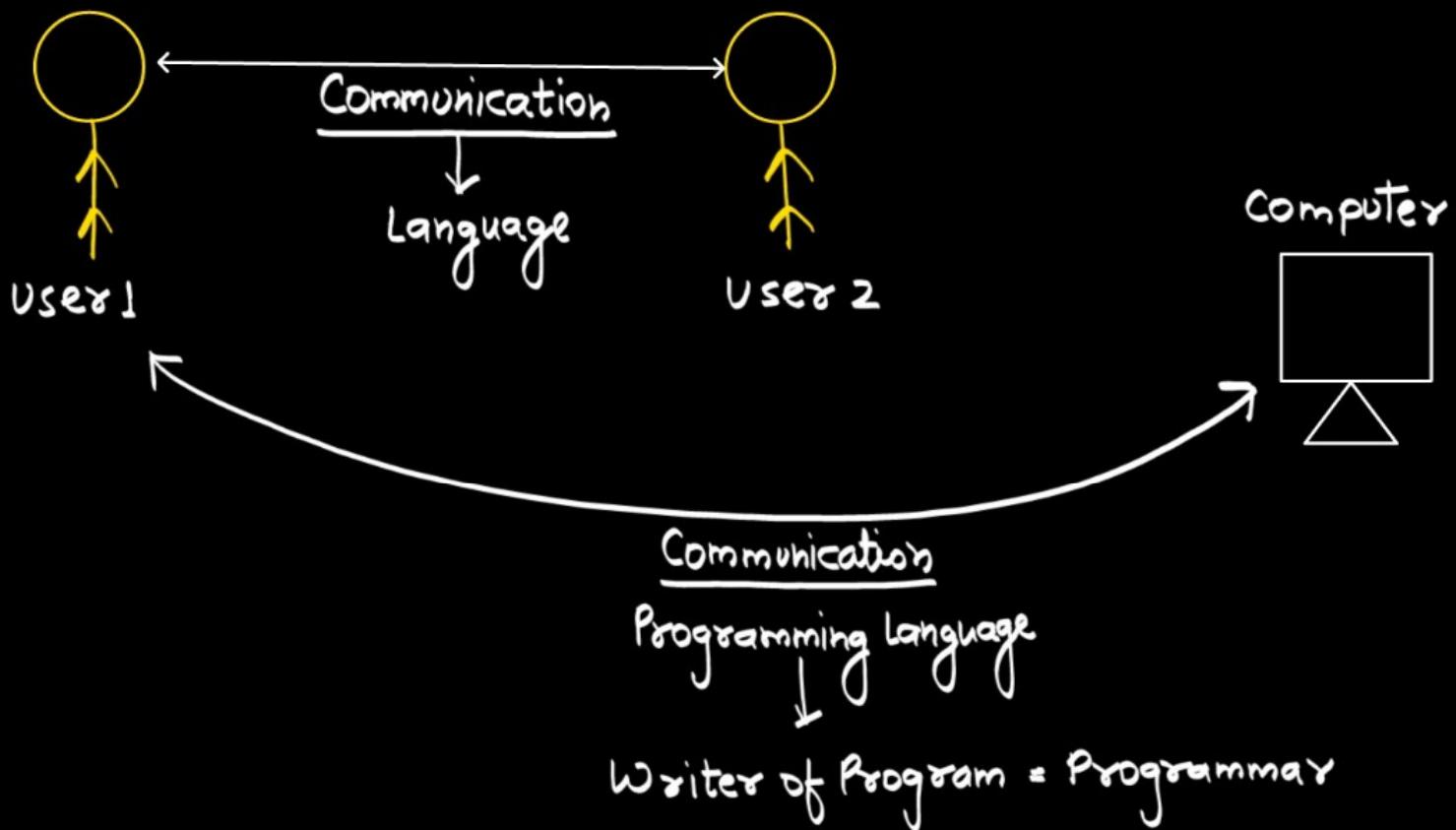


## "Programming Concepts"



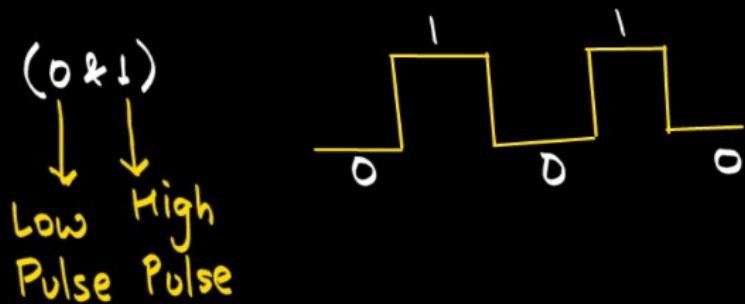
## \* Program ⇒

- Set of instructions to do any task.
- Passive set of instructions / Not active.

## \* Process ⇒

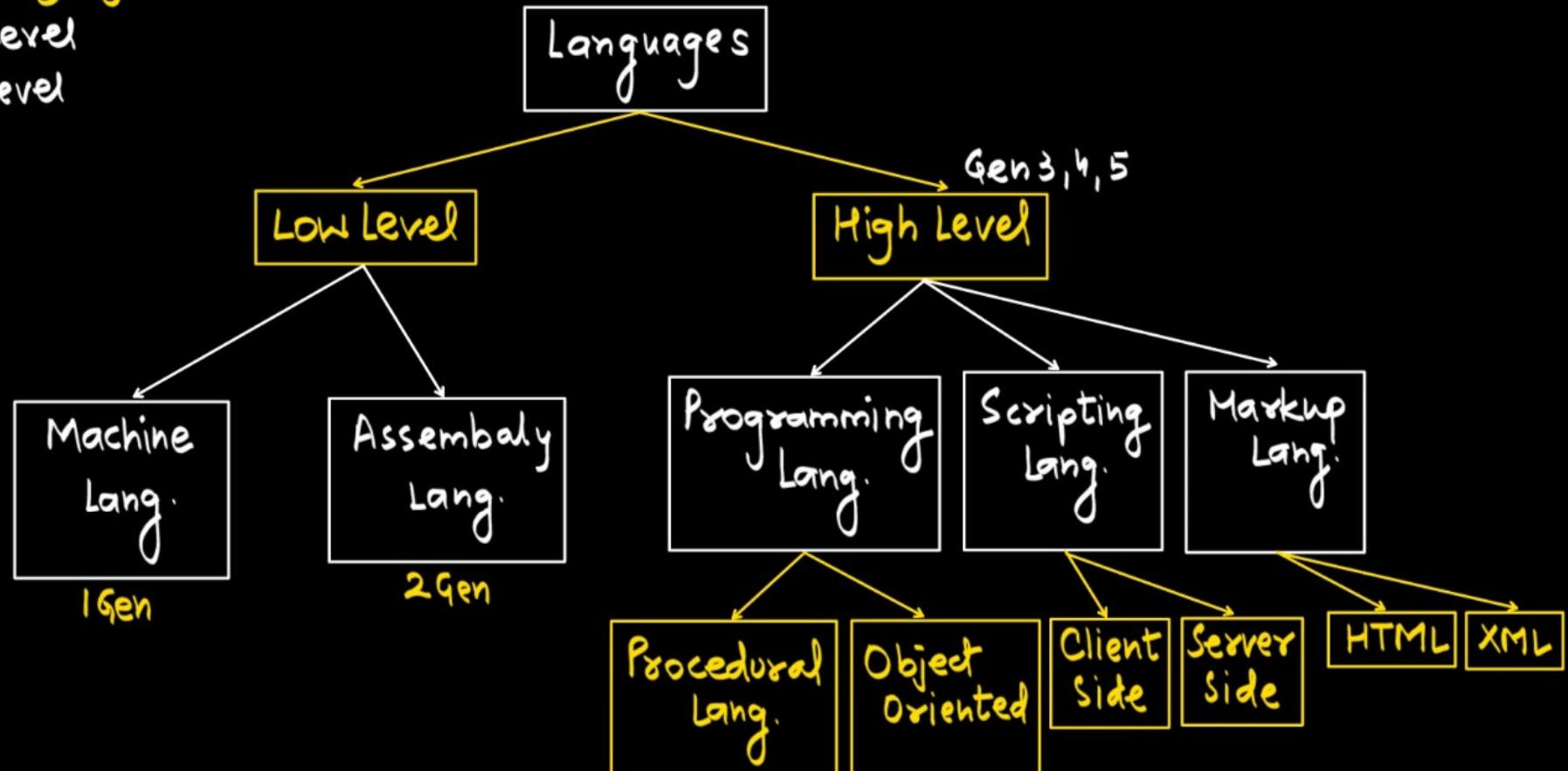
- An active set of instructions to do any task.
- Any program in running mode.
- All the processes are programs but all the programs can't be a process.

\* Computer can only understand Machine Language. (0 & 1)



## \* Types of Languages ⇒

- A. Low Level
- B. High Level



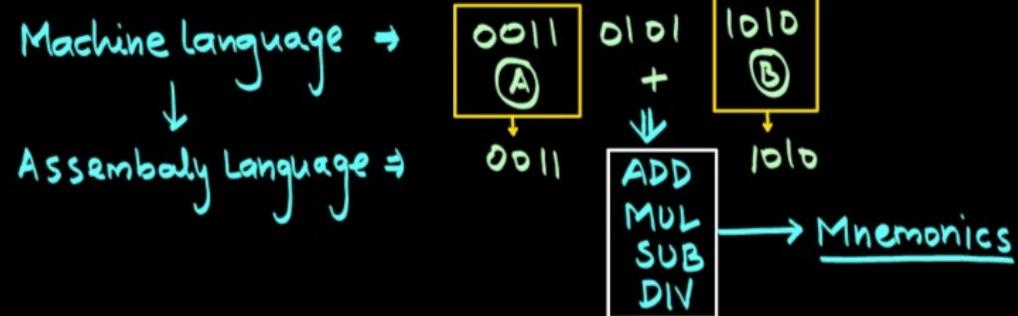
## \* Machine Language ⇒

- First language used for programming.
- Machine language में Programming 0 & 1 की String में की जाती है।  
↓  
Group of characters
- String = 101100110011011
- Only language जिसे Computer बिना किसी Translator के understand करता है।
- It is now outdated.
- Error find करना Tough.

## \* Assembly Language \*

- Programming Languages के field में first development.
- Invention in = Year 1949
  - ↓
    - By Maurice Wilkes & David Wheeler
    - 1949 में EDSAC Computer बनाया।
    - Electronic Delay Storage Automatic Calculator
    - First computer in which program was run.

- Assembly Language में Instructions को English Language में change कर दिया गया, जिन्हें Mnemonics कहते हैं।



इन Mnemonics को Machine Language में Translate करने के लिए Assembly बनाया गया।

## \* High Level Languages →

- ऐसी Language जियका code English Language / User Understandable language में लिखा जाता है, उसे High Level Language कहते हैं।
- Close / Near to user.
- We use Interpreter & Compiler to translate code of High Level Languages to Machine Language.

## \* Interpreter v/s Compiler ↗

Difference	Interpreter	Compiler
1. Working	Statement by statement / Line by Line Conversion	Whole program at a time convert.
2. Speed	यह Translation के बाद बनी वाली object File को save नहीं करता है, इसलिए Every Time Object File बनानी पड़ती है, इसलिए यह <u>Slow</u> है।	यह Translation के बाद बनी वाली object File को Hard Disk में save कर लेता है, इसलिए बार-2 Object File नहीं बनानी पड़ती है, इसलिए यह <u>Fast</u> है।
3. Hard Disk	<u>कम use</u> , ज्योंकि यह object file save नहीं करता है।	<u>अधिक use</u> , ज्योंकि यह object file को save करता है।
4. RAM	<u>अधिक use</u> , ज्योंकि यह प्रत्येक बार program को translate करता है।	<u>कम use</u> , ज्योंकि यह program को केवल सक बार translate करता है।

5. Debug  
↓  
Delete Bug

6. Security

7. Examples

Partially Error Detect

Less Secure क्योंकि प्रत्येक बार Program को  
RAM में लाना पड़ता है।

Python, Perl, Ruby, Java, Javascript,  
LISP, MATLAB, R, PHP, Lua, ADA.

Fully Detect

More Secure

C, C++, C#, Swift.

## "C Language"

- Appeared in = 1972
- Developed by = Dennis Ritchie
- Developed At = AT & T Bell Labs  
American Telephone & Telegraph  
↓  
Renamed As = Bell Laboratories
- Acquired by Nokia & Renamed As Nokia Laboratories
- Developed On = DEC PDP-11  
→ Digital Equipment Company Programmed Data Processor-11  
(इसी Machine पर Unix OS Rewrite किया गया था)  
↓  
Originally written in Assembly language
- 1971 में Ken Thompson ने Launch किया।

- Designed for =  $\frac{\text{Unix OS}}{\downarrow}$   
C Language में लिखा गया Largest Program
- Execution Sequence = Top to Bottom (Procedural Oriented Programming Language - Pop)
- Extension = .c (Program)  
.h (Header File)
- B Language की Successor थी, परन्तु इसमें B & BCPL दोनों की properties होती है।

## \* Skeleton / structure of C Program ⇒

1. Documentation Section → Optional
2. Preprocessor Directives / Link section - Required
3. Definition Section → Optional
4. Global Declaration Section → Optional
5. Main function → Required
6. Sub program Section → Optional

## 1. Documentation Section →

- Program तथा इसके features के बारे में Programmers को बताने के लिए use.
- इस section में Comment line का use होता है।

## \* Comment Line →

- Program को describe करते में useful.
- Compiler इसे Read नहीं करता है।
- 2 प्रकार
  - A. Single Line
  - B. Multi Line
- C language में initially Multi Line Comment line का use किया जाता था, इसके बाद Single Line Comment line start की गई।

→ Which comment line was used in starting of C language?  
⇒ Multi line

- Single Line Comment Line " // " से start होती है तथा इसे बन्द करने की Need नहीं होती है।
- Multi Line Comment Line " /\* " से start होकर " \*/ " पर End होती है।

## 2. Preprocessor Directives / Link Section →

- इस section में Header Files का use होता है।
- Program के अन्येक Keyword/ Reserve words को Header files में store किया जाता है तथा अन्येक Header file की Library file में रखा जाता है।
- Program में Header Files को Library Files से Link करने के लिए इस section का use होता है।
- Preprocessor directive = # (Hash)
- Library file Reserve words, Functions, Macros का Precompiled Collection होता है।

### 3. Definition Section →

- इस section में #define का use कर Parameters को define किया जाता है।
- Parameter एक Variable ही होता है, लेकिन इसका कोई DataType नहीं होता है।
- इसका use Macro को define करने में किया जाता है।  
↓  
A code snippet जिसे Program में बार-<sup>2</sup> use किया जाता है।

### 4. Global Declaration Section →

- main() function के बाहर declare किया जाने वाला Variable "Global Variable" कहलाता है।
- इसे पुरे Program में कहीं भी Access किया जा सकता है।
- Global Variable की Value change नहीं होती है।

## 5. Main Function ↳

- main() function of Program.
- इसमें Program की सारी properties define की जाती है।
- Soul of Program
- प्रत्येक Program में केवल 1 "main()" हो सकता है।

## 6. Sub Program Section ↳

- main() function के अलावा Program में use में लिए जा रहे functions को Subfunctions कहते हैं
- user defined functions.

Ex

```
// Introduction of C
/* Written by
   whoshubhamsir */
} Documentation Section

#include <stdio.h>
#include <conio.h>} Preprocessor / Link section

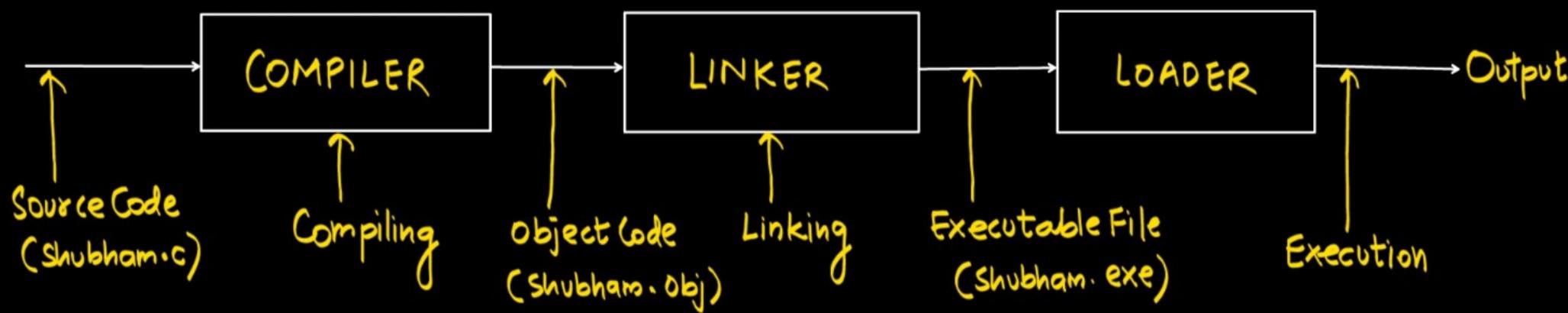
#define MAX 100 } Definition Section

int a=5; } Global Variable

Void main()
{
} } main function

Void add()
{
} } Sub Program Section
```

## \* Compilation Process of C Program ⇒



Shubham.c → Shubham.obj → Shubham.exe

→ C Program की Compilation Process में 3 Steps होती हैं -

- A. Compiling
- B. Linking
- C. Loading

→ सर्वप्रथम Programmer .c Extension की Source File देता है, जिसे Compiler Object Code .obj extension में Translate कर देता है।

→ इस .obj extension वाले Code को Linker Executable File (.exe) में Convert करता है।

→ Last में Loader इस .exe file को Run कर Output देता है।

## \* C Token $\Rightarrow$

→ Smallest entity of program.

→ किसी भी Program की smallest Entity जिसे further divide नहीं किया जा सकता है।

→ 4 प्रकार -

- A. Identifier / Variable
- B. Constants
- C. Keywords / Reserve Words
- D. Data Types

## A. Identifiers / Variables ↗

→ Program में user defined entities को एक नाम दिया जाता है, ताकि उसे पूरे Program में individually identify किया जा सके, इस नाम को Identifier कहते हैं।

### \* Rules of identifier ↗

1. Max length = 31 characters
2. Case sensitive (A & a are different)
3. Identifier के नाम में Alphabets (A-Z, a-z), Numbers (0-9) & underscore (-) ना use किया जा सकता है।
4. Identifier हमेशा Alphabet या underscore से Start होता | (Number से नहीं)
5. Identifier के नाम में Space, Special characters (#, @, \$), Keywords allowed नहीं हैं।

Ex

Abc ✓

abc ✓

abc123 ✓

123abc ✗

abc-123 ✓

\_abc123 ✓

\_123abc ✓

abc@ ✗

@abc ✗

abc# 123 ✗

abc 123 ✗

void ✗

Switch ✓

If ✓

switch ✗

if ✗

For ✓

for ✗



## B. Constant $\Rightarrow$

- दैर्से Variables जिनकी Values को Program के execution के दौरान change नहीं किया जा सकता है।
- प्रकार  $\Rightarrow$ 
  - (i) Integer Constants
  - (ii) Character Constants
  - (iii) String Constants
  - (iv) Floating Constants

### (i) Integer Constants $\Rightarrow$

$\rightarrow$  3 प्रकार

Octal Constants  $\longrightarrow (754)_8$

Decimal Constants  $\longrightarrow (794)_{10}$

Hexadecimal Constants  $\longrightarrow (79A)_{16}$

$\rightarrow$  By default रूप से Compiler सभी integer constants को Decimal count करता है।

$\rightarrow$  Language में Octal Constant देने के लिए Number से पहले 0 लगाया जाता है।

$\rightarrow$  Language में Hexadecimal Constant देने के लिए Number से पहले 0X लगाया जाता है।

उत्तर-  $456 = \text{Decimal}$

$0456 = \text{Octal}$

$0x456 = \text{Hexadecimal}$

\* 456 will be treated as which constant in C by default?

- A. Decimal      B. Octal  
C. Hexa            D. None

Q. In C 426X will be counted as which constant?

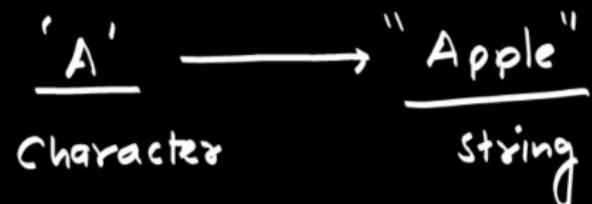
- A. Octal      B. Decimal      C. Hexa       D. NOTA

## (ii) Character Constant

- Language में character को Single inverted comma (' ') के बीच रखा जाता है।  
जैसे - 'A'
  - Language में compiler characters को directly read नहीं करता है, इसलिए character को सबसे पहले ASCII (American Standard Code for Information Interchange) में Convert किया जाता है तथा उसके बाद ASCII के Equivalent Binary findout करता है।
  - A → Z ⇒ 65 → 90
  - a → z ⇒ 97 → 122
  - NULL ⇒ 0
- \* Keyboard से किसी भी Key को Press करने पर उसके Keystroke को Binary में interchange करने के लिए ANSI standard नया ASCII Coding System का use किया जाता है।

### (iii) String Constant

→ Group of characters is known as string.



→ language में string को double inverted comma (" ") के बीच रखा जाता है

जैसे "ABC" ✓      "A" - X = Minimum 2 characters required

"abc" ✓      "Abc123" ✓

"123" ✓

#### (IV) Floating Constant

→ A number must contain floating point i.e. `.'

Ex- - 42.65

### 3. Keywords / Reserve Words →

- प्रत्येक Language में कुछ Predefined Words को Reserve रखा जाता है, जिनका एक Predefined meaning है, उन्हें Reserve words / Keywords कहते हैं।
- इन्हें Identifier / Variable के रूप में Use नहीं किया जाता है।
- Keywords हमेशा Small letters में होते हैं।
- C Language में Total 32 Keywords होते हैं -

auto	do	float	register	struct	volatile
break	default	for	return	switch	while
const	double	goto	short	typedef	
char	else	if	signed	union	
case	enum	int	sizeof	unsigned	
continue	extern	long	static	void	

#### 4. DataType ⇒

→ महं Data के Format & Property को बताता है।

→ 2 प्रकार

A. Built-in / Machine Provided / Primitive / Primary

B. Derived

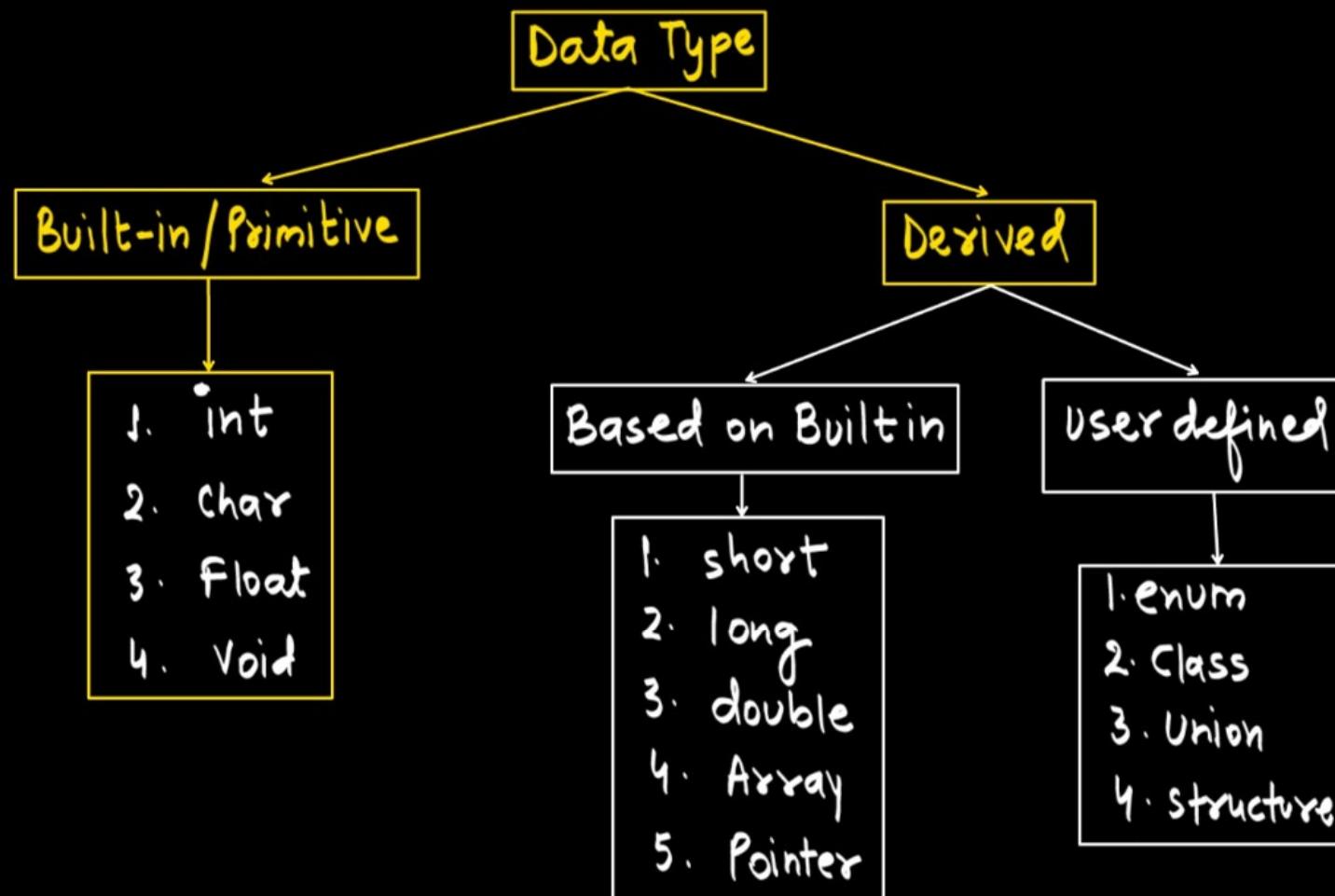
\* What does a data type show?

A. Type of Data

B. Pattern of Data

C. Type of Value

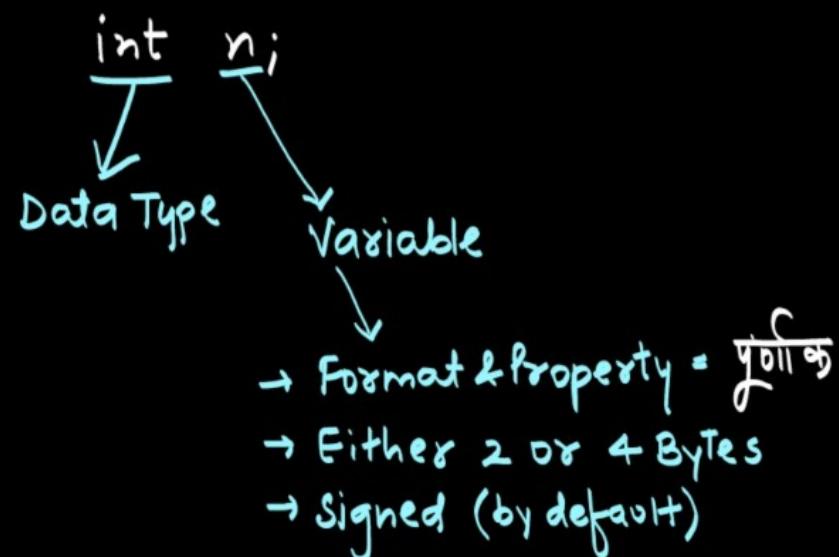
D. Type of format & property of Data



Data Type	Limit	Size
1. char	-128 to +127	1 Byte
2. Unsigned char	0 to 255	1 Byte
	<ul style="list-style-type: none"> <li>* ये Data Type जो + - दोनों में values लेता है, वह Signed तथा जो केवल + लेता है, वह Unsigned होता है।</li> <li>* by default all datatypes are signed.</li> </ul>	
3. int	<ul style="list-style-type: none"> <li>-32768 to +32767 (short int)</li> <li>-2147483648 to +2147483647 (long int)</li> </ul>	<ul style="list-style-type: none"> <li>2 Byte</li> <li>4 Byte</li> </ul>
	<ul style="list-style-type: none"> <li>* by default int की size compiler/Machine dependant होता है।</li> </ul>	

4. unsigned int	0 to 65535 (short int) 0 to 4294967295 (long int)	2 Byte 4 Byte
5. float	$3.4 \times 10^{-38}$ to $3.4 \times 10^{+38}$ $3.4 \times 10^{-38}$ to $3.4 \times 10^{+38}$	4 Byte
6. double	$2.3 \times 10^{-308}$ to $1.7 \times 10^{+308}$ $2.3 \times 10^{-308}$ to $1.7 \times 10^{+308}$	8 Bytes
7. long double	$3.4 \times 10^{-4932}$ to $1.1 \times 10^{+4932}$	10 Bytes

Ex



unsigned int n;  
 ↓  
 Unsigned declare करना पड़ता है।

Q = If  $\text{int } a = 5$  then what will be the Range of  $a$ ?

- A. 0 - 65535
- B. -32768 - +32767
- C. -128 - +127
- D. 0 - 32767
- E. NOTA

\* Size/Range of integer is Compiler or Machine dependant.

IA 2018 ⑩  
 $\text{int } a[5][2];$   
 ✓ Size of  $a = \underline{\text{NOTA}}$

Q. :-

int a = 10;      } Initialization - किसी Variable को declare करते ही Value दे देना।

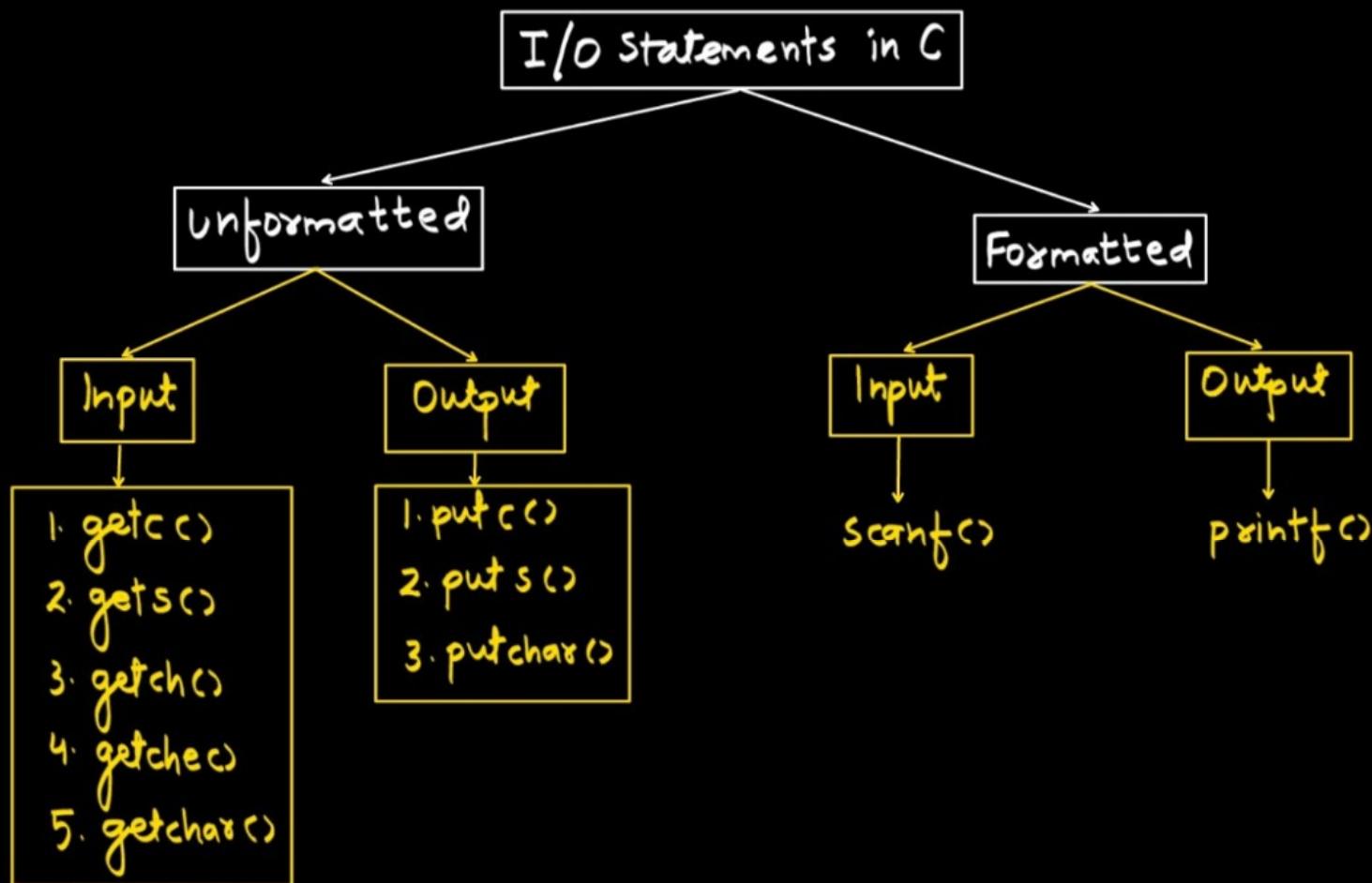
int a;  
a = 10;      } Assignment - किसी Variable को declare करने के बाद Next Line  
                        में Value देना।

\* Compiler takes more time in Assignment than Initialization.

\* Initialization is better than Assignment.



## \* Input-Output Statements



## \* Unformatted I/O Statements $\Rightarrow$

$\rightarrow$  इसी statements जिनकी Formatting पहले से fix होती है, इसमें हम कोई भी change नहीं कर सकते हैं।

Ex :-

char x;

a =  $x = \text{getchar}();$  Ram-Singh  $\leftarrow$

b =  $x = \text{getch}();$   $\xrightarrow{\text{Echo}}$  Ram-Singh  $\leftarrow$

c =  $x = \text{getche}();$   $\xrightarrow{\text{Reflexion}}$  Ram-Singh  $\leftarrow$

d =  $x = \text{getc}(\text{input}, \text{FileName})$

$\downarrow$   
File से एक character को Read करेगा।

Ans = केवल R

Ans = कुछ भी display नहीं

Ans = केवल R

- getch() & getche() दोनों keyboard से 1 character input में लेते हैं।
- getch() उस character को output में display नहीं करता है जबकि getche() में e= echo/reflection होता है अर्थात् getche() Read किये गए character को output में display करता है।
- getch() & getche() एक character को read करने के बाद Enter या Backspace की permission नहीं देते हैं।
- getchchar() & getc() दोनों Keyboard से एक character को Read करते हैं, लेकिन इनके साथ Enter press करना जरूरी है।

Q3 मानि `ch = getchar();` हो और keyboard से Kautilya type कर Enter press किया जाए, तो `ch` की value क्या होगी ?

A. Kautilya

B. a

C. K

D. MOTA

E. NOTA

Kautilya ↲  
↙

→ `getcc` 2 entities लेता है, जिनमें पहली Input तथा दूसरी File-Name होता है, जहाँ  
से Read करना होता है।

`x = getc ( A, stdio )`

↓

इस file से A को  
Read किया जाएगा।

↓

if A is not available  
then it displays an error.

Q. Which unformatted input function use filename as a second entity?

OR

Which unformatted input function use two variables?

A. `getc()`

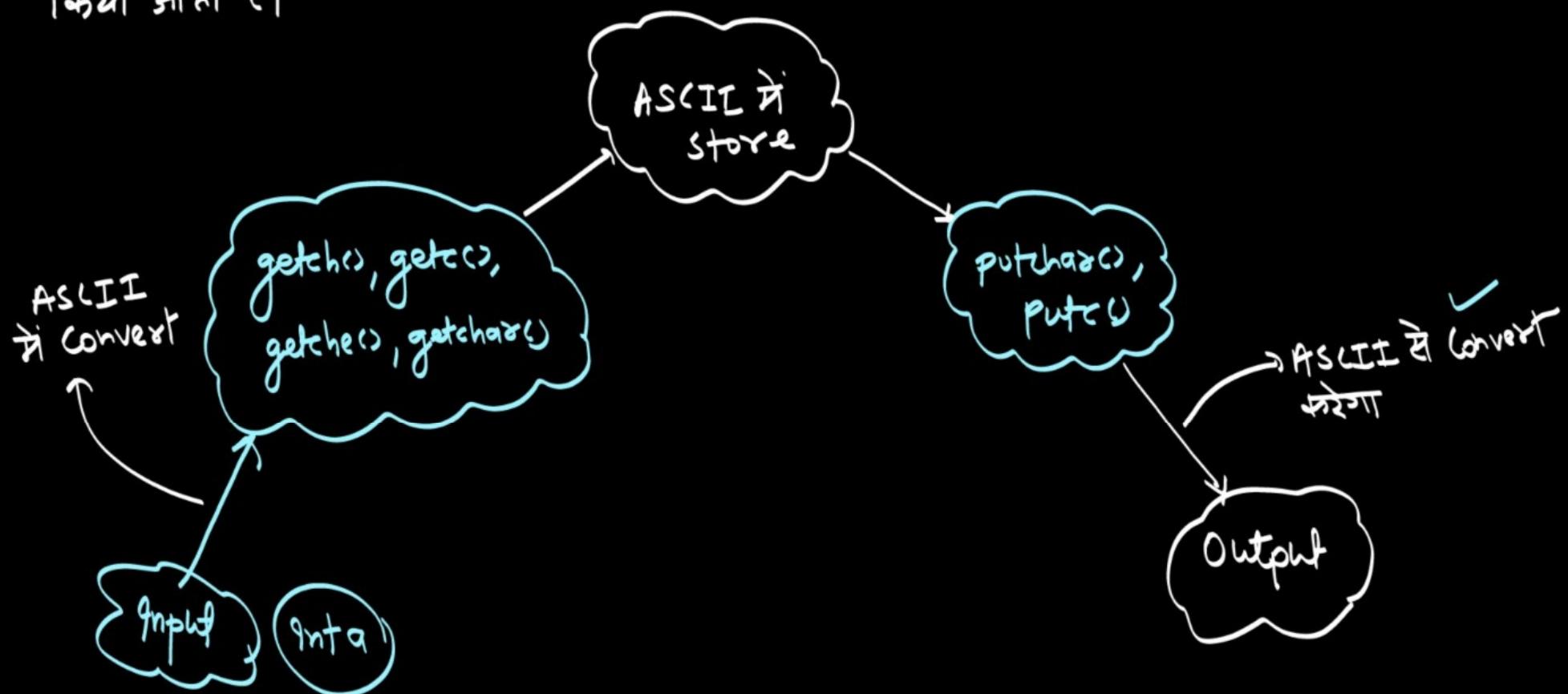
\* `getch()`, `getche()`, `getchar()` & `getc()` यारों functions ASCII Read करते हैं।

Q⇒ `char R;`      } यदि उक्त Code को 16Bit की Machine 'ABC' Input के रूप में दिया जाए, तो  
`R=getchar();`      } R की Value क्या होगी ?

A. ABC      C. 16ABC      E. B

B. A      D. ABC16

→ putchar() & putc() का use एक character को output ने display करने के लिए किया जाता है।



Q

```
char u;  
x = getchar();  
putchar(x);
```



65 ग्राह कर  
Output में A  
Display करेगा।

Input  
A  
↑  
 $u = 65$   
Store दोगा,  
जिसकि A की  
ASCII 65 दोती है

Q⇒

char  $x = 'A'$ ;

A.  $\text{putchar}(x) = A$

B.  $\text{putchar}('A') = A$

C.  $\text{putchar}(A) = \text{Error}$

D.  $\text{putchar}('5') = 5$

E.  $\text{putchar}('ss') = \text{Error}$

F.  $\text{putchar}(' ') = \text{No display}$

G.  $\text{putchar}('A' + 2) = \text{Error}$

H.  $\text{putchar}('A' + '2') = C$

$$65 + 2 = 67$$

$\text{putchar}("ss") \rightarrow s$

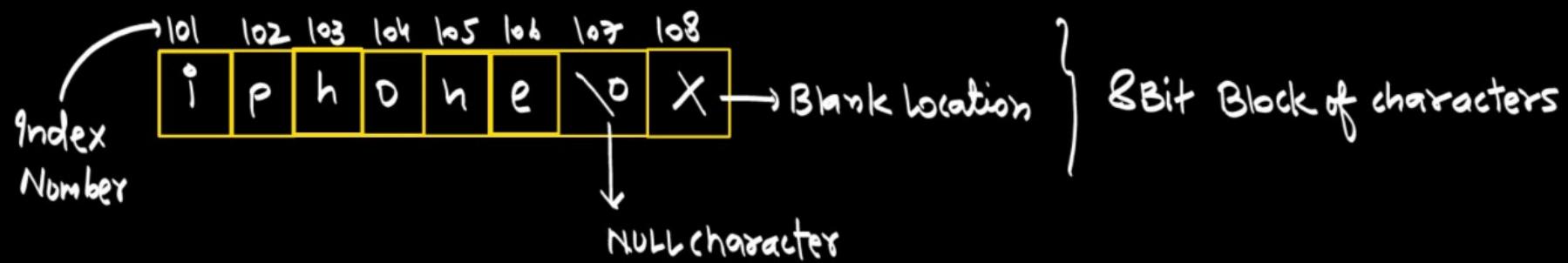
Memory में Space की ASCII = 32  
Store करेगा।

## \* String Based I/O functions ⇒

A. gets() - string Read

B. puts() - string Output में display करता है।

Ex :- `x = "iphone";`



\* ये Memory में First Index Number को  
Point करता है।

## \* NULL character

→ Memory location में string के last में \0 को default रूप से add कर दिया जाता है, ताकि Compiler को यह लगे, कि string end हो गई है।

Ex ⇒ `gets(" a");` → gets() function " a" को Double inverted Comma के कारण string के रूप में Read करेगा।

101	102	103	104	105	106	107	108
a	\0	X	X	X	X	X	X

\* Single character 'a' होने पर भी 8 bit का Block Memory में Allocate होगा।

Q.2 If gets("Ram") the who will add Null character after Ram?

- A. gets()
- B. Compiler
- C. Ram
- D. "
- E. NOTA

### \* NULL character ⇒

→ Memory location में string के last में \0 को default रूप से add कर दिया जाता है, ताकि Compiler को यह ले जाए, कि string end हो गई है।

Ex ⇒    `gets(" u");`    → gets() function " u" को Double inverted Comma के कारण string के रूप में Read करेगा।

101	102	103	104	105	106	107	108
u	\0	X	X	X	X	X	X

\* Single character 'u' होने पर भी 8 bit का Block Memory में Allocate होगा।

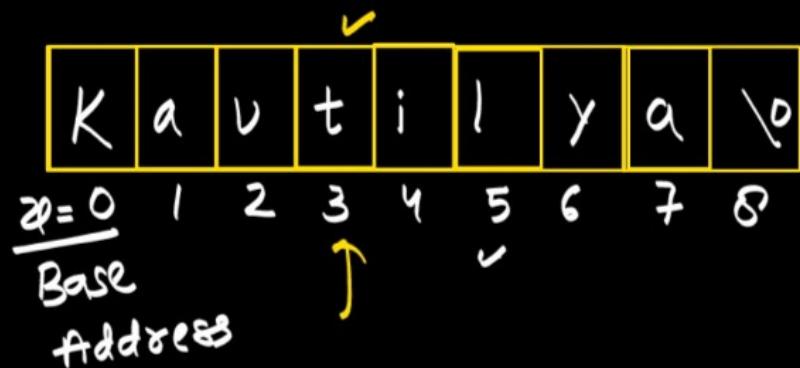
Q.⇒ If gets("Ram") the who will add Null character after Ram?

- A. gets()
- B. Compiler
- C. Ram
- D. "
- E. NOTA

Q = Find the Output, if -

```
char s[8];
s = "Kautilya";
// gets(s) = "Kautilya";
```

- A. puts(s) = Kautilya
- B. puts(s+3) = tilya
- C. puts(s+5) = lya
- D. puts("chippkey", +3) = nkey



## \* Formatted I/O Statements ↳

- ऐसी statements जिनके द्वारा लिये गये input या दिए गये output के format को define करना पड़ता है।
- format को define करने के लिए format specifier का use किया जाता है।
- इन्हें % का प्रयोग करके define किया जाता है।
- Important format Specifiers ↳

1. char = %.c
2. int = %.d
3. float = %.f (•.nnnnnn)  
    %.g (general)  
    %.e (exponential/Power)

4. Unsigned = %u

5. short = %h

6. long = %li

7. double = %.lf

8. long double = %.Lf

9. string = %.s

→ formatted Input Statement का function "scanf()" होता है तथा formatted Output statement का function "printf()" होता है।

→ scanf() के साथ "& (Ampersand)" Symbol का use किया जाता है।

Ex

Char a = 'c';

→ c की ASCII 99 को store करेगा।

A. printf ("%d", a);

⇒ 99

B. printf ("%c", a);

⇒ C

C. printf ("%c%d", a, a);

⇒ C99

D. printf ("A");

⇒ A

\* printf() & scanf() के साथ multiple format Specifiers को use में लेकर multiple values output में display करवाई जा सकती है।

\* printf() के साथ अदि इसी को Double Inverted Comma (Except format specifiers) के बीच लिखा जाये तो as it is output में display होगा।  
=printf ("Shubham"); - o/p = Shubham

Ex :-

float a = 15.45;  
printf("%f", a); → printf("%g", a)

(A) 15.45

(B) 15

(C) 15.4

(D) 15.450000

(E) 15.450

%f = n.nnnnnn  $\Rightarrow$  15.45

\* %f dot के बाद default हम से 6 digits  
लेना है।

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    .
    float n1 = 15000000;
    float n2 = -15000000;
    double n3 = 25.2132;
    double n4 = 0.00001302;

    printf("n1 (%.f) - %f: \n", n1);
    printf("n1 (%.E) - %E: \n", n1);

    printf("n2 (%.f) - %f: \n", n2);
    printf("n2 (%.E) - %E: \n", n2);

    printf("n3 (%.f) - %f: \n", n3);
    printf("n3 (%.E) - %E: \n", n3);

    printf("n4 (%.f) - %f: \n", n4);
    printf("n4 (%.E) - %E: \n", n4);

    exit(EXIT_SUCCESS);
}
```

Output:

```
n1 (%f) - 15000000.000000:
n1 (%E) - 1.500000E+07:
n2 (%f) - -15000000.000000:
n2 (%E) - -1.500000E+07:
n3 (%f) - 25.213200:
n3 (%E) - 2.521320E+01:
n4 (%f) - 0.000013:
n4 (%E) - 1.302000E-05:
```

## \* scanf() ⇒

- यह Input के रूप में Arguments लेकर उन्हें Memory location पर store कर देता है।
- printf() doesn't need enter key but scanf() needs.
- scanf() के साथ & का use होता है।

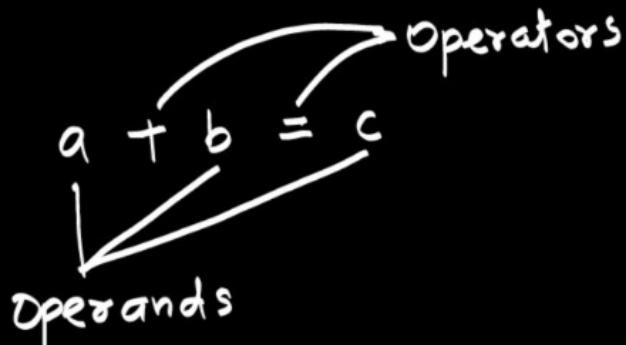
Ex ⇒

```
int a;  
scanf("%d", &a);  
printf("%d", a);
```

use के value लेकर  
a में store करेगा।

a की value को display  
जाएगा।

## \* Operators in C $\Rightarrow$



- Mathematical Symbols which are used to perform various operations on operands are known as Operators.
- Types of Operators  $\Rightarrow$

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Conditional Operators
6. Bitwise Operators
7. Other Operators

## I. Arithmetic Operators ⇒

Unary ⇒

- A. Unary +
- B. Unary -
- C. sizeof
- D. typecast
- E. addressof (&)
- F. Value at (\*)
- G. increment (++)
- H. Decrement (--)

Binary ⇒

- A. Binary +      D. Binary %
- B. Binary -
- C. Binary \*

## \* Unary Operators $\Rightarrow$

$\rightarrow$  ऐसे Operators जो किसी एक Operand पर कार्य करते हैं।

उदाहरण -

int a=5, b;

b+=a;  $\rightarrow$  compound Assignment



b=b+a;

$=0+5 = \boxed{b=5}$

int a=10, b=5;

a-=b;

$a=a-b;$

$a=5$

int x = -10, y;

y=-x;

$y=-(+10);$

$y=10$  ✓

int p=10, q=5;

p-=q;

$p=p-q;$

$p=5$

## \* Sizeof Operator ↴

→ datatype की size की जाता है।

→ sizeof operator में size का अर्थ कोई Variable / DataType के द्वारा Memory में Occupied Bits से होता है।

प्रैक्टि- int a=5;  
sizeof(a);  
(A) 2  
(B) 4  
(C) 5  
**✓(D) NOTA**

a = integer  
↓  
Machine  
Dependant

---

int n = sizeof("Lata");  
**✓n=4**

char x = "Ram";  
sizeof(x);

✓ **(A) 1** ✓

- (B) 2  
(C) 3  
(D) 4

int n = sizeof(457);  
printf("%d", n);

**n=3**

यदि sizeof के द्वारा  
केवल Number हो तो  
Total Digits Output  
में display होती।

## B. Relational Operators $\Rightarrow$

→ Total 6

1. Less than - <
2. Greater than - >
3. Equals to - ==
4. Not Equals to - !=
5. Less than Equals to - <=
6. Greater than Equals to - >=

\* Which of the following operators are not Relational?

X A.  $\leq$       X B.  $\geq$

X C.  $\neq$       D. NOT A

E. NOT A

Q = if  $a=5, b=6, c;$

$c = (a < b);$

$c = ?$

A. 5      B. 6

C. 1      D. MOTA

E. NOTA

Q = int  $a=5, b=6, c=7, d;$

$d = (a < b) + (b > c) + (c < a);$

$d = ?$

A. 1      B. 2      C. 3

D. MOTA    E. NOTA

\* Relational Operators का output हमेशा 0 या 1 की तौर पर होता है।

\* यदि Condition True हो तो output = 1  
तथा Condition False होने पर output = 0  
आयेगा।

Q = int  $a=7, b;$

$b = (7 == 7);$

$b = ?$

A. 7      B. 0      C. 1

D. MOTA    E. NOTA

$a = b$

\* a को b की value  
assign की जा रही है।

= Assignment Operator

$a == b$

\* a और b की value को  
Compare किया जा रहा है।

= = Relational Operator

Q:- int x=5, y=7, c;

A  $\rightarrow$  C =  $\frac{5}{x} = \frac{7}{y}$   $\Rightarrow$  0

B  $\rightarrow$  C =  $\frac{x < y}{\checkmark}$   $\Rightarrow$  1

C  $\rightarrow$  C =  $\frac{!}{\checkmark} \mid = y$   $\Rightarrow$  1

D  $\rightarrow$  int a =  $\frac{x}{0} = \frac{!}{0} \mid = y$   $\Rightarrow$  1

E  $\rightarrow$  int a =  $\frac{c}{x} = \frac{0}{x = y}$   $\Rightarrow$  0

F  $\rightarrow$  int p =  $\frac{c}{c} = \frac{y}{x} >= x$   $\Rightarrow$  1

### 3. Logical Operators $\Rightarrow$

- इन operators का use Logical Expressions के साथ किया जाता है।
- इनका Output अर्थात् 0 या 1 होता है।
- 3 प्रकार-
  - A. Logical AND ( $\&$ )
  - B. Logical OR ( $||$ )
  - C. Logical NOT ( $!$ )

## A. Logical AND (&&)

→ इस Operator में दोनों Inputs/Variables True होने पर की output True होगा।

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

Q ⇒ int a = 5, b = 6, c;  
c =  $\frac{(a < b)}{\checkmark ①} \&\& \frac{(b > a)}{\checkmark ①}$ ;  
c = ?  
A. 0      B. 1      C. 5  
D. MOTA    E. NOTA

## B. Logical OR (||) ⇒

→ इस Operator में दोनों Variables/ Inputs/Values में से कोई भी एक True होने पर Output True होगा।

A	B	A    B
0	0	0
0	1	1
1	0	1
1	1	1

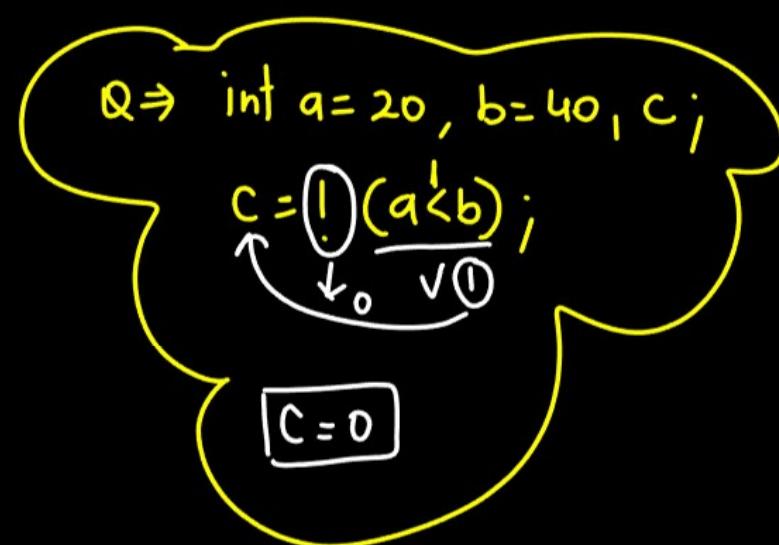
Q ⇒ int a = 10, b = 20, c;  
c = ( $\frac{a > b}{\times 0}$ ) || ( $\frac{a < b}{v 1}$ );  
c = ?  
A. 0      B. 1      C. 10  
D. MOTA    E. NOTA

### C. Logical NOT (!) $\Rightarrow$

→ Complimentor

→ यदि Input TRUE हो तो Output FALSE और यदि Input FALSE हो तो Output TRUE होगा।

A	A'
0	1
1	0



Q. On what kind of operators logical operators can be used?

- A. Bitwise
- B. Relational
- C. Arithmatic
- D. MOTA
- E. NOTA

\* Logical Operators का use  
Relational Operators के साथ  
किया जाता है, क्योंकि इन्हीं Operators  
का output 0 या 1 होता है, जिसे  
Logical Operators में input के रूप  
में use में लिया जाता है।

#### 4. Assignment Operator $\Rightarrow$

$\rightarrow$  किसी Variable को Value assign करना।  
उद्य- int a = 5;

#### 5. Conditional Operator $\Rightarrow$

$\rightarrow$  Ternary Operator अर्थात् यह 3 operands पर काम करता है।

1<sup>st</sup> Operand = Condition

2<sup>nd</sup> Operand = True Part of Condition

3<sup>rd</sup> Operand = False Part of Condition

} (condition) ? True : False ;

Ex

int a=5, b=7, c;

c = (a < b) ? a : b ;

c = ?

A. 0    B. 1    C. 5

D. MOTA    E. NOTA

\* Conditional Operators user friendly नहीं होते हैं, इसलिए  
इन Control Statements के साथ  
use के लिए जाना है।

Ex

int a=7, b=52, c;

c = (a > b) ? a++ : b++ ;

c = ?

A = 52    B = 7    C = 53

D = MOTA    E = NOTA

} b++ ऐ Postfix Increment  
{, that means first use  
than increment.

## 6. Bitwise Operator $\Rightarrow$

→ Supports Bitwise Operations.

→ 6 गुणार्थ -

- A. Bitwise AND ( $&$ )
- B. Bitwise OR ( $|$ )
- C. Bitwise XOR ( $^$ )
- D. Bitwise Not ( $\sim$ )
- E. Bitwise Right Shift ( $>>$ )
- F. Bitwise Left Shift ( $<<$ )

l = Pipe  
 $^$  = Caret  
 $\sim$  = Tild

## A. Bitwise AND (&) $\Rightarrow$

→ इसमें दोनों Inputs True होने पर ही Output True आयेगा।

A	B	$A \& B$
0	0	0
0	1	0
1	0	0
1	1	1

Ex  $\Rightarrow$  int a=2, b=3, c;  
c = (a & b);  
c = ?  
C = 2

Ex = int a= 5, b= 8, c;

c = (a & b);

C = 0

$$\begin{array}{r} .0101 \\ \times 1000 \\ \hline 0000 \end{array}$$

## B. Bitwise OR (|) $\Rightarrow$

→ इक्सें दोनों inputs में से एक भी input True होने पर Output True आयेगा।

A	B	A B
0	0	0
0	1	1
1	0	1
1	1	1

$$\text{Ex: } \text{int } a=2, b=3, c;$$

$$c = (a|b);$$

$$\boxed{C = 3}$$

$$\begin{array}{r} 2 = 0010 \\ 3 = 0011 \\ \hline 0011 \end{array}$$

$$\text{Ex: } \text{int } a=5, b=8, c;$$

$$c = (a|b);$$

$$\boxed{C=13}$$

$$a=5 = 0101$$

$$b=8 = 1000$$

$$\boxed{1101}$$

### C. Bitwise XOR (^) $\Rightarrow$

→ इसमें दोनों Inputs अलग-अलग होने पर ही output True होता, otherwise False.

A	B	$A \wedge B$
0	0	0
0	1	1
1	0	1
1	1	0

Ex = int a=2, b=3, c;  
 $c = (a \wedge b);$   
C = 1

$$\begin{array}{r} 2 = 0010 \\ 3 = 0011 \\ \hline 0001 \end{array}$$

Ex = int a=5, b=8, c;  
 $c = (a \wedge b);$   
C = 13

$$\begin{array}{r} 5 = 0101 \\ 8 = 1000 \\ \hline 1101 \end{array} \checkmark$$

#### D. Bitwise Not ( $\sim$ ) $\Rightarrow$

- $\rightarrow$  Input True = Output False
- $\rightarrow$  Input False = Output True

#### E. Bitwise Right shift ( $>>$ ) $\Rightarrow$

- $\rightarrow$  यह दिये गये नम्बर की Bits को Right side में Shift करता है।

Ex  $\Rightarrow$   $C = a >> 2;$

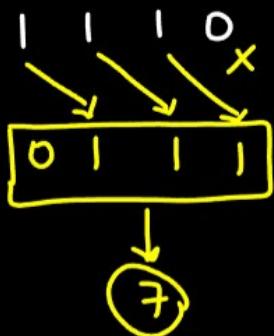
$\downarrow$   
a की Value को 2 बार Right Shift

1 = 1		
2 = 10		
3 = 11		
4 = 100		
5 = 101		
6 = 110	14 = 1110	
7 = 111	15 = 1111	
8 = 1000		
9 = 1001		
10 = 1010		
11 = 1011		
12 = 1100		
13 = 1101		

Ex = int a=14, b;  
 b=a>>1;  
 b=?

Sol = 14 की Binary =

$$\boxed{b=7}$$



Ex = int a=80, b;  
 b=a>>3;  
 $\boxed{\sqrt{b=10}}$

$$\frac{80}{2 \times 2 \times 2} = \textcircled{10}$$

\* Right shift जितनी बार करता हो,  
 उन्हीं बार 2 का गांज दे दिया  
 जाता है।

\* int a=60, b;  
 b=a>>2;

$$\boxed{b=15}$$

$$\frac{60}{\cancel{2} \times \cancel{2}} = 15$$

## F. Bitwise left shift (`<<`) :-

- यह दिए गये नम्बर की Bits को left side में shift करना है।
- Number को जितनी बार left shift करना है, उतनी बार 2 से गुणा कर दिया जाता है।

Ex :-    `int a=7, b;`  
              `b = a << 1;`       $(7 \times 2 = 14)$   
               $b = 14$

Ex :-    `int a=10, b;`  
              `b = a << 3;`       $\frac{10 \times 2 \times 2 \times 2}{(80)}$   
               $b = 80$

$a = 0010$   
 $1101$   
 $\underline{0011}$   
 main.c  $\boxed{-3}$

Last Bit  
 ↓ Same

$$a \& b = \begin{array}{r} a = 0010 \\ b = 0011 \\ \hline 0010 = 2 \end{array}$$

$$b >> 1 = \boxed{0010} \quad \boxed{1}$$

$a = 0011$   
 $1100$   
 $\boxed{0010}$   
 Last Same

Run

$\boxed{-2}$

Output

```

1 #include <stdio.h>
2 int main()
3 {
4     int a = 2, b = 3;
5     printf("a&b = %d\n", a & b); ✓
6     printf("a|b = %d\n", a | b); ✓
7     printf("a^b = %d\n", a ^ b); ✓
8     printf("~a = %d\n", ~a);
9     printf("b<<1 = %d\n", b << 1); ✓
10    printf("b>>1 = %d\n", b >> 1); ✓
11    return 0;
12 }
```

$$a|b = \underline{0011} = 3$$

$$a^b = \underline{0001} = 1$$

$$b << 1 = \boxed{0011} \quad \boxed{6}$$

$$b << 1 = 3 \times 2 = \boxed{6}$$

$$b >> 1 = \frac{3}{2} = \boxed{1} \quad \cancel{\boxed{1}}$$

int a=3, b;  
 $b = (\sim a);$

==== Code

65535

```
int a=5, b;  
b=(~a);
```

5 = 0101  
1010  
|  
0100  
↙ [-4]

## \* Increment & Decrement Operator

- किसी भी Variable की Value को +1 करने के लिए Increment Operator तथा किसी भी Variable की Value -1 करने के लिए Decrement Operator का use किया जाता है।
- Increment & Decrement दोनों operators को Prefix & Postfix way में use किया जा सकता है।
- Prefix = operator Operand ; (First increment then use)
- Postfix = Operand operator ; (First use then increment)

Ex  $\Rightarrow$  int  $a=5, b;$

$$b = \boxed{a++} + \boxed{++a};$$

$b = ?$       ⑥       $6+1$

$$5+7$$

✓  $\boxed{b=12}$

Ex  $\Rightarrow$  int  $a=7, b;$

$$b = \boxed{--a} + \boxed{a--};$$

$b = ?$       ⑥       $6+6$

$$6+6$$

$\boxed{b=12}$

Ex  $\Rightarrow$  int  $p=11, q;$

$$q = \boxed{p--} + \boxed{--p};$$

$q = ?$       ⑩

$$q = 11 + 9$$

$\boxed{q=20}$

Ex  $\Rightarrow$  int  $x=9, y;$

$$y = \boxed{--x} + \boxed{++x};$$

$y = ?$       ⑧       $8+9$

$\boxed{y=17}$

## \* Control Structure =>

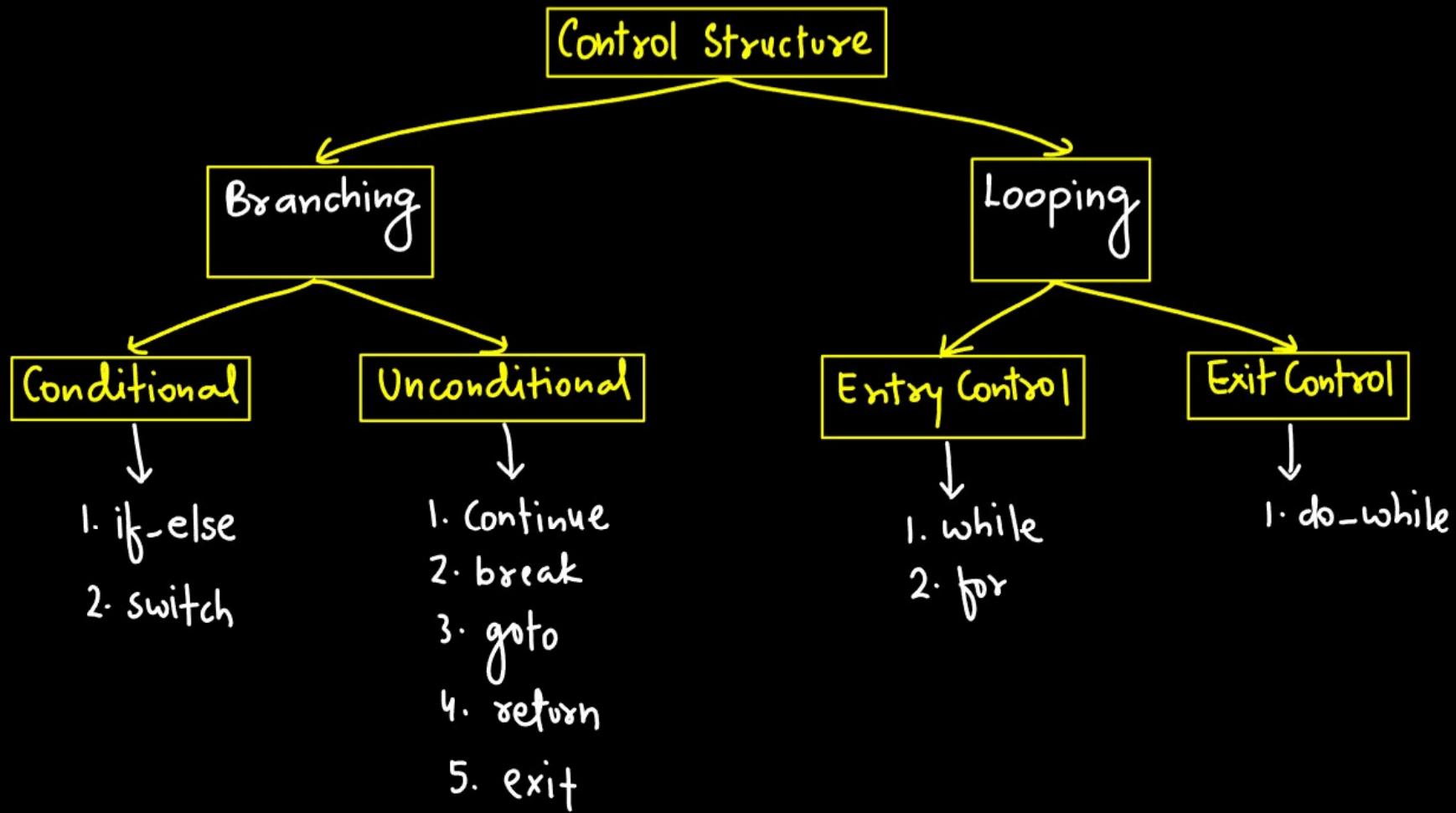
→ Program के flow को Control करने के लिए use में  
जी जाने वाली statements.

→ 2 प्रकार -

- A. Branching / Decision Making
- B. Looping / Iteration

Q ⇒ किसी भी Language में  
program का flow by  
default कैसा होता है

- A. Top to Bottom
- B. Bottom to Top
- C. Sequential
- D. MOTA
- E. NOTA



## A. Branching / Decision Making $\Rightarrow$

- Program के flow को किसी specific statement पर ब्रेंजने के लिए use.
- 2 Types -
  - 1. Conditional
  - 2. Unconditional

### I. Conditional $\Rightarrow$

- दोस्ती statements जो program के flow को किसी condition के आधार पर control करती हैं।
- 2 प्रकार-
  - a. if-else
  - b. switch

Q = How many types of if-else in c ?

- A. 3
- B. 4
- C. 5
- D. MOTA
- E. NOTA

## a. if - else $\Rightarrow$

$\rightarrow$  4 Types -

- (i) Simple if
- (ii) if - else
- (iii) Nested if
- (iv) ladder if

### (i) Simple if $\Rightarrow$

- $\rightarrow$  Simplest form of if - else.
- $\rightarrow$  if without else.
- $\rightarrow$  इसमें else part नहीं होता है, जिससे अगर Condition wrong हो, तो output नहीं आता है।

Syntax  $\Rightarrow$

```
if (condition)
    statement;
```

```

1 #include <stdio.h>
2 int main()
3 {
4     int n = 19;
5     if (n < 10)
6     {
7         printf("%d < 10", n);
8     }
9     return 0;
10 }

```

==== Code Execution Successful ====

int main()
{ int age=29;
if (age<18) → condition is not true,
 printf("you can't vote");
 return 0;
}

So no output

AutoPlay

Shubham (H:)

There's a problem with

Ex ⇒ { int a=15, b=6, c;  
if (a>b)  
 printf("%d", a);  
 printf("%d", a+b);  
} return 0;

15  
21

11

if के  
Scope से बाहर } }

## (ii) if - else $\Rightarrow$

- इस structure में if = true part of statement & else = false part of statement होता है।
- Condition true होने पर if part & condition false होने पर else part execute होता।

Syntax  $\Rightarrow$

```
if (Condition)
    statement;
else
    statement;
```

```
#include <stdio.h>
int main()      void main()
{
    int n = 10;
    if (n > 5)
    {
        .....
        printf("%d is greater than 5",n);
    }
    else
    {
        .....
        printf("%d is less than 5",n);
    }
    return 0;
}
```

10 is greater than 5

==== Code Execution Successful ===

### (iii) Nested-if

- एक statement में दूसरी statement को प्रयोग में लेना Nested statement कहते हैं।
- Similarly यदि हम एक if-else statement में दूसरी if-else statement का use करें, तो इसे Nested-if कहा जाता है।

Syntax →    if (Condition)  
          {  
            if (Condition)  
              Statement;  
            else  
              Statement;  
          }  
          else  
            { if (Condition)  
              Statement;  
            else Statement;  
          }

```
1 #include <stdio.h>
2 int main(){
3     int i = 10;
4     if (i == 10) {
5         if (i < 18)
6             printf("Still not eligible for vote");
7         else
8             printf("Eligible for vote\n");
9     }
10    else {
11        if (i == 20) {
12            if (i < 22)
13                printf("i is smaller than 22 too\n");
14            else
15                printf("i is greater than 25");
16        }
17    }
18    return 0;
19 }
```

i=17 → No output

✓ Still not eligible for vote

==== Code Execution Successful ===

Ex int a = 5, b = 4, c = 2;

if (a < b) X  
    5 < 4 X

    if (a < c)  
        printf ("a is small");  
    else  
        printf ("c is small");  
}

else :

{ if (b < c) X  
    4 < 2 X  
    printf ("b is small");  
    else  
        printf ("c is small");  
}

D/p = c is small

#### (IV) Ladder if →

→ इस structure में condition के सभी दोनों पर output आ जाता है, लेकिन Condition के गलत होने पर Else part में एक और condition को check करवाया जाता है।

Syntax →

```
if (condition)
    statement;
else if (Condition)
    statement;
else if (condition)
    statement;
else
    statement;
```

```
1 #include <stdio.h>
2
3 int main() {
4     int i = 20; ✓
5
6     // If else ladder with three conditions
7     if (i == 10)
8         printf("Not Eligible");
9     else if (i == 15)
10        printf("wait for three years");
11     else if (i == 20)
12         printf("You can vote");
13     else
14         printf("Not a valid age");
15
16     return 0;
17 }
```

You can vote

==== Code Execution Successful ===

```
int a=5,b=6,c=7;
if (a<b)
    printf("a"); ✓
else if (b<c)
    printf("b");
else
    printf("c");
```

D/p=a

## 2. switch - case - default :-

- Switch एक carrier होता है, जो condition को carry करता है।
- case = All possible outputs.
- default = switch की universal statement कहते हैं, क्योंकि यदि उन्हें सभी case सही नहीं तो default Statement execute हो जाती है।
- **Switch-case-default** का use multiple statements में से किसी उन्हें को choose करने में किया जाता है।

Q. what is switch in C?

- A. Condition
- B. Carrier**
- C. option
- D. MOTA
- E. NOTA

Syntax ⇒

```
switch (condition)
{
    case 1 : statement;
    break;

    case 2 : statement;
    break;

    ...
    default : statement;
    break;
}
```

Q ⇒ if we use break in switch,  
what happens with compiler?

- A. work slowly
- B. works fast**
- C. No effect
- D. MOTA
- E. NOTA

→ Switch में multiple statements होने के कारण condition/case match ही जाने के बाद भी मारे cases को read किया जाता है, जिससे compiler को ज्यादा Time लगता है, इसलिए प्रत्येक case के बाद break लगाया जाता है, जिससे किसी भी case के condition से match होने पर flow of program loop से बाहर चला जाता है।

```

1 #include <stdio.h>
2 int main() {
3     int var = 1;
4     switch (var) {
5         case 1:
6             printf("Case 1 is Matched.");
7             break;
8         case 2:
9             printf("Case 2 is Matched.");
10            break;
11        case 3:
12            printf("Case 3 is Matched.");
13            break;
14        default:
15            printf("Default case is Matched.");
16            break;
17    }
18    return 0;
19 }

```

Case 1 is Matched.

== Code Execution

last  
 $a=12$   
 $b=-5$

```

int a=5, b=7;
switch(15)
{
    case a: a++; b--;
    break;
    case b: --a; ++b;
    break;
    default: a=a+b; b=b-a;
}
    
```

## \* Unconditional Control Statement ⇒

→ ऐसी Control Statements जो program के flow को बिना Condition के control करते हैं।

→ 5 प्रकार ⇒

1. break
2. continue
3. goto
4. return
5. exit

## J. break =>

→ program के flow को किसी Loop या Block से बाहर फैक्ना।

```
1 #include <stdio.h>
2 int main()
3 {
4     int i;
5     for (i = 0; i < 10; i++)
6     {
7         if (i == 4)
8         {
9             break;
10        }
11        printf("%d\n", i);
12    }
13    return 0;
14 }
```

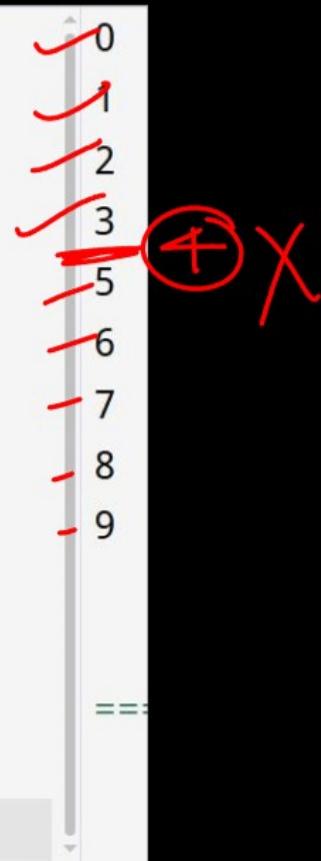
0  
1  
2  
3  
==

## 2. continue →

→ loop के Next iteration में जाना।

```
1 #include <stdio.h>
2 int main()
3 {
4     int i;
5     for (i = 0; i < 10; i++)
6     {
7         if (i == 4)
8         {
9             continue;
10        }
11        printf("%d\n", i);
12    }
13    Break;
14 }
```

next iteration



#### 4. return ↳

→ किसी function से Value return करवाना।

#### 5. exit ↳

→ Program को end करना।

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main()
4 {
5     for(int i=0;i<=7;i++)
6     {
7         if(i==5)
8             exit(0);
9         else
10            printf("%d\n",i);
11    }
12
13 }
```

0  
1  
2  
3  
4  
=====

0  
1  
2  
3  
4

A handwritten note 'x' is located at the bottom left of the slide.

## \* Looping / Iteration ⇒

→ किसी i statement को बार-<sup>2</sup> Run करने से useful.

→ 2 प्रकार -

1. Entry Control Loop
2. Exit Control Loop

## \* Counter Variable ⇒

→ ऐसा variable जिसकी Value loop के प्रत्येक iteration पर change होती है।

(can)

Q- How many Counter variables must be there in any loop?

- A. 1 B. 2 C. ∞ D. NOTA  
E. NOTA

Q. Which loop gives atleast one output whether condition is false or not?

- A. while B. for  
C. dowhile  
D. NOTA  
E. NOTA

\* किसी भी Loop में 3 Parts होते हैं -

- (i) Initialization = loop start कर्ता से करना है।
- (ii) Condition = Loop end कर्ता करना है।
- (iii) Increment / decrement =  $\frac{\text{Execution}}{\text{change in initial value}}$

## 1. Entry Control Loop

- ऐसा loop जिसमें Program का flow "करते समय ही Condition को check कर लेना है, जिससे यदि Condition wrong हो, तो एक भी output नहीं आएगा है।
- जैसे - while & for loop.

Enter

## 2. Exit Control Loop

- ऐसा loop जिसमें program का flow Exit होते समय Condition की check करता है, जिससे यदि condition wrong ही हो, तो भी atleast एक output आयेगा।
- जैसे - do-while.

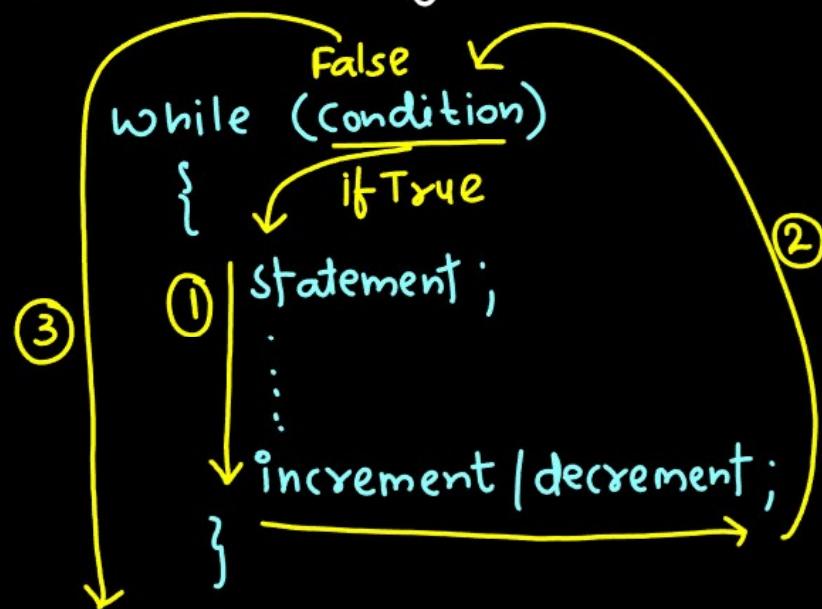
Q ⇒ In which of the following statements program will be executed once?

- A. while
- B. do-while
- C. for
- D. MOTA
- E. NOTA

## \* while loop →

- Entry Control Loop.
- Program को flow loop में entry करते समय condition check करता है।
- No output if condition is false.

Syntax →



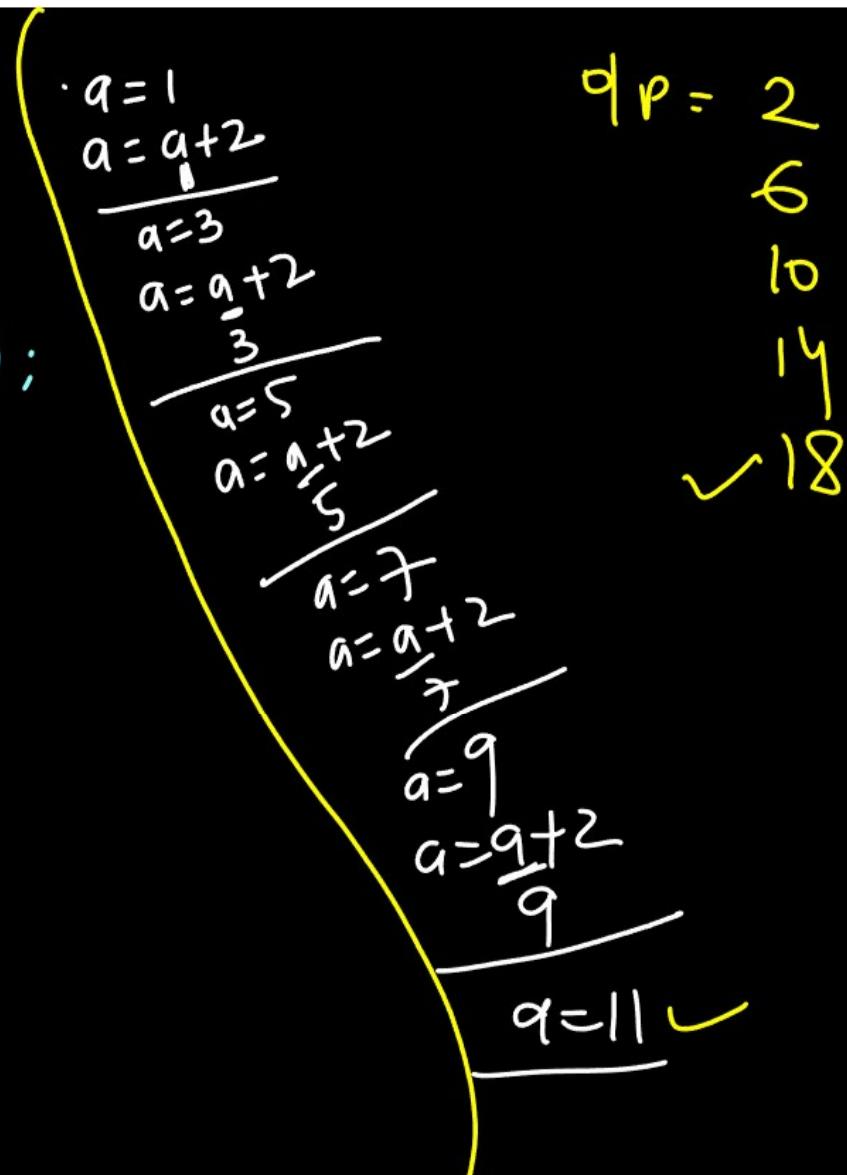
\* while loop 3 steps  
में work करता है।

Ex-1

```
int a = 1;  
if  
while (a <= 10)  
{  
    printf ("%d", a * 2);  
    a += 2;  
}
```

Output  $\Rightarrow$  2 6 10 14 18

Counter Variable = a



Ex int a=5;  
while (a<10)  
{  
 printf ("%d", a);  
}

\* No increment in initial value.  
\* program execute  
\* It will run infinite times.

- A. 5 6 7 8 9
- B. 5 6 7 8
- C. 5 6 7 8 9 10
- D. MOTA
- E. NOTA

Ex=2

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

\* Infinite Times Run  
\* No condition

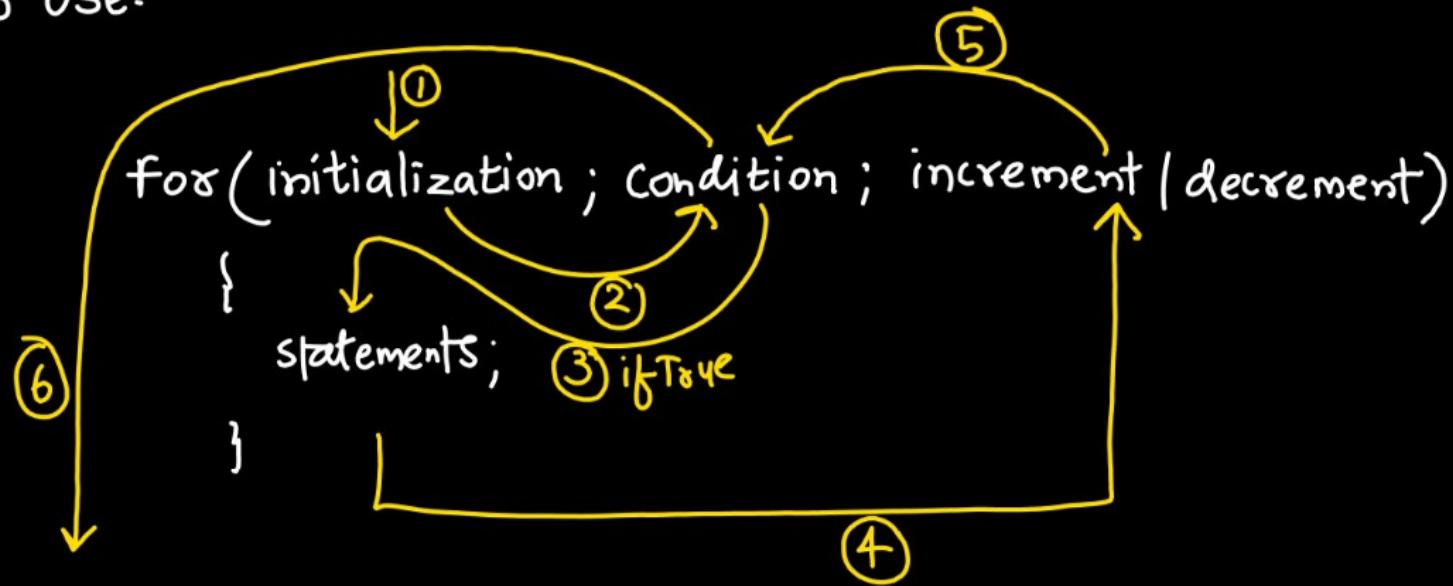
- A. 5 6 7 8 9
- B. 5 6 7 8
- C. 5 6 7 8 9 10
- D. MOTA
- E. NOTA

## ★ for ⇒

→ Most Efficient & User friendly Loop

→ सर्वाधिक Use.

### Syntax ⇒



\* Steps 3, 4 व 5 Repeat होंगी।

\* Step 5 के बाद Repeat.

Q. Which of the following is wrong declaration of for loop?

- ✗ A. for( )
- ✗ B. for( ; )
- ✓ C. for( ; ; )
- ✓ D. for( int a=1; ; )
- ✓ E. for( ; a<5; )
- ✓ F. for( ; ; a++ )

6. MOTA

H. NOTA

\* for loop के तीनों parts empty  
होने पर भी यह चल सकता है,  
केवल semicolon होना चाहिए।

Q = int i=1;  
 ✓for( ; i<10; )  
 {     printf("%d", i);  
 }     break;

How many will this loop run?

- A. 9
- B. 10
- C. 8
- D. MOTA
- E. NOTA



∞

if break is used before printf(),  
 then how many <sup>times</sup> compiler will read  
 for( ; i<10; ) statement?

A. 1

B. 2

C. 3

D. MOTA

E. NOTA

↓  
what will be  
the output?

A. 1

B. 2

C. 3

D. MOTA

E. NOTA → No Output

Q. How many times initial value in for loop will be checked?

A. 1

B. 2

C. 3

D. MOTA

E. NOTA

```
Q = int i=1;  
for(i=1; i<10; i--)  
{  
    printf("%d", i);  
}
```

How many i will be printed?

A. 10

B. 9

C.  $\infty$

D. MOTA

E. NOTA

```
Q = int i=1;  
for(i=1; i<=25; i++)  
{  
    break;  
    printf("%d", i);  
}
```

What will be the last value of i?

A. 25

B. 20

C. 1

D. 2

E. NOTA

## \* do-while loop ⇒

→ Exit control Loop

→ इस loop में program का flow loop से बाहर निकलते समय condition को check करता है, इसलिए यह loop condition के गलत होने पर भी Minimum 1 output देने की guarantee देता है।

Syntax ⇒

```
do
{
    statements;
    increment/decrement;
} while (condition);
```

Ex⇒

```
int a=5;  
do  
{  
    printf ("%d", a/6++);  
    a--;  
} while (a<10);
```

A. 56789.....∞

B. 54321.....∞

C. 55555.....∞

D. MOTA

E. NOTA

## \* Array $\Rightarrow$

- Group of Homogenous Data Types.
- समान प्रकार के Data Types का समूह।

## Syntax $\Rightarrow$

data-type array-name [size/No.of Elements];

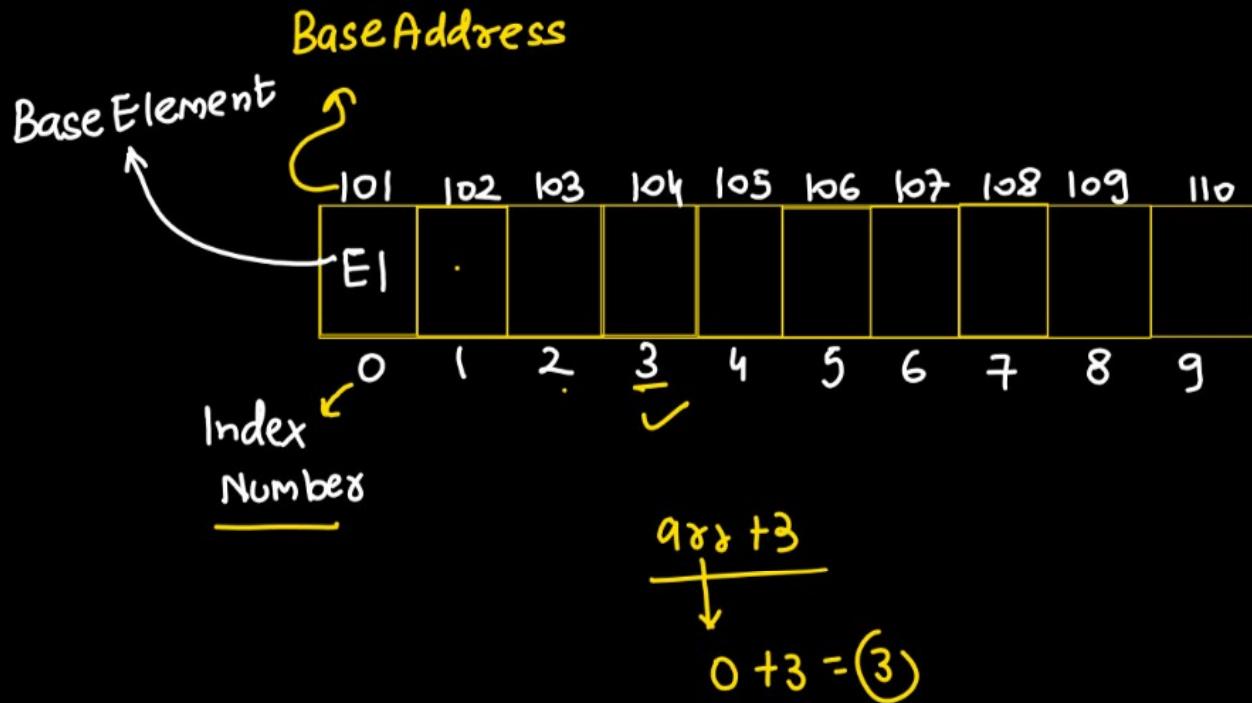
↓              ↓              ↓  
Type of Elements    Name of Collection    Total Elements

int arr [5];



[ ] = Array Subscript

\* Memory Allocation of Array  $\Rightarrow$  int arr[10]; } Memory Allocation = Sequentially



- ()  $\Rightarrow$  Function Call
- []  $\Rightarrow$  Array Subscript
- First Index No. - 0
- First Address - 101
- First Element का Address - Base Address
- सभी Data Elements Sequentially store होते हैं।

## \* Types of Array ⇒

1. One Dimensional Array (1'D)
2. Two Dimensional Array (2'D)

### I. One Dimensional Array ⇒

- Single Direction में ध्लता है।
- 1 Array Subscript का use.
- Total Elements की सख्ता Subscript की size के बराबर होती है।

Syntax ⇒      data-type array-name [size];

int arr [5];

## 2. Two Dimensional Array

- 2 Directions / Axis में चलता है।
- 2 Subscript का use.
  - पहला = Row की size
  - दूसरा = Column की size
- Total Elements = Row X Column

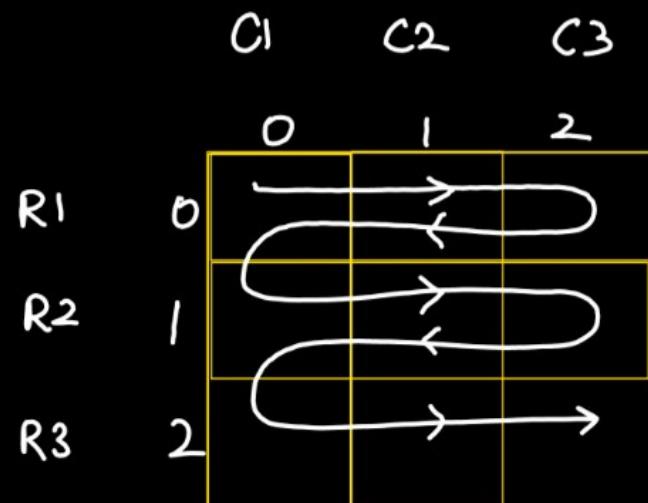
Syntax

data-type array-name [Row] [Column];

int arr[3][3];

↓  
Total Elements =  $\frac{3 \times 3 = 9}{9 \text{ cells}}$

→ Memory Allocation = Matrix के रूप में



Arr [0][0] = पहली Row  
पहली Column

Arr [1][0] = दूसरी Row  
पहला Column

\* 2'D Array में Memory Allocation  
Rowwise होता है, That means एक  
Row के सभी columns जिन्हें के  
बाद ही कुम्भरी Row Allocate होती।

## Examples ⇒

1. int arr[5][7];

What will be storage size of this array?

- A. 35    B. 70  
C. 12    D. MOTA    E. NOTA

→ Size = Machine Dependant

2. int arr[10];

printf("%d", arr[1]);

Output = ?

- A. 0    B. 1    C. 10  
D. MOTA    E. NOTA

→ Garbage Value

$$\text{Total Elements} = \underline{5 \times 7 = 35}$$

0 1 2 — 9

$$\begin{array}{r} \text{arr} = 0 \text{ Index} \\ \hline \text{arr} + 3 \\ 0 + 3 = 3 \end{array} \checkmark$$

3. int arr[10];

printf("%d", arr+3);

kis Index Number wali Element ko display  
dega?

- ✓ A. 3    B. 4    C. 2  
D. MOTA    E. NOTA

$\text{arr} \Rightarrow$  First Index Number (0) = 1<sup>st</sup> value

$\text{arr}[1] \Rightarrow$  1 index Number पर Value = 2<sup>nd</sup> value

$$\frac{\text{arr} + 3}{0 + 3} = \frac{3^{\text{rd}} \text{ index}}{4^{\text{th}} \text{ value}}$$

$\text{arr} = \text{index}$

$\text{arr}[0] = \text{पहली value}$

## \* String ⇒

- Group of characters / Array of characters
- String से related functions "string.h" header file में होते हैं।

## \* String functions ⇒

### 1. strlen() ⇒

- String की length बताता है।
- Space भी count करता है।

```
#include <stdio.h>
#include <string.h>✓
int main()
{
    ✓
    char str[] = "whoshubhamsir";
    int len = strlen(str);
    printf("Length of string is : %d", len);
    return 0;
}
```

Output:

Length of string is : 13

## 2. strcpy() :-

→ एक String में दुखरी String को Copy करता।

Syntax :-

strcpy (s1, s2);

s2, s1 में copy होगी।

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[] = "Who Shubham Sir";
    char s2[50];
    strcpy(s2, s1);
    printf("%s", s2);
    return 0;
}
```

Output:

Who Shubham Sir

### 3. strcat() ⇒

- दो strings को जोड़ता।
- पहली string के last से दूसरी string को add कर देता है।
- Space का usage करता है।

Syntax ⇒    `strcat(destination, source)`

```
#include <stdio.h>
#include <string.h>
int main()
{
    char S1[100] = "Who";
    char S2[100] = "Shubham Sir";
    strcat(S1,S2);
    printf("%s\n", S1);
    return 0;
}
```

Output:

WhoShubham Sir

4. strcmp() ⇒

→ दो strings को आपस में compare करता |

```
#include <stdio.h>
#include <string.h>
int main()
{
    char* s1 = "whoshubhamsir";
    char* s2 = "whoshubhamsir";
    printf("%d", strcmp (s1, s2));
    return 0;
}
```

Output:

0 ✓

%s = string  
%d = Number

## \* String =>

- Group of characters / Array of characters
- String से related functions "string.h" header file में दीते हैं।

## \* String functions =>

### 1. strlen() =>

- String की length बताता है।
- Space भी count करता है।

```
#include <stdio.h>
#include <string.h>✓
int main()
{
    ✓
    char str[] = "whoshubhamSir";
    int len = strlen(str);
    printf("Length of string is : %d", len);
    return 0;
}
```

Output:

Length of string is : 13

## 2. strcpy() =

→ एक String में दुखरी String को Copy करना।

Syntax =

strcpy(s1, s2);

s2, s1 में copy होगी।

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[] = "Who Shubham Sir";
    char s2[50];
    strcpy(s2, s1);
    printf("%s", s2);
    return 0;
}
```

Output:

Who Shubham Sir

### 3. Strcat()

- दो strings को जोड़ता है।
- पहली string के last में  
दूसरी string को add कर  
देता है।
- Space का usage करता है।

Syntax :-    `strcat(destination, source)`

```
#include <stdio.h>
#include <string.h>
int main()
{
    char S1[100] = "Who";
    char S2[100] = "Shubham Sir";
    strcat(S1,S2);
    printf("%s\n", S1);
    return 0;
}
```

Output:

WhoShubham Sir

;

4. strcmp() ⇒

→ दो strings को आपस में compare करता |

```
#include <stdio.h>
#include <string.h>
int main()
{
    char* s1 = "whoshubhamsir";
    char* s2 = "whoshubhamsir";
    printf("%d", strcmp(s1, s2));
    return 0;
}
```

Output:

0 ✓

\* Characters की ASCII की Read करता है।

0 = Equals  
- = S1 < S2  
+ = S1 > S2

%s = String  
%d = Number ✓

### 5. strrev() ⇒

→ String को reverse करके display करता।

Ex ⇒

```
char s1[10] = "Ram";
printf("%s", strrev(s1));
```

O/P ⇒ mar

7. strlwr() ⇒ String को  
small letters में display  
करता।

strlwr ("RAM")  
= ram

6.strupr() ⇒ String को capital letters में convert करके display करता।

strupr ("Ram") = RAM

## 8. strstr()

→ किसी एक substring की availability का पता करना।

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str1[20] = "whoshubhamsir";
    char str2[20] = "sir";
    printf("Found : %s", strstr(str1, str2));
    return 0;
}
```

Output:

✓ Found : sir ✓

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str1[20] = "whoshubhamsir";
    char str2[20] = "sir";
    printf("Found : %d", strstr(str1, str2));
    return 0;
}
```

Output:

Found : -476582166 ✓

## \* Function →

- C Language में किसी भी प्रोग्राम को Basic Building Blocks में divide कर दिया जाता है, जिनमें function कहते हैं।
- प्रत्येक function में Multiple statements होती है, जिनमें Parenthesis ({ }) के बीच रखा जाता है।

```
function ()  
{  
    Body;  
}
```

- C language में Modularity & Reusability का feature होने के कारण function को बार-<sup>2</sup> use में लिया जा सकता है।

Modularity = Divide in Modules / Reusability = use many times

→ C language में Function की procedure & subroutine भी कहते हैं।

### \* Advantages of Function ⇒

1. Same code को बार-<sup>2</sup> Repeat करने की Need नहीं।
2. एक function को infinite times call किया जा सकता है।
3. किसी बड़े program को छोटे-<sup>2</sup> multiple functions में divide किया जा सकता है।

## \* Aspects of Function →

A. Function Declaration

B. Function Call

C. Function Definition

### A. Function Declaration ⇒

return-type      function-name (Arguments);  
                ↓                ↓                    ↓  
            Void add (int a, int b);

## B. Function Call $\Rightarrow$

function-name (Arguments);

add (5, 6);

## C. Function Definition $\Rightarrow$

return-type function-name (Arguments)  
{  
    statements;  
}

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

void add (int a, int b); — Declaration

add(5,7); — call

void add (int a, int b) {  
 int c = a + b;  
 printf("add", c); } } — Definition

## \* Type of functions →

A. Built-in / Library functions

B. User defined functions

### A. Built-in / Library functions →

→ Predefined functions in header files of C.

Ex = getch(), putch(), printf(), scanf(),  
main(), gets(), puts() etc.

## B. User Defined Functions

→ ऐसे functions जिन्हें कोई user अपनी ज़रूरतों के आधार पर multiple usage के लिए बनाता है।

## \* Type of functions ↗

A. Built-in / Library functions

B. User defined functions

### A. Built-in / Library functions ↗

→ Predefined functions in header files of C.

Ex = getch(), puts(), printf(), scanf(),  
main(), gets(), puts() etc.

## B. User Defined Functions

- ऐसे functions जिन्हें कोई user अपनी ज़रूरतों के आधार पर multiple usage के लिए बनाता है।
- प्रत्येक user defined function किस प्रकार की value return करेगा, उह बनाने के लिए function-name से पहले उसका return-type बताना ज़रूरी होता है।

जैसे-

```
int add()  
{  
    statements;  
}  
return  
type
```

Void add();

→ उह वास्तव में कोई return-type नहीं होता है, उह केवल printf वाली value को display करता है।  
→ Void = Null

## \* Call by Value v/s Call by References ⇒

Call by value	Call by Reference
1. इसमें function को call करते समय Variable की value pass की जाती है।	1. इसमें function को call करते समय variable का Address/ Reference Pass की जाती है।
2. इस Method में calling function के प्रत्येक variable की value called function के dummy variables से pass की जाती है।	2. इस Method में calling function के प्रत्येक variable का address called function के dummy variables में copy किया जाता है।

3. Called function के variables से changes करने से calling function के variables पर कोई भी Effect नहीं पड़ता है।

3. Called function के variables में changes करने से calling function के variables पर भी Effect पड़ता है।

```

#include <stdio.h>
void swap(int x, int y)
{
    int temp = x;
    x = y;
    y = temp;
}
int main()
{
    int a = 40, b = 50;
    printf("Before swap, a = %d and b = %d\n", a, b);
    swap(a, b);
    printf("After swap, a = %d and b = %d\n", a, b);
    return 0;
}

```

Called = दुसरा function  
value देता है  
Calling = दुखरे function  
को values  
देता है

Output:

Before swap, a = 40 and b = 50

After swap, a = 40 and b = 50

```
#include <stdio.h>
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
int main()
{
    int x = 5;
    int y = 10;
    printf("Before swap: x = %d, y = %d\n", x, y);
    swap(&x, &y);
    printf("After swap: x = %d, y = %d\n", x, y);
    return 0;
}
```

Output:

Before swap: x = 5, y = 10

After swap: x = 10, y = 5

## \* Dynamic Memory Allocation $\Rightarrow$

int arr [5];  
↓

\* Static Memory Allocation / Compile Time

\* By Default

### Static

1. Compile Time पर
2. इस Allocation में size compilation के Time पता लग जाती है।
3. LIFO का use होता है। (Stack)
4. More Memory
5. Fast Execution

जैसे-      int a=5;

### Dynamic

1. Run Time पर
2. इस Allocation में size compilation के Time पता नहीं लगती है।
3. Assignment का order नहीं होता है।
4. Less Memory
5. Slow Execution

जैसे-      int a;



A. malloc() ⇒

→ Memory Allocation

→ प्रदृ उपर की requested size का memory block allocate करती है।

char()

B. calloc() ⇒

→ Contagious Allocation

→ User को variable length की size का memory block allocate करती है

varchar()

→ malloc() की तुलना में Memory wastage कम।

C. realloc() ⇒

→ Re-Allocation

→ malloc() तथा calloc() के द्वारा allocate की गई size/Block को  
Reallocate करना।

D. free() ⇒

→ malloc() तथा calloc() के द्वारा allocate किये गये blocks को free/ release  
करना।

\* Pointer =>

→ ऐसा variable जो दूसरे variable के Address को store करता है।

int a;      Normal Variable

int \*a;      Pointer Variable  
                Pointer to integer

Ex => int a = 5;

int \*b = &a;

↓  
Address of Operator

\* b में a का address store.

```

#include <stdio.h>
int main()
{
    int var = 10;
    int *ptr = &var; // Address of var to ptr
    printf("%d", ptr);      printf("%d", *ptr);
    return 0;
}

```

$\text{ptr} = \text{address}$   
 $\text{display}$   
 $\text{करेगा।}$   
 $*\text{ptr} = \text{Address से}$   
 $\text{value display}$   
 $\text{करेगा।}$   
 $= 10$

## \* Pointer to function $\Rightarrow$

→ ऐसा variable जो किसी function के Address को store करता है।

```
#include <stdio.h>
int add(int a, int b)
{
    return a + b;
}
int main()
{
    int (*fptr)(int, int);
    fptr = &add;
    printf("Add", fptr(10, 5));
    return 0;
}
```

O/p = 15