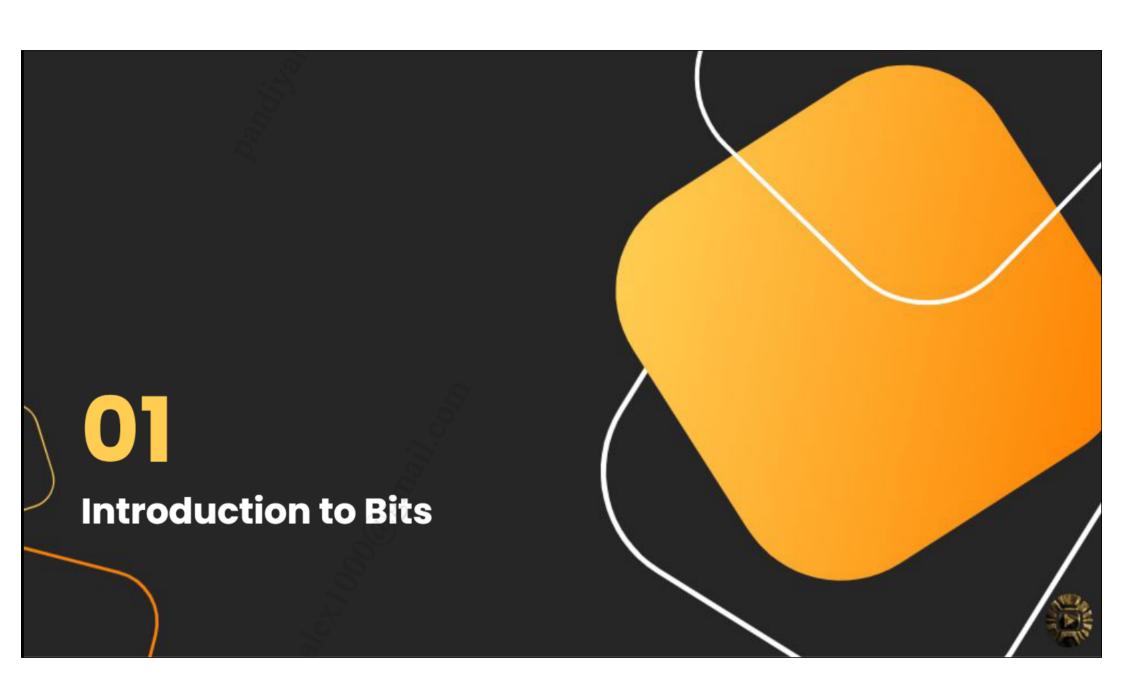


CONTENTS

- 1. Introduction to Bits
- **2.** Bitwise Operators and Mask
- **3.** Applications of Bits
- 4. Activity





• Introduction to Bits

What is Bit Manipulation?

- > Performing operations directly on bits of numbers
- ➤ Works at the **lowest level** of data (binary level)
- Used for optimization, encoding, flags, etc.



• Introduction to Bits

Why Bit Manipulation?

- Super fast (direct hardware support)
- Saves memory (can store multiple states in 1 integer)
- ➤ Essential in:
 - o Competitive programming
 - o Embedded systems
 - Cryptography
 - o Game dev, OS internals





➤ Operators

- o OR (|)
- o AND (&)
- o XOR (^)
- Complement (~)
- o Right Shift (>>)
- o Left Shift (<<)



Bitwise AND (&):

- Returns 1 only if both bits are 1
- Used for masking bits, clearing flags

Truth Table:

Α	В	A&B
0	0	0
0	1	0
1	0	0
1	1	1

Example :- 5 & 3 = 0101 & 0011 = 0001 \rightarrow 1



Bitwise OR (|):

- Returns 1 if atleast one bit is 1
- Used for setting bits

Truth Table:

Α	В	A B
0	0	0
0	1	1
1	0	1
1	1	1

Example :- 5 | 3 = 0101 | 0011 = 0111
$$\rightarrow$$
 7



Bitwise XOR (^):

- Returns 1 only if the bits are different
- Used for **toggle** or finding differences

Truth Table:

A	В	A^B
0	0	0
0	1	1
1	0	1
1	1	0

Example :- 5 ^ 3 = 0101 ^ 0011 = 0110 \rightarrow 6



Bitwise NOT (~):

- Inverts all bits
- Unary operator

Truth Table:

Α	~A
0	1

A (binary)	~A (binary)	A (decimal)	~A (decimal)
0000	1111	0	-1
0001	1110	1	-2
0010	1101	2	-3
0101	1010	5	-6



Bitwise NOT (~):

- Inverts all bits
- Unary operator

Truth Table:

Α	~A
0	1

A (binary)	~A (binary)	A (decimal)	~A (decimal)
0000	1111	0	-1
0001	1110	1	-2
0010	1101	2	-3
0101	1010	5	-6



Left Shift (<<)

- Shifts bits **left**, fills with 0s on the right
- Equivalent to num × 2ⁿ

Shift Table:

Input	Shift	Output
0001	<< 1	0010
0011	<< 1	0110
0101	<< 2	10100

Example :-
$$5 << 1 = 0101 << 1 = 1010 \rightarrow 10$$



Right Shift (>>)

- Shifts bits **right**, discards rightmost bits
- Signed: fills with sign bit (Java/C++)

Shift Table:

Input	Shift	Output
0100	>> 1	0010
0110	>> 1	0011
1010	>> 1	1101 (signed shift)

Example :- 5 >> 1 = 0101 >> 1 = 0010
$$\rightarrow$$
 2



Masks

Masks (particular combination of bits) can be created and used in conjunction with the Bitwise Operators to:

- Set a particular bit
- Get a particular bit
- Clear a particular bit

```
Get a particular bit

java

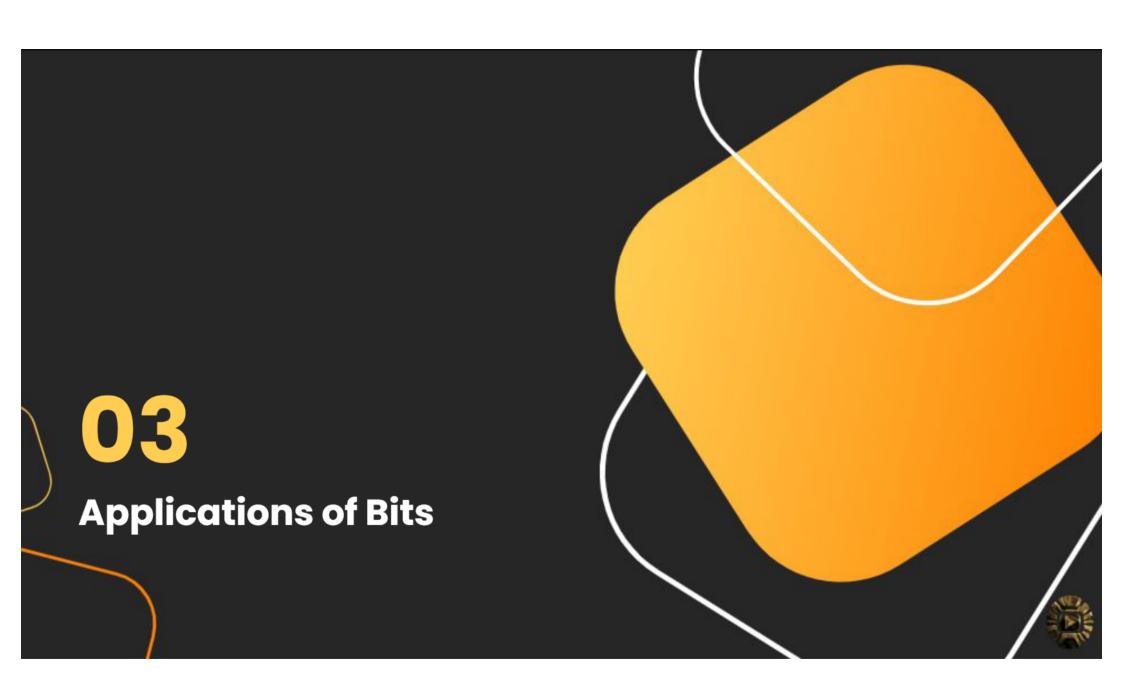
○ Copy ▷ Edit

int num = 5; // 0101

int pos = 2;

int mask = 1 << pos;

int bit = (num & mask) >> pos; // result: 0101 & 0100 = 0100 → bit = 1
```



Applications of Bits

- Media Files (e.g., MP3, MP4, JPG use bit-level encoding)
- Networking (e.g., MP3, MP4, JPG use bit-level encoding)
- Security (e.g., AES, RSA, hashing)
- ➤ Gaming (e.g., Collision maps, status toggles, and item collections are stored as bits)
- > Flag handling (Use bit masks to set, clear, and toggle flags efficiently)
- Compression (e.g., compressing large arrays)
- Optimization (Fast math operations like multiplication/division by powers of 2 (x << n, x >> n))
- > Searching / Algorithms (e.g., Used in Bitmask DP, Trie-based search, and subset generation)





Activity Graph

```
Single Number

Find the Duplicate Number

Counting Bits

Number of 1 bits

Missing Number

Reverse Bits

Add Binary
```

