

Create your First Program

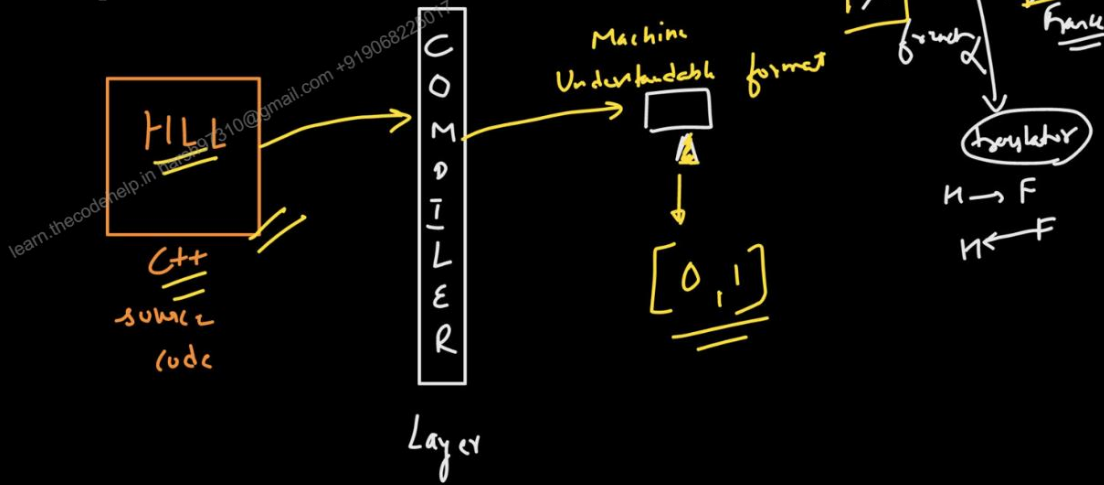
Instructor: Love Babbar

Programming Languages

Why we need it ?

- A Language using which, we can instruct the computer to carry out real life tasks and computations is called a programming language. It acts as a language in which we could easily express our thoughts to the machine.
- Like natural languages, programming language has a fixed set of rules according to which programs could be written in it. These programs are then converted into a language which machines can understand. This task is carried out by a special software called compiler.
- Every language has its own compiler/Interpreter → Refrah
- Once a program is compiled and linked, its executable is created and the computer can run our program now.

Compilation Process



IDE:

Setup
↳ Replit

Code Editor
or

IDE's

↳ Code Blocks

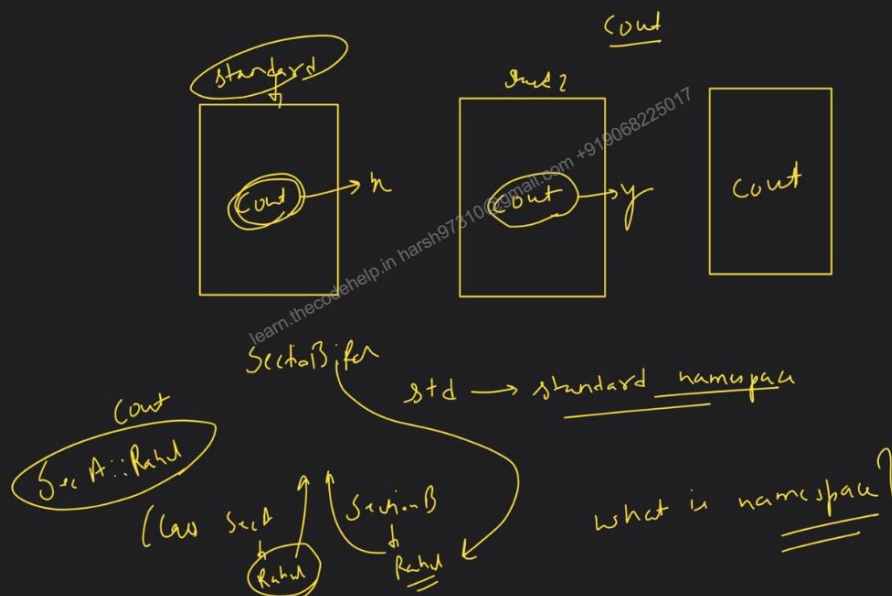
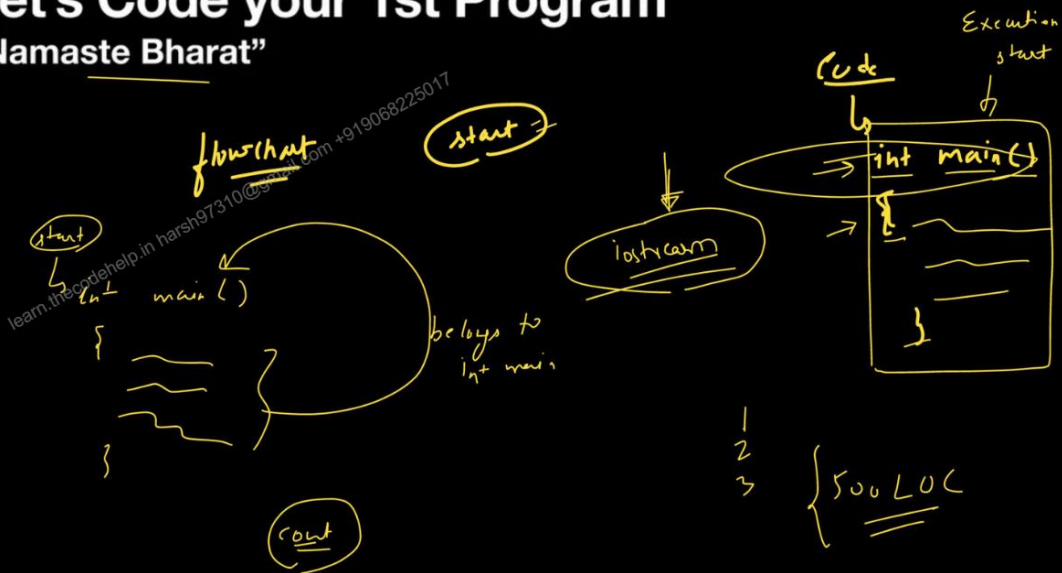
↳ VS Code

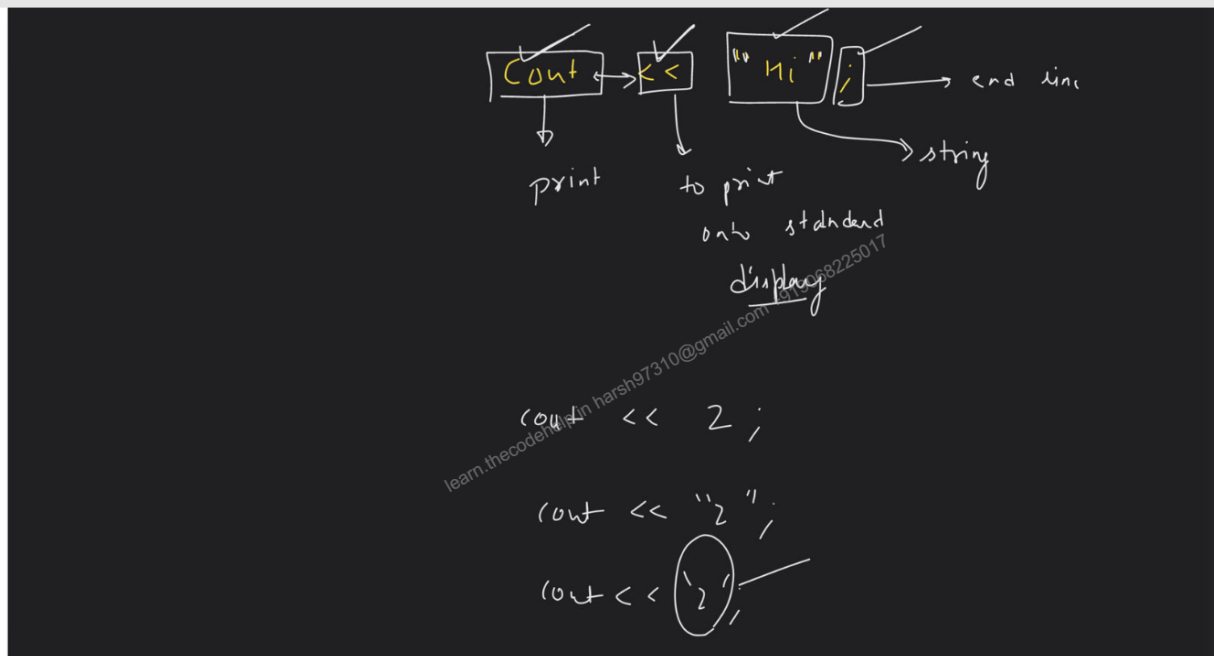
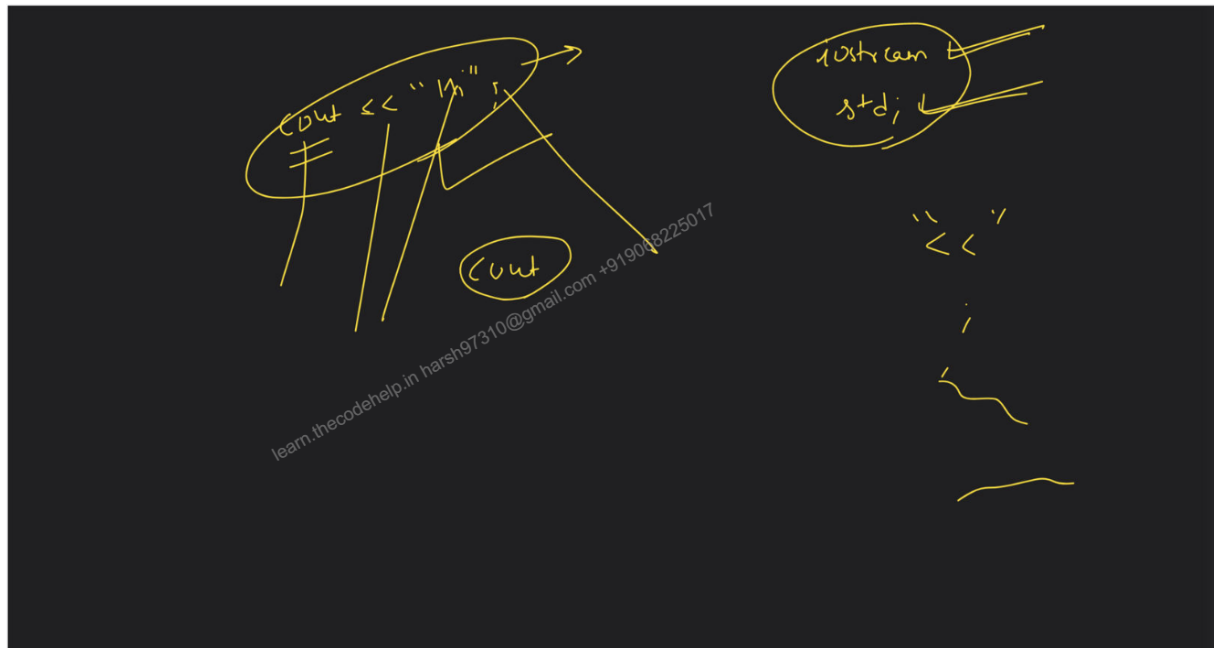
↳ Y Code

↳ Sublime

Let's Code your 1st Program

"Namaste Bharat"





Taking Input in C++

print → cout
input → cin

SKIP

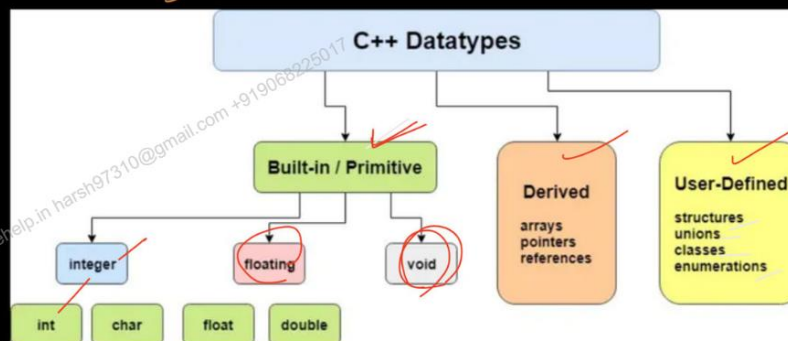
Datatypes & Variables

A variable in C is a memory location associated with some name in order to store some form of data and retrieve it when required & data types are used to tell the variables the type of data they can store.

int b = 7
7
b
type of data

void ?

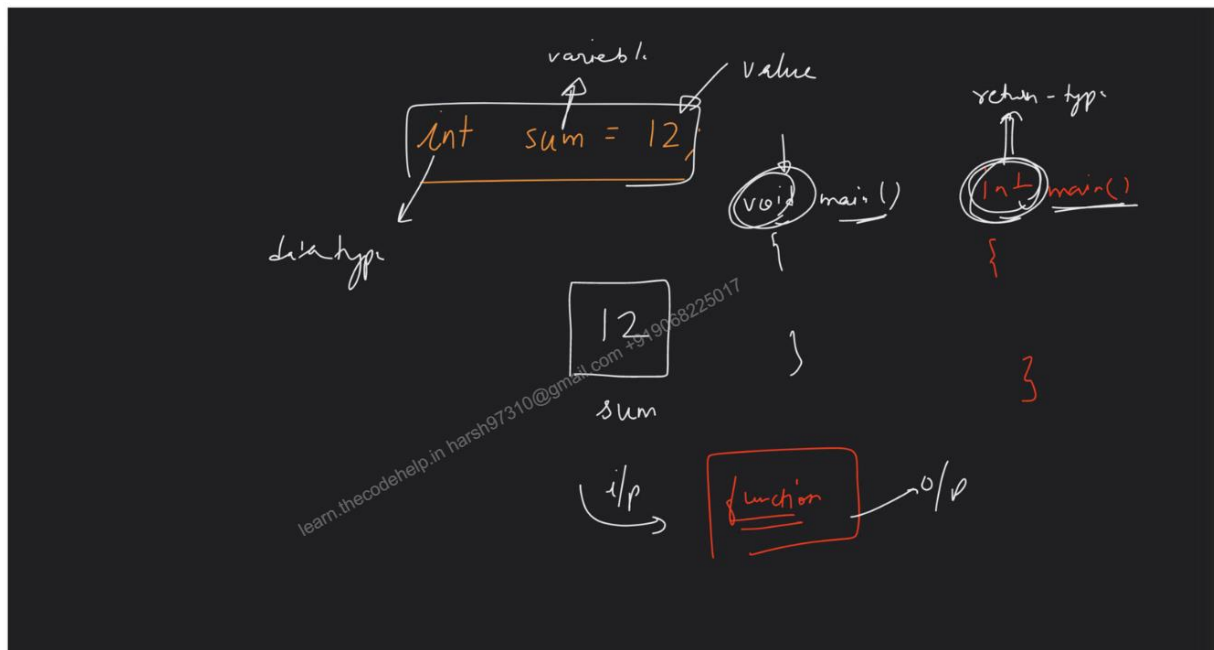
variable ?



3
sum

int a = 5

5



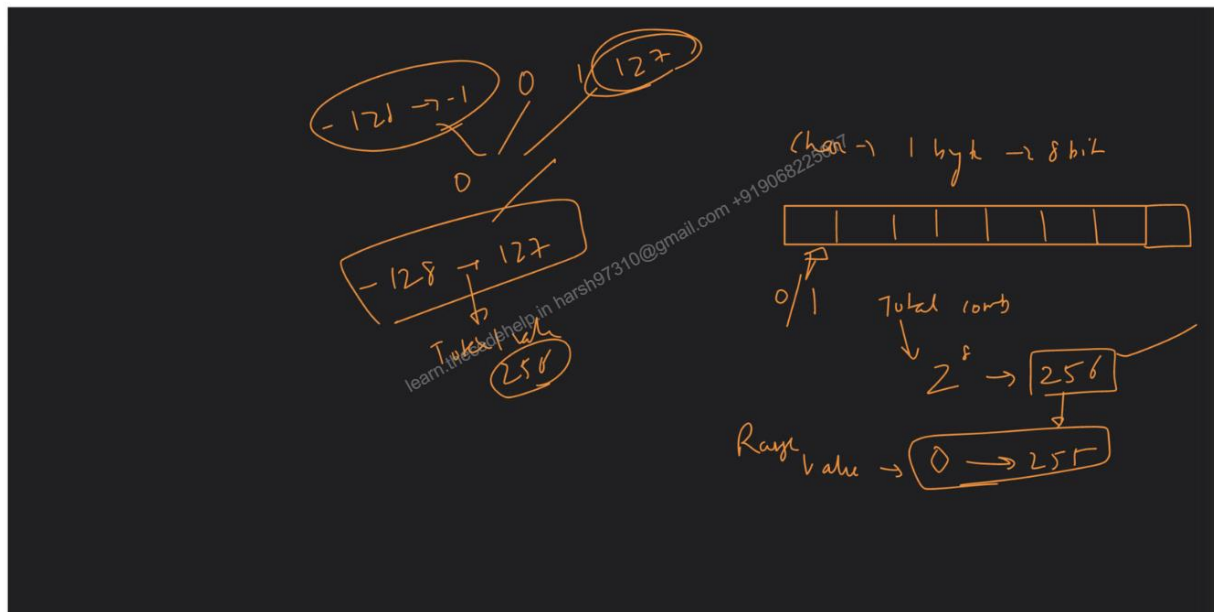
DataTypes: → which type of data u will store

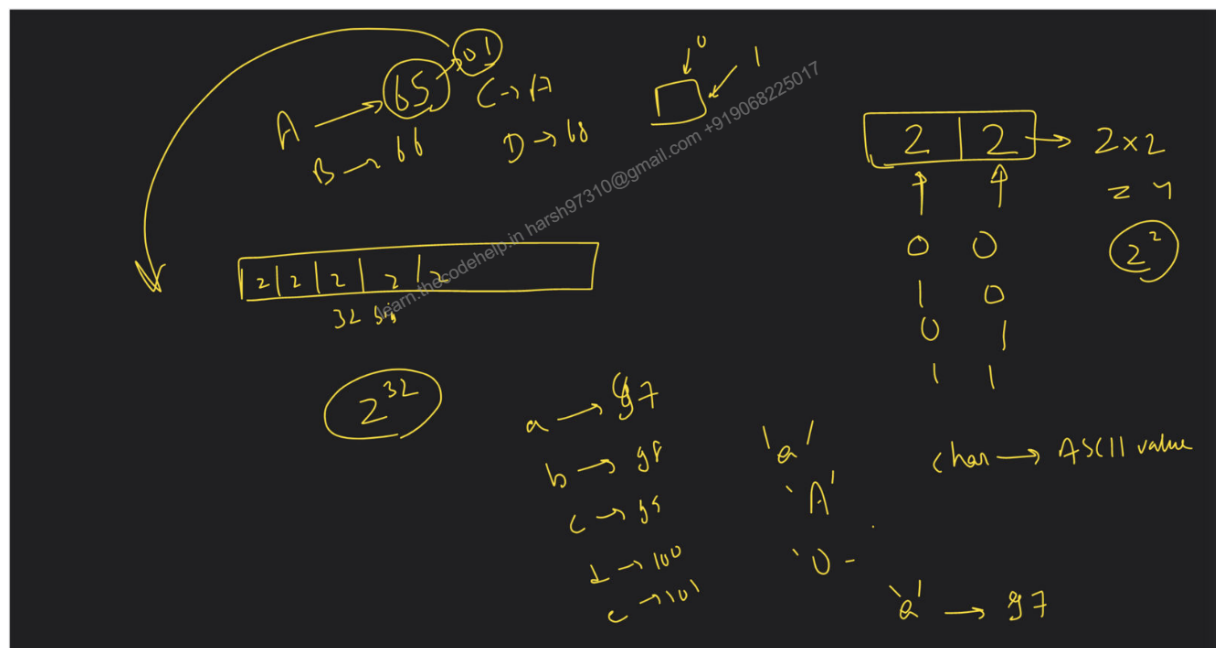
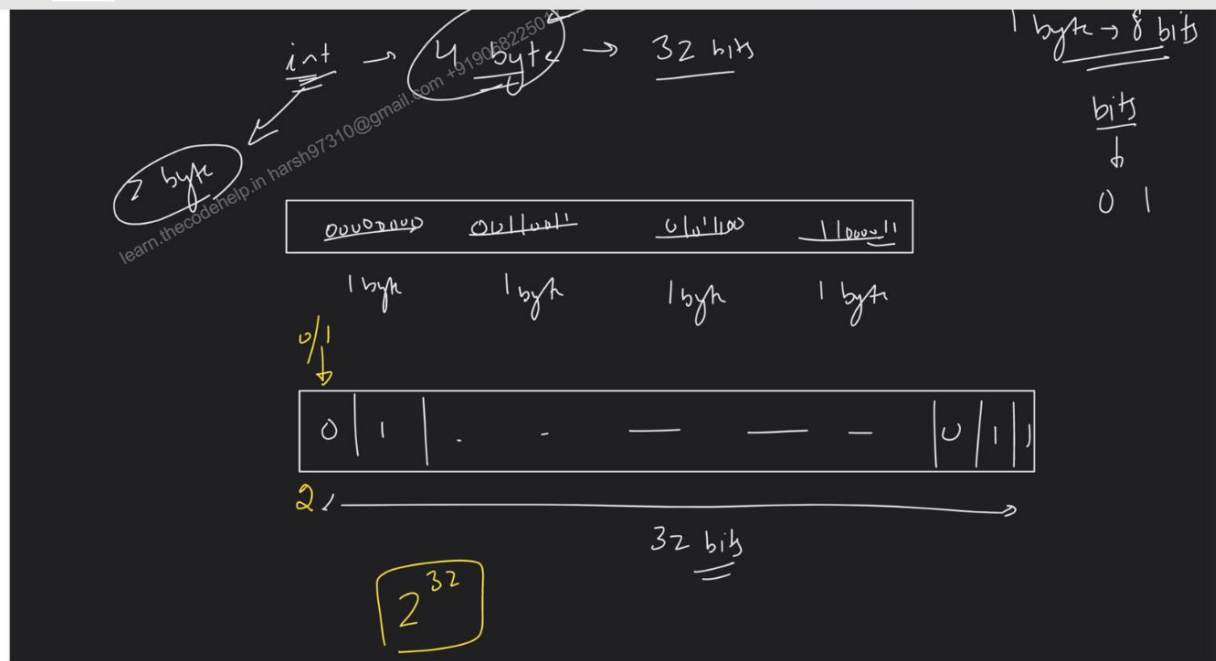
Handwritten notes: 255, 32-bit vs 64-bit CPU, void printName { int <= 4; } long long <= 17+3; int → 4 → 32 bit long long → 8 → 64 bit

C Basic Data Types	32-bit CPU	64-bit CPU
	Size (bytes)	Range
char	1	-128 to 127
short	2	-32,768 to 32,767
int	4	-2,147,483,648 to 2,147,483,647
long	4	-2,147,483,648 to 2,147,483,647
long long	8	9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4	3.4E +/- 38
double	8	1.7E +/- 308

Handwritten notes on the right side of the table:

- void printName { int <= 4; }
- long long <= 17+3;
- int → 4 → 32 bit
- long long → 8 → 64 bit
- 2³² - 1





char ch = 256

Explor

ASCII table

Chg' Chg'

g7

60/07
Ch

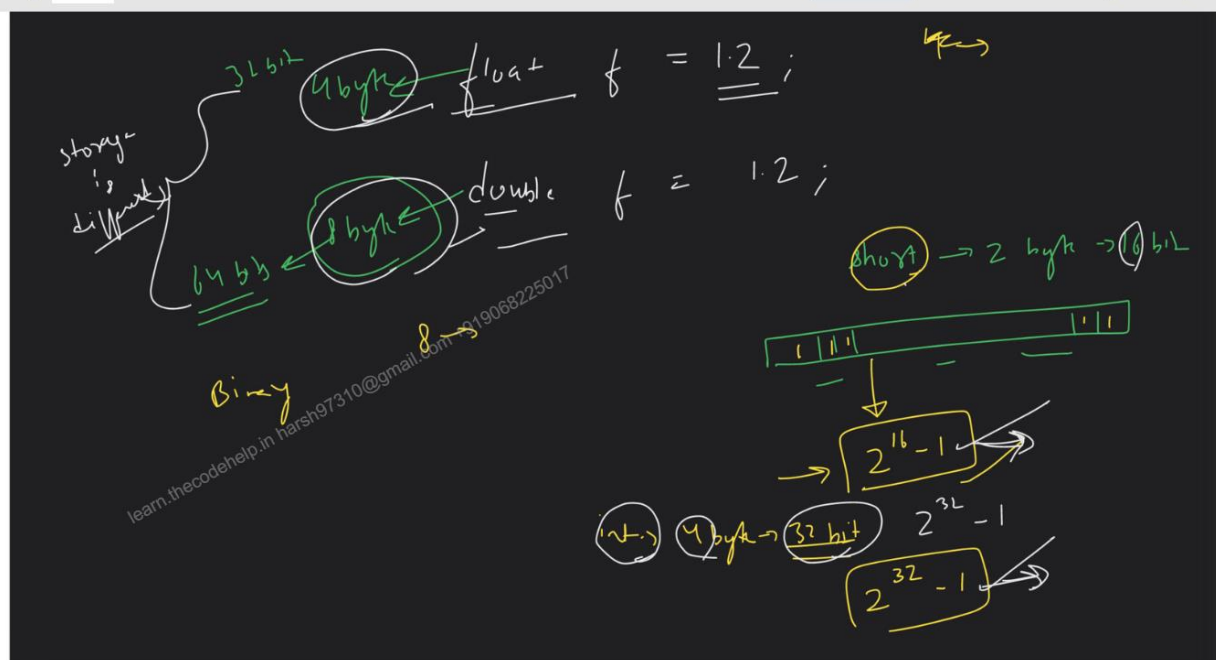
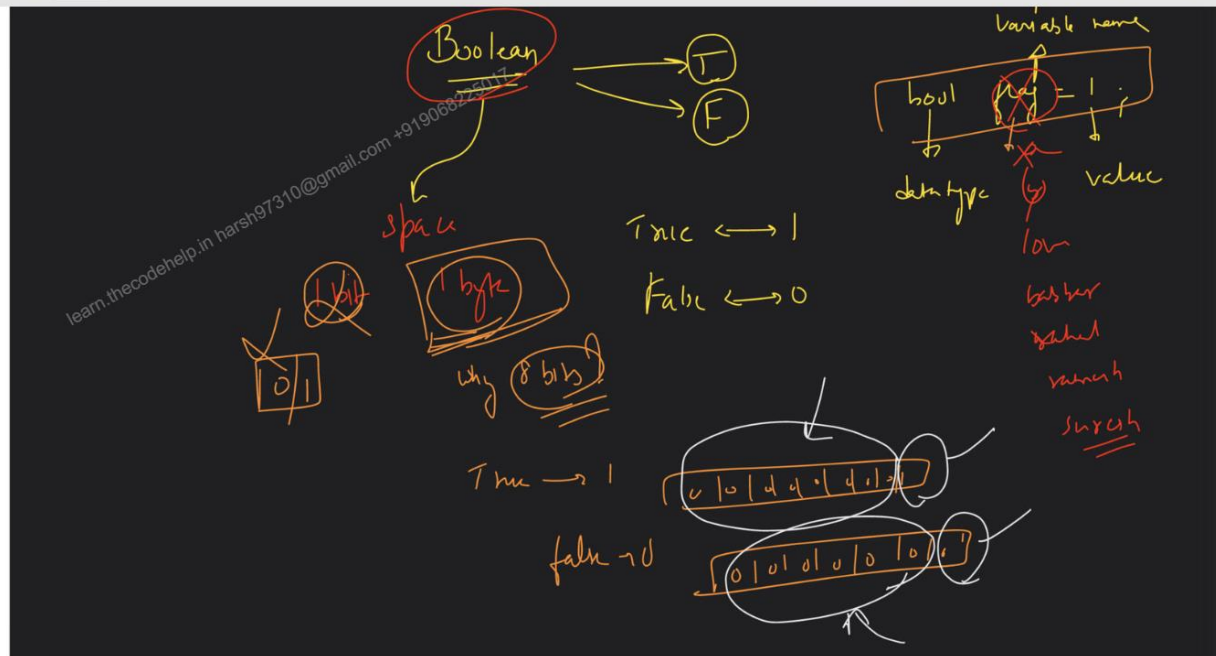
chan

$2^8 - 1 = 255$

(char -) 256

The diagram illustrates the memory layout and address calculation for a character array `ch`.

- Memory Layout:** A horizontal array of 10 cells is shown. The first 9 cells contain the value `2`, and the 10th cell contains `0`. Below the array, the address range `2x2x2x2x2x2x2x2x2x2` is written, indicating a 10x2x2x2x2x2x2x2x2x2x2 structure. To the left, the values `0/1 0/1` are written with arrows pointing to the first two cells.
- Character Declaration:** A box labeled `char ch = 'a';` is shown. An arrow points from this box to the 10th cell of the array.
- Address Calculation:**
 - The address `28` is circled, with an arrow pointing to the 10th cell.
 - The address `29` is circled, with an arrow pointing to the 10th cell.
 - The address `210` is circled, with an arrow pointing to the 10th cell.
 - The address `28` is circled, with an arrow pointing to the 10th cell.
 - The address `28` is circled, with an arrow pointing to the 10th cell.
 - The address `28` is circled, with an arrow pointing to the 10th cell.
 - The address `28` is circled, with an arrow pointing to the 10th cell.
 - The address `28` is circled, with an arrow pointing to the 10th cell.
 - The address `28` is circled, with an arrow pointing to the 10th cell.
 - The address `28` is circled, with an arrow pointing to the 10th cell.
- Final Address:** The address `512` is circled, with an arrow pointing to the 10th cell.



Handwritten notes on a blackboard background showing binary representations and calculations.

Left side calculations:

- $2^2 - 1 \rightarrow 11$
- $2^3 - 1 \rightarrow 111$
- $2^4 - 1 \rightarrow 1111$

Right side calculations:

- $111 \rightarrow 7$
- $1111 \rightarrow 15$
- $11111 \rightarrow 31$
- $111111 \rightarrow 63$
- $1111111 \rightarrow 127$
- $11111111 \rightarrow 255$
- $111111111 \rightarrow 511$
- $1111111111 \rightarrow 1023$
- $11111111111 \rightarrow 2047$
- $111111111111 \rightarrow 4095$
- $1111111111111 \rightarrow 8191$
- $11111111111111 \rightarrow 16383$
- $111111111111111 \rightarrow 32767$
- $1111111111111111 \rightarrow 65535$
- $11111111111111111 \rightarrow 131071$
- $111111111111111111 \rightarrow 262143$
- $1111111111111111111 \rightarrow 524287$
- $11111111111111111111 \rightarrow 1048575$
- $111111111111111111111 \rightarrow 2097151$
- $1111111111111111111111 \rightarrow 4194303$
- $11111111111111111111111 \rightarrow 8388607$
- $111111111111111111111111 \rightarrow 16777215$
- $1111111111111111111111111 \rightarrow 33554431$
- $11111111111111111111111111 \rightarrow 67108863$
- $111111111111111111111111111 \rightarrow 134217727$
- $1111111111111111111111111111 \rightarrow 268435455$
- $11111111111111111111111111111 \rightarrow 536870911$
- $111111111111111111111111111111 \rightarrow 1073741823$
- $1111111111111111111111111111111 \rightarrow 2147483647$

Variable Naming Conventions [Explore]

Handwritten notes on a blackboard background showing variable naming conventions.

Left side:

- $2^2 - 1 \rightarrow 11$
- $2^3 - 1 \rightarrow 111$
- $2^4 - 1 \rightarrow 1111$

Right side:

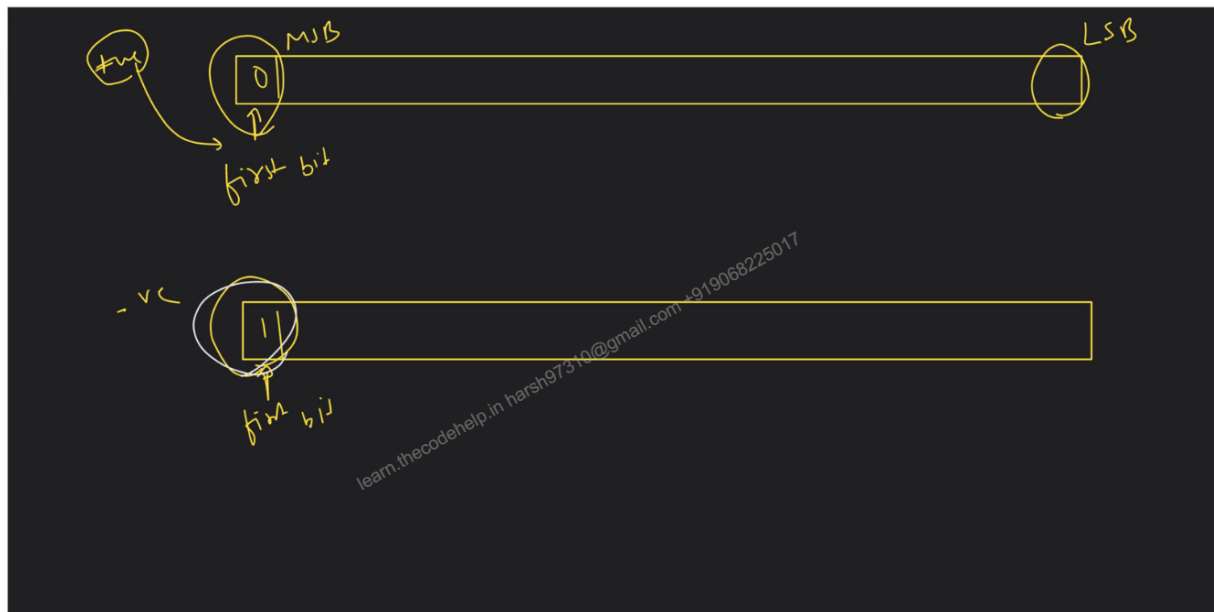
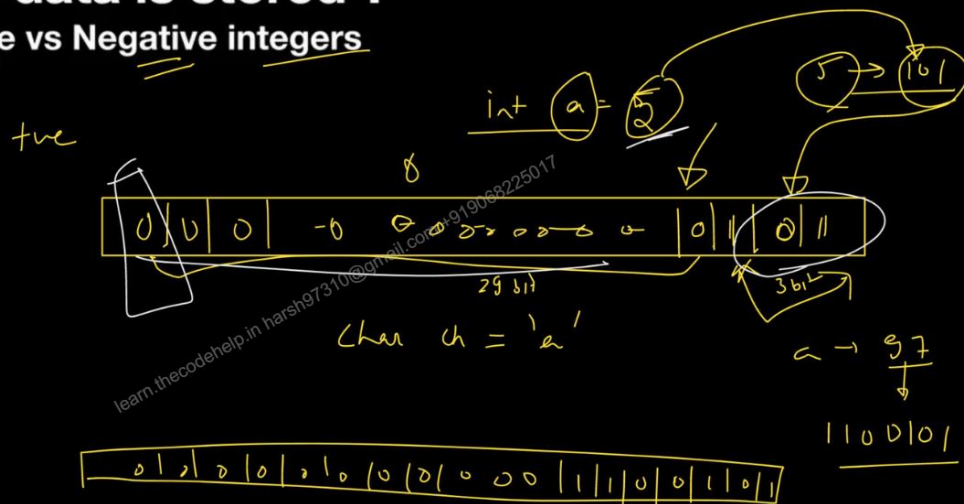
- $111 \rightarrow 7$
- $1111 \rightarrow 15$
- $11111 \rightarrow 31$
- $111111 \rightarrow 63$
- $1111111 \rightarrow 127$
- $11111111 \rightarrow 255$
- $111111111 \rightarrow 511$
- $1111111111 \rightarrow 1023$
- $11111111111 \rightarrow 2047$
- $111111111111 \rightarrow 4095$
- $1111111111111 \rightarrow 8191$
- $11111111111111 \rightarrow 16383$
- $111111111111111 \rightarrow 32767$
- $1111111111111111 \rightarrow 65535$
- $11111111111111111 \rightarrow 131071$
- $111111111111111111 \rightarrow 262143$
- $1111111111111111111 \rightarrow 524287$
- $11111111111111111111 \rightarrow 1048575$
- $111111111111111111111 \rightarrow 2097151$
- $1111111111111111111111 \rightarrow 4194303$
- $11111111111111111111111 \rightarrow 8388607$
- $111111111111111111111111 \rightarrow 16777215$
- $1111111111111111111111111 \rightarrow 33554431$
- $11111111111111111111111111 \rightarrow 67108863$
- $111111111111111111111111111 \rightarrow 134217727$
- $1111111111111111111111111111 \rightarrow 268435455$
- $11111111111111111111111111111 \rightarrow 536870911$
- $111111111111111111111111111111 \rightarrow 1073741823$
- $1111111111111111111111111111111 \rightarrow 2147483647$

Variable Naming Conventions:

- $int\ a = 5$ (underlined a)
- $int\ studentCount = 5$ (underlined $studentCount$)

How data is stored ?

Positive vs Negative integers



Now -ve no are stored in memory!

2's complement

1's complement + 1

1's complement

00101011

1's complement

11010100

+1

2's complement

11010101

1's complement → flip bits

0 → 1

1 → 0

7 → 111

7 → 0000111

1's → 1111000

+1

2's

1111001

MSB or first bit

0 → +ve

1 → -ve

int a = -5

decimal
 $\frac{1}{2} \rightarrow 10$
 $\frac{1}{2} \rightarrow 10$

1 + 1 → 10

Binary ⇒

1
1
10

0000 1111
 + 1
 0001 0000

① ignore -ve sign
 5

② Binary equivalent
 0000 1111

③ 2's complement

000 - - - 00 101

1's C → 111 - - - 11 010

+ 1

2's → 111 - - - 11 011

Read → 2's complement

1's comp → 000 - - - 00 100

+ 1

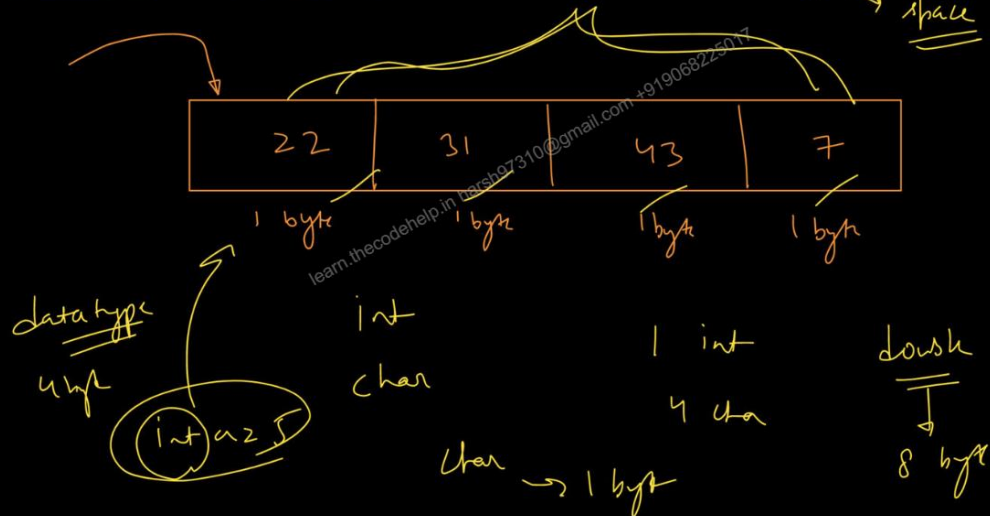
2's → 000 - - - 00 101 ⇒ 5

(out < 6)

Interesting Problem:

Easy

datatype → type of data
space



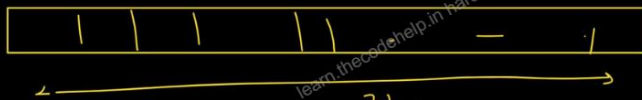
Signed vs Unsigned data:

1 byte → 8 bit

int a = 5

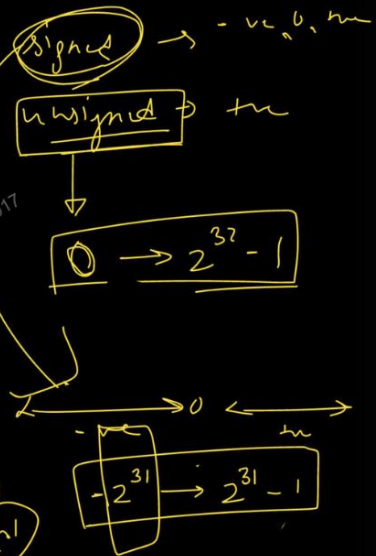
a = 5

int → 4 byte → 32 bit



Total Combs → 2^{32}

$\frac{2^{32}}{2} = 2^{31}$



int \rightarrow 4 byte \rightarrow 32 bit

Total →
Combination

$$\frac{2^{32}}{2} = \frac{2^{32}}{2^1} = 2^{32-1} = (2^{31})$$

int \rightarrow signed \rightarrow -ve 0 +ve

உங்களுக்கு

A diagram of a horizontal beam with a point load P acting downwards at a distance a from the left end. A reaction force R acts upwards at the right end. The beam is supported by a pin support at the right end. The distance from the left end to the reaction force is $a + b$.

$$0 \rightarrow 2^{32} - 1$$

$$\frac{2^{32}}{2} \neq 2^{16}$$

$$2^{\frac{32}{2}} = 2^{16}$$

$$\frac{2^a}{2^b} = 2^{a-b}$$

$$-2^{31} \rightarrow 0 \rightarrow 2^{31} - 1$$

$$\frac{2^{32}}{2^1} = 2^{32-1} = 2^{31}$$

~~1 byte~~ shift \rightarrow 2 byte \rightarrow 16 bit

$nyz \rightarrow 6 \text{ byte} \xrightarrow{\times 8} 48 \text{ bits}$
 Total Comb $\rightarrow \boxed{2^8} \quad 2^{41}$
 $\frac{2^{48}}{2} = 2^{47}$
 unsigned $\rightarrow 0 \rightarrow 2^{48} - 1$
 signed $\rightarrow -2^{47} \rightarrow 2^{47} - 1$

n bits
 signed $\rightarrow (-2^{n-1} \rightarrow 2^{n-1} - 1)$
 unsigned $\rightarrow \boxed{0 \rightarrow 2^n - 1}$

TypeCasting

Implicit vs Explicit

- Type casting refers to the conversion of one data type to another in a program. Typecasting can be done in two ways: automatically by the compiler and manually by the programmer or user. Type Casting is also known as Type Conversion.

Implicit Type Casting or Implicit Type Conversion

- It is known as the automatic type casting.
- It automatically converted from one data type to another without any external intervention such as programmer or user. It means the compiler automatically converts one data type to another.
- All data type is automatically upgraded to the largest type without losing any information.
- It can only apply in a program if both variables are compatible with each other.

automatically
↳ implicitly
type conversion

Explicit Type Casting or Explicit Type Conversion

- It is also known as the manual type casting in a program.
- It is manually cast by the programmer or user to change from one data type to another type in a program. It means a user can easily cast one data to another according to the requirement in a program.
- It does not require checking the compatibility of the variables.
- In this casting, we can upgrade or downgrade the data type of one variable to another in a program.
- It uses the cast () operator to change the type of a variable.

Operators:

Arithmetic

Relational

Assignment

Logical

Bitwise

0, +, -, *, /, %

>, <, >=, <=, !=, ==

int a = 5;

Precedence table

2's Complement Based

>, <, >=, <=, ==, !=

!(false) → true

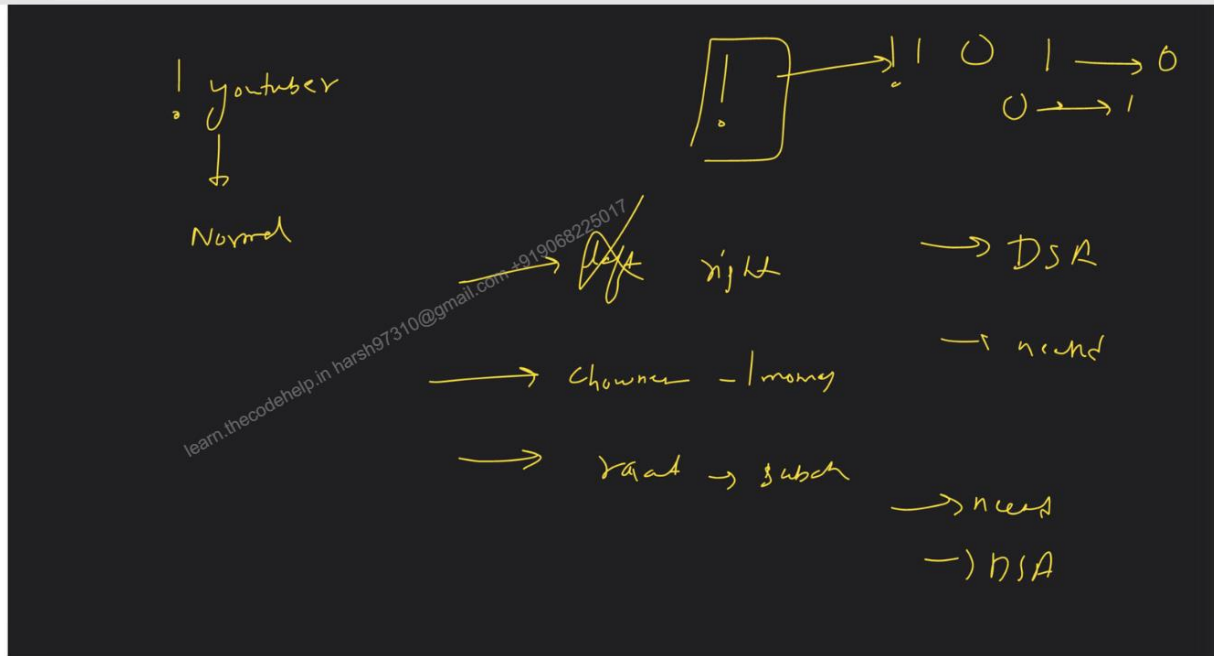
!(a > 5)
!(true) → false

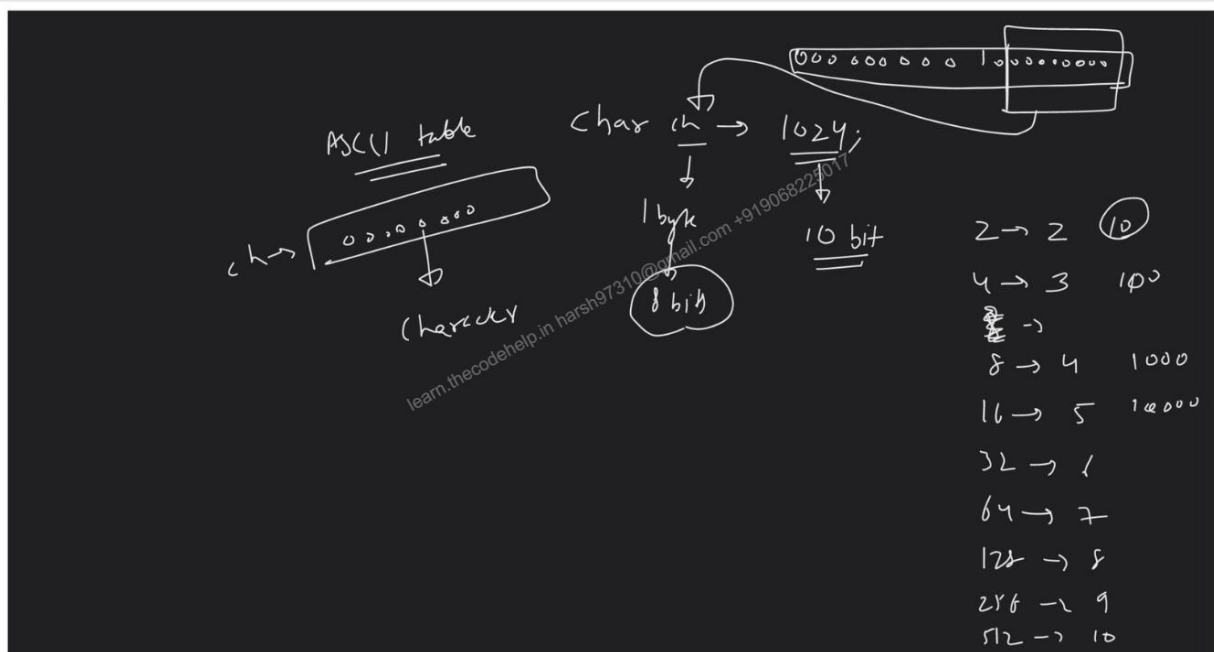
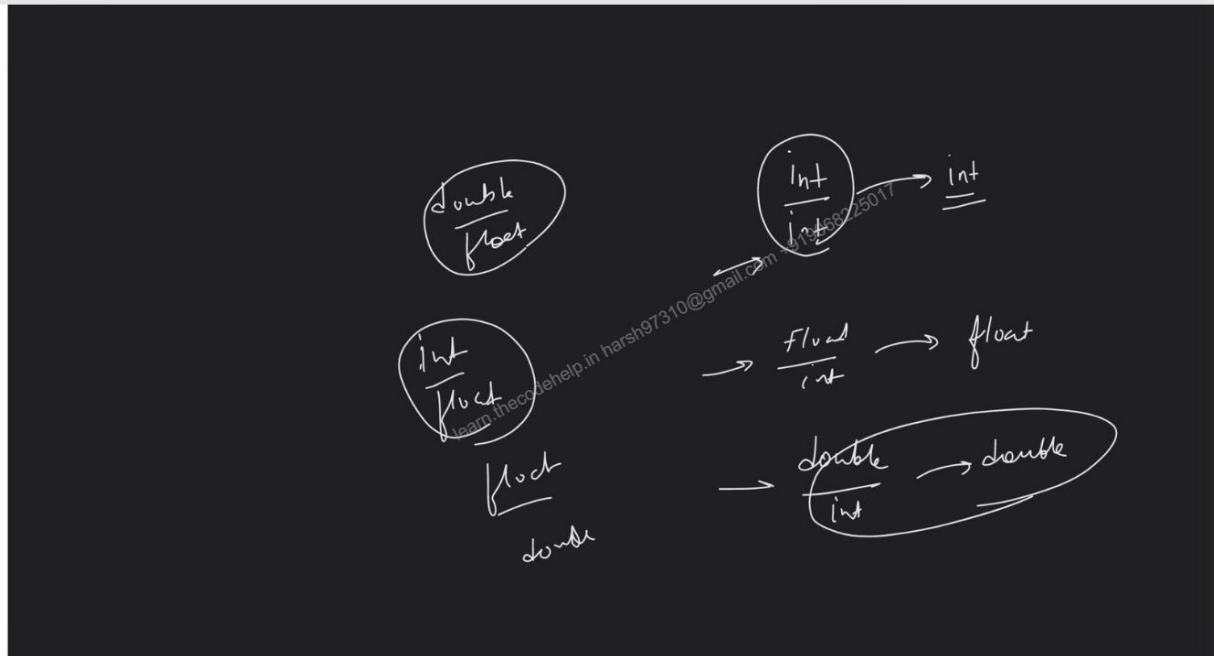
char ch = 234342

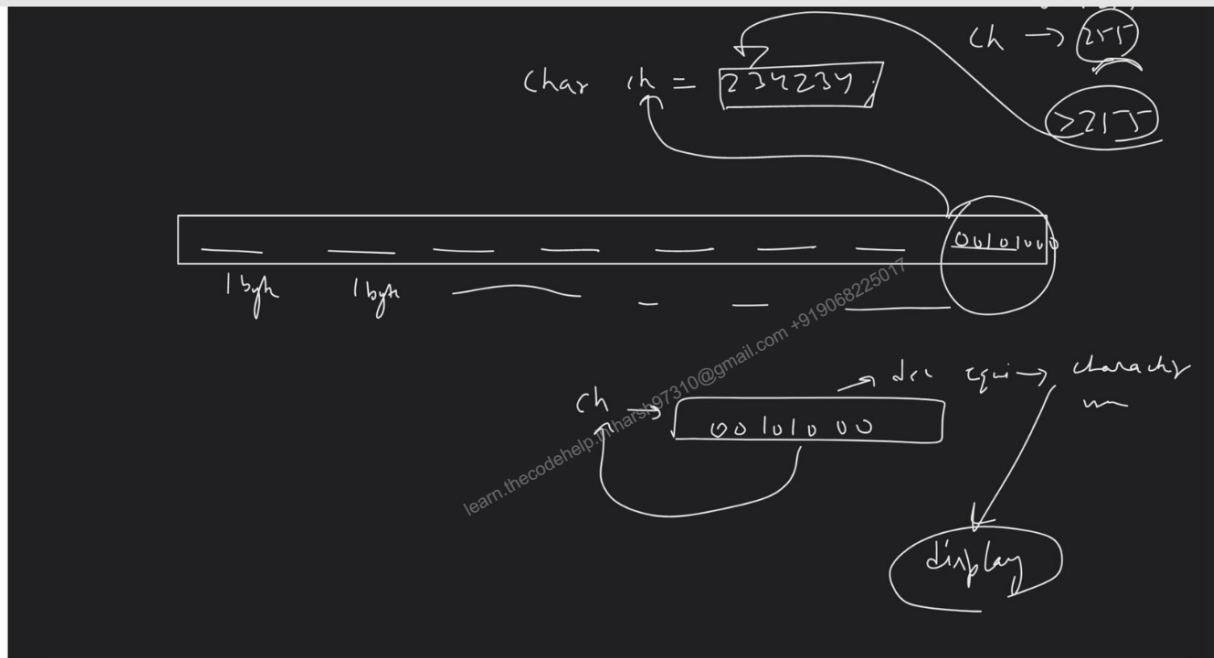
1 byte
8 bit
2's complement

a > 5
5 > 5
a > 5

T
F







Conditionals

learn.thecodehelp.in harsh97310@gmail.com +919068225017

Loops:

learn.thecodehelp.in harsh97310@gmail.com +919068225017

30 Dec

spam

dund
server

