(Problem Solving)

Problem Statement

Conditions —

↳ Solution → Time ↓ (execute)

Space ↓ (code) ✓

} optimal solution

Indirect

(Logic) → Pattern ✓

→ Sorting Search ?

(Sum of 2 no)

↓

given (a, b)

↳ (+)

direct ✓

Problem St → Sol ?

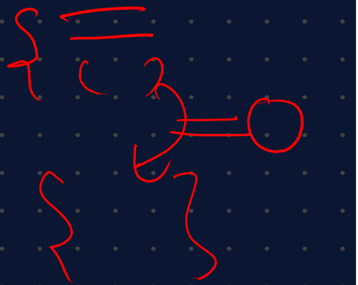
Understand : Problem ? → (given values) ✓

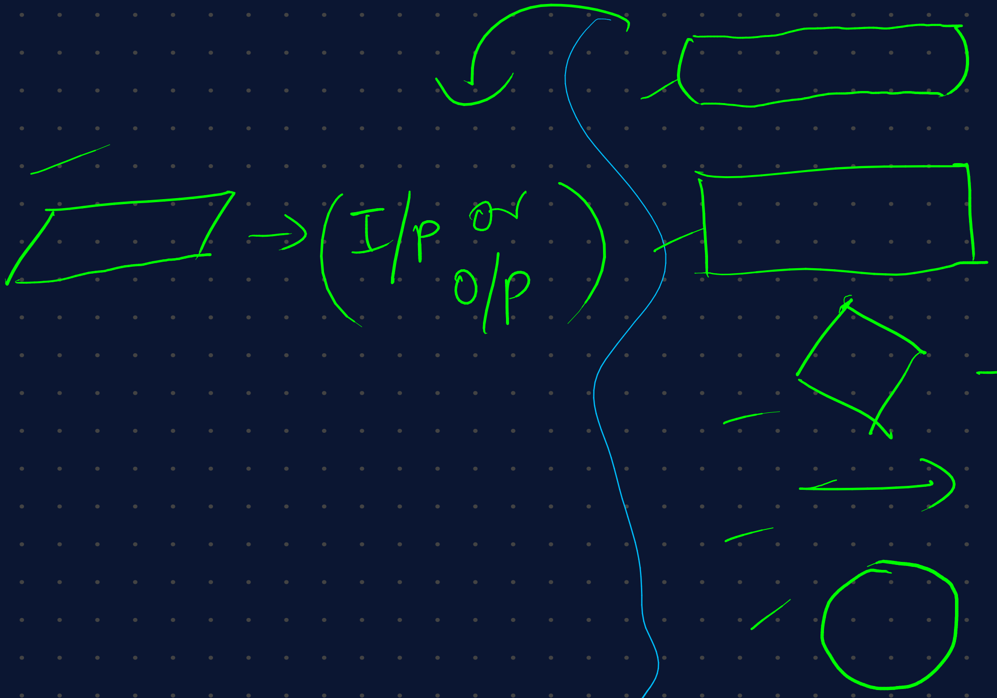
Rough idea/ Logic → Code → Solution

Rough work ? → flowchart / Pseudocode } way of representing Code / logic.

(flowchart) → (Components)

→ Diagram rep. Logic





→ (Start/end)

→ Process

→ Condition

→ Arrow (flow)

→ function (connect) ✓

→ (I/p or o/p)

Program

generic way

↓

Pseudocode

(Rough code) → X ~~C, C++~~

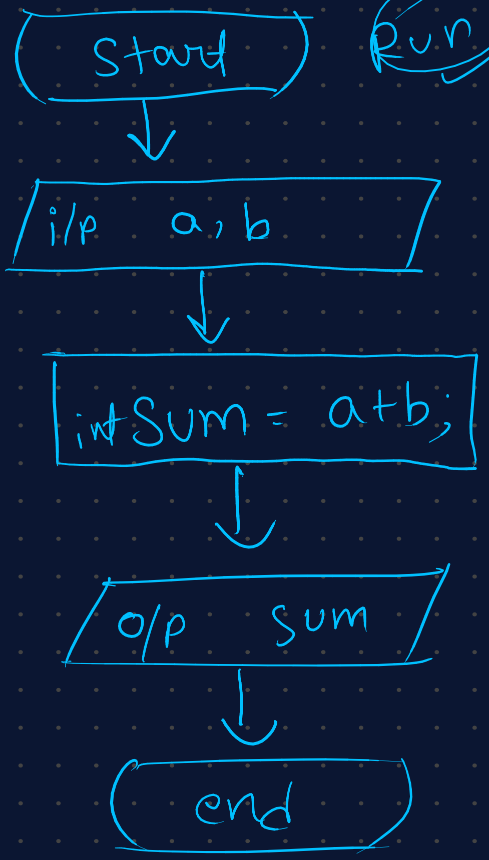
flowchart

Logic

Rough work

(Sum of 2 NO)

(Sum of 2 NO) ✓



Run

(Sum = a+b)

↓

calculation

↓

process

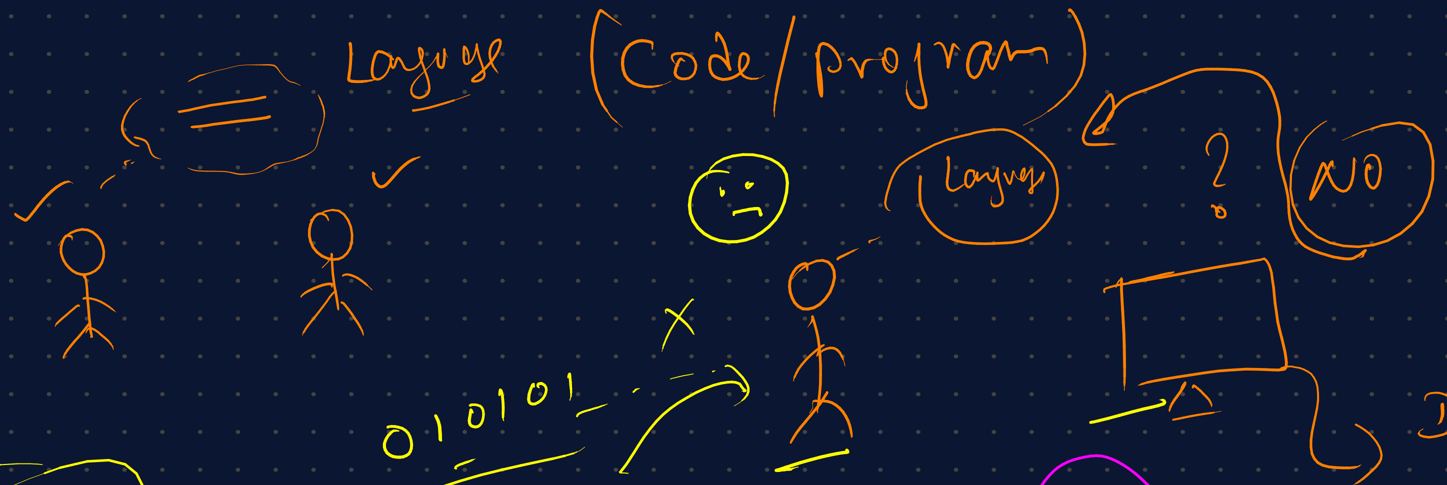
- Start
- Take a, b i/p
- sum = a+b;
- o/p sum
- Stop

(✓ ×)

Logic ✓

- Start
- initialise a, b
- Take in

English ✓



Diff Language

Way → Human ⇄ Computer

Language → Suitability

Hman

Source Code, C, C++,
Java,
python

Coding (English)

Language → (High level language
/middle level)

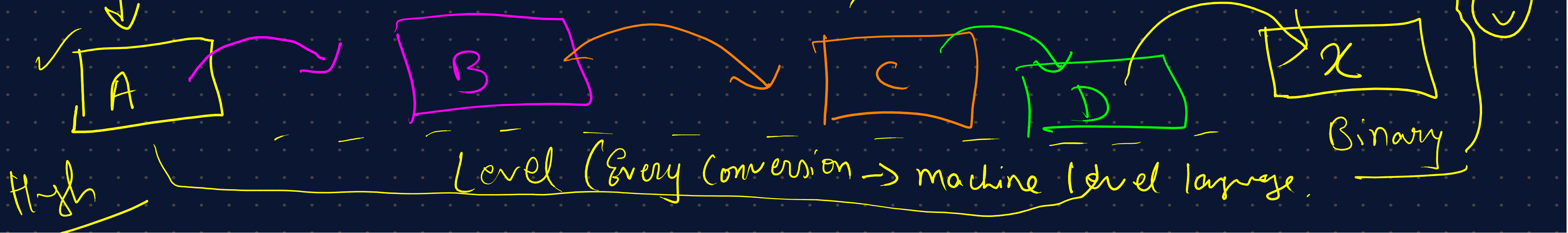
on, off
true, false
0 → false
16 → true
-1 → true

Bin → (2) → Bits
0, 1

Decim → (10)

Hex → (16)

Computer



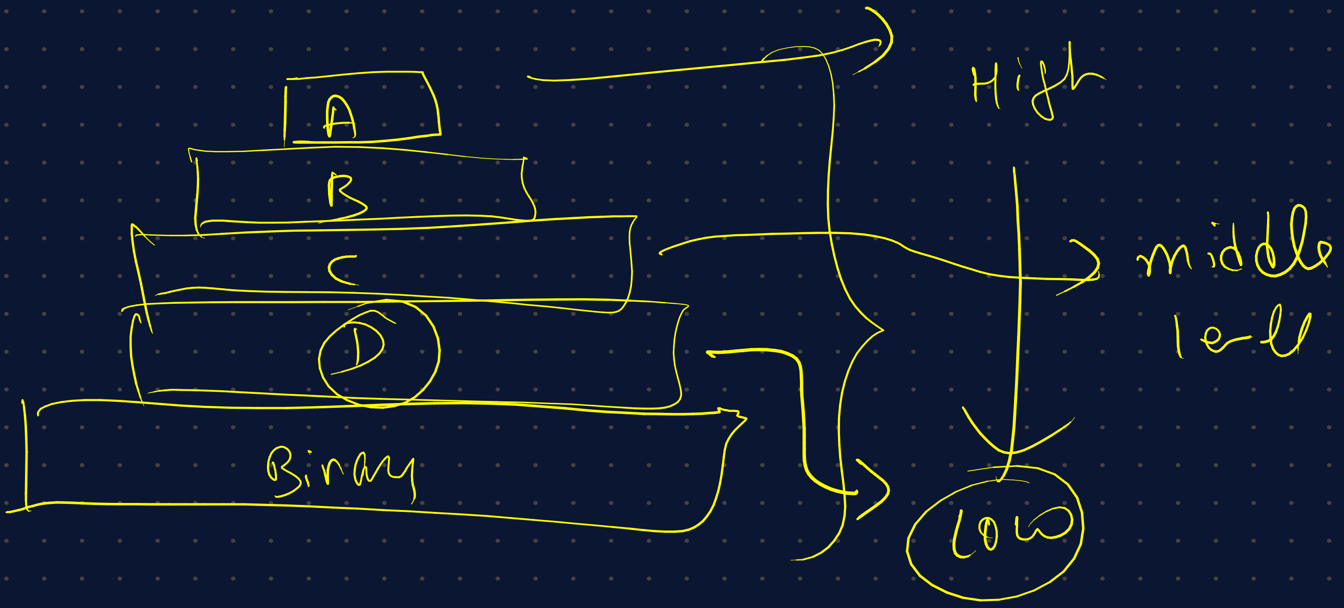
middle level

↓

Assembly -> Low level

↓

Binary -> machine level



(Sum of 2 no) -> chart / Pseudocode

$S1 \Rightarrow \left(\frac{p \times r \times t}{100}\right) \longrightarrow$

(a>b) ->

Problem -> Rough -> pseudocode

flowchart

A. Convenience

(Github)

1 to N

even

odd

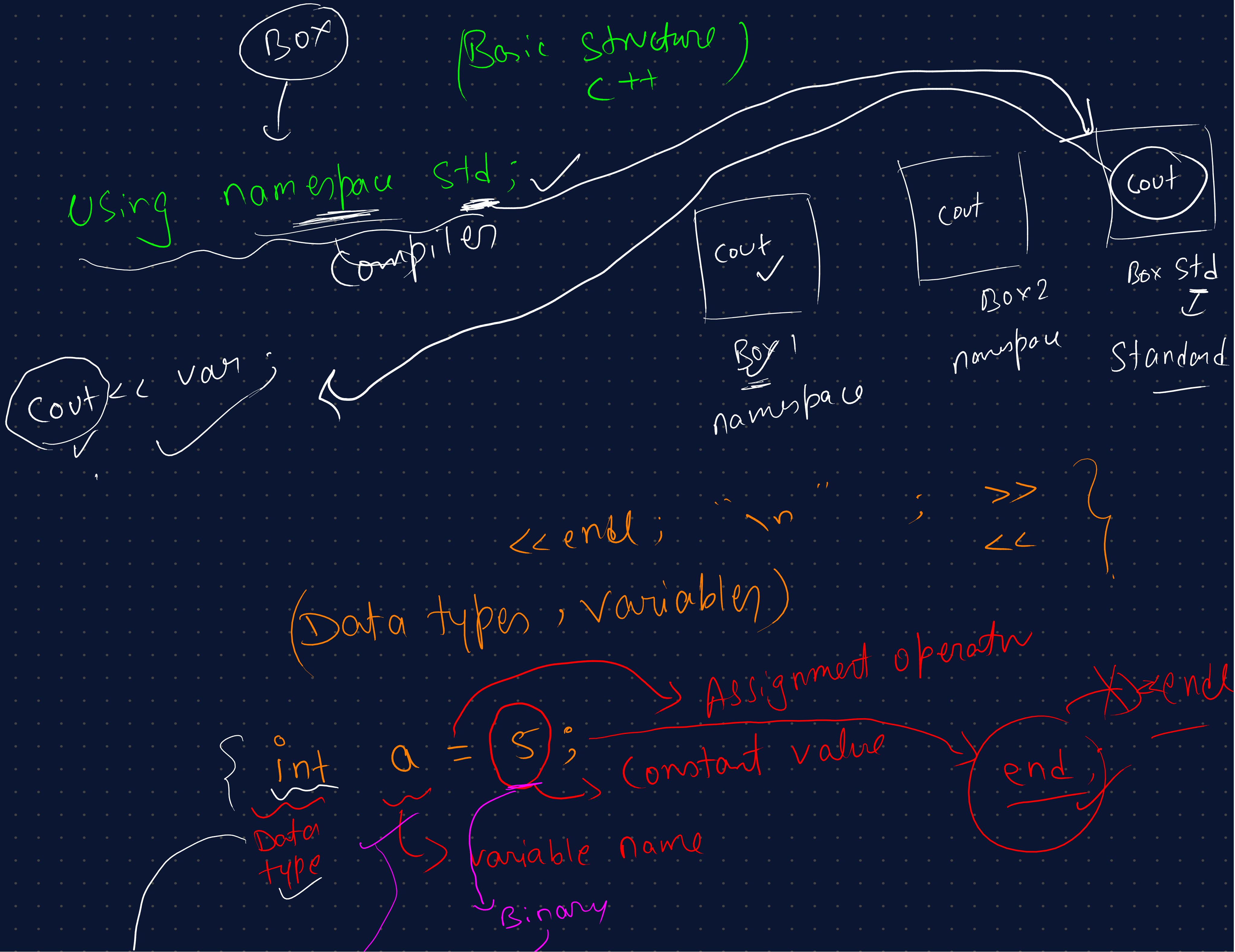
A > B

S 1

github.

Link

download



(Basic structure C++)

Box

using namespace std; ✓

Compiler

cout ✓

Box 1 namespace

cout

Box 2 namespace

cout

Box std

Standard

cout << var ;

<< endl ; "\n" ; >> <<

(Data types , variables)

{ int a = 5 ;

Data type

variable name

Binary

Assignment operator

constant value

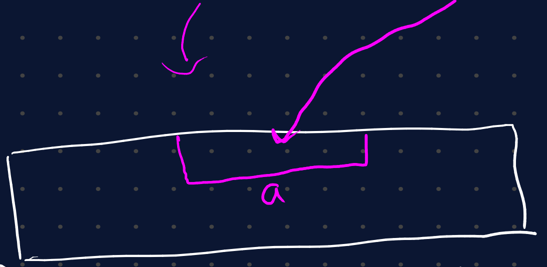
end

~~<<endl~~

Architecture

32 bit → 2 bytes

64 bit → 4 bytes



Mem → RAM → 4 GB
8 GB
16 GB

size

byte

1 byte = 8 bit

1 Kb = 1024 bytes

1 Mb = 1024 Kb

, ,

Important concept

Bits ⇒ 0, 1

(3 bit rep)

0 → 000

1 → 001

2 → 010

⋮

7 → 111

→ (8 values)

3 bit → 8

3 bit → $2^3 \Rightarrow 8$ ✓

2 bits

$2^2 \Rightarrow 4$

00

01

10

11

4

$2^2 \Rightarrow \boxed{4}$

$2^{\boxed{3}} \Rightarrow \boxed{8}$

Range

Integer → 4 bytes ⟹ $8 \times 4 \Rightarrow \boxed{32}$ bits

$2^{\boxed{32}}$ Values

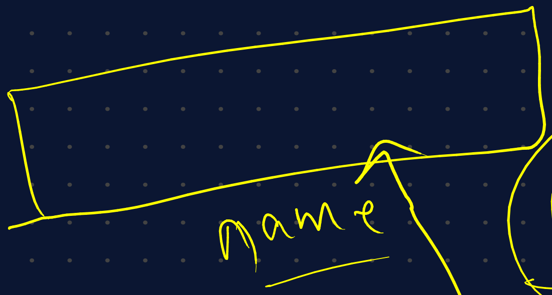
Integer Max value → 2^{32} ✓

Min value ⇒ $\left(-2^{32} - \underset{0}{1}\right)$

exclude

Char name = 'a';

↳ 8 bits → (1 byte) ✓



ASCII

64, 67

Binary

4 byte

↳ Int

signed → -v ✓

$\left(-2^{32} - 1 \le x \le 2^{32}\right)$

double

float, etc

bool → C ✗

bool → C++

int roll = 42;

name

↳ (rules)?

1 bit => 0
=> 1

① (Reserved word X C++)

↳ switch

↳ for

↳ if X

② Number se start XX

③ _ , a, A ✓

④ Name, Case sensitive

name Name

(Run it yourself)

sizeof( );

✓ → particular Architecture

```
int a = 5;
cout << size_of(a);
```

Arch x64 → 4 bytes ✓

Arch x32 → 2 bytes

Imagine -> Personal Architecture

int -> (6 bits)

32	16	8	4	2	1
0	0	0	0	0	1
0	0	0	0	1	0

-> ①

-> ②

+ve value

✓ 6 bits

char ch = 'a';

ASCII

6bytes

Store

(-ve values)

(6 bits)

(-2) -> (-ve signed) ignore

original value

2

0 0 0 0 1 0 } store

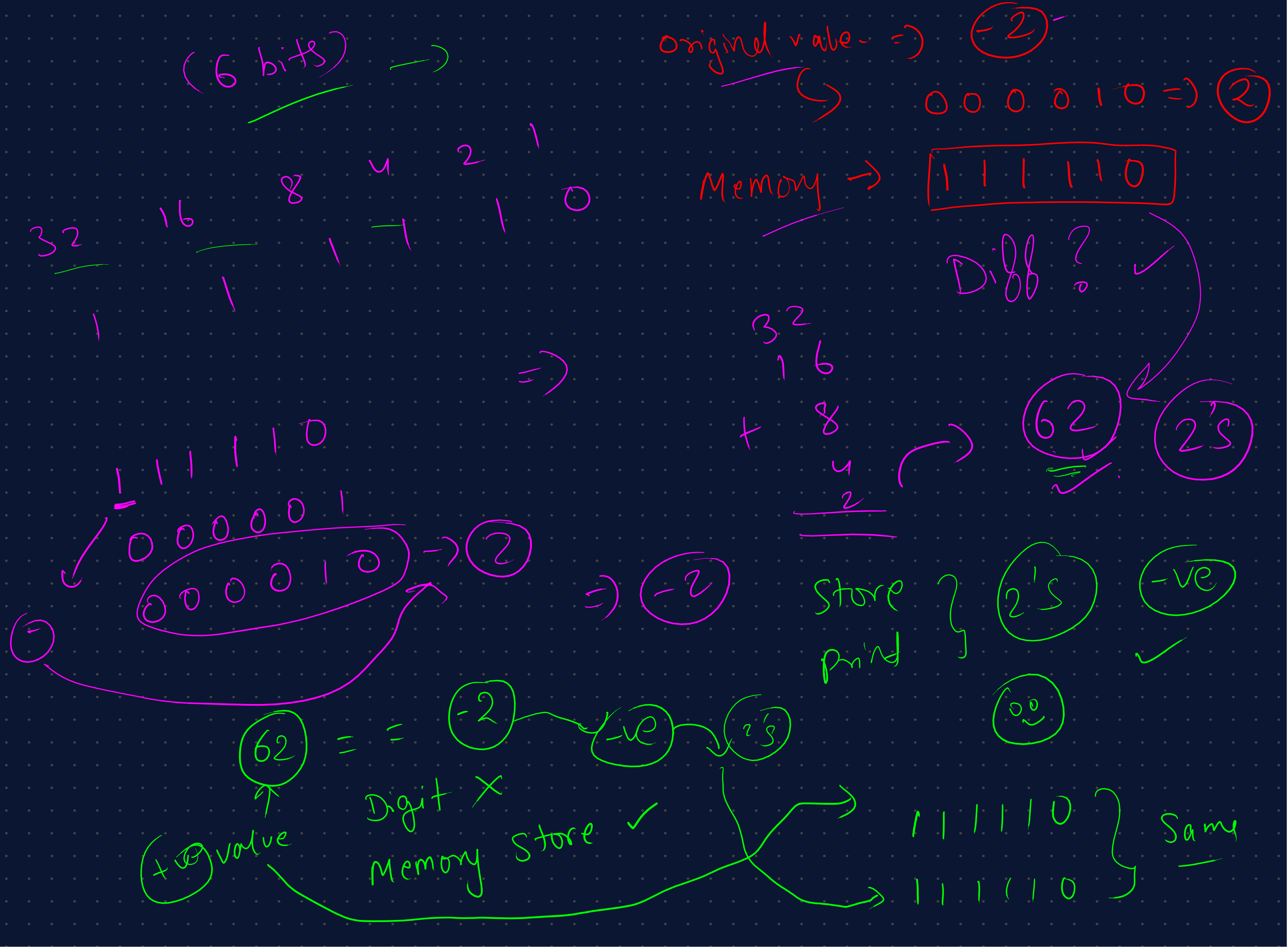
1's 1 1 1 1 0 1

2's 1 1 1 1 1 0 -> memory -> 2's

Print

0 -> value +ve

1 -> value (-ve)


(6 bits) →
32 16 8 4 2 1
1 1 1 1 1 0
=>
1 1 1 1 1 0
0 0 0 0 0 1
0 0 0 0 1 0 → 2
-
=> -2
original vale => -2
→ 0 0 0 0 1 0 => 2
Memory → 1 1 1 1 1 0
Diff ?
✓
32
16
+ 8
4
2
→ 62 ✓
2's
Store
print
2's
-ve
✓
62 == -2 -ve 2's
(+ve) value
Digit ✗
Memory Store ✓
1 1 1 1 1 0
1 1 1 1 1 0
Same

=> $\boxed{-2}$ → find out what store in memory ?

Bits => 6.

<-> $2^6 + (-2)$ ✓ 64 - 2 => $\boxed{62}$

Typecasting.

(int) + (Syntax)

{ implicit
Explicit } ✓

char ch = 97;
cout << ch;

ch = 1 2 3 4 5 6

64 , @ ✓

ch =>

cout << (~~char~~ int) ch;

explicit

Kanjus

1 extra space waste Bachane ke liye

2's ✓

(fact) →

-ve

?

6 bit ✓

0 → (0 0 0 0 0 0)

-0 = 0 → (1 0 0 0 0 0) mem

(1 0 0 0 0 0) => (0 0 0 0 0 0) → alag ?

(0 0 0 0 0 0)

(Operator)

Arithmetic → (+ , - , * , / , %)

2)5(2
4
$\boxed{1}$

5 % 2 => 1

int = $\boxed{2}$ $\frac{5}{2}$ => 2)5($\boxed{2}$.~~5~~
4
1 0

3 % 2 => 1

2)3(

float => $\boxed{2.5}$ $\frac{5.0}{2}$

2 % 3 => $\boxed{2}$

3)2(0
0
$\boxed{2}$

Relational => Relation?

| $\boxed{=}$ Assignment

(Logical operator)

(&& , || , !) ✓

$\boxed{\text{use}}$ ✓

increment, derement

++ --

>
<
>=
<=
==
!=

i++ → Post increment) ?

++i → Pre increment

(i-- --i)

→ i Ka jo value hai, usko use Karlo, fir increment Kardo.

→ Increment Karenge, fir use Karenge.

i

~~5~~ → Print

~~6~~ → Print

7

(Shorthand)

i = i++ ⇒ i += 1;

```
i = 5
i++;          → 6
cout << i;
cout << ++i;
```

O/p 6

(Condition & loops)

Decision making

Scenario a —> krdo

b —> Dusra

"

```
if (condition)
{

}
```

true
or
false

0 -> false

1 -> true

x ->

```
a = 2
if (a == 3)
{
    cout << a;
}
else {
    cout << "Hi";
}
```

kuch no

Execte ☺

o/p —> Hi

```
int a = 2;
if (a) —> true
{
    cout << "Hi";
}
```

$\overset{2}{a}$

o/p —> "Hi"

if elseif ladder;

```
if ( age >= 18)
{ cout << "vote";
}
else if( age >= 16)
{ cout << "kid";
}
else {
      cout << "bye";
}
```

o/p

(vote Kid)

Program -> (18 >= vote)

(18 > n >= 16)

Kiddo

```
✓ X { if ( age >= 18) ✓
      {
         cout << "you can vote";
      }
X ✓ { else {
         cout << "Kiddo");
      }
```

age => 18

Market

-> 3 Mango ✓

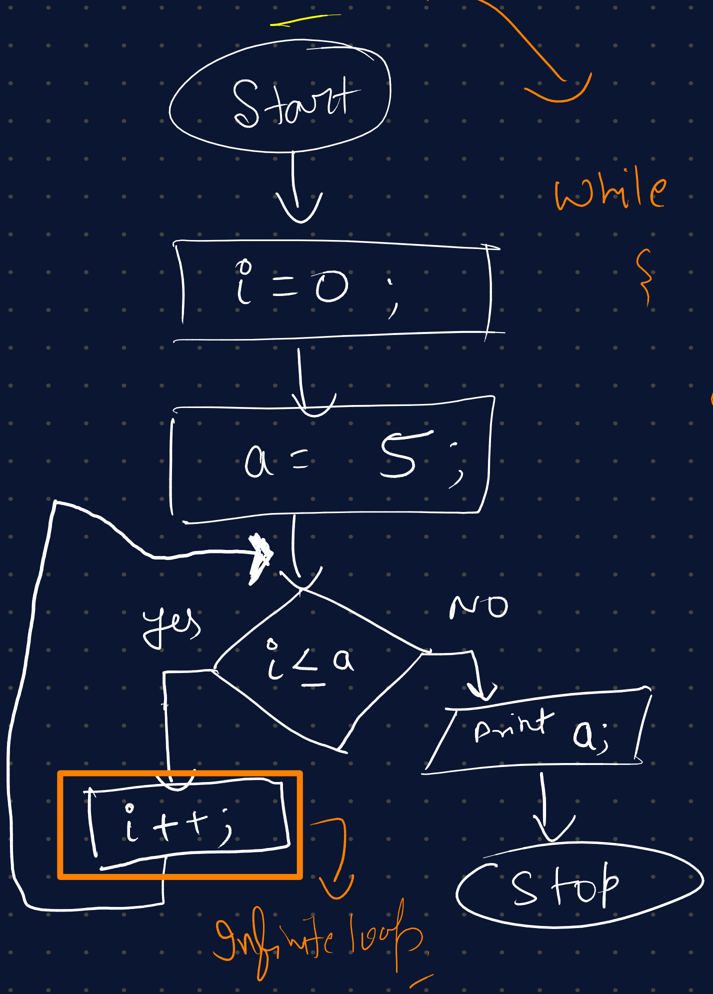
false -> else if { 4 Choclate ✗

(18 >= 18)

18 >= 16

True

(While)



```
while ( )
{
    ═
}
```

entry control

(Do while)

```
do {
    ═
} while ( )
```

exit control

(for loop)

```
for( int i=0 ; i<=n; i++ )
{

}

for( ; ; )
{ if ( )
    b
}
```

(Basics

clean)

n=3

```
* * *
* * *
* * *
```

```
111
222
333
```

```
123
123
123
```

j+1

```
9 8 7
6 5 4
3 2 1
```

n=3

```
1 2 3
4 5 6
7 8 9
```

```
1
2 2
3 3 3
```

Done

Move

```
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
```

n i j Logic

i j

0 -> 3

9 ≤ 0

```
*
* *
* * *
```

i=3

j ≤ i

```
1
1 1
1 1 1
```

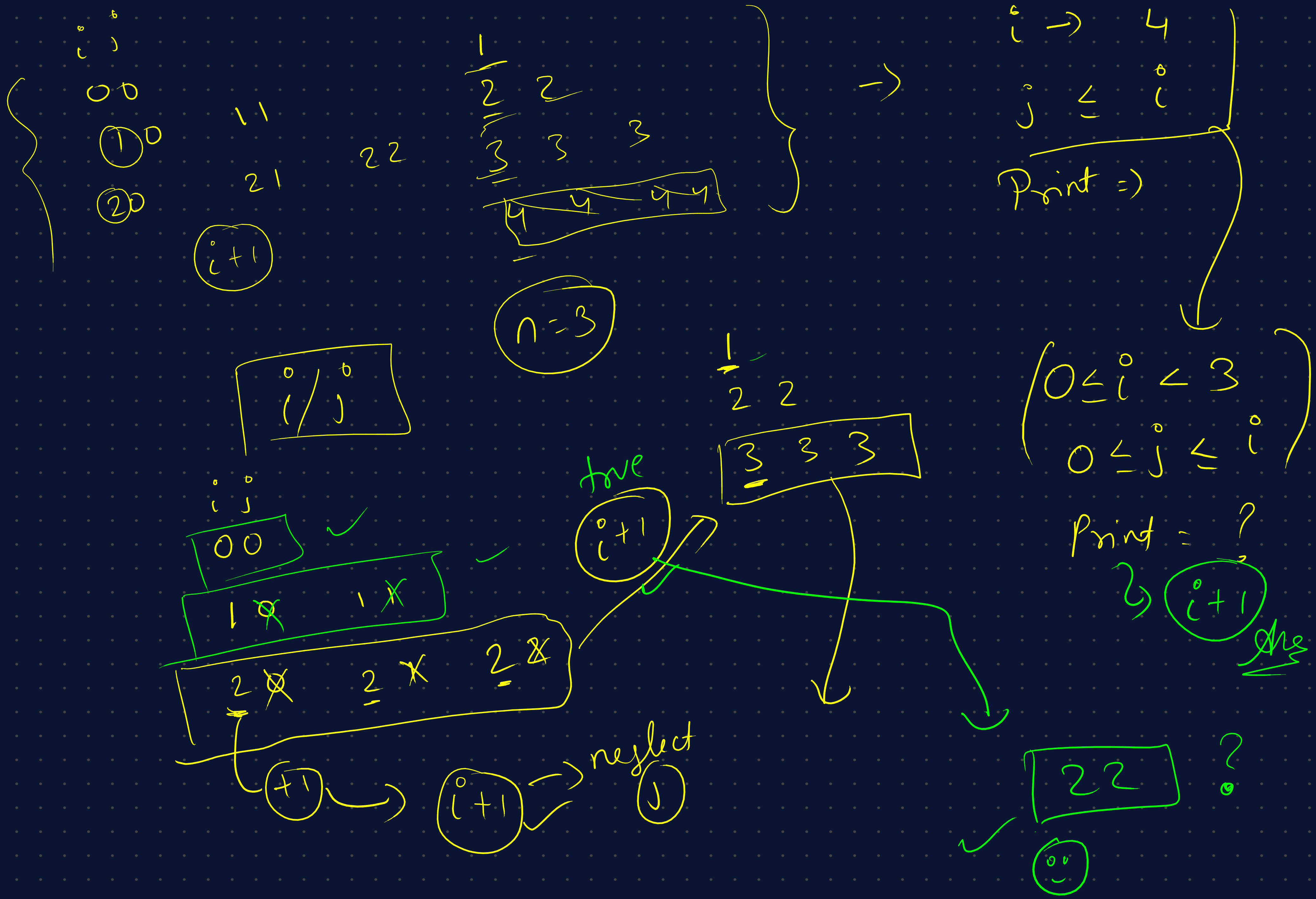
n

for(int i=n+4 ≤ > i--)

≤ ++

fomola build

diff


i j
0 0
1 0
2 0
1 1
2 1
2 2
i+1
1
2 2
3 3 3
4 4 4 4
n=3
i -> 4
j ≤ i
Print =)
i / j
i j
0 0
1 0 1 0
2 0 2 0 2 0
true
i+1
1
2 2
3 3 3
i+1
i+1
neglect
j
0 ≤ i < 3
0 ≤ j ≤ i
Print = ?
i+1
Ans
2 2
?

```
int count = 1

for( int i=0; i<4; i++)
{
    for( int j=0; j<=i; j++)
    {
        cout << count;
        count ++;
    }
    endl;
}
```

1
2 3
4 5 6
7 8 9 10

$0 \le i < 4$

$0 \le j \le i$

Lazy.

count = 1.

(++);

Print

cout

(Next page)

1
2 3
3 4 5
4 5 6 7

under ✓
given
Karma (logic)

$0 \le i < 4$

$0 \le j \le i$

Print = ?

n = 4

i / j

(New row)

i+1 ? ✓

++

1
2 3
3 4 5
4 5 6 7

00
10 11
20 21 22
30 31 32 33

i+1 → row ↓
New colm ++

(Logic)

int value = i;

```
for (int i = 0; i < n; i++)
 {
    int value = i;
    for (int j = 0; j <= i; j++)
    {
        cout << value + 1 << " ";
        value++;
    }
    cout << endl;
 }
```

0 <= 0

1 < 4

1 < ~~=~~ 0

0 <= 1

1 + 1

i	~~0~~	1	
j	~~0~~	1	
n	4		
value	0	~~1~~	1

1

2

(Pattern -> 6)

tk khtm

~~value = i~~

formula

Build to print Same pattern using formula