# Class – 9 (Tuesday, May 21, 2024)

(C → 9) → (Array) ☑

Story →

10 variable, int → data ☑

50 data, int type?

(50,0000) ↳ (50,0000) → (Loop) → variable X

In other words;

Collection of Same data types ka value → Array

Diktha kaya?

```
int arr[15];
```

- int → Datatype
- arr → Name of array
- 15 → Size of array

Ek is blocks reserve kr do memory me (int store hoga);

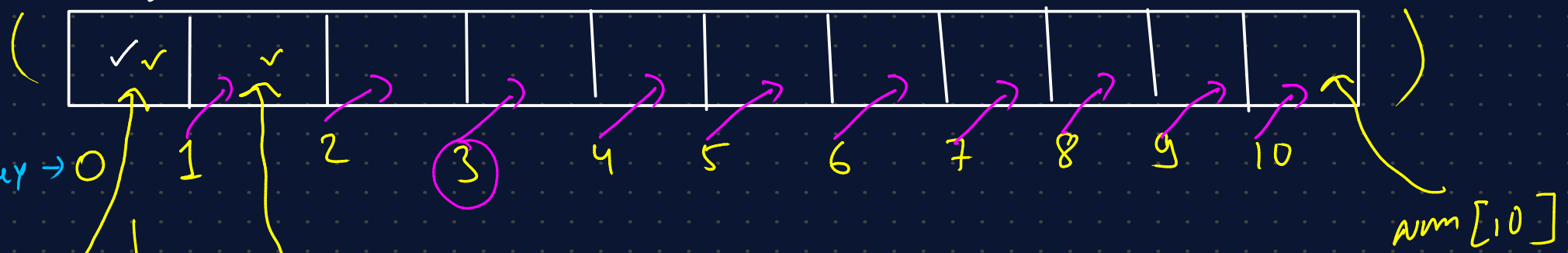
int x = (2); initialisation?

Declare → banana → int Num[11];

initialise → value dena

rep

Name → Num → represent 1st box.




Num[0]

Num[1]

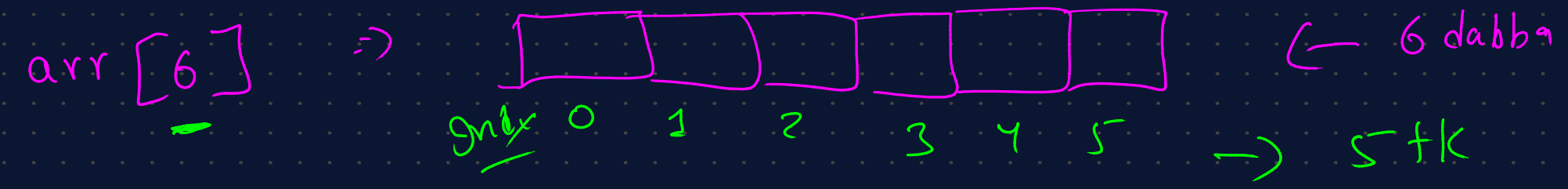
Memory → Continous block allocate ? num.

11 Blocks ? → Garbage value;

cout << Num[0] ✓

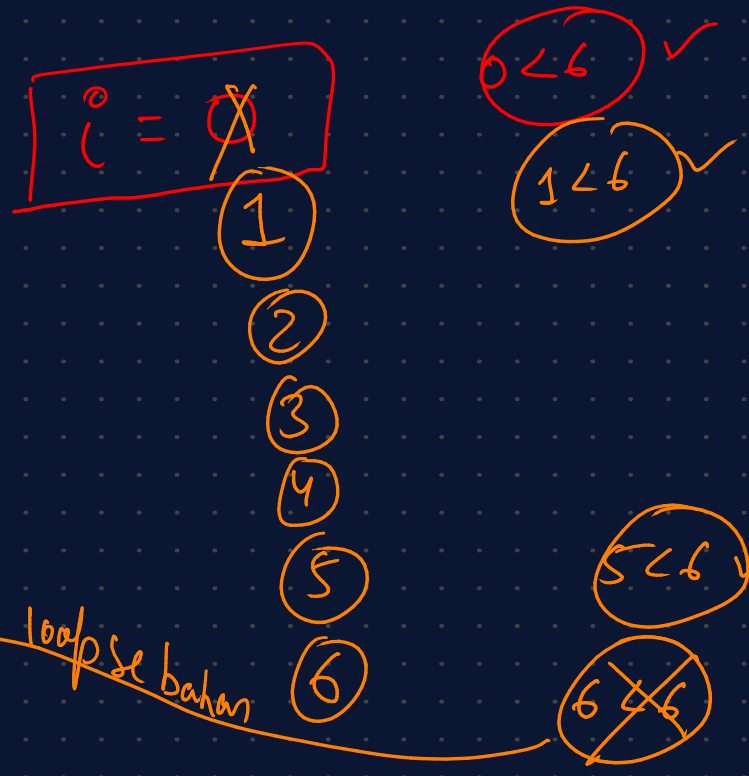
int a;

a → a = 1 → a → 1




```
for (int i = 0; i < 6; i++)
    {
        cout << roll[i] << " ";
    }
    cout << endl;
```

{ Max index -> (size - 1)
Min index -> 0.




f(n) name

```
fill(random, random + 6, -11);
```

arr name

arr size

dalna kya hai?

age samjahenge

int arr[6];

Size -> 6
Index -> 0 to 5 -> 6


Index 0 1 2 3 4 5

(value already hai)

So, no input value only print ☑

suppose value hai,

output

```
for(int i=0; i<3; i++)
{
cout << arr[i];
}
```

0 to 2 index tk he print krna?

Mem Add.

int num = 10;

random assigned

CPU → Hexadecimal

Address (Memory) ↕ → CPU jaake lyega.

Memory

main()

10

num

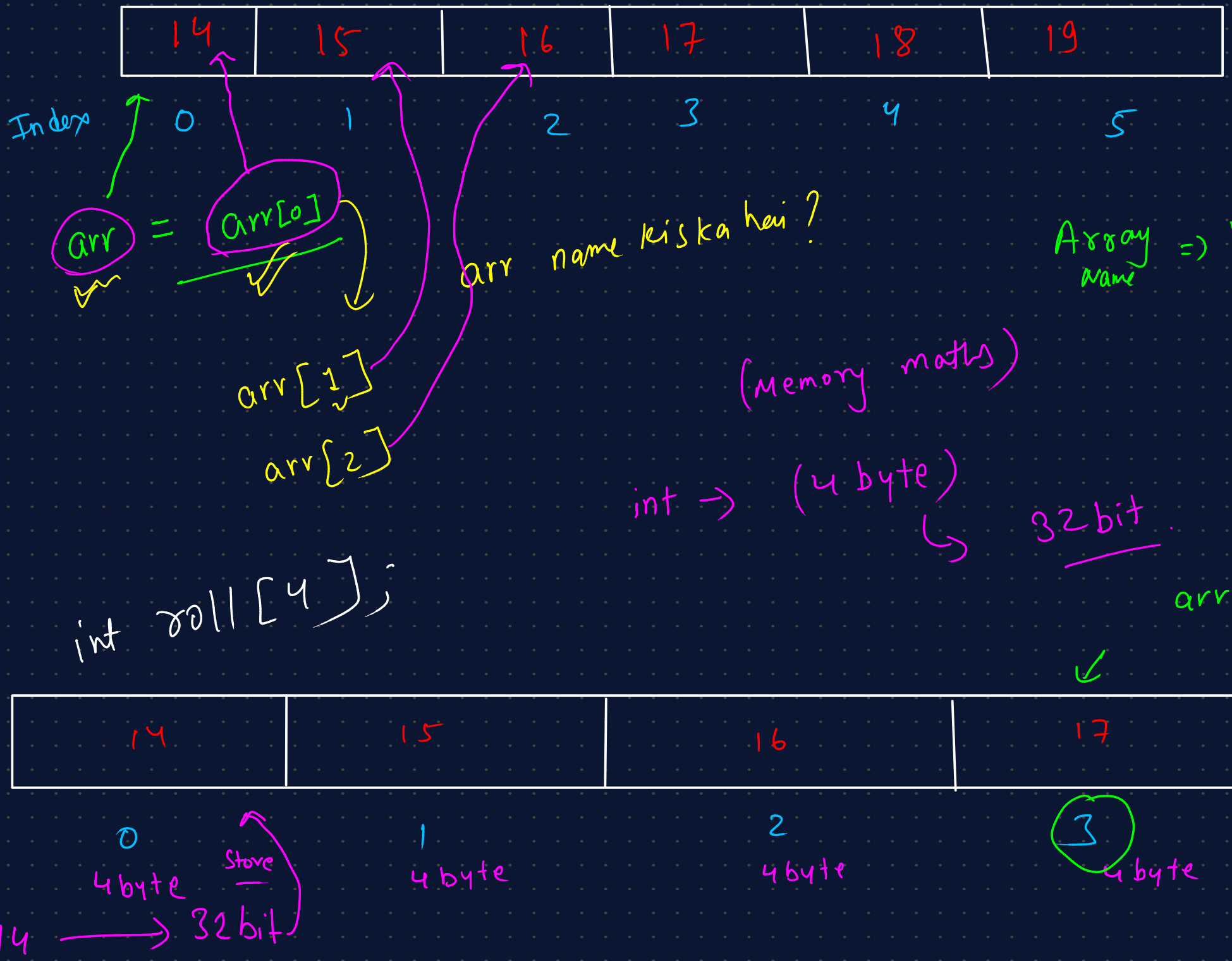
0x×1×x2x3x → Hexadecimal

Dabba ka Address → Memory address.

print()

10

num


14
15
16
17
18
19
Index
0
1
2
3
4
5
arr
=
Arr[0]
arr[1]
arr[2]
arr name kiska hai?
Array name => "arr"
(Memory maths)
int => (4 byte)
32 bit
int roll[4];
arr[3]
14
15
16
17
0
4byte
Store
14 ⟶ 32 bit
1
4byte
2
4byte
3
4byte

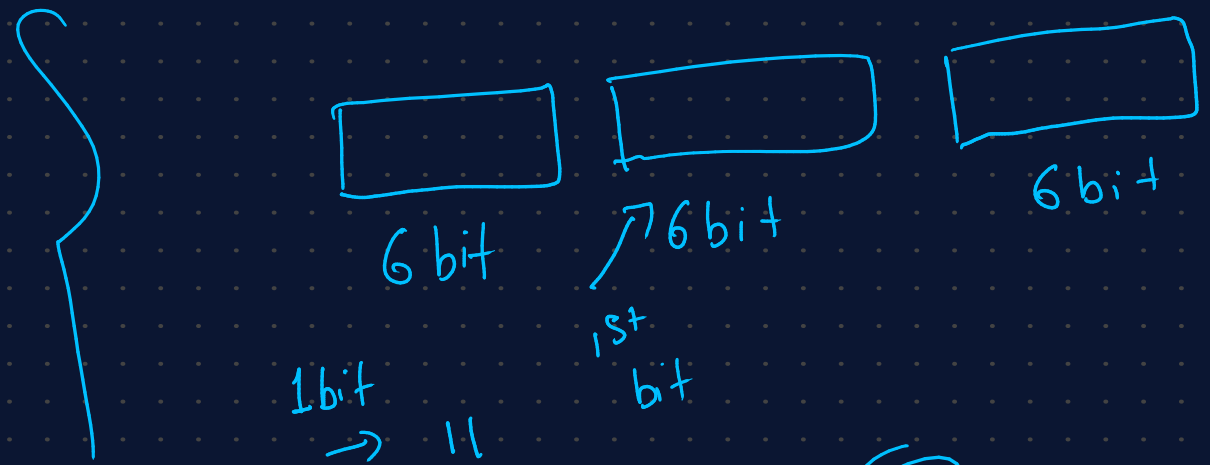
Memory me → Agr arr[0] ka address → 56 hai
Suppose

memory

arr[1] " " → 60 hoga

arr[2] " " → 64 hoga

Memory ke andar




11+6 => (17)

arr[0]

Upar ka math

arr[1] → arr[0] + 1

arr[2] → arr[0] + 2

- - - - - - - -

function ✓

Definition

function (Parameter)

↳ array?

array 😳? why


arr = arr[0]

Pass X (arr, size)

arr[]

↳ arr[0] 0 wala dabba

arr[0+5]

5 index

(arr + 5)

Size

ko wo add krle rha hai

nCr → main

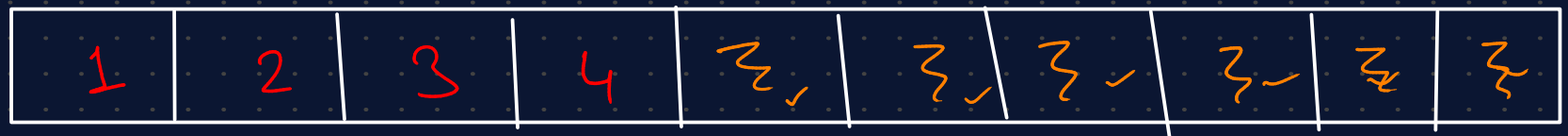
(factorial)

$$nC_r = \frac{n!}{r! \times (n-r)!}$$

```
fact(n) {
    //
    //
}
```

```
main()
{
    result = findnCr(n, r);
    cout << result;
}
```

```
findnCr(int n, int r)
{
    int num = fact(n);
    int deno = (fact(r) X
                fact(n-r));
}

return num/den;
```




value

arr size ✗ given

find ?

( int arr[] , int size )

( arr , 5 )

int arr[10]

(10 blocks)

int arr_

1 → 4 bytes

4 × 10 => 40 bytes

arr[10]

no of elemt.

$\frac{40}{4}$ => 10

int arr[]

char arr → string

float arr

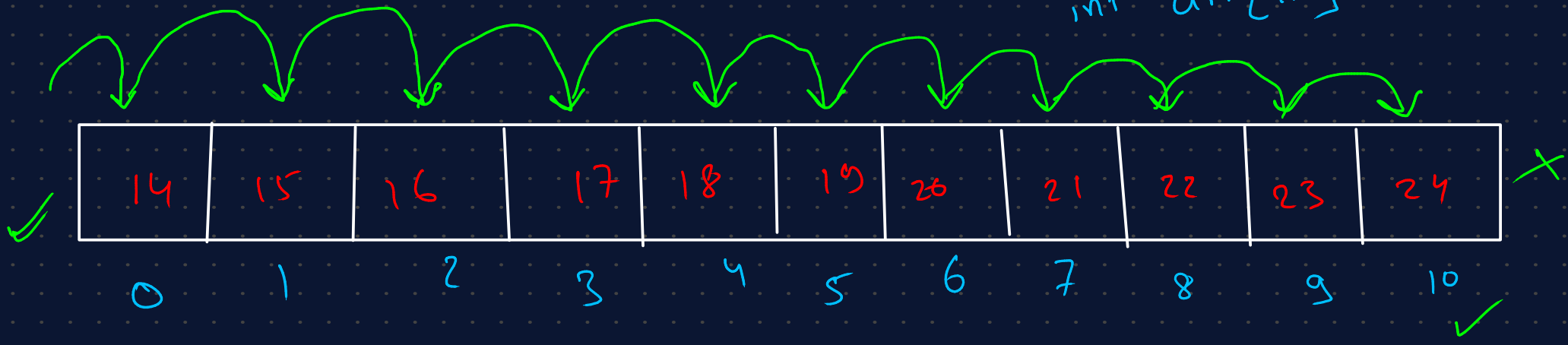
double arr

long long

bool arr[]

# (Linear Search)

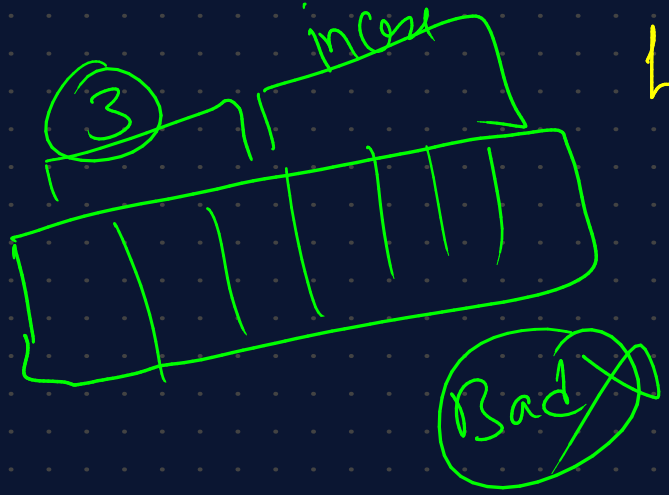
int arr[11]




(1) arr me value input karna.

Linear search

(2)
- i/p -> n = 14
- o/p -> value is present at index 'x'
- o/p -> value is not present in array.




(Traversing array max -> 1 time)

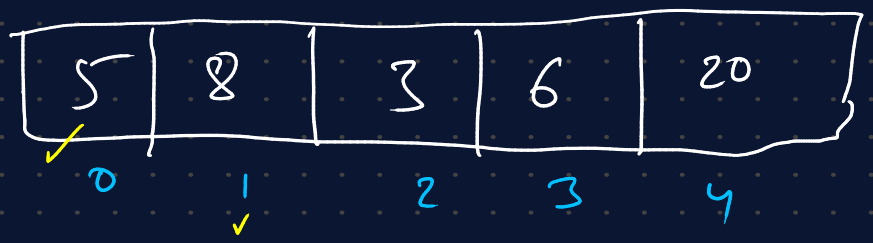
flow of program

return 0;

isko v leke jao

(Min & Max) in (arr)

int arr[5]



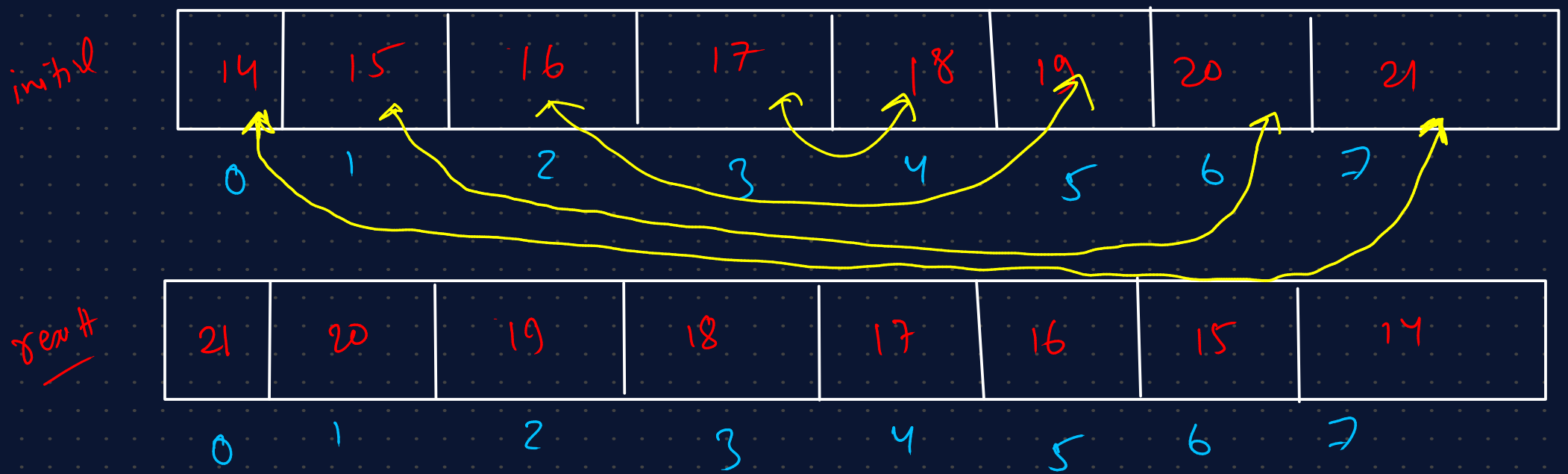

```
main()
{ o/p -> fun()
  min()
  max()
}
```

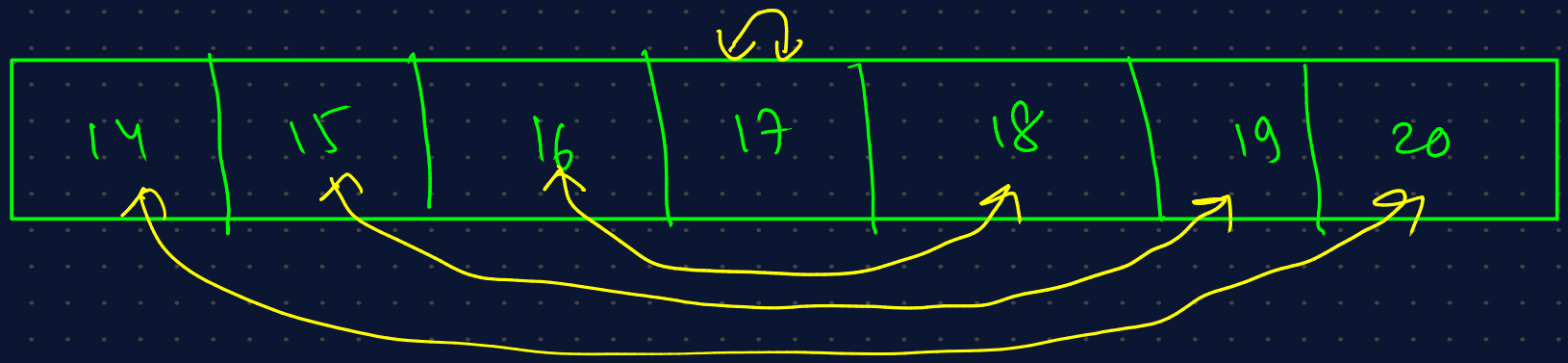
temp

```
Min() {
  int value = 50;   // Assume for now

  (arr[0] < value) -->  5 < 50 -> true
                        value = arr[0]

  (arr[1] < value) -> 8 < 5 -> false

  (arr[2] < value) -> 3 < 5 -> true
                        -> val = arr[2]
                        mt update loop

  (arr[3] < value) -> 6 < 3 -> false

  (arr[4] < value) -> 20 < 3 -> false

  return value;
}
```

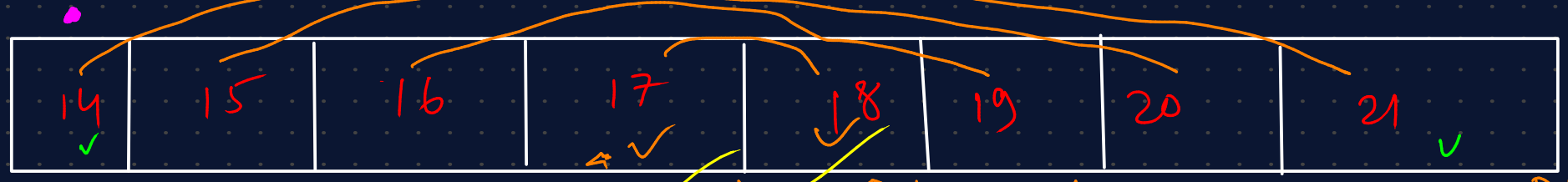
(reverse integer)

int arr[8];




Dost ✓




index { Dost 1 => 0 ;
Dost 2 => (Size - 1) ;

18 , 17

(Dost 1 ≤ Dost 2) 17, 18

while ( Dost 1 ≤ Dost 2 )
{
swap ( arr[Dost 1] , arr[Dost 2] )
After swap;
Dost 1 ++ ;
Dost 2 -- ;
}

Array swap?
1 same
2 Swap

(*)

Sum of all elmts in array → user input

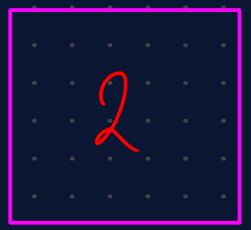
value → Sum

}

}

int a = 2;

cout << a;




Print

a

0x16x----

cout << &a;

?

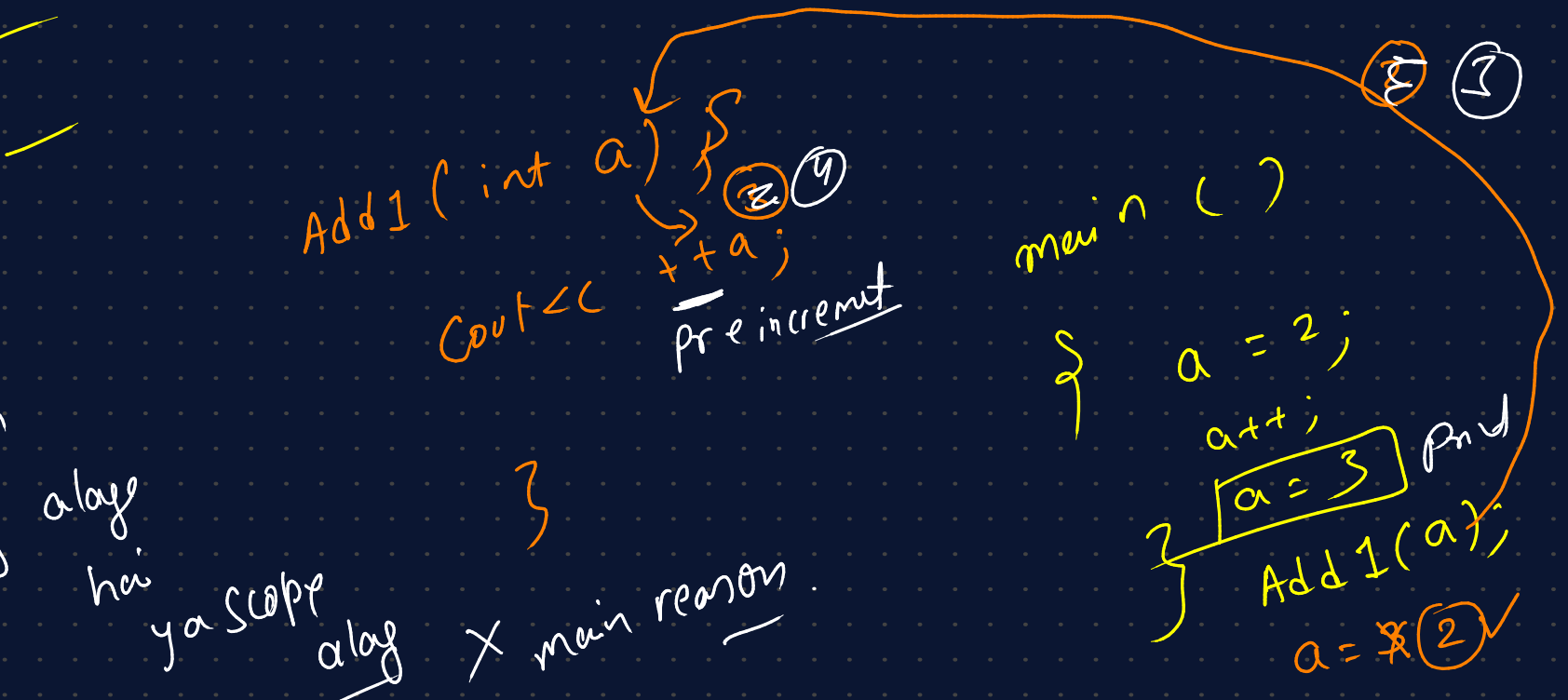
Address of

Binary (0, 1)

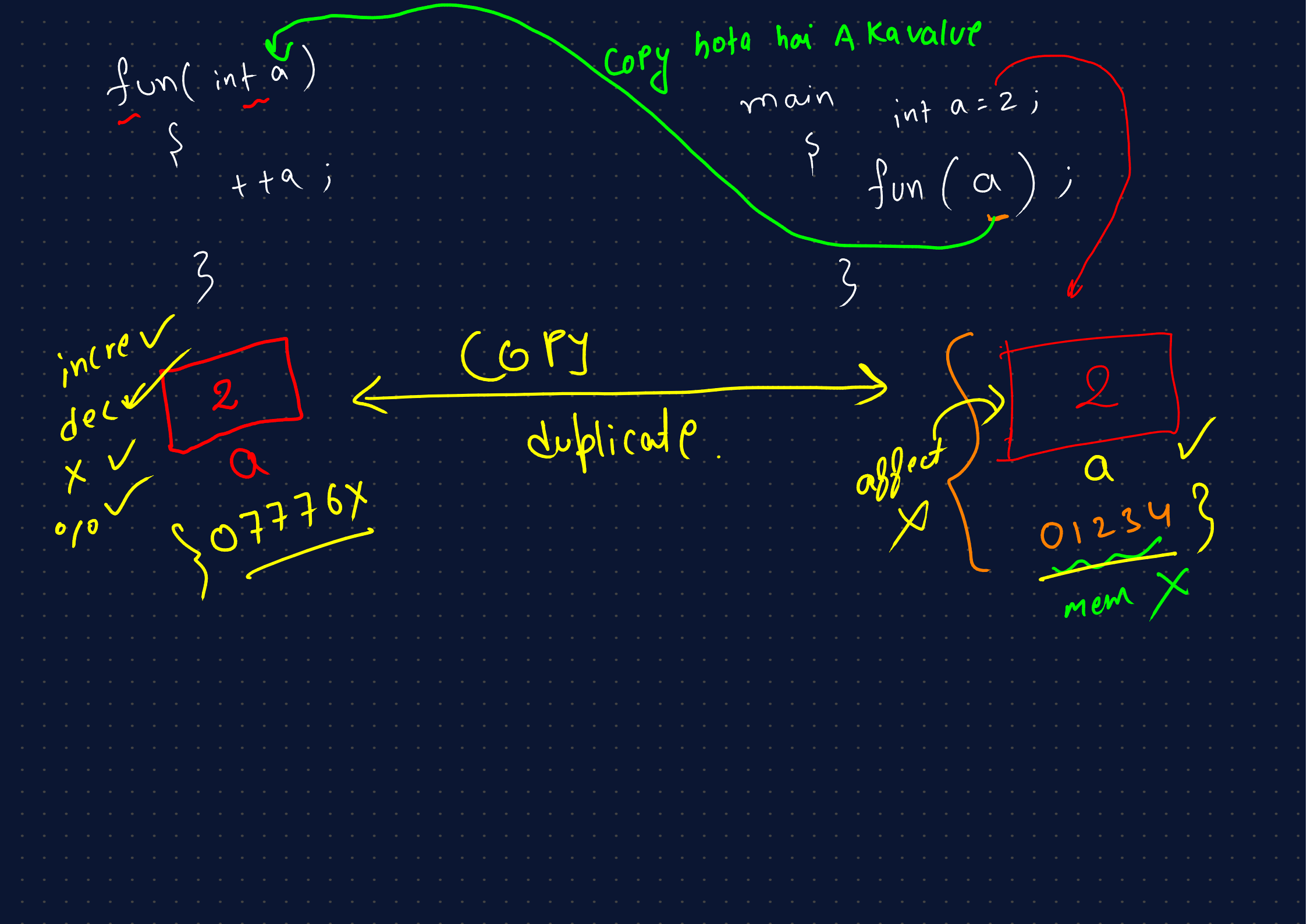
Decimal (0 to 9)

Octal —

Hex —




function alag alag hai ya scope alag X main reason.


fun( int a )
{
++a ;
}
Copy hota hai A Ka value
main
{
int a = 2 ;
fun ( a ) ;
}
increv
dec
x
o/o
2
a
07776X
Copy
duplicate.
affect
2
a
01234
mem

# (Array)

int arr[4]


Reason?

fac(a); → fun(&a);

Function Param

arr

original address

Param

original

Function kis: ✓

value change hogi
↳ har jagah reflect hogi

copy

value a (&a)

fun

int a

int &a

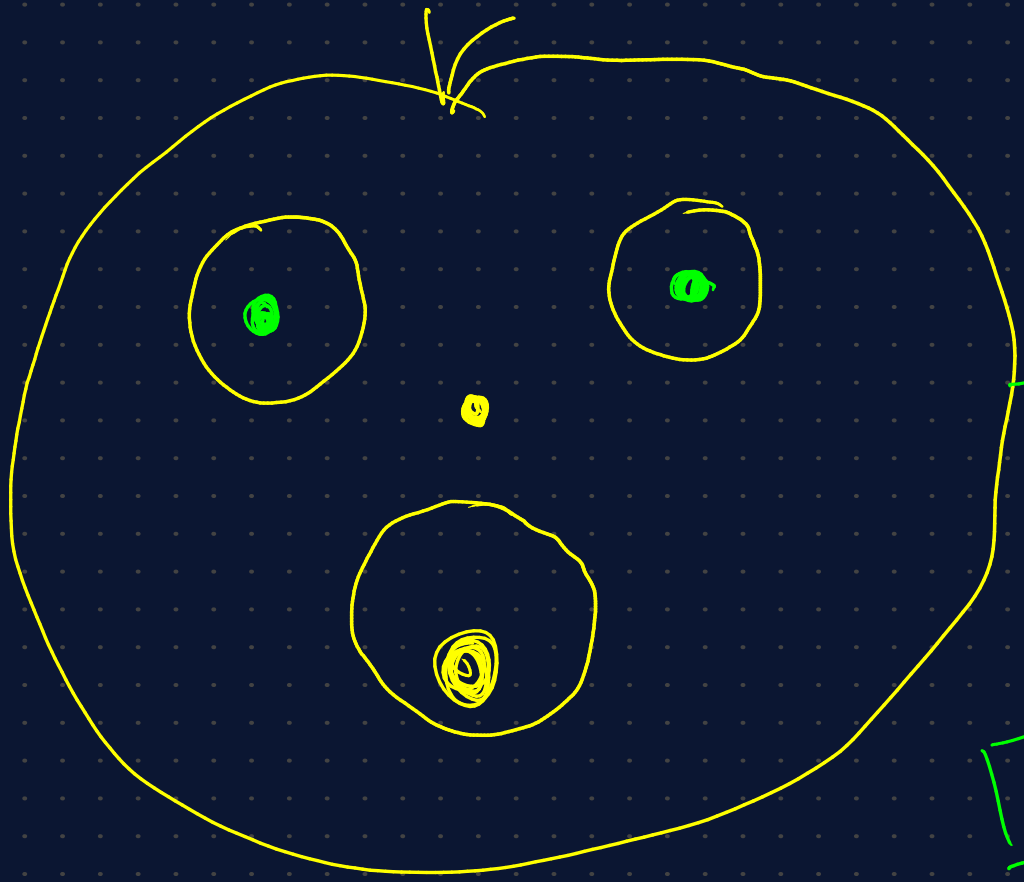
No copy

Pass by value

value copy

Data duplicate.

Pass by reference
↳ original address.
original data.



→ phase 1 complete.

IS phase incomplete.