

- 4 → update  
2 → condition → will repeat until it fails.

```

inhalisation
while (condition) {
    loop work
    update:
}

```

The diagram illustrates the execution flow of a for loop. It starts with a box labeled 'for' containing three parts: 'initialisation' (labeled 1), 'condition' (labeled 2), and 'update' (labeled 4). Below the 'condition' is 'loopwork' (labeled 3). The flow is as follows:
 

- From 'initialisation' (1), an arrow points down to 'loopwork' (3).
- From 'loopwork' (3), an arrow points up to 'condition' (2).
- From 'condition' (2), an arrow points right to 'update' (4).
- From 'update' (4), an arrow points left back to 'condition' (2), forming a loop.
- From 'condition' (2), an arrow points left to a box labeled 'will get executed only once (i)'.
- From 'will get executed only once (i)', an arrow points down to a box labeled 'exit loop'.

Diagram illustrating a for loop structure with annotations:

```

for (int i = 1; i <= 10; i++) {
    sop(i);
}

```

- Initialization:** `int i = 1` is annotated with `st → 1` (start) and `10 = p.` (point).
- Condition:** `i <= 10` is annotated with `11 > 10` (indicating the loop ends) and `↓ T` (indicating the condition is true for the first iteration).
- Body:** `sop(i);` is annotated with `↓ T` (indicating the body is executed).
- Increment:** `i++` is annotated with `↑` (indicating the increment operation).
- Exit:** A red arrow points from the end of the loop body to the text `exit loop.`

$i = 1 \rightarrow 2 \rightarrow \dots \rightarrow 10 \rightarrow 11$       o/p  $\Rightarrow 1\ 2\ \dots\ 10$

$\downarrow$   
exit loop.

Q → write a for loop to print odd Number from 1 to N; (int N = sc.nextInt();)

1 → 10

i <= 10

i <= 20

⋮

i <= N

1 → 10

10 → 1

(i <= 10; i >= 1; i++)

N = 10 → 1 3 5 7 9 ← O/P.

```
for (int i = 1; i <= N; i++) {
    if (i % 2 != 0) {
        sop(i);
    }
}
```

exit loop.

2 % 2 (0 != 0) ↓ false.

i = 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 O/P ⇒ 1-3-5-7-9-

→ 9 → 10 → 11

```
for (int i = 1; i <= N; i += 2) {
    sop(i);
}
```

exit loop

1 <sup>+2</sup> ⇒ 3 <sup>+2</sup> ⇒ 5 <sup>+2</sup> ⇒ 7 <sup>+2</sup> ⇒ 9 <sup>+2</sup> ⇒ 11  
 O/P 1 3 5 7 9  
 exit loop.

Quiz

1 → Initialisation (only once);  
 2 → condition  
 3 → loop work  
 4 → update

① 1 → 2 → 3 → 4 then stop X

② 1 → 2 → 3 → 4 → 1 and stop. X

③ 1 → 2 → 3 → 4 → start from 3 X

④ 1 → 2 → 3 → 4 → then again start from 1 ✓

Que-2

1)  $0 \rightarrow N$  ~~X~~  
sop(i);

2)  $i=1 \rightarrow i=N$  (i++)  
sop(i)  $\rightarrow$  odd num ~~X~~

3) int i  
 $i=2$   $i=N$   $i++$  ~~X~~

3, 4, 5, 6

4)  $i=2 \rightarrow i=N = i+=2$   
2 4 6 8 10...

Q23  $\rightarrow$  (Pm)

A  $\Rightarrow$  (int i);  $\rightarrow$  Declaration  
for (i=N; i >= 1; i--)  $\rightarrow$  scope of i  
sop(i);  $\rightarrow$   
sop(i)  $\rightarrow ?$  (Pm)

initialising variable

Q 4]

psvm (string array) &

→ for (int i = 1; i <= 3; i++) {  
    cout << i << endl;  
}

→ `sopl(i)`; → error → C-E  
i not defined.

Q 5)

psvm (String arr) d

→ int i=1;

→ for (int i = 1; i <= 3; i++) {  
    cout << i << " ";  
}

9  
sopln (i)

$$\text{int } i = 1$$

i already declared  
main method.

Q67

psvm (strong ar) d

↪ int i;

$\rightarrow$  for (i = 1; — i <= 3 ; i++) {  
    sopln(i);  
}

 $\log \ln(i)$ 

exit loop  $\rightarrow$  stop in  $C_i$  ;

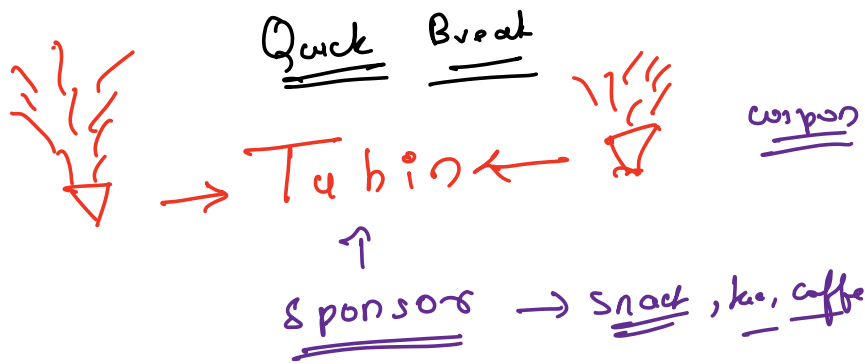


10000

$$i = 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$$

o/p

- 1
- 2
- 3
- 4



20:27 ⇒ 10:40 pm

Q ⇒ Print first and last digit of a number

$N = 6175$   
 ??? = first digit ←   → 5 = last digit →  $N \% 10$

- 1) declare  $fd$ ;
- 2) store  $d \rightarrow$  digit inside  $fd$ .
- 2) outside loop print  $fd$ ;

$fd \rightarrow$  first digit

getting rid  
of last digit

o/p 6 → fd  
5 → ld.   int fd;

$617(5) \leftarrow$   
 $d = n \% 10 \rightarrow 5$   
sep(d);    $fd = d$ ;

$n = n / 10. \Rightarrow 617$

o/p 5 7 1 (5) → fd  
 ↳  
 sep(fd);  
 sep(ld);

```
int fd = 0;
```

```
int fd = 0;
```

```
int fd = 0;
```

```
int d = 17.10;
```

$$f(d) = d;$$

3

$$\text{stop}(\underline{f(d)}); \rightarrow L'$$

→  $\text{sop}(d);$

 $i = i(10')$ 

4

(17.10)

$\begin{array}{c} 0 \\ 1 \end{array}$      $i \geq 0$      $d$      $f d$      $i = 110$   
 $1432$     T     $2 \rightarrow 2$      $\begin{array}{r} 143 \\ \hline \end{array}$   
 $143$     T     $3 \rightarrow 3$      $14$   
 $14$     T     $4 \rightarrow 4$      $\textcircled{1} \leftarrow \text{copy}$   
 $1$     T     $1 \rightarrow 1$      $0$

0 → f → exit loop

Old

fd7!

$$k_d = 2$$

$N = 6123 \Rightarrow 3216$

715  $\Rightarrow$  517.

~~Print~~

```
int n = 421
```

```
int rev = 124;
```

$$\angle C \overset{\text{bring}}{\overbrace{=}} X$$

int n = 143  
 $\downarrow$   
 $14 \times 10$   
 $\downarrow$   
 $140 + 3$   
 $\downarrow$   
 143

$\Rightarrow \underline{\underline{143}}$

n = 61257  
 $\downarrow$   
 $6125 \times 10$   
 $\downarrow$   
 $61250 + 7$   
 $\downarrow$   
 61257

$$\boxed{rev = rev \times 10 + d;}$$

Adding digit to a number at the last

3

$N = 6143 \Rightarrow \boxed{\text{int rev} = 0}; \text{int d;}$

$$\boxed{rev = rev \times 10 + (d);}$$

$$d = N \% 10; \rightarrow 3$$

$$\begin{aligned} rev &= 0 \times 10 + 3 \\ &= 0 + 3 \\ &= 3. \end{aligned}$$

$\Rightarrow$  approach  $\Rightarrow$

6143  
 $\Rightarrow \underline{\underline{3416}}$

1)  $rev = 0;$

2)  $d = n \% 10$

3)  $rev = rev \times 10 + d$

4)  $n = n / 10;$

Condition

```
int n = sc.nextInt();
```

```
int rev = 0;
```

```
for (int i = n; i > 0; i = i / 10) {
```

```
    int d = n % 10;
```

```
    rev = rev * 10 + d;
```

}

```
    System.out.println(rev);
```

Output 2413

Input 3142



Doubt

→ Armstrong number

$$\begin{aligned} \hookrightarrow \underline{153} &\Rightarrow 1^3 + 5^3 + 3^3 \\ &\Rightarrow 1 + 125 + 27 \\ &\Rightarrow \underline{153} \end{aligned}$$

(num == sum) (limited 3 digits)

↓  
armstrong number.

1 3 2 2 2 → 4

① ②  
12  
—  
3

int N = sc.nextInt();

int num = 1; - 2 - 3 - ...

while (num != N)

int sum = 0;

while (num > 0)

int d = num % 10;

sum = sum + (d \* d \* d);

num = num / 10;

if (num == sum)

sc.println("Armstrong number");

num++;

1) take a loop from 1 → N

2) take a loop inside the above loop.

3) use the inner loop to find all digits

int sum = 0;

cube

sum = sum + (d \* d \* d);

loop → 1 → N

inner loop → (loop 2 to check sum)

→ if (num == sum) { print num }

num++

5) compare sum with num

if true  $\rightarrow$  Print if false - ignore

```
long bal = sc.nextLong();
int T = sc.nextInt();
while (T > 0) {
    int op_type = sc.nextInt();
    long amt = sc.nextLong();
    if (op_type == 1) {
        bal = bal + amt;
        sop(bal);
    } else {
        if (bal > amt) {
            bal -= amt;
        } else {
            sop("Insufficient");
        }
    }
    T--;
}
```

```
int A = 5;
int B = 10;
while (A < B) {
    sop(A + " ");
    A++;
    sop(B);
}
```

( 5 \_ 6 \_ 7 \_ 8 \_ 9 \_ 10 )

$$N = \underline{\underline{35}} \leftarrow \underline{\underline{2}} \rightarrow 1$$

$$35 \neq 2 \rightarrow \frac{17}{2} \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

( ① - ② - ③ - ④ - ⑤ )

steps 0; n = 35  
 while (n != 1) {  
 n = n / 2;  
 steps++;  
 }  
 return steps;

35 - 17 → 8  
 4  
 2  
 1

for(int i = 0) only once i < 10 ; i++ }d

4

→ int i = 0 ; only once  
{ while ( i < 10 ) {  
    printf("%d\n", i);  
    i++;  
}