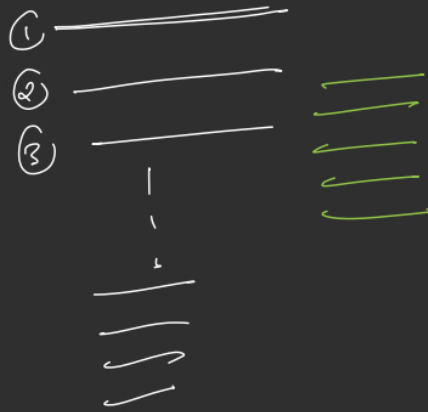


A hand-drawn diagram of a stack data structure. It consists of a vertical rectangle with a curly brace on the left side and three horizontal lines inside, representing the stack's contents. The number '3' is written below the rectangle.

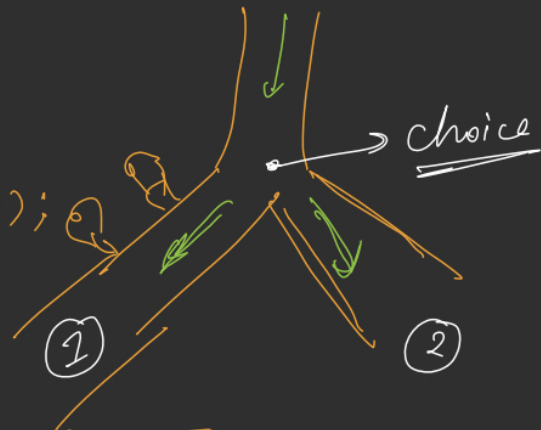


Selection ✓

- ↳ if()
- ↳ if-else()
- ↳ switch()

Iteration

Jumps


$$\{C, C, C, C\}$$

3

```

if (expression) {
    // do something.
}

```

True / False
boolean.

```

boolean b = true;
if (b == true) {
    // do something
}

```

False

b == true

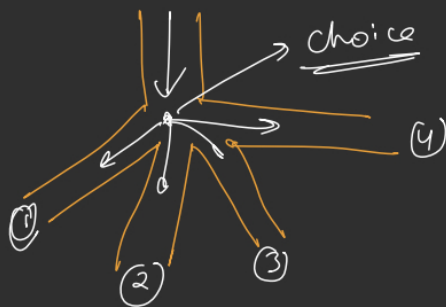
```

if ( ) {
}
else {
}

```

Switch statements

- ↳ if-else-if ladder
- ↳ More optimized.



```

if (choice == 1) {
    // go to 1
}
else if (c == 2) {
    // go to 2
}

```

Switch v/s if-else-if ladder

- ① Switch can only test equality but if-else can test both equality & inequality.
- ② Switch is not efficient than if-else-if ladder.

int i = 2;

Switch (i) {

Case 1 :
// do something
break;

Case 2 :
// do something
break;

Case 3 :
// do something
break;

default :
// do something
break;

}

if (i == 1) {
// do something.
}

else if (i == 2) {
// do something
}

else if (i == 3) {
// do something
}

else {
// do something.
}

i → 2

Jump tables

C-1 L3	C-2 L3	C-3 L3	L-53
0	1	2	3

Jump table ✓✓

↳ Jump tables are not always efficient

↳ $i = 3$

Switch(i) {

Case 1: _____

Case 1000: _____

} Case: 1,000,000 _____

C-1	null	null	2	---	C-1000
0	1	2	3		1000

Sparse
✓ values

Jump table

Table-switch ✓✓

0	1	2	3
0	1	2	3

Jump to

lookup-switch

C-1	1000	1,000,000
0	1	2

$i = 3000$

Case 1:

Case 1000:

Case 2000:

Case 10,000:

⋮

Case 1,000,000

Switch Statements

↳ Nested switch.

```
if ( ) {  
    if ( ) {  
    }  
}
```

```
switch ( ) {  
    switch ( ) {  
    }  
}
```