

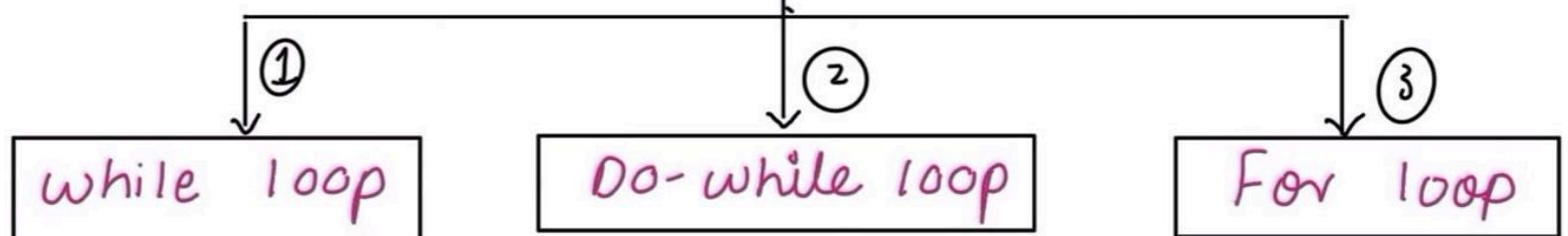
C++

Chapter 6:

Control Statements (part 2)
Looping Control Statements,
while loop, Do-while loop,
for loop, Infinite loop,
Nested loop, Summary.

looping - control statements

used to execute a statement
'n' times.
also called iterative statements.



① while loop: Do it until the condition is true.

Syntax:

```
while (condition) {  
    // statements  
}
```

while loop property: It may execute zero or more times.

Example:

// value initialization

int i = 1;

// apply while loop

while (i <= 5) {

cout << i << endl;

i++;

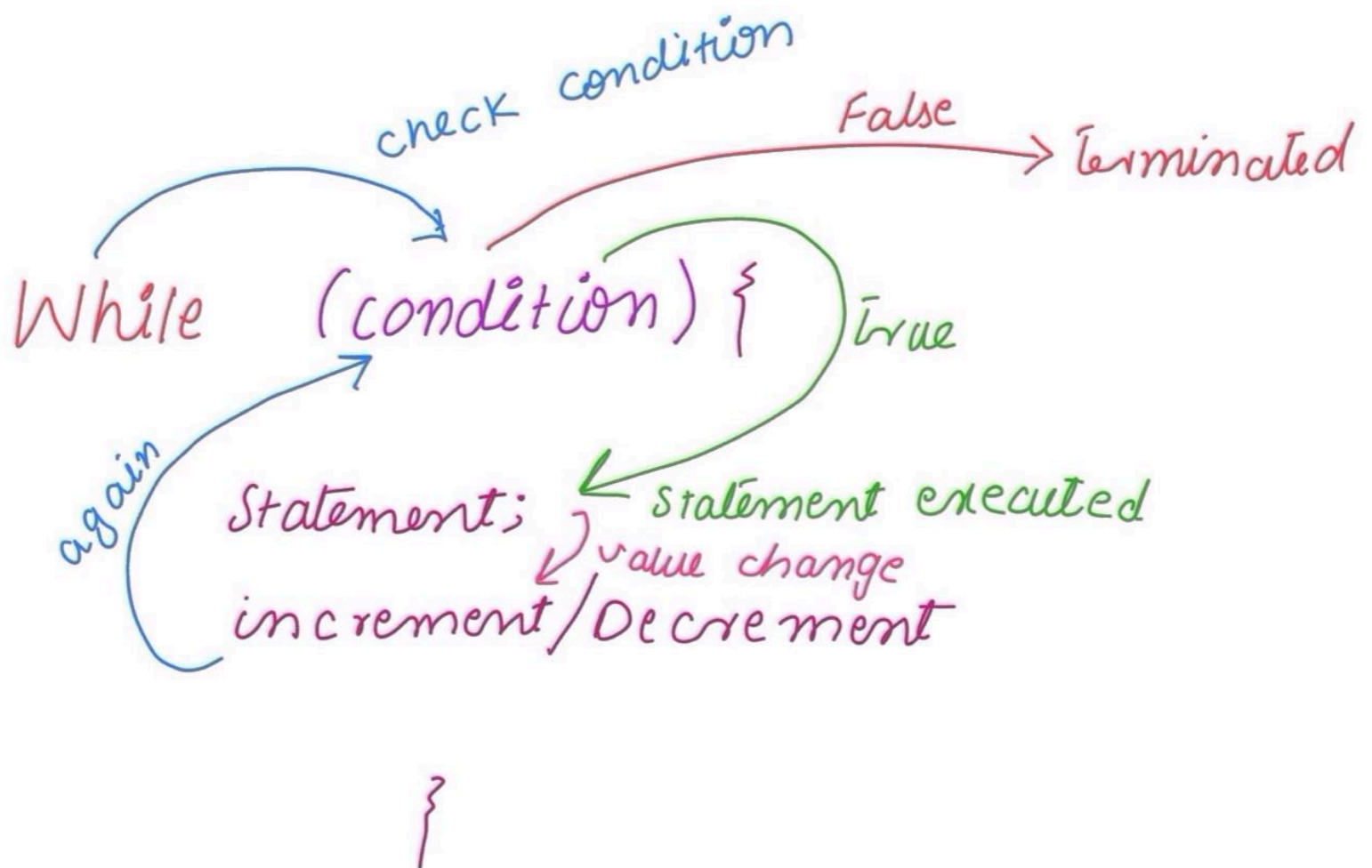
}

increment value by 1.

Condition

↓
False → loop terminated

True → loop executed.



The condition is tested first and the statements in the body are executed only when the condition is true.

② Do-while loop: Firstly (at the start) execute the statements without checking condition and then do it until the condition is true.

Syntax:

```
do {  
    statements;  
}  
while (condition);
```

Property: Do-while loop execute at least once or more times.

Condition Test: Condition is tested after execution of a loop body (means at last).

Example:

// value initialization

int i = 1;

// apply do while loop.

do {

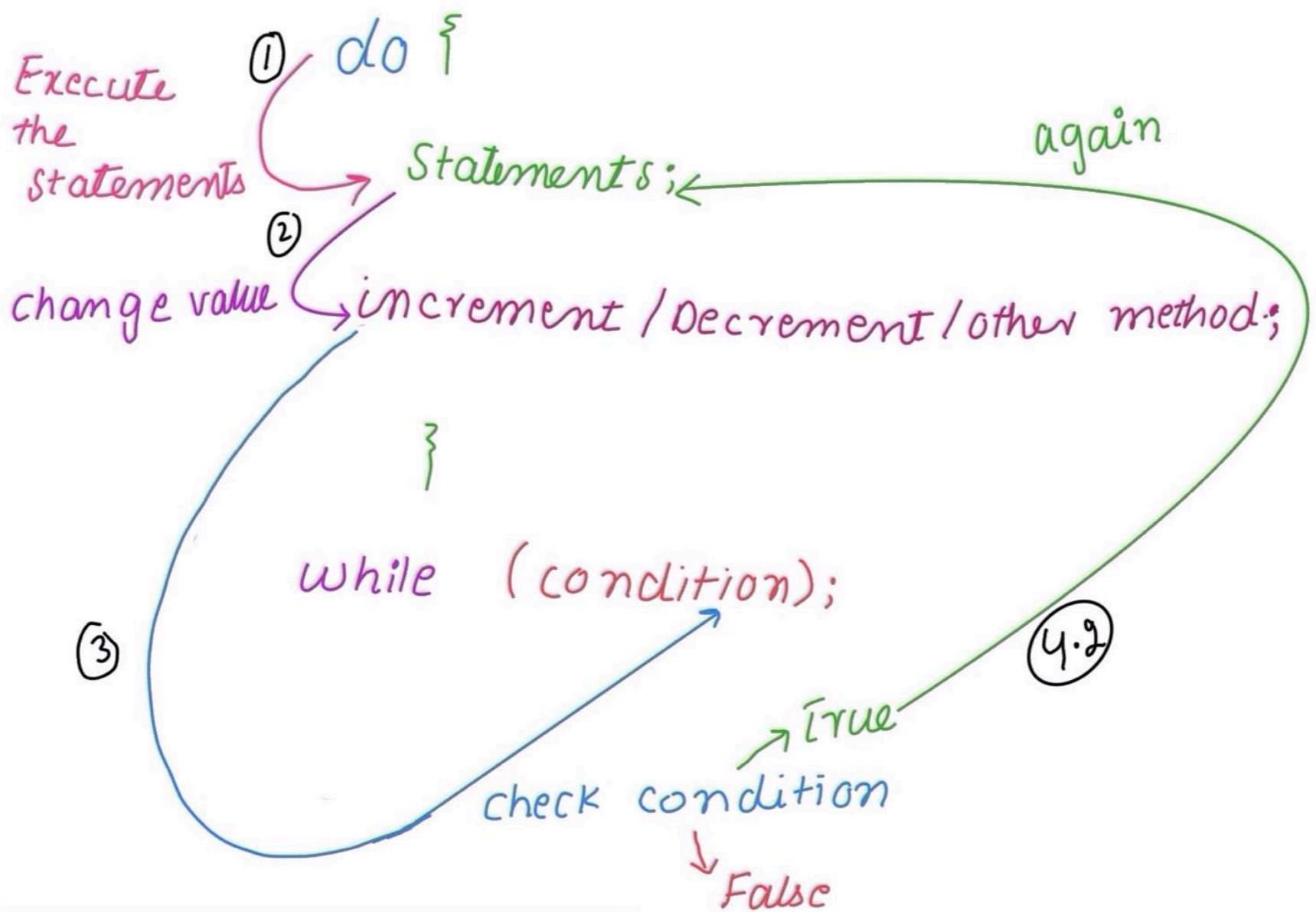
Statement
increment
value by 1

← cout << "Hello world!";

← i++;
}

while (i <= 5);

↓
condition



① Execute statement

② change value.

③ check condition

④.1 If false → Terminate

④.2 If true execute statement repeat

④.1

loop terminated

④ For loop: Repeats a block of code a specific number of times.

Syntax:

```
for (initialization; condition; increment/dec){
```

```
    Statement 1;
```

```
    Statement 2;
```

```
    "        "
```

```
}
```

Property: For loop execute zero or more times.

Condition Test: condition is tested before the execution of a loop body.

Example:

`for (int i=1; i<5; i++){`

initialization condition increment value

`cout << i;` → statement for execution.

}

"Or"

`int i = 1;` → initialization outside the loop.

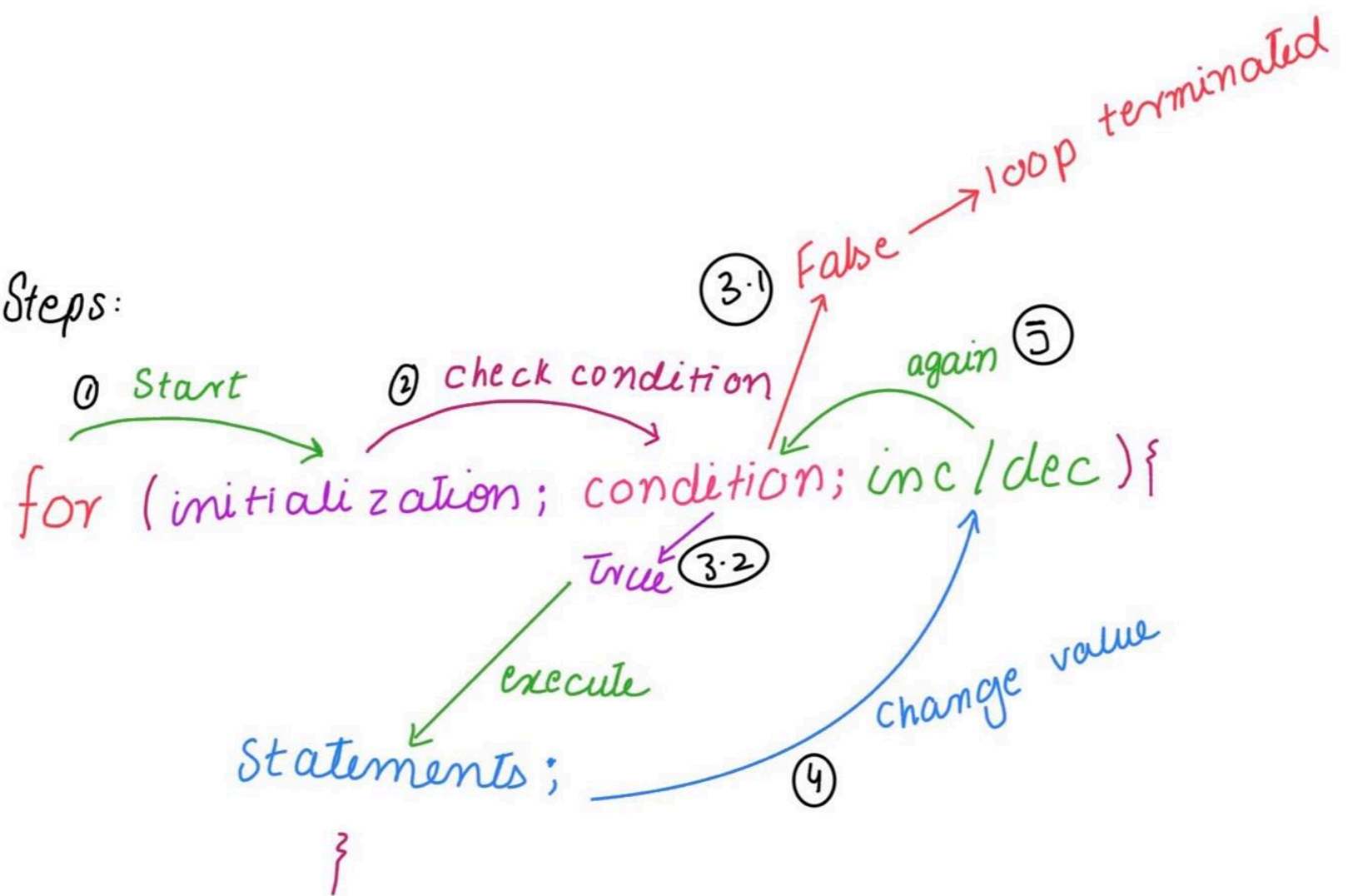
`for (i; i<5; i++){`

mention name
of variable

`cout << i;`

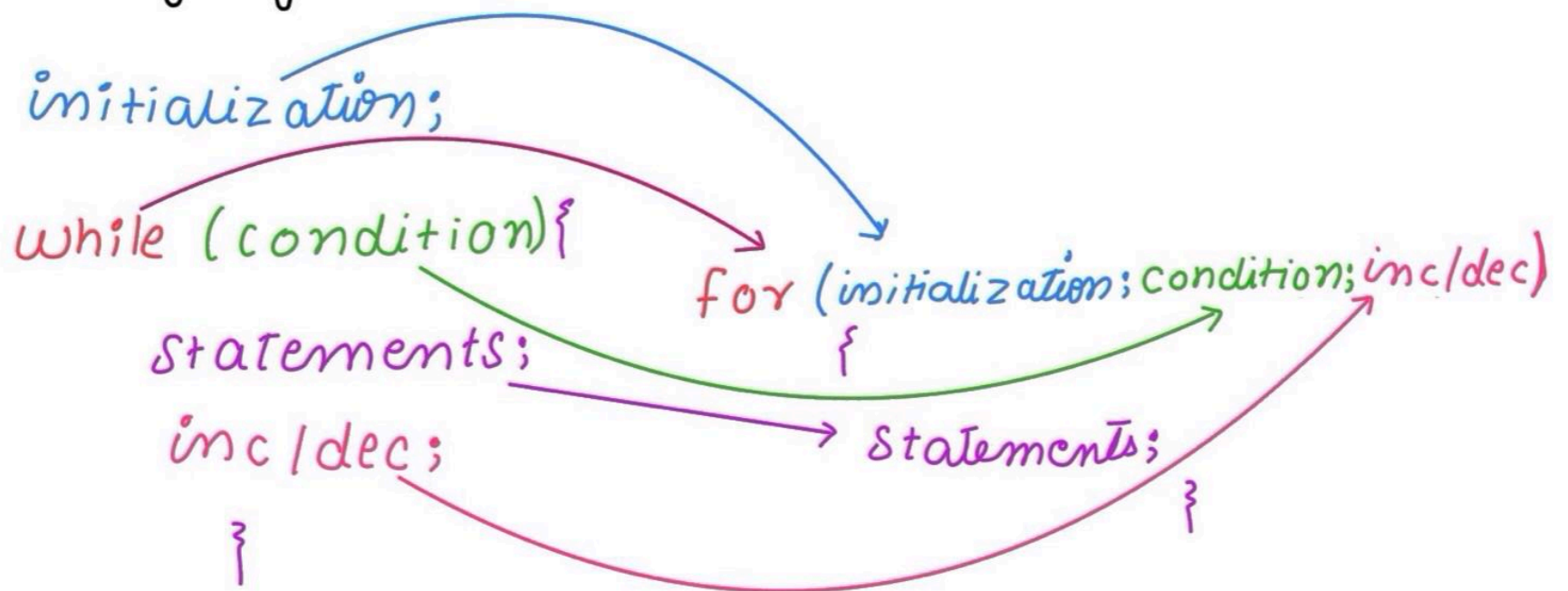
}

Steps:

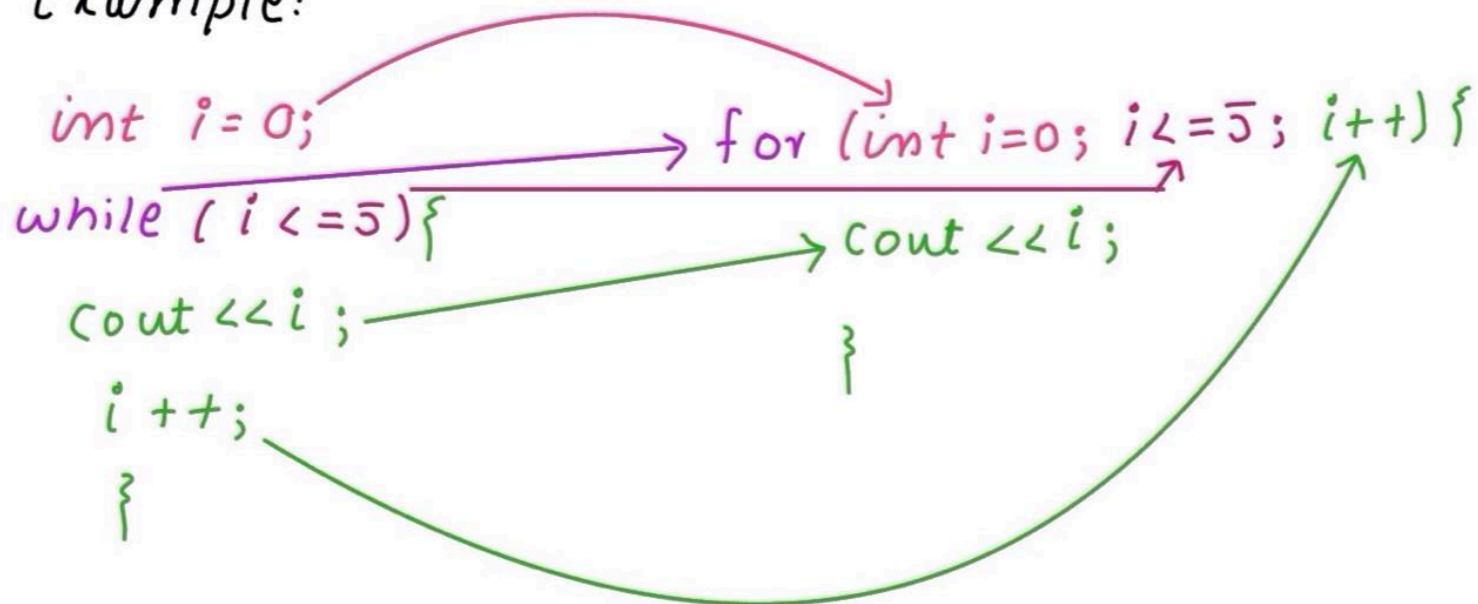


1. Start loop / initialize variable.
2. check condition
- 3.1. If condition is false terminate the loop.
- 3.2. If condition is true execute the statements.
4. change value by increment/decrement or other way.
5. Again repeat from step 2.

changing while loop \rightarrow for loop.



Example:



Infinite loop: Loop in which the condition is always true.
→ never stops

Examples:

For loop:

```
for (int i=1; i>=1; i++) {  
    cout << i << endl;  
}
```

→ always true.

while loop:

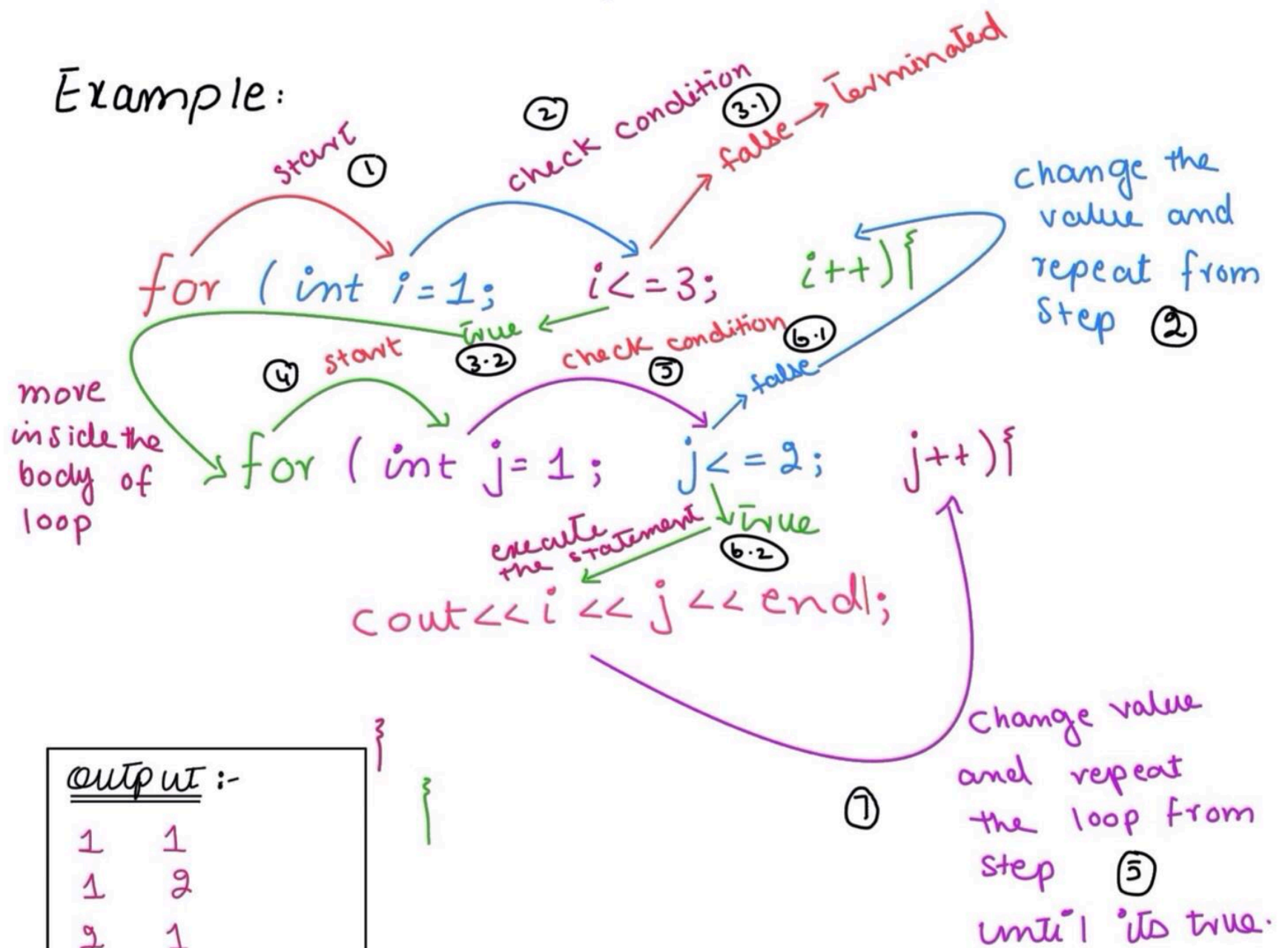
```
while (true) {  
    cout << "Hello" << endl;  
}
```

do-while loop:

```
int j=0;  
do {  
    cout << j << endl;  
}  
while (j >= 0);
```

Nested Loop:- Loop inside another loop.

Example:



output:-

1	1
1	2
2	1
2	2
3	1
3	2

1st Loop

- ① start the loop.
- ② check the condition.
- ③.1 If condition is false → Terminate the loop.
- ③.2 If condition is true → move to nested loop.

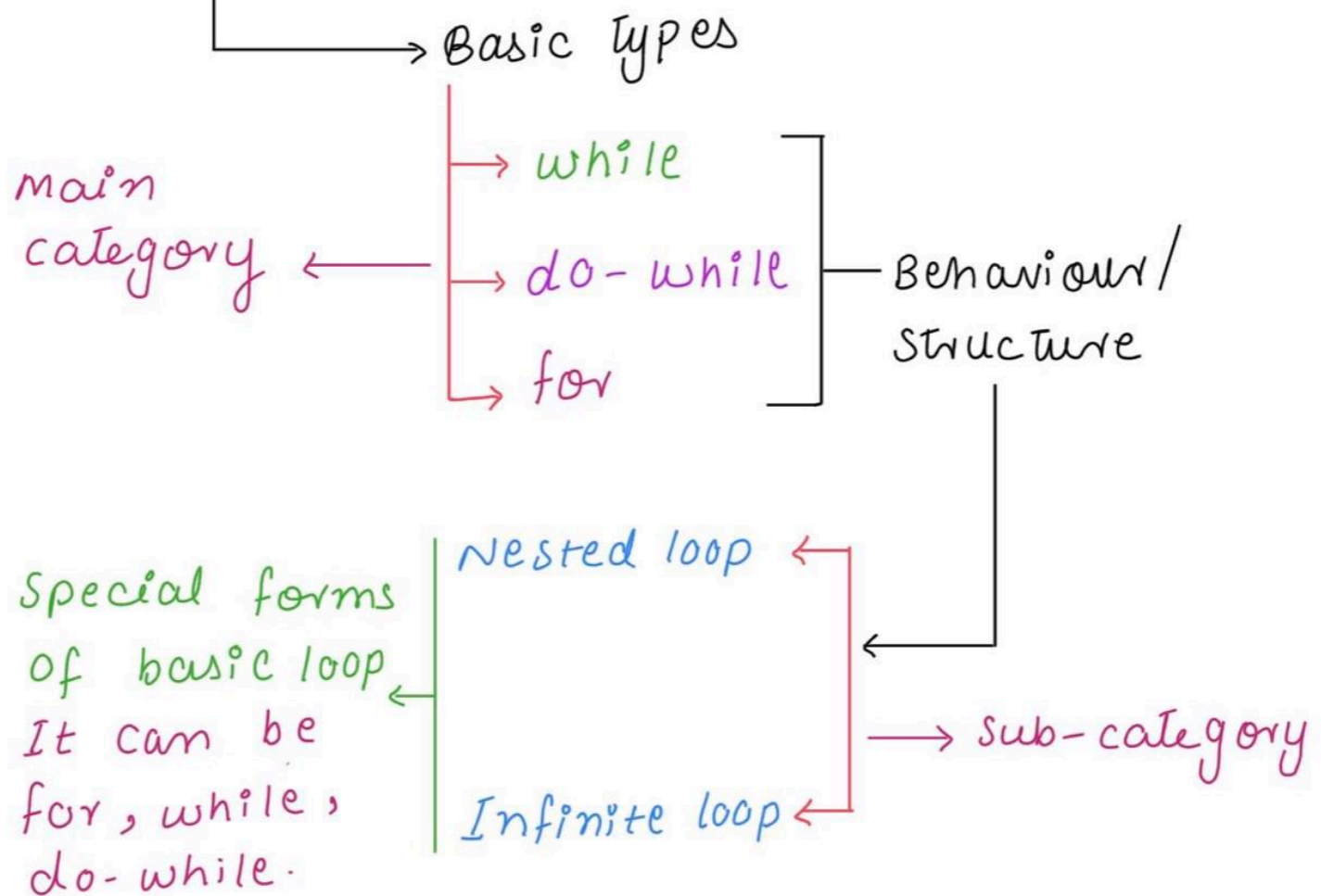
Nested Loop

- ④ Start the loop.
- ⑤ check the condition
- ⑥.1 If condition is false → move to 1st loop, change the value and repeat from step ②
- ⑥.2 If condition is true → execute the statement inside nested loop.
- ⑦ change value by inc/dec and repeat the nested loop from step ⑤

NOTE:

For the execution of nested loop the condition of main (1st) loop must be true.

Loops



* Summary

Loop	Syntax	when to use
while	while (condition) { statements; }	run until condition is true unknown end.
do-while	do { statements; } while (condition);	must run at least once. whether condition is true or false.
for	for (initialization; condition; change val) { statements; }	Repeat a known numbers of time. known end.
Infinite	ANY OF THE ABOVE	Non stopping. condition is always true.
Nested	ANY of the ABOVE loop 1 { main loop loop 2 nested loop }	to run loop inside loop.

C++

Chapter 7:

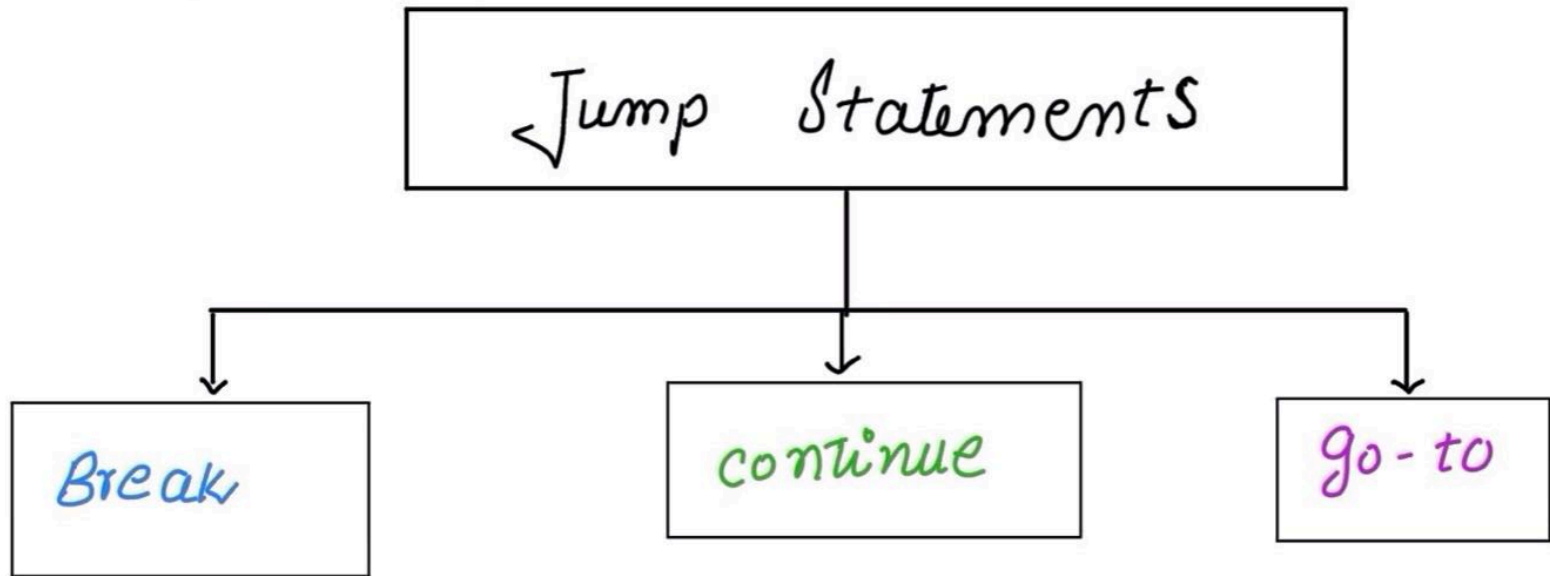
control statements (part 3)

Jump statements, Break

continue, go-to.

For complete beginners

@helloWorld-275.



1. Break: Immediately stop a loop or switch statements.
The program jumps out of the loop or out of the switch block, skipping the remaining code inside.

Syntax: `break;`

Example:

```
for (int i=1; i<=10; i++){  
    if (i==5){  
        break;  
    }  
    cout << i << endl;  
}
```

output:

```
1  
2  
3  
4
```

loop break
at 5

Example 2:

```
Switch (grade){
```

```
Case 'A':
```

```
    cout << "outstanding";  
    break;
```

```
case 'B':
```

```
    cout << "Excellent";  
    break;
```

```
case 'C':
```

```
    cout << "Good";  
    break;
```

```
case 'D':
```

```
    cout << "Average";  
    break;
```

```
Case 'E':
```

```
    cout << "Poor";  
    break;
```

```
case 'F':
```

```
    cout << "Fail";  
    break;
```

```
default:
```

```
    cout << "Enter grade from A-F";  
}
```

check if condition match then execute the statement and break the switch means jump out of switch and skip the remaining case.

if condition not matches then move to next case for checking.

2. Continue: Skip the current iteration of a loop and move directly to next iteration.

loop does not stop.
code written after continue is ignored for that iteration.

Syntax: `continue;`

Example:

```
for (int i=1; i<=5; i++){  
    if (i==3){  
        continue; // skip when i is 3.  
    }  
    cout << i << endl;  
}
```

output:

```
1  
2 // 3 is not  
  printed.  
4  
5
```



3. Go-to: Transfer the flow of execution of a program. "or"

Jump directly to another part of a program.

use label for location.

Syntax:

`goto label;`

any name → where you want to go.

Example:

```
int age;
```

```
Start: //label.
```

```
cout << "Enter your age";  
cin >> age;
```

```
if (age < 0) {
```

```
    cout << "Invalid age. Try Again!"
```

```
    goto Start;
```

```
}
```

```
cout << "Age: " << age;
```



C++

Chapter 8:

variable Naming Rules,
Size of operator,
Operators precedence.

Variable Naming Rules:

1. variable name can start only with:

- A letter (A-z).
- An underscore (-).

Cannot start with:

- A number (0-9).
- Any symbol (@, #, %, &, *)

2. After the first character, you can use:

- letter (A-z).
- Digits (0-9)
- underscore (-)

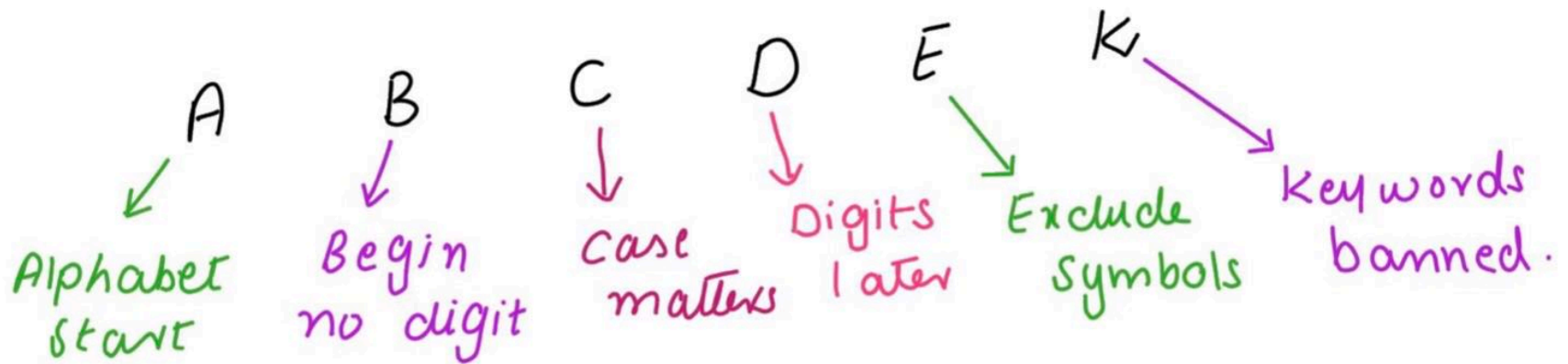
Still cannot use symbols.

3. Variable name is case-sensitive

Age \neq age \neq AGE

4. No space allowed.
5. Keywords cannot be used as identifier.
6. choose meaningful names.

Trick 1:



Examples:

```
int age; ✓  
int -marks; ✓  
int 5score; X Can't start with digit  
int @date; X can't use symbols.
```



int total9; ✓

int Student_name; ✓

int amount\$; ✗ no symbol allowed.

int na#me; ✗ no symbol allowed.

int my_name; ✓

int my name; ✗ no space allowed.

int else; ✗ no keyword allowed.

int Else; ✓

String x; | Both are correct but
String name; | name is clear and meaningful.

int A_1; ✓

int A-1; ✗ minus (-) not allowed.

Sizeof operator:

- How much memory a datatype or variable use.
- output shows no of bytes. (number)

Syntax:

sizeof (variable name / datatype).

Example:

sizeof (int);
sizeof (float);

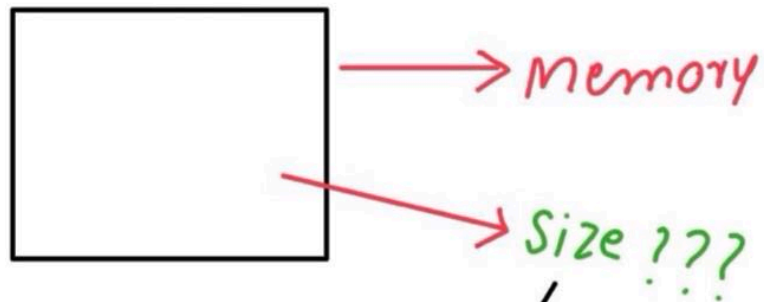
| datatype.

int a;

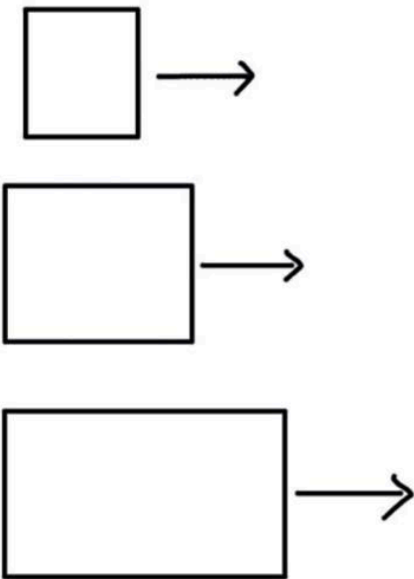
String name;

sizeof (a);
sizeof (name);

| variable names



Find by using operator `sizeof`



All the three boxes have different size.

Similarly in memory different variables/datatypes have different size.

Operators Precedence:

- * which operator is evaluated first.
- * Multiple operators in one expression.
- * Some are solved before other.
- * Sequence of operators evaluation.

Rules:

1. Bracket first $\Rightarrow () \rightarrow$ inside solve first.
2. unary next $\Rightarrow ++, --, !, \text{sizeof}$.
3. Multiply, divide before add sub $\Rightarrow *, /, \% \rightarrow +, -$
4. comparison after math: $\Rightarrow <, >, <=, >=$.
5. Equality $\Rightarrow !=, ==$.
6. Logical $\Rightarrow \&\&, \|\|$.
7. Assignment last $\Rightarrow =, +=, -=, *=, /=, \%=$.

Example 1:

$$3 + 4 * 2.$$

1st step: multiply

$$4 * 2 = 8$$

$$3 + 8$$

2nd step: addition

$$3 + 8$$

$$11$$

$$3 + 4 * 2$$

$$7 * 2$$

$$14$$

X

wrong

Example 2:

$$a + b > c \ \&\& \ d == 0.$$

① $a + b$

② $>$ comparison

③ $==$ comparison (Equality).

④ $\&\&$ logical.

— (Summary) —

Priority	operators	Meaning
1 (Highest)	()	Brackets first
2	++, --, !	unary operators
3	*, /, %	Multiply, Division, Modulus
4	+, -	Addition, Subtraction
5	<, >, <=, >=	Comparisons
6	==, !=	Equality check
7	&&	logical AND
8		Logical OR
9 (lowest)	=, +=, -= etc	Assignment