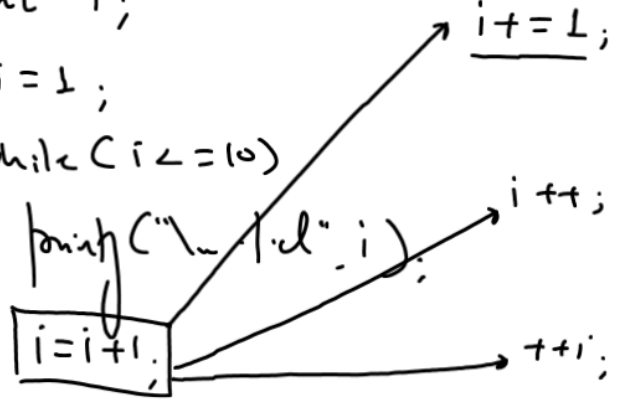
Using Shorthand & Increment / Decrement op

```
int main()
{
    int i;
    i = 1;
    while(i<=10)
    {
        printf("\n %d", i);
        i=i+1;
    }
    return 0;
}
```



① `a = a + b;`

or

`a += b;`

② `i = i * j;`

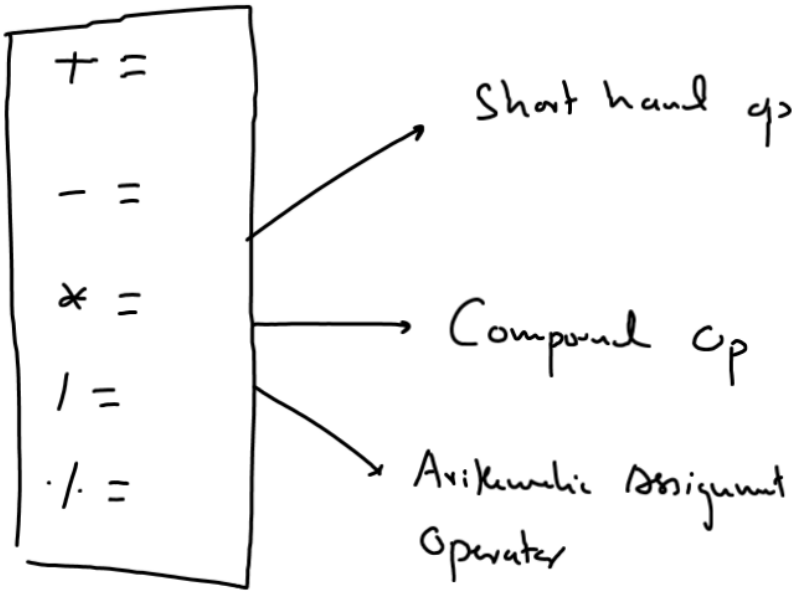
or

`i *= j;`

③ `x = x - y;`

or

`x -= y;`

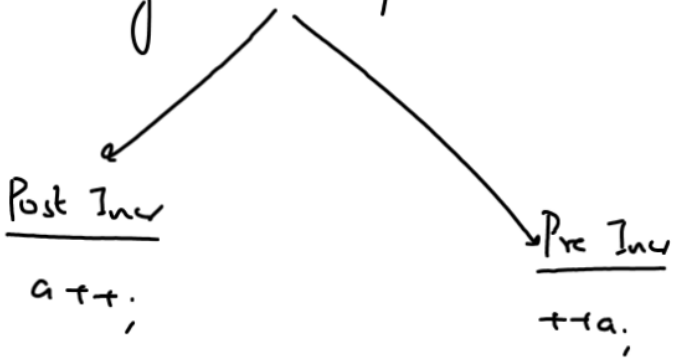



```
int a;

a = 10;

a = a + 1;      a++;
                ++a;

printf("%d", a);
```

Using Increment Op

++

Unary Incr Op




```
int b=10;

b++;

printf("%d", b);
```

```
int b=10;

++b;

printf("%d", b);
```

```
int a, b=10;

a = b++;
```

a = b;

b++;

```
printf("%d %d", a, b);
```

10 11

```
int a, b=10;

a = ++b;
```

++b;

a = b;

```
printf("%d %d", a, b);
```

11 11

```
int x=2, y, z;
```

~~2~~ 3 4	2	4
x	y	z

```
y = x++;
```

y = x;

x++;

```
z = ++x;
```

++x

z = x

```
printf("%d %d %d", x, y, z);
```

~~2~~ 3 4	2	3
~~3~~ 2 4		
~~4~~ 2 3		
4 2 4 ✓		

~~2~~ 3 3	3
3 3 4	
~~3~~ 2 4	

~~2~~ 3 4	4
~~2~~ 3 2	

int x = 2, y, z;

x	y	z
~~234~~	~~73~~	~~73~~

y = ++x;

→ ++x;

→ y = x;

z = x++;

→ z = x;

→ x++;

printf(" %d %d %d", x, y, z);

✓ 4 3 3

① Pre

② Arithmetic

③ Assignment

④ Post

int a = 5, b = 10, c;

c = a++ + b++;

printf(" %d %d %d", a, b, c);

6 11 15

```
int a=5, b=10, c;

c = ++a + b++;

printf("%d %d %d", a, b, c);
        6  11  16
```

Whenever we use incr/decr operator multiple times on the SAME VAR in the SAME EXPR then the behaviour becomes totally COMPILER DEPENDENT

```
int a=10, b;
                                  TC
b = a++ + a++;

printf("%d %d", a, b);
        12  20
```

```
int a=10, b;
b= a++ + ++a;
printf("%d %d", a, b);
```

TC

12 22

In MingW

① L -> R

② Post: Use and then incr

③ Pre: keep on incr & after exp ends then use the value

12 (++)
a

7
b

```
int a=10, b;
```

Mistake

```
b= a++ + a++;
   10  + 11
```

```
printf("%d %d", a, b);
        12  21
```

```
int a=10, b;
```

10 11 12
a

```
b= ++a + a++;
   12  + 11
```

```
printf("%d %d", a, b);
        12  23
```

++ Vs +

++	+
① Inc op	① Arithmetic op
② Unary op	② Binary op
③ Always changes the value of its opnd	③ Never changes the value of its opnd.

Based Upon

Types of Operator (No of operands)

Unary
(Req only 1 opnd)
① a++;
② ++a;

Binary
(Req 2 opnd)
① x+y

Terna
(Req 3 opnd)
① (test_cond) ? (st) : (st);

	Operator	Name	Category
①	=	Assignment	Binary
②	==	Equality Comp	Binary
③	&	Addr of op	Unary
④	+	Arithmetic +	Binary
⑤	-	Unary - OR Negation op	Unary

```
int a;
a = -6;
```

```
int a;
a = 7 - 6;
    ↓
  7 + (-6)
```

① int a = 10, b;

b = a++;

10 11

② int a = 10, b;

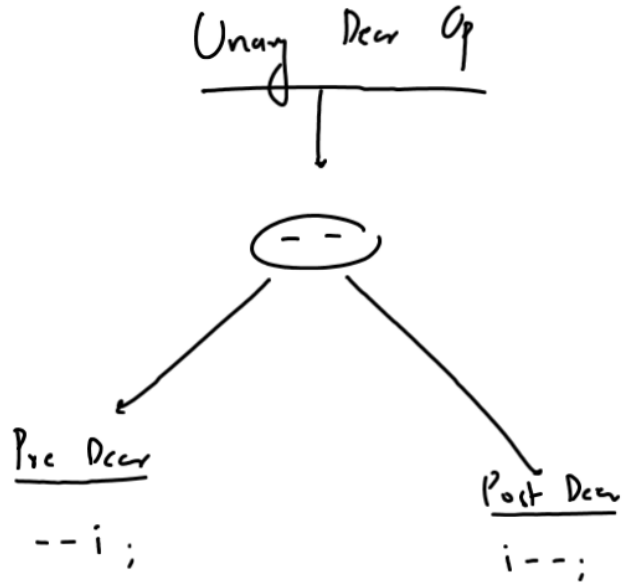
b = a + 1;

11 10

③ int a = 10, b;

b = ++a;

11 11



a++;

X a++++;

X ++a++;