

Using Short hand & Increment / Decrement op

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

Diagram illustrating the transformation of the increment operation `i = i + 1;` into its shorthand forms:

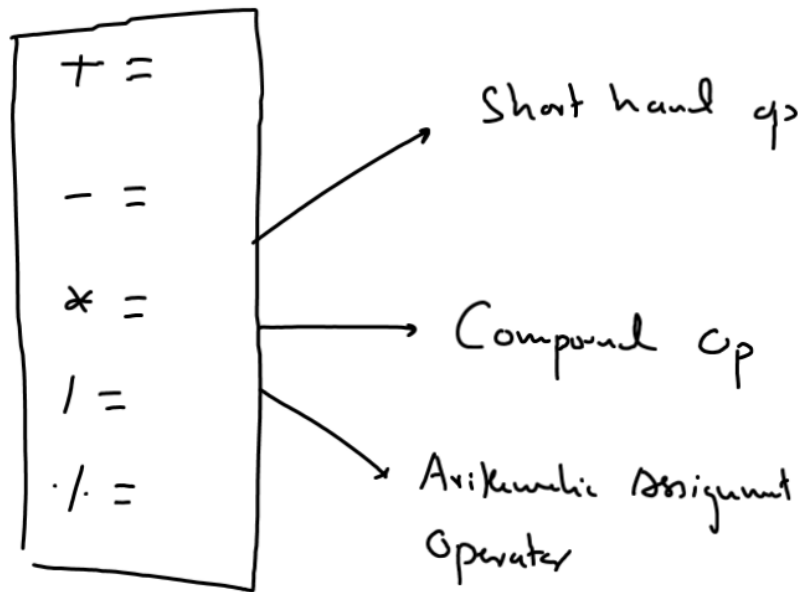
- `i++`
- `++i`
- `i += 1;`

① `a = a + b;`
or
`a += b;`

③ `x = x - y;`

or
`x -= y;`

② `i = i * j;`
or
`i *= j;`



int a;

a = 10;

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

printf("%d", a);

Using Increment Op

++

Unary Incr Op

Post Incr

a++;

Pre Incr

++a;

int b=10;

b++;

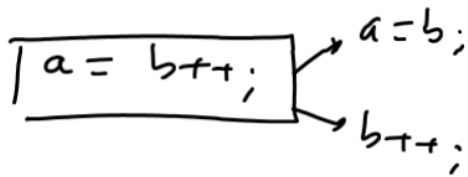
printf("%d", b);
↳ 11

int b=10;

++b;

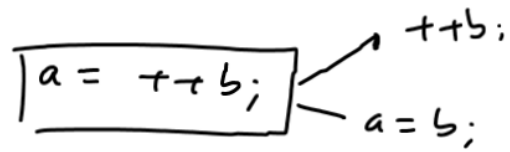
printf("%d", b);
↳ 11

int a, b=10;



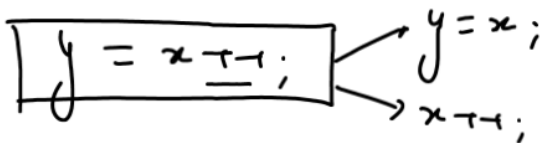
mainly (" ./d ./d", a, b);
10 11

int a, b=10;

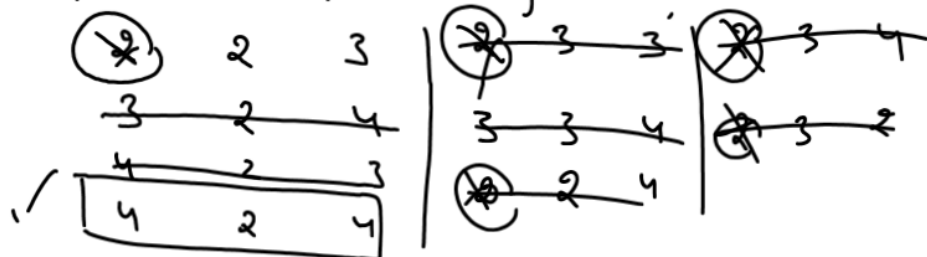


mainly (" ./d ./d", a, b);
11 11

int x=2, y, z;



mainly (" ./d ./d ./d", x, y, z).



$$\boxed{\begin{array}{r} -231 \\ \times \end{array}} \quad \boxed{\begin{array}{r} 73 \\ \times \end{array}} \quad \boxed{\begin{array}{r} 73 \\ \times \end{array}}$$

$y = t + x;$ \rightarrow $t + x;$
 $y = x;$

$z = x \tau \tau ;$ $\rightarrow z = x ;$
 $\rightarrow x \tau \tau ;$

1. $\text{min} ("1.2.3", x, y, z);$
 ✓

| | | |
|---|---|---|
| 1 | 2 | 3 |
|---|---|---|

① P_{re}

② Anikuntic

③ Assignment

④ Post

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

$$c = a_{t+1} + b_{t+1};$$
$$\text{Prinly}(\begin{matrix} \cdot & \cdot & \cdot & \cdot \\ & 6 & 11 & 15 \end{matrix}, a, b, c);$$

```
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;
```

7C

```
b = a++ + a++;
```

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

int a=10, b;

Tc

b = a++ + ++a;

printf("%d %d", 4, 5);
12 22

In MingW

① L → R

② Post: Use and then inc

③ Pre: Keep on inc & after exp ends
then use the value

int a=10, b;

$\begin{array}{|c|} \hline 12 \\ \hline \end{array}$
 $\begin{array}{|c|} \hline \text{++} \\ \hline \end{array}$
 $\begin{array}{|c|} \hline 7 \\ \hline \end{array}$
 $\begin{array}{|c|} \hline a \\ \hline \end{array}$ $\begin{array}{|c|} \hline b \\ \hline \end{array}$

Might

$b = \begin{array}{|c|} \hline a++ \\ \hline \end{array} + \begin{array}{|c|} \hline a++ \\ \hline \end{array};$
 $\begin{array}{|c|} \hline 10 \\ \hline \end{array} + \begin{array}{|c|} \hline 11 \\ \hline \end{array}$

printf("%d %d", a, b);
 $\begin{array}{|c|} \hline 12 \\ \hline \end{array}$ $\begin{array}{|c|} \hline 21 \\ \hline \end{array}$

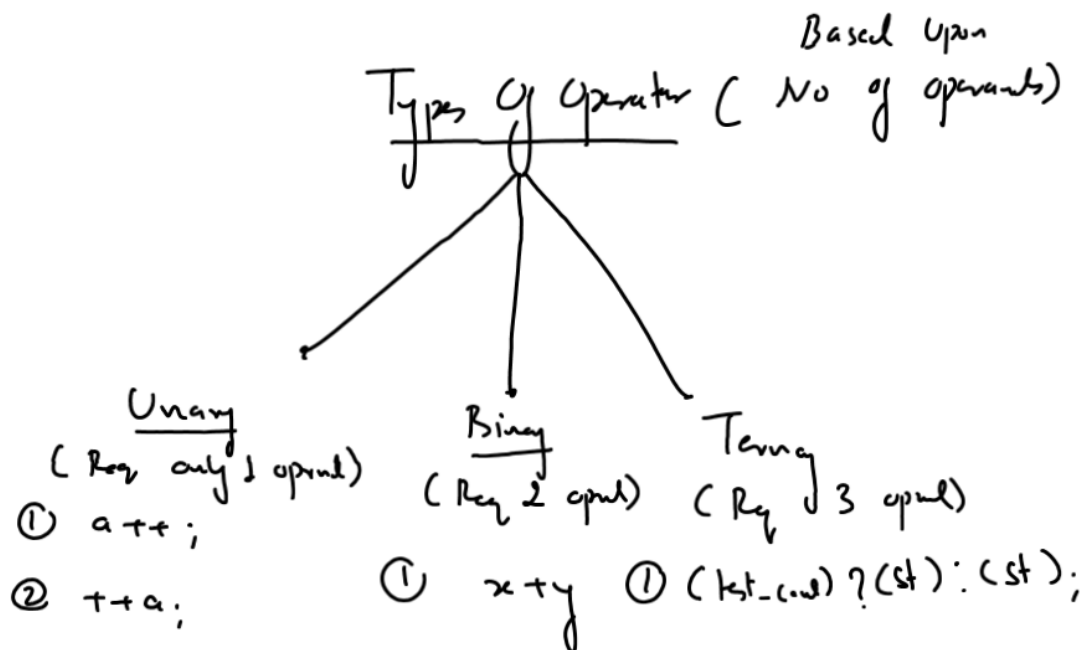
int a=10, b;

$\begin{array}{|c|} \hline 10+12 \\ \hline \end{array}$
 $\begin{array}{|c|} \hline a \\ \hline \end{array}$

$b = ++a + a++;$
 $\begin{array}{|c|} \hline 12? \\ \hline \end{array} + \begin{array}{|c|} \hline 11 \\ \hline \end{array}$

printf("%d %d", a, b);
 $\begin{array}{|c|} \hline 12 \\ \hline \end{array}$ $\begin{array}{|c|} \hline 23 \\ \hline \end{array}$

| ++ | Vls | + |
|---|---|---|
| ① Incr op | ① Arithmetic op | |
| ② Unary op | ② Binary op | |
| ③ Always changes the value of its operand | ③ Never changes the value of its operand. | |



| <u>Operator</u> | <u>Name</u> | <u>Category</u> |
|-----------------|---------------------------|-----------------|
| ① = | Assignment | Binary |
| ② == | Equality Comp | Binary |
| ③ & | Addr g op | Unary |
| ④ + | Arithmetic + | Binary |
| ⑤ - | Unary - OR Negation op | Unary |

```
int a;
```

```
a = -6;
```

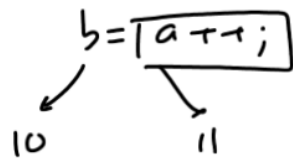
```
int a;
```

```
a = 7 - 6;
```

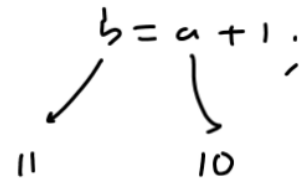
```
↓
```

```
7 + (-6)
```

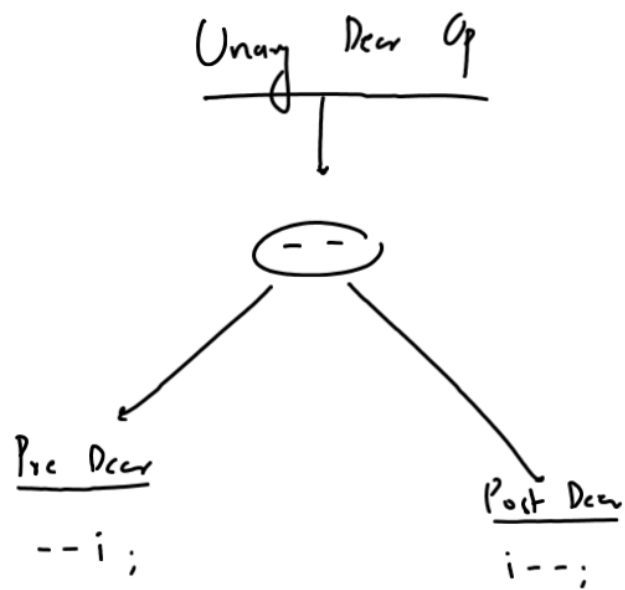
① `int a=10, b;`



③ `int a=10, b;`



② `int a=10, b;`



$$a\tau\tau;$$

$$\chi \ a\tau\tau\tau\tau;$$

$$\chi \ \tau\tau a\tau\tau;$$