# Operators (1A)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

# Pre / Post Increment

Pre Increment

```
++x;
```

```
x = x + 1;
```

Post Increment

```
x++;
```

```
x = x + 1;
```

Assignment w/ Pre Increment

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

*Increment before assigning*

```
x = x + 1;
y = x;
```

Assignment w/ Post Increment

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

*Increment after assigning*

```
y = x;
x = x + 1;
```

# Pre / Post Decrement

Pre Decrement

```
--x;
```

```
x = x - 1;
```

Post Decrement

```
x--;
```

```
x = x - 1;
```

Assignment w/ Pre Decrement

```
y = --x;
```

*Increment before assigning*

```
x = x - 1;
y = x;
```

Assignment w/ Post Decrement

```
y = x--;
```

*Increment after assigning*

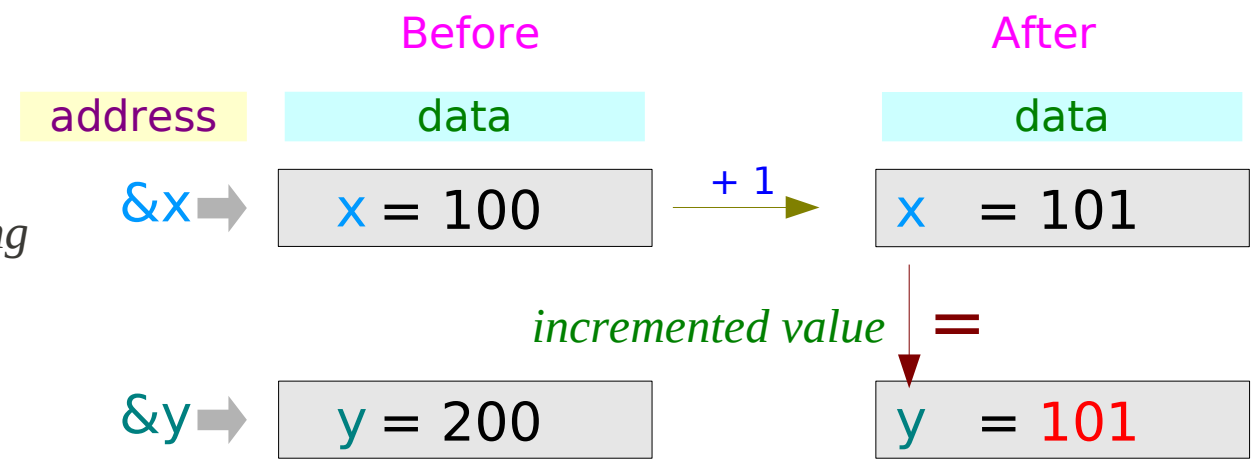
```
y = x;
x = x + 1;
```

# Pre / Post Increment Example

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

*Increment before assigning*

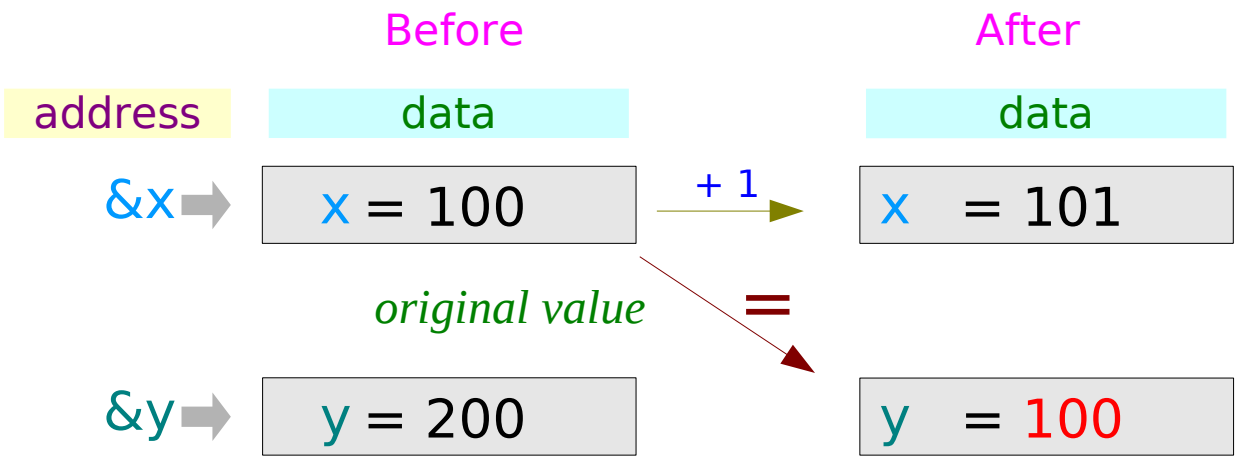
```
x = x + 1;
y = x;
```




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

*Increment after assigning*

```
y = x;
x = x + 1;
```



# Pre / Post -Increment Pointer Variable

```
p = &x;
y = *++p;
```

```
p = &x;
y = ++(*p);
```

⬌

```
p = &x;
y = ++*p;
```

```
p = p + 1;
y = *p;
```

```
*p = *p + 1;
y = *p;
```

```
p = &x;
y = *p++;
```

```
p = &x;
y = (*p)++;
```

```
y = *p;
p = p + 1;
```

```
y = *p;
*p = *p + 1;
```

```
int x = 100;
int y = 200;
int *   p;
```

*++, −− higher precedence than **

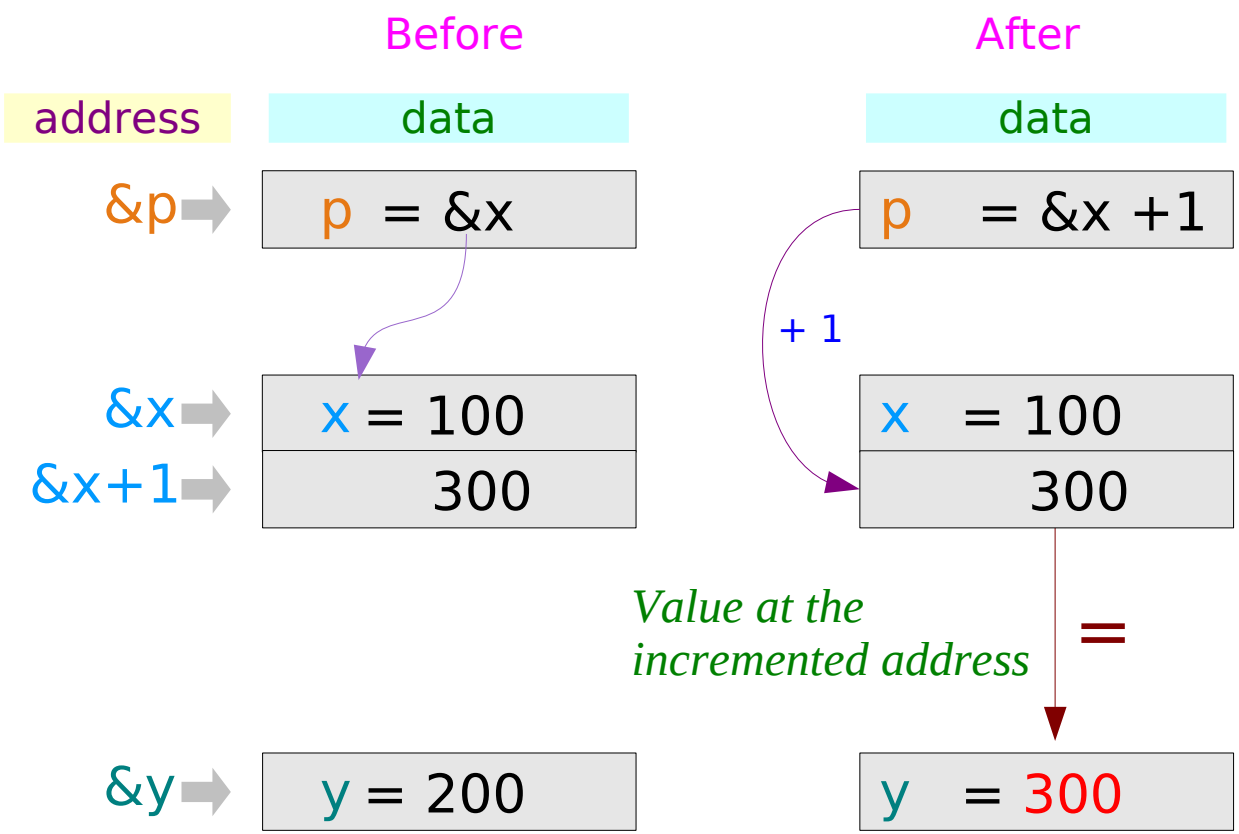
# Pre-Increment Example (1)

```
int x = 100;
int y = 200;
int *   p;
```

```
p = &x;

y = *++p;
```

```
p = p + 1;
y = *p;
```



# Pre-Increment Example (2)

```
int x = 100;
int y = 200;
int *   p;
```

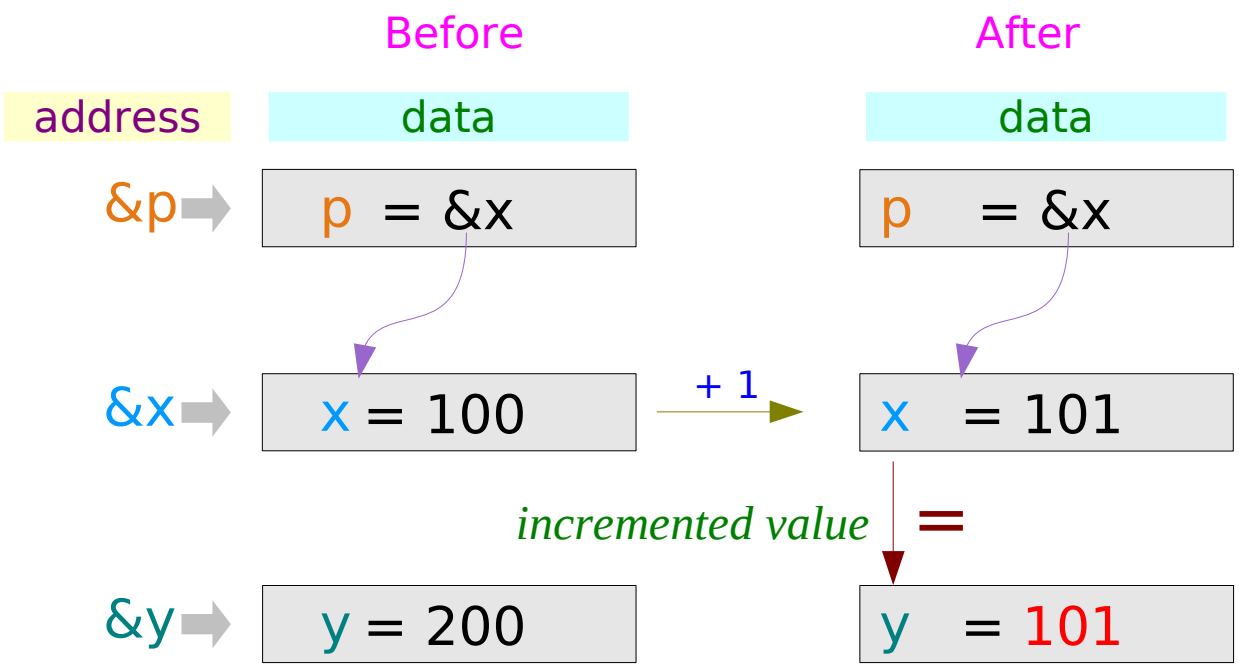
```
p = &x;

y = ++(*p);
```

⟺ `y = ++*p;`

```
*p = *p + 1;
   ⬇
 y = *p;
```

++, −− *higher precedence than* *

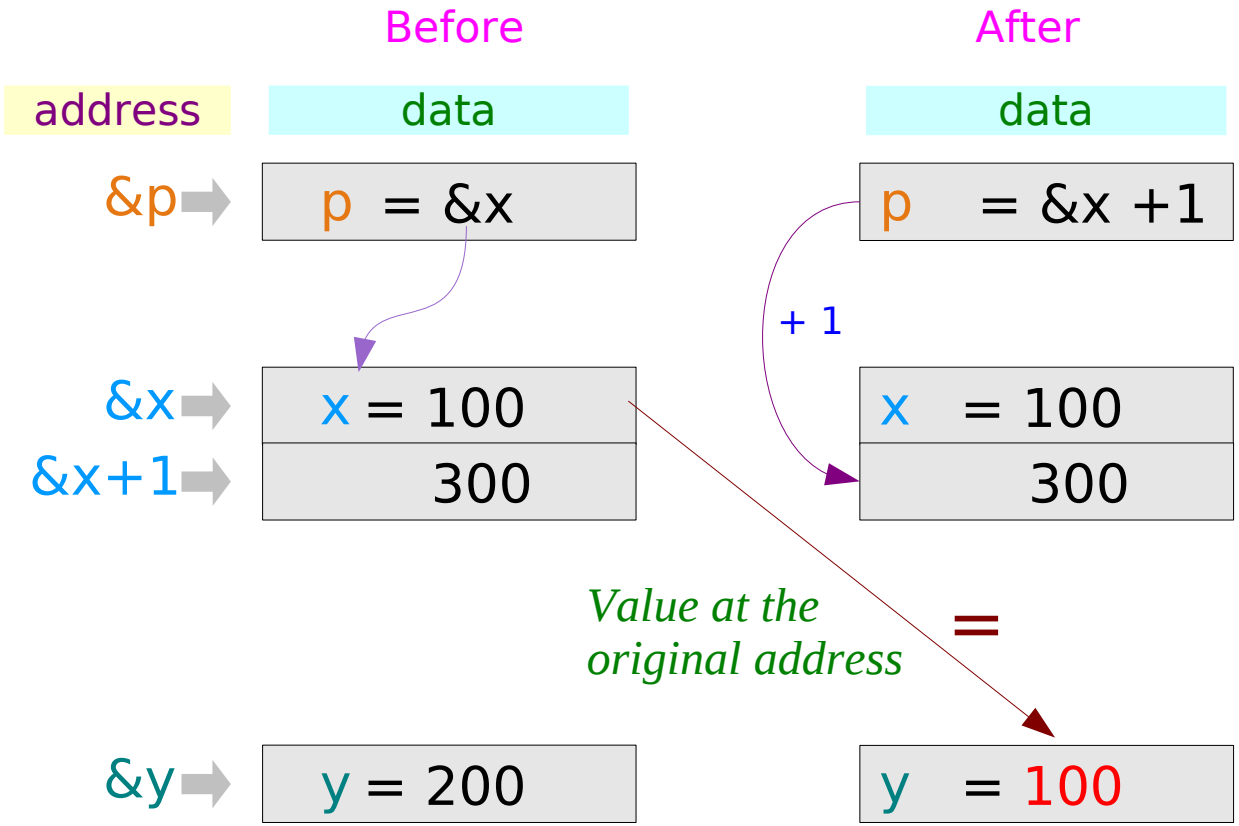


# Pre-Increment Example (3)

```
int x = 100;
int y = 200;
int *   p;
```

```
p = &x;
y = *p++;
```

```
y = *p;
p = p + 1;
```

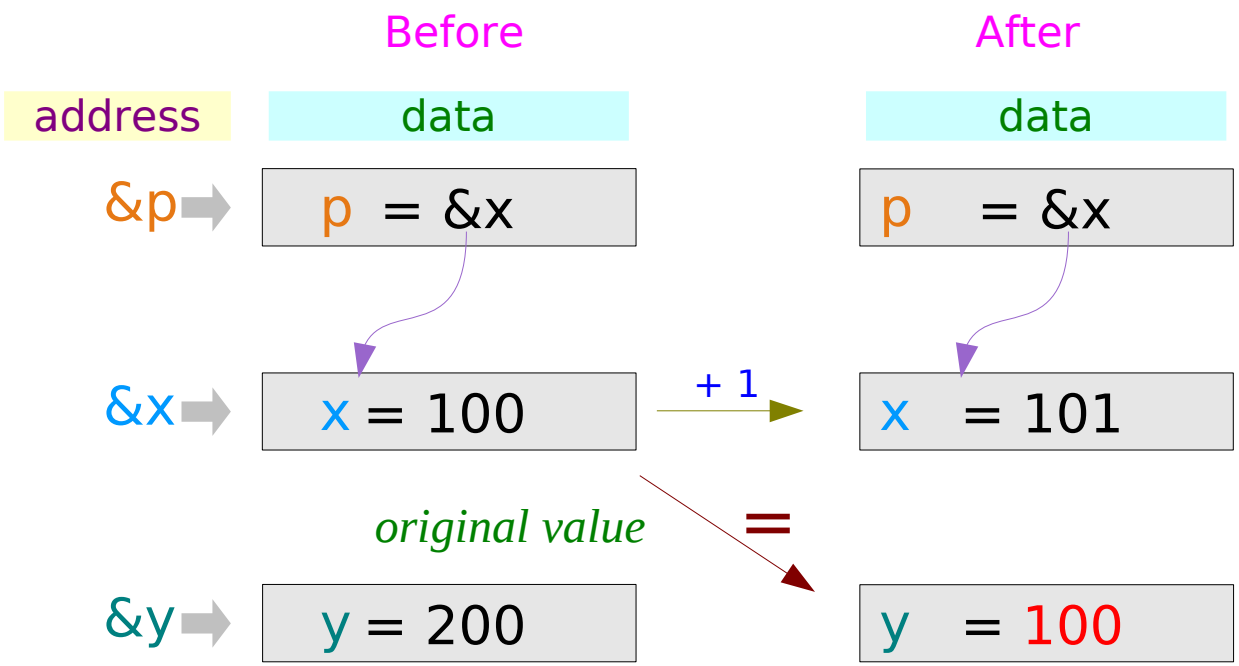


# Post-Increment Example (4)

```
int x = 100;
int y = 200;
int *   p;
```

```
p = &x;
y = (*p)++;
```

```
y = *p;
*p = *p + 1;
```



# Example Code

```
int main (void)
{
        int x[3] = {300, 100, 400};
        int y   = 200;
        int *p;

        // & has higher priority than []

        p = &( x[1] );          // p = &x[1];
                                // p = x + 1


        printf("&x[1]=%p &y=%p &p=%p\n", &x[1], &y, &p);
        printf("x=%d y=%d *p=%d p=%p\n", x[1],y,*p,p);

        printf("&x=%p &y=%p &p=%p\n", &x[1], &y, &p);
        printf("x=%d y=%d *p=%d p=%p\n", x[1],y,*p,p);


        return 0;
}
```

```
y = *++p;
y = *p++;
y = ++(*p);
y = (*p)++;
y = ++*p;
```

# Array

## References

[1] Essential C, Nick Parlante
[2] Efficient C Programming, Mark A. Weiss
[3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
[4] C Language Express, I. K. Chun