

Operators (1A)

Copyright (c) 2010-2013 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

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

This document was produced by using OpenOffice.

Pre- and Post- Increment / Decrement

x = a ++;

x = a --;

Access
First

x = a ++;

x = a --;

Update
Next

x = a ++;

x = a --;

x = ++ a ;

x = -- a ;

Update
First

x = ++ a ;

x = -- a ;

Access
Next

x = a ++;

x = a --;

Pre- and Post- Increment / Decrement

```
int a = 3;
```

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

```
a --;      a = a - 1;
```

```
cont int a = 3;
```

```
a ++;
```

```
a --;
```

```
double b = 3.1;
```

```
b ++;      b = b + 1;
```

```
b --;      b = b - 1;
```

```
const double b = 3.1;
```

```
b ++;
```

```
b --;
```

Pointers with ++ and -- (1)

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

`x = * (p --);` `x = *p--;`

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

`x = * (-- p);` `x = *--p;`

Access
First

`x = * (p ++);`

`x = * (p --);`

Update
First

`x = * (++ p);`

`x = * (-- p);`

Update
Next

`x = * (p ++);`

`x = * (p --);`

Access
Next

`x = * (++ p);`

`x = * (-- p);`

Pointers with ++ and -- (2)

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

```
x = (* p) --;
```

Access
First

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

```
x = (* p) --;
```

Update
Next

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

```
x = (* p) --;
```

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

```
x = -- (* p);    x = --*p;
```

Update
First

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

```
x = -- (* p);
```

Access
Next

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

```
x = -- (* p);
```

Pre and Post Increment / Decrement

$v = *p++;$

$v = *p$ (access first)
 $p = p+1$ (increment later) (**pointer** increment)

$v = (*p)++;$

$v = *p$ (access first)
 $*p = *p+1$ (increment later) (**value** increment)

$v = *++p;$

$p = p+1$ (increment first) (**pointer** increment)
 $v = *p$ (access later)

$v = ++*p;$

$*p = *p+1$ (increment first) (**value** increment)
 $v = *p$ (access later)

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