

## Week 5-2 : Name Spaces, Operator Overloading

### Part2. Operator Overloading

operators(+,-,\*,/,=...) were defined as function name  
It can be look like operate objects.

Operators that can be overloaded							
+	-	*	/	%	^	&	
~	!	=	<	>	+=	-=	*=
/=	%=	^=	&=	=	<<	>>	>>=
<<=	==	!=	<=	>=	&&		++
--	->*	,	->	[]	()	new	delete
new[]	delete[]						

Operators that cannot be overloaded			
.	.*	::	?:

Only member function operator : =, (), [], ->

## 1. Two forms of the operator overloading

### ① Member function

```
#include <iostream>
using namespace std;
class Point
{
private:
    int x,y;
public:
    Point(int argX,int argY);
    void ShowPosition();
    Point operator+(Point &p);
};

Point::Point(int argX,int argY)
{
    x=argX;
    y=argY;
}

void Point::ShowPosition()
{
    cout<<x<<" "<<y<<endl;
}

Point Point::operator+(Point &p)
{
    Point temp(x+p.x,y+p.y);
    return temp;
}

int main(void)
{
    Point p1(1,1);
    Point p2(2,2);
    Point p3=p1+p2;
    p3.ShowPosition();

    return 0;
}
```

p1                    +                    p2  
↙①                    ↓②                    ↘③  
p1                    .operator+                    (p2)

① object which calls member function

② function name

③ function's parameter

## ② Global function

```
#include <iostream>
using namespace std;
class Point
{
private:
    int x,y;
public :
    Point(int argX,int argY);
    int getX();
    int getY();
    void ShowPosition();
    friend Point operator+(Point &p);
};

Point::Point(int argX,int argY)
{
    x=argX;
    y=argY;
}

void Point::ShowPosition()
{
    cout<<x<<" "<<y<<endl;
}

int Point::getX()
```

```
{
    return x;
}
int Point::getY()
{
    return y;
}
Point operator+(Point &p1,Point &p2)
{
    Point temp(p1.x+p2.x,p1.y+p2.y);
    return temp;
}

int main(void)
{
    Point p1(1,1);
    Point p2(5,2);
    Point p3=p1+p2;
    p3.ShowPosition();

    return 0;
}
```

friend Point operator+(Point &p);  
using 'friend' to access 'p' declared by private

p1                    +                    p2  
operator+            (p1                    ,p2)

## 2. Single operator's overloading

prefix increment	postfix increment
++p	p++
<p><b>p.operator++()</b></p> <pre>Point&amp; operator++() { xpos+=1; ypos+=1; return *this }</pre>	<p><b>p.operator++(int)</b></p> <pre>const Point operator++(int) { const Point retobj(xpos, ypos); // const Point retobj(*this); xpos+=1; ypos+=1; return retobj; }</pre>
<p><b>operator++(Point &amp;ref)</b></p> <pre>Point&amp; operator--(Point &amp;ref) { ref.xpos-=1; ref.ypos-=1; return ref; }</pre>	<p><b>operator++(Point &amp;ref, int)</b></p> <pre>const Point operator++(Point &amp;ref, int) { const Point retobj(ref); ref.xpos+=1; ref.ypos+=1; return retobj; }</pre>

①

return \*this;

this : pointer which point object itself

\*this : refer pointed object = return itself

②

prefix and postfix increment are distinguished by int parameter

③

prefix increment return : reference value

postfix increment return : value, temporary object

### Case3. cout, cin Overloading

```
#include <iostream>
using std::endl;
using std::cout;

using std::ostream;

class Point{
private:
    int x,y;
public:
    Point(int _x=0, int _y=0):x(_x), y(_y) {}
    friend ostream& operator<<(ostream& os, const Point& p);
};

ostream& operator<<(ostream& os, const Point& p){
    os<<"("<<p.x<<","<<p.y<<")"<<endl;
    return os;
}

int main() {
    Point p(1,3);
    cout<<p;
    return 0;
}
```

cout : ostream's object/ cin : istream's object

#### [Exercise]

1.Satisfy the following conditions by modifying below code, using overloading by global function.

- Add z member function to the point class.
- Change the x, y, z member variable to the “private” and make the getX(), getY(), getZ() for the access to the variable.
- Declare operator - and operator\* additionally. Operations should use accessors because member variable is “private”.
- Declare namespace point and also declare Point p1(3,3,3), p2(2,2,2),p3; in the namespace. Print the p3 member variables by

declaring print() function.

- Write the code that prints the all operations(+,-,\*) result. (ex. p1+p2 =>p3)

```
#include <iostream>
using namespace std;
class Point
{
public:
    int x,y;
    Point(int argX,int argY);
    int getX();
    int getY();
    void ShowPosition();
    friend Point operator+(Point &p);
};

Point::Point(int argX,int argY)
{
    x=argX;
    y=argY;
}

void Point::ShowPosition()
{
    cout<<x<<" "<<y<<endl;
}

int Point::getX()
{
    return x;
}

int Point::getY()
{
    return y;
}


Point operator+(Point &p1,Point &p2)
{
    Point temp(p1.x+p2.x,p1.y+p2.y);
    return temp;
}

int main(void)
{
    Point p1(1,1);
    Point p2(5,2);
    Point p3=p1+p2;
    p3.ShowPosition();

    return 0;
}
```

2. Make a program of managing book. Print out formatted number automatically using operator overloading.

-format : [012] 222.3012

참고 

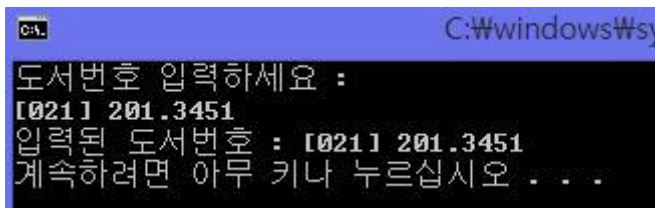
1. using setw() for setting field width.  
ex. input >> setw( 3 ) >> number.areaCode;
2. (,),space ignore  
ex. input.ignore( 2 ); // skip ) and space

### BookNumber.h

```
#include <iostream>
#include <string>
using namespace std;

class BookNumber
{
    friend ostream &operator<<( ostream &, const BookNumber & );
    friend istream &operator>>( istream &, BookNumber & );
private:
    string bookArea;
    string number1;
    string number2;
};
```

### 출력결과



```
C:\windows#sy
도서번호 입력하세요 :
[021] 201.3451
입력된 도서번호 : [021] 201.3451
계속하려면 아무 키나 누르십시오 . . .
```