C++ 스터디 4주차

과제

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

class Point
{
    int x, y, z;
public:
    Point(int a = 0, int b = 0, int c = 0);
    Point& Print();
    Point& Add(int a = 0, int b = 0, int c = 0);
    void Add(Point&);

    friend Point& Print(Point&);
};

Point::Point(int a, int b, int c) : x(a), y(b), z(c) {}

Point& Point::Print()
{
    cout << "멤버 함수 Print : (" << x << ", " << y << ", " << z << ")" << endl;

    return *this;
}

Point& Point::Add(int a, int b, int c)
{
    x += a;
    y += b;
    z += c;

    return *this;
}
void Point::Add(Point& p)
{
    x += p.x;
    y += p.y;
    z += p.z;
}

Point& Print(Point& p)
{
    cout << "전역 함수 Print : (" << p.x << ", " << p.y << ", " << p.z << ")" << endl;

    return p;
}

int main(void)
{
    Point p1;
    Point p2(1, 2, 3);

    Print(p1).Add(1, 1, 1).Print().Add(1, 1, 1);

    p1.Add(p2);

    p1.Print();

    return 0;
}
```

과제

```cpp
#include<iostream>
using namespace std;
class Circle;

class Point
{
    int x, y;
    static int count;
public:
    Point(int a = 0, int b = 0) : x(a), y(b) { count++; }
    ~Point() { count--; }

    static void Print()
    {
        cout << "Count : " << count << endl;
    }

    friend class Circle;
};
```

```cpp
class Circle
{
    Point middle;
    double radius;
public:
    Circle(int a = 0, int b = 0, double r = 0) : middle(a, b), radius(r) {}

    void Move(int a, int b)
    {
        middle.x += a;
        middle.y += b;
    }

    void SetRadius(double r)
    {
        radius = r;
    }

    void Print()
    {
        cout << "(" << middle.x << ", " << middle.y << ") - ";
        cout << "radius : " << radius << endl;
    }
};

int Point::count = 0;

```

복사 생성과 대입

복사 생성
복사 생성
객체 대입

```
Point p1(3, 4);
Point p2 = p1;
Point p3(p1);

Point p3;
p3 = p1;
```

복사 생성자 호출

대입 연산자 호출

따로 정의하지 않았다면 둘의 동작 방식은 동일(멤버 단위 복사)

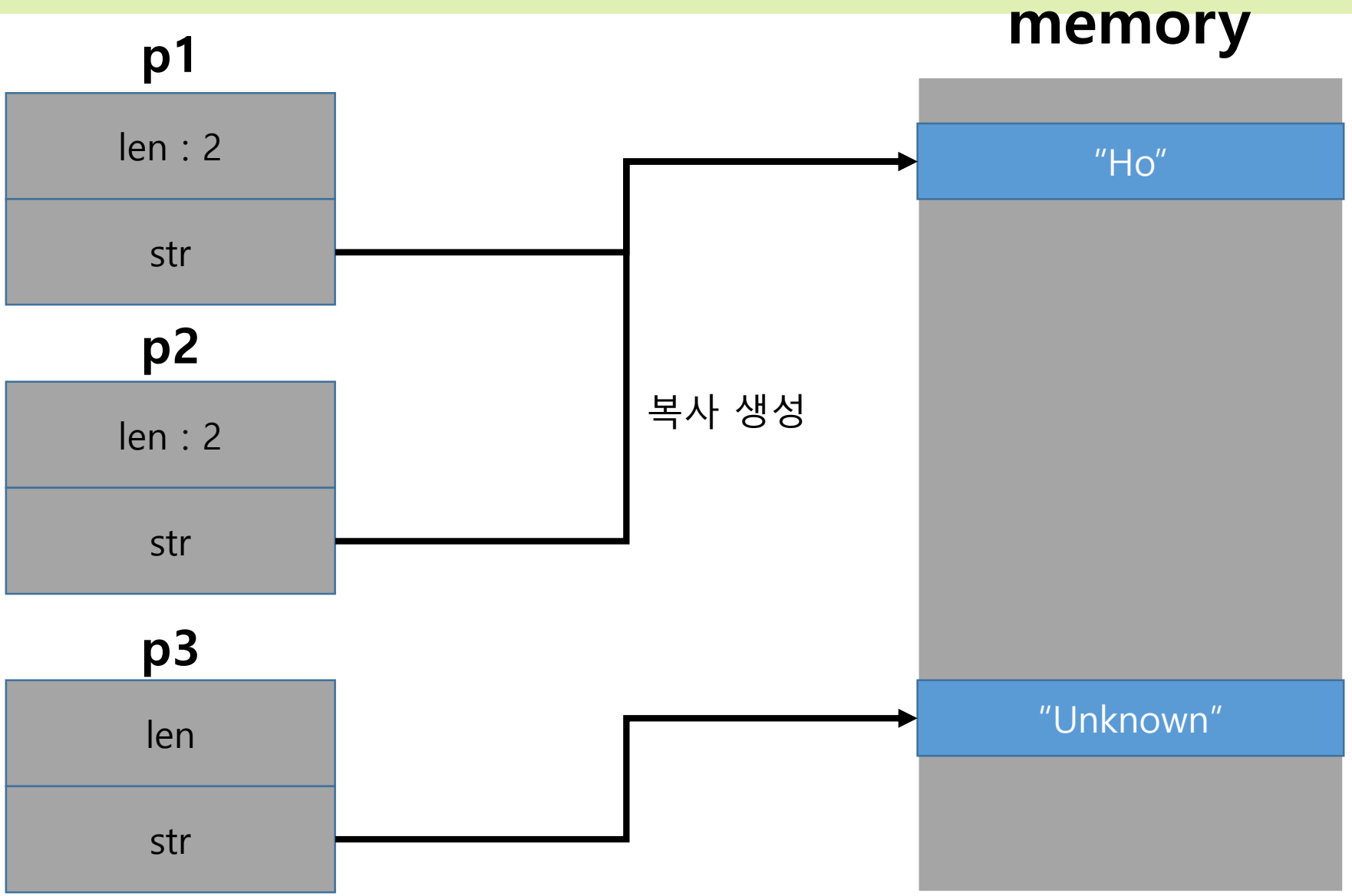
복사 생성과 대입

```
class Str
{
        int len;
        char* str;
public:
        Str(const char* s = “Unknown”)
        {
                len = strlen(s);
                str = new char[len + 1];
                strcpy(str, s);
        }
        ~Str()
        {
                delete[] str;
        }
};
```

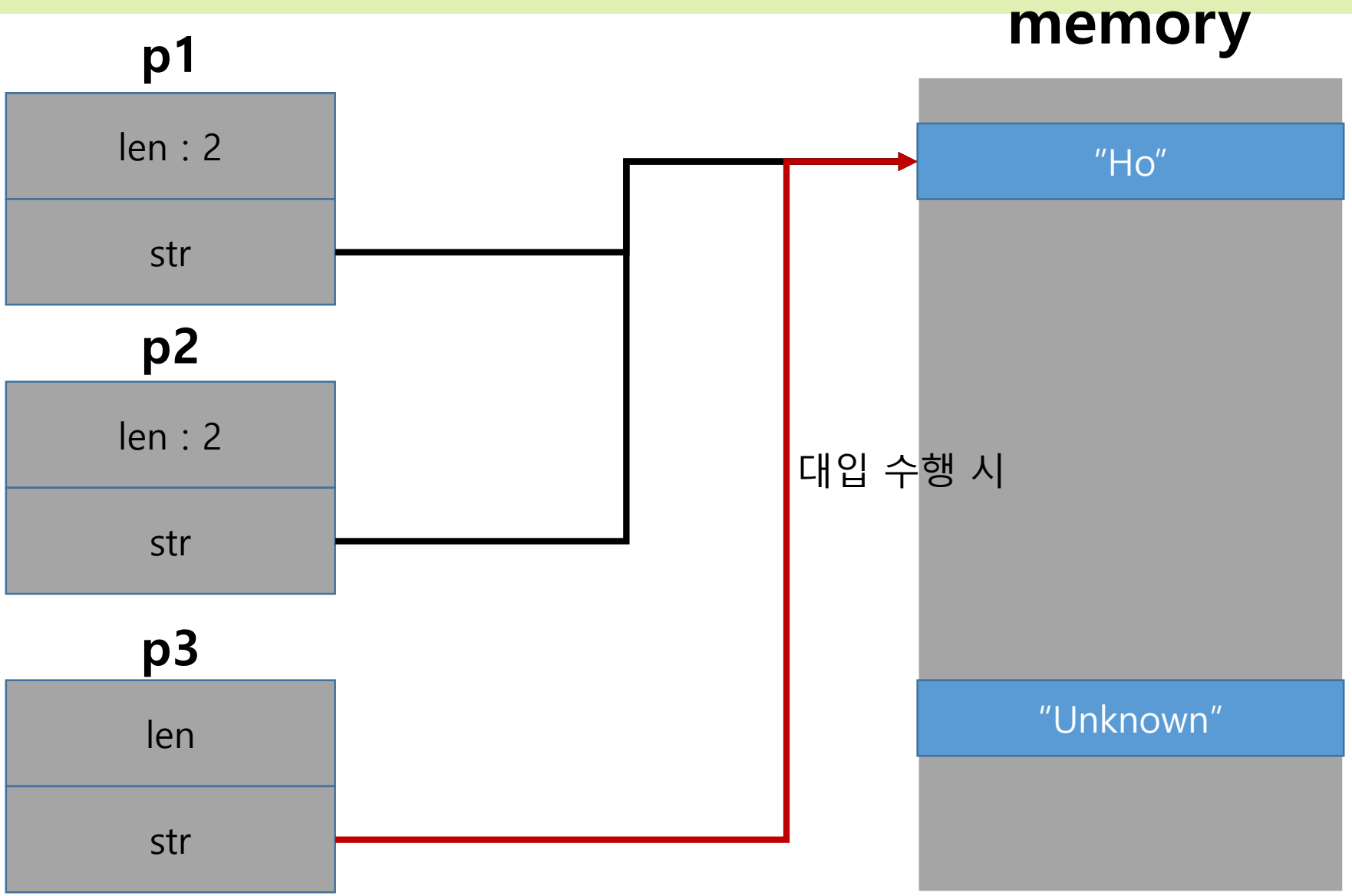
```
int main(void)
{
        Str s1(“Ho”);
        Str s2(s1);
        Str s3;

        s3 = s1;
}
```

문제 발생

복사 생성과 대입



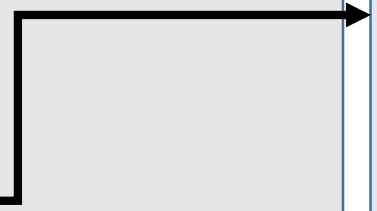
복사 생성과 대입



복사 생성

```
int main(void)
{
        Str s1("Ho");

        Print(s1);
}
```

```
void Print(Str s)
{
        cout << s.str << endl;
}
```



값의 의한 호출에서 객체를 매개변수로 넘겨줄 시에도
복사 생성자 동작

복사 생성자

- 생성자 중 특이한 생성자
- [클래스 이름](같은 클래스 참조형 매개변수)
  ex : Str(Str& s);
- 복사 생성, 값의 의한 호출 시 사용

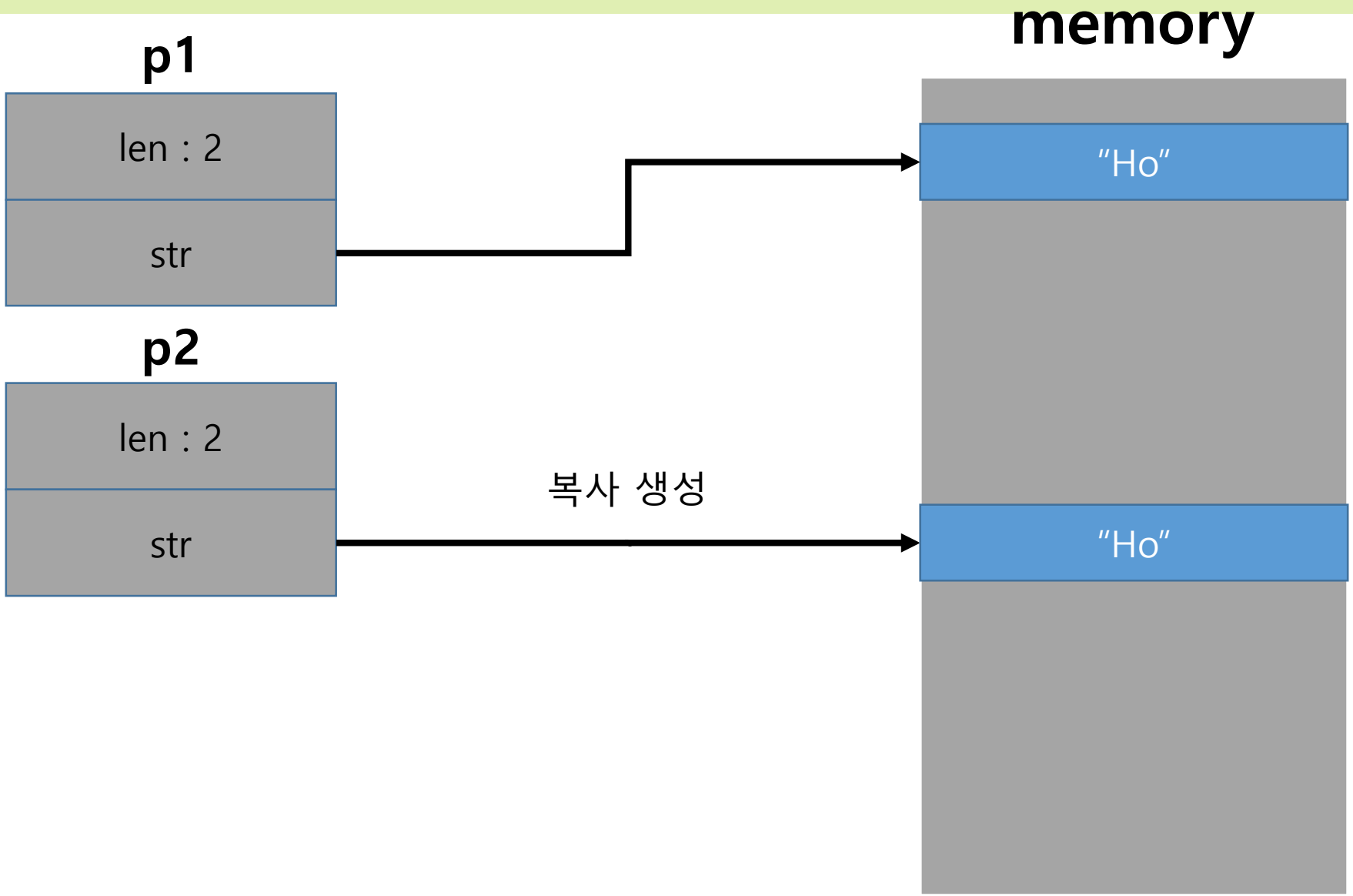
디폴트 복사 생성자

```
Str(Str& s)
{
      len = s.len;
      str = s.str;
}
```

디폴트 복사 생성자(멤버 단위 복사)

명시적으로 정의하지 않으면 디폴트 복사 생성자가 작동

복사 생성자



실습

- 지금까지 한 Str 클래스가 잘 작동하도록 복사 생성자를 만들어보자.

```
[redacted]localhost study]$ a.out
Ho
Ho
Unknown
```

```
int main(void)
{
    Str s1("Ho");
    Str s2(s1);
    Str s3;

    Print(s1);
    Print(s2);
    Print(s3);

    return 0;
}
```

실습

```
Str(const Str& s)
{
    len = s.len;
    str = new char[len + 1];
    strcpy(str, s.str);
}
```

Print 함수 friend 선언도 필요!

연산자 오버로딩

- 멤버 함수를 통한 오버로딩
- 전역 함수를 통한 오버로딩

연산자 오버로딩(멤버 함수)

강의록 6장
임시 객체 참조

더하기 연산자 오버로딩

```
class Point
{
        int x, y;
public:
        Point(int a = 0, int b = 0) : x(a), y(b) {}
        Point operator+(Point& p)
        {
                return Point(x + p.x, y + p.y);
        }
};

int main()
{
        Point p1(1, 2);
        Point p2(3, 4);
        Point p3 = p1 + p2;
}
```

연산자 오버로딩(멤버 함수)

- 일반 함수와 같이 반환형, 매개변수 존재

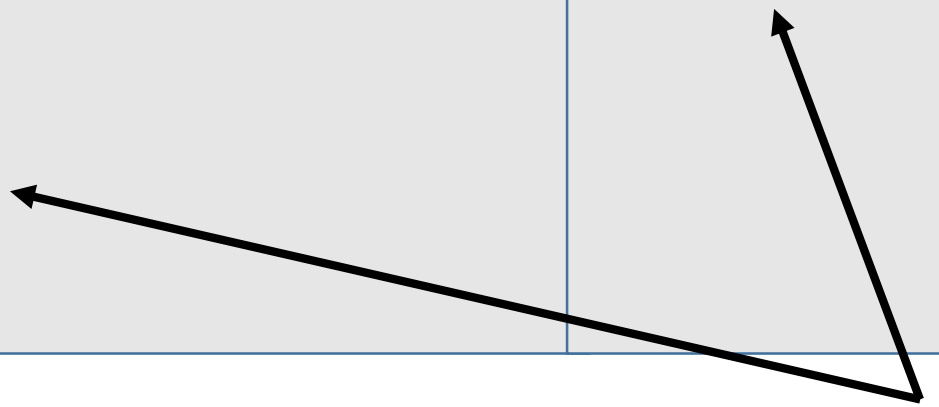
- [반환형] [오버로딩할 연산자](매개변수)
- ex : int operator*(int a);
  -> Point p1; int a = p1 * 3;

- 명시적 호출 가능 -> int a = p1.operator*(3);

연산자 오버로딩(전역 함수)

```
class Point
{
        int x, y;
public:
        Point(int a = 0, int b = 0) : x(a), y(b) {}

        friend Point operator+(Point&, Point&);
};

int main()
{
        Point p1(1, 2);
        Point p2(3, 4);
        Point p3 = p1 + p2;
}
```

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



더하기 연산자 오버로딩

연산자 오버로딩(전역 함수)

- 멤버 함수와 전역 함수 둘 다 존재 시, 멤버 함수 우선 호출

- 명시적 호출 가능 -> `Point p3 = operator+(p1, p2);`