



C++ 스터디 4주차

과제

```
1 #include<iostream>
2 using namespace std;
3
4 class Point
5 {
6     int x, y, z;
7 public:
8     Point(int a = 0, int b = 0, int c = 0);
9     Point& Print();
10    Point& Add(int a = 0, int b = 0, int c = 0);
11    void Add(Point&);
12
13    friend Point& Print(Point&);
14 };
15
16 Point::Point(int a, int b, int c) : x(a), y(b), z(c) {}
17
18 Point& Point::Print()
19 {
20     cout << "멤버 함수 Print : (" << x << ", " << y << ", " << z << ")" << endl;
21     return *this;
22 }
23
24
25 Point& Point::Add(int a, int b, int c)
26 {
27     x += a;
28     y += b;
29     z += c;
30
31     return *this;
32 }
```

```
34 void Point::Add(Point& p)
35 {
36     x += p.x;
37     y += p.y;
38     z += p.z;
39 }
40
41 Point& Print(Point& p)
42 {
43     cout << "전역 함수 Print : (" << p.x << ", " << p.y << ", " << p.z << ")" << endl;
44
45     return p;
46 }
47
48 int main(void)
49 {
50     Point p1;
51     Point p2(1, 2, 3);
52
53     Print(p1).Add(1, 1, 1).Print().Add(1, 1, 1);
54
55     p1.Add(p2);
56
57     p1.Print();
58
59     return 0;
60 }
```

과제

```
1 #include<iostream>
2 using namespace std;
3 class Circle;
4
5 class Point
6 {
7     int x, y;
8     static int count;
9 public:
10     Point(int a = 0, int b = 0) : x(a), y(b) { count++; }
11     ~Point() { count--; }
12
13     static void Print()
14     {
15         cout << "Count : " << count << endl;
16     }
17
18     friend class Circle;
19 };
```

```
21 class Circle
22 {
23     Point middle;
24     double radius;
25 public:
26     Circle(int a = 0, int b = 0, double r = 0) : middle(a, b), radius(r) {}
27
28     void Move(int a, int b)
29     {
30         middle.x += a;
31         middle.y += b;
32     }
33
34     void SetRadius(double r)
35     {
36         radius = r;
37     }
38
39     void Print()
40     {
41         cout << "(" << middle.x << ", " << middle.y << ") - ";
42         cout << "radius : " << radius << endl;
43     }
44 };
45
46 int Point::count = 0;
47
```

복사 생성과 대입

복사 생성
복사 생성

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

}

복사 생성자 호출

객체 대입

```
Point p3;  
p3 = p1;
```

대입 연산자 호출

따로 정의하지 않았다면 둘의 동작 방식은 동일(멤버 단위 복사)

복사 생성과 대입

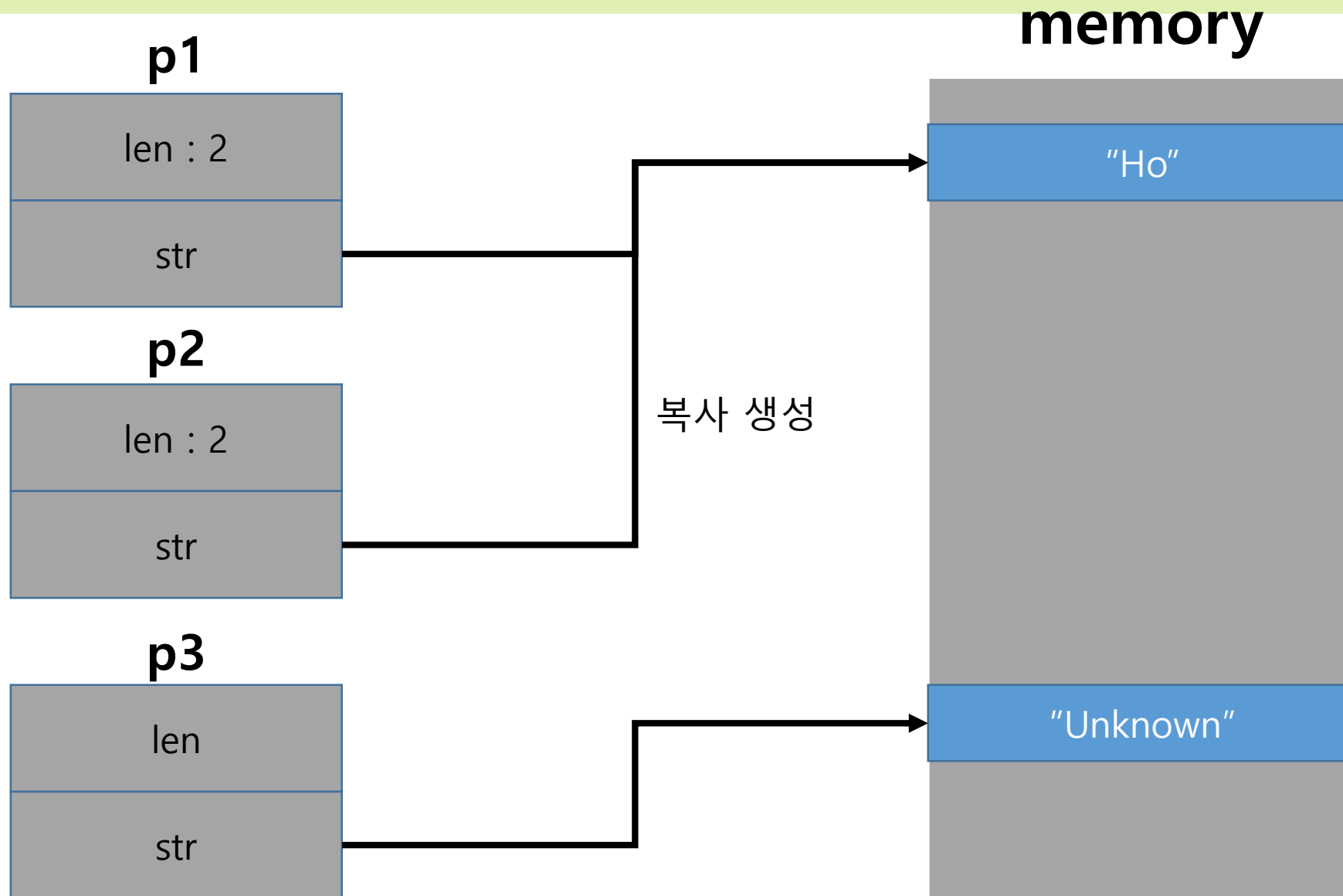
```
class Str
{
    int len;
    char* str;
public:
    Str(const char* s = "Unknown")
    {
        len = strlen(s);
        str = new char[len + 1];
        strcpy(str, s);
    }
    ~Str()
    {
        delete[] str;
    }
};
```

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

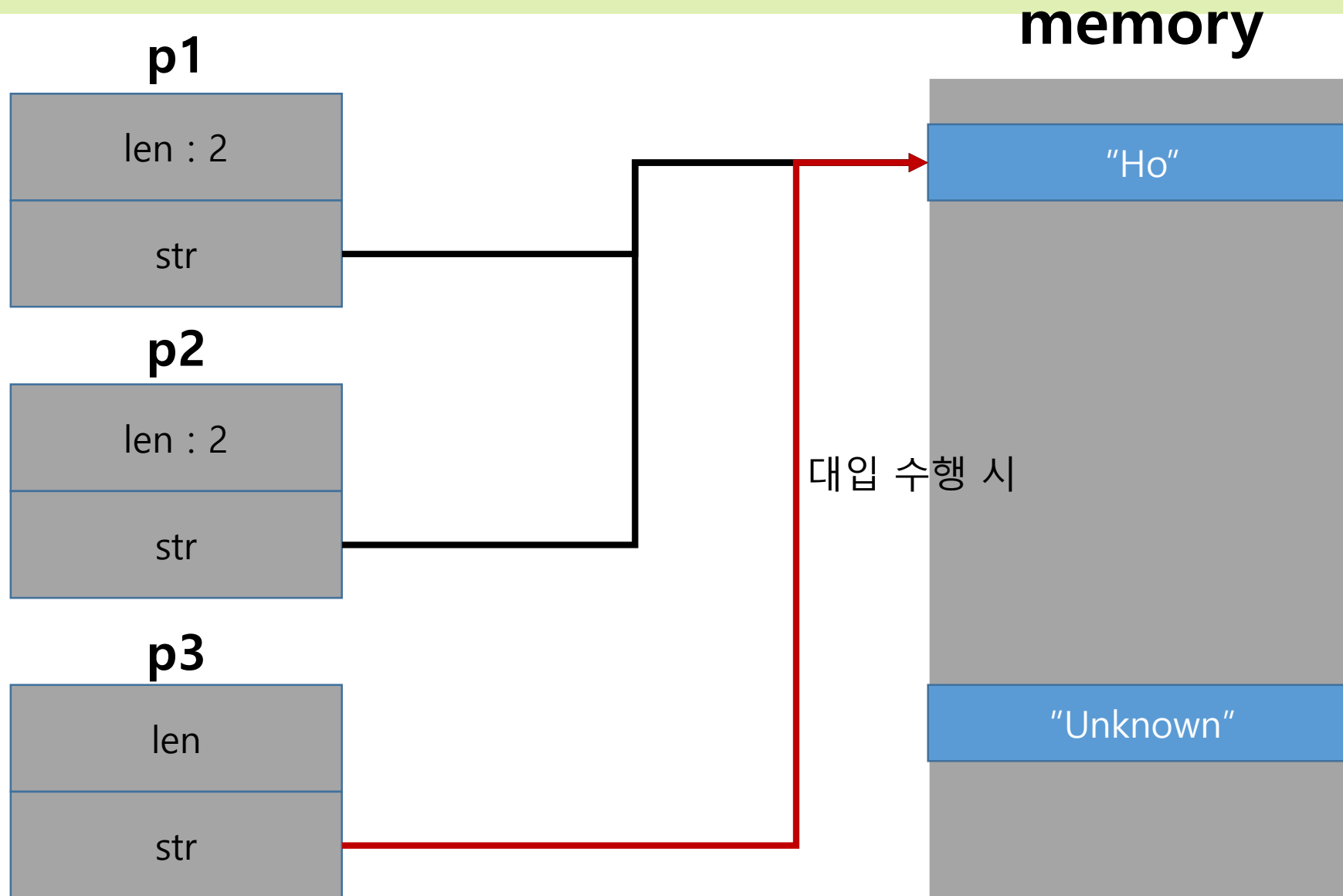
    s3 = s1;
}
```

문제 발생

복사 생성과 대입



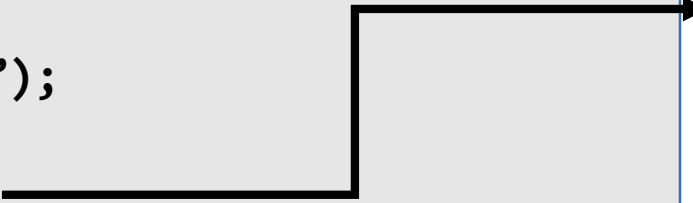
복사 생성과 대입



복사 생성

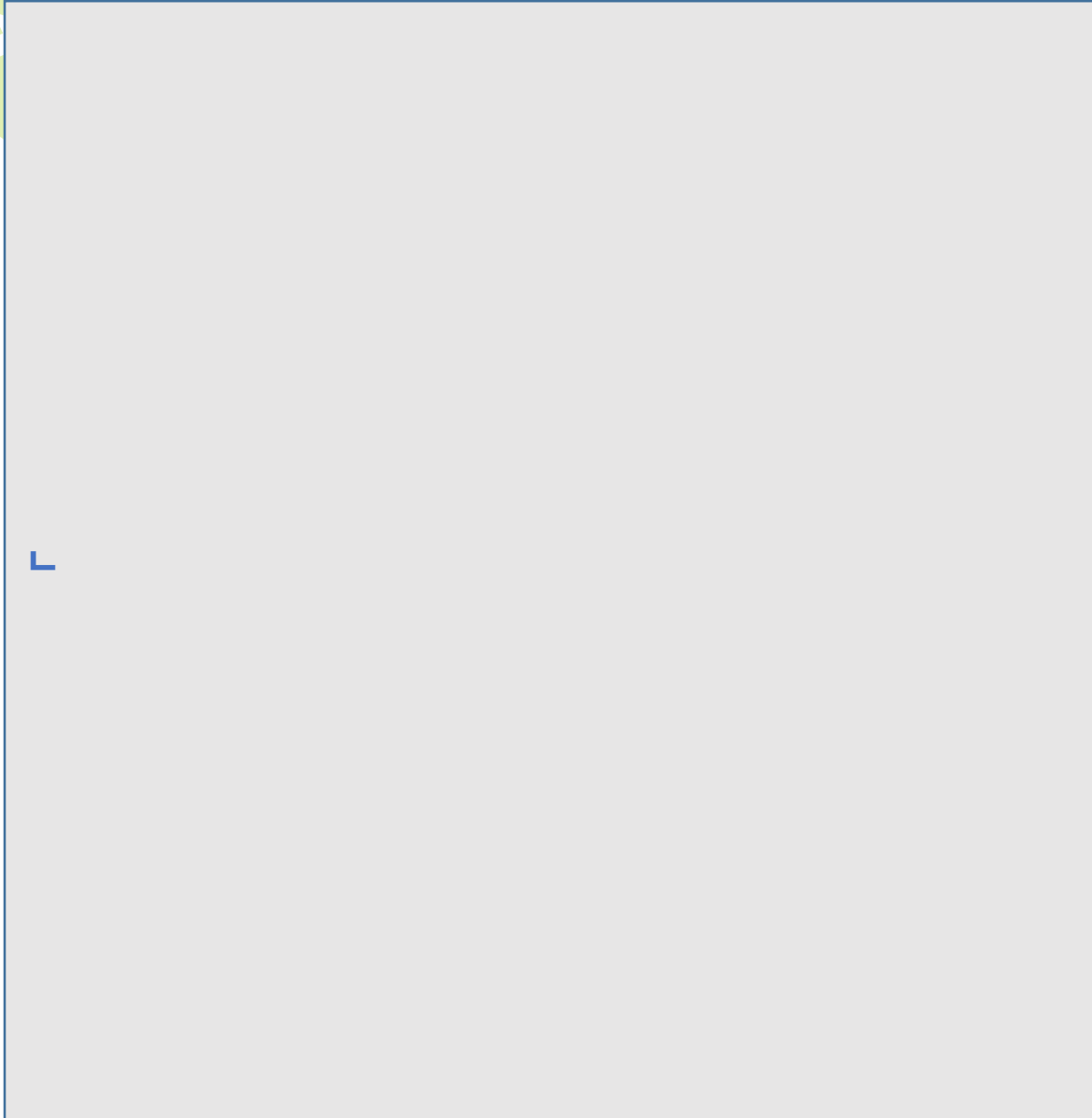
```
int main(void)
{
    Str s1("Ho");
    Print(s1);
}
```

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



값의 의한 호출에서 객체를 매개변수로 넘겨줄 시에도
복사 생성자 동작

복사 생성자



복사 생성자!

디폴트 복사 생성자

```
Str(Str& s)
```

```
{
```

```
    len = s.len;
```

```
    str = s.str;
```

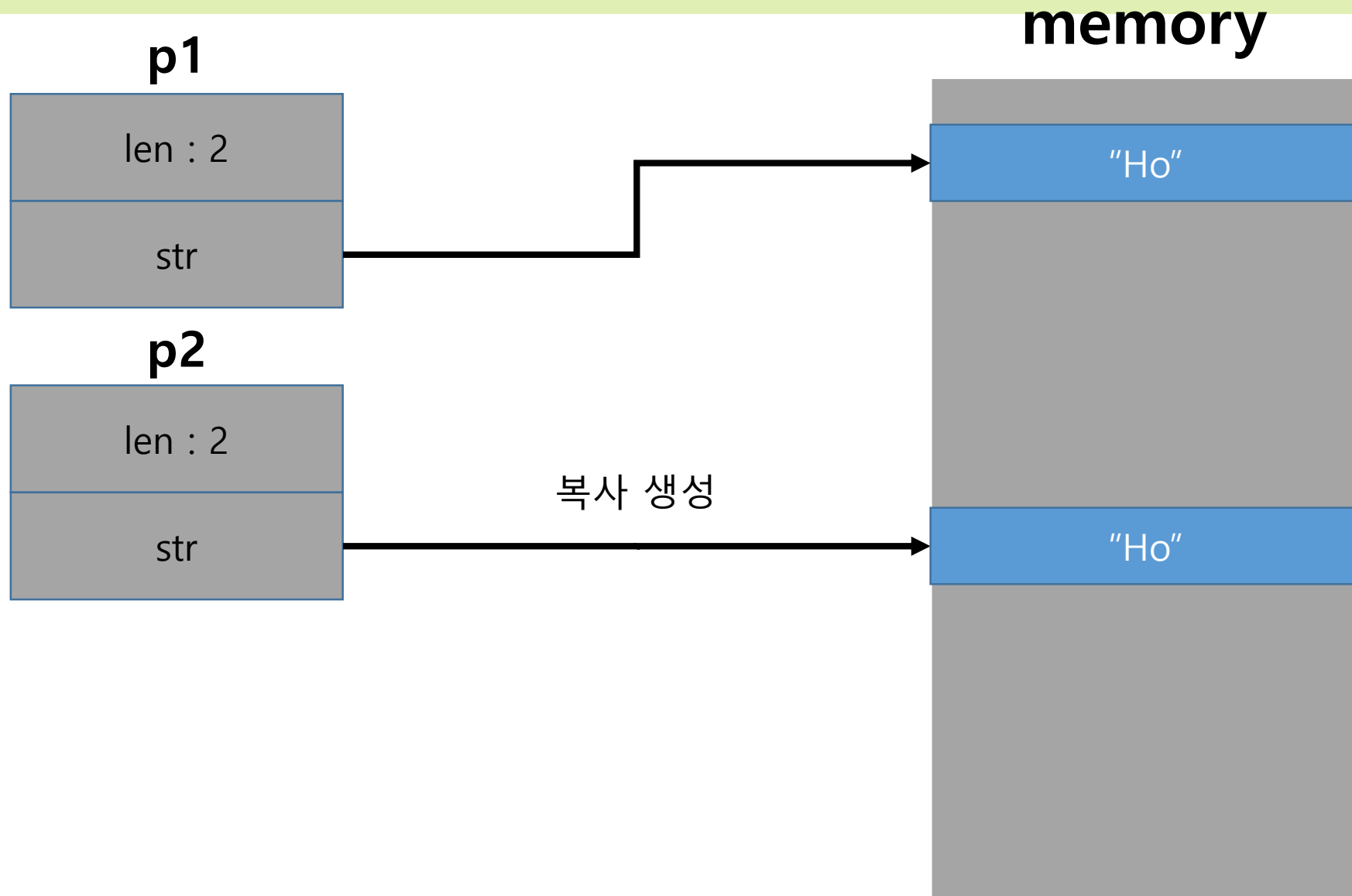
```
}
```



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.

```
[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)

```
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;
}
```

강의록 6장
임시 객체 참조

더하기 연산자 오버로딩

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);`