4. 가상함수(Virtual function)

C++에서 Overriding된 멤버함수를 부모클래스 형(type)의 포인터를 이용하여 접근할 수 있도록 하려면 **가상함수**로 정의하여야 한다.(부모클래스의 포인터가 자식클래스의 객체를 가리킴)

```
class A
{
  public : virtual int voo(int x, int y){ return x + y; }     //가상함수(virtual)로 정의
};
class B : public A
{
  public : virtual int voo(int x, int y){ return x * y; }     //재정의(overriding)
};
void main()
{
  A *p = new B();              //부모클래스의 객체 포인터 p가 자식클래스 객체를 가리킴
  cout<<p->voo(4, 5)<<"\n";  //출력 : 20 → 자식클래스에 정의된 메서드가 호출됨
}
```

- 만약, 클래스 A에 정의된 voo()에 **virtual**이 없으면 **9**가 출력된다. 가상함수가 아니므로
- 부모클래스의 메서드에 **virtual**을 기술하면, 하위클래스에는 **virtual**을 생략해도 **가상함수**가 된다.

// 부모클래스의 객체 포인터 p가 자식클래스 객체를 가리킬 때

- A *p = new B();
- 자식클래스에 재정의된 메서드가 있으면, 자식클래스에 재정의된 메서드가 호출된다.
- 이 원리는 Java와 같다. Java에서 많이 설명하였다.
- 단지, Java에서는 **virtual**을 기술할 필요가 없고, C++에서는 **virtual**을 기술해야 한다.
- 이는 Java의 메서드는 기본적으로 가상함수로 작동한다는 것이다.

// 함수의 바인딩

정적바인딩 (static binding)	• **컴파일시간**(프로그램 수행 전)에 함수의 호출주소를 결정한다.**(선결합)** • C++의 일반함수는 정적바인딩으로 처리된다.
동적바인딩 (dynamic binding)	• **프로그램 수행 도중**에 어느 함수를 호출할 것인지를 결정한다.**(후결합)** • 컴파일시간에는 **빈칸**으로 두었다가 실행시간에 점프할 주소를 결정한다. • 실행코드에 점프할 주소가 없기 때문에 **'가상'**이라는 말을 사용 • C++의 가상함수는 동적바인딩을 한다.(키워드 **virtual**로 정의) • Java의 일반메서드는 가상함수이며 동적바인딩을 한다.(**virtual** 불필요)

기출문제 분석

1. 다음 C++ 프로그램의 실행 결과는? [2010년 국가 9급 유형]

```
#include <iostream>
using namespace std;
class A{
  public : int n;
           A() { n = 0; }
           virtual int get(){ return ++n; }
           void print(int k){ cout<<"A : print "<<k<<endl; }
};
class B : public A{
  public : int get(){ return n++; }
           void print(int k){ cout<<"B : print "<<k<<endl; }
};
void main() {
  A *p = new B();
  p->print(p->get());
}
```

① A : print 0　　② A : print 1　　③ B : print 0　　④ B : print 1

♤ 일반함수 / 가상함수

```
virtual int get(){ return ++n; }                  //가상함수(동적바인딩)
void print(int k){ cout<<"A : print "<<k<<endl; }  //일반함수
```

↓상속

```
int get(){ return n++; }                          //가상함수(재정의), n=0을 반환하고 1 증가
void print(int k){ cout<<"B : print "<<k<<endl; }  //일반함수
A *p = new B();          //부모타입의 포인터 p가 자식객체를 참조
p->print(p->get());      //자식클래스에서 재정의된 멤버함수 get()이 호출된다.
```

↳ 멤버함수 print()는 가상함수가 아니므로 부모클래스 A에 정의된 것이 호출된다.

• 해서, '**A : print 0**'이 출력된다.

정답 : ①