항목 3: 복사는 컨테이너 안의 객체에 맞게 비용은 최소화하며, 동작은 정확하게 하자

컨테이너의 본분은 객체를 담아 관리하는 것입니다. 컨테이너에서 객체를 가져오려면 우선 객체를 넣어야 합니다. 그렇다면, 컨테이너에 객체를 넣을 때 객체 원본이 들어갈까요? 아니면 복사해서 들어갈까요?

**정답은. “복사해서 들어간다.” 입니다.**

따라서 컨테이너에 넣은 객체를 다시 꺼낼 때는 자신이 넣어둔 객체가 아닌 것 입니다. 이럴 경우 여러 가지 문제가 발생 할 수 있습니다.

* **속도 문제**

복사에 드는 비용이 큰 객체를 컨테이너에 담을 때 속도가 크게 저하 됩니다.

* **파생 클래스 slice 문제**

기본 클래스를 담는 컨테이너에 파생 클래스 객체를 넣을 때 기본 클래스의 복사생성자가 호출되어 들어가게 되는데 이때 파생 클래스 부분은 잘려나갑니다.

이러한 문제들을 해결 하는 아주 간단한 방법이 있습니다.

**객체 컨테이너가 아닌, 포인터(스마트 포인터)의 컨테이너 입니다. 즉, 객체를 담는 것이 아니라 포인터를 담는 것입니다.** 포인터의 복사는 속도가 빠르며, 정확히 동작하며, 데이터가 잘려나가지 않고 원본을 그대로 가져올 수 있습니다.

자칫 STL은 복사를 많이 하는 라이브러리 라고 느낄 수 있습니다. 하지만 그렇지 않습니다. 불필요한 복사는 하지 않습니다. 예를 들어보겠습니다.

Widget w[amount]; // amount개 만큼 Widget 객체를 만듭니다. 이 말은 amount번 생성자를 호출 한다는 것 입니다.

vector<Widget> vw; // Widget객체를 담을 vector를 생성합니다. 하지만 Widget객체는 없습니다. Widget객체 생성자 또한 호출하지 않았습니다. 사용자가 필요할 때 객체를 생성하여 담아서 사용하므로 꼭 필요한 만큼 생성자를 호출 합니다.