항목 26: 변수 정의는 늦출 수 있는 데까지 늦추는 근성을 발휘하자

객체를 생성하면 반드시 물게 되는 비용이 두 개 있습니다. 바로 생성자와 소멸자 입니다. **문제는 객체를 생성해 놓고 사용하지 않은 경우 입니다.** 어떤 경우에 발생 할 수 있는지 예제를 보여드리겠습니다.

*string 함수(string& password)*

*{*

*string객체 A생성*

*if( ) // 예외 체크*

*{*

*return;*

*}*

*객체 A 사용*

*}*

위 에서 보시는 봐와 같이 만약 if문에서 예외가 감지 되어 함수가 종료 되면 만들어 둔 객체 A는 쓸모 없이 생성되었다가 소멸되는 비용이 발생 합니다. **예외 체크가 된 후에 객체 A를 생성하는 것이 좋습니다.**

다음은 생성자/대입연산자 모두 부르지 말고 복사생성자만 호출 하는 것 입니다.

*string 함수(string& s)*

*{*

*string A객체 // 생성자 호출*

*A = s; // 대입연산자 복사*

*}*

위 함수의 문제는 비용을 들여서 생성자를 호출 했지만 s값으로 복사하기 때문에 생성자를 통해 초기화 된 값이 모두 없어졌습니다. **s값만 사용한다면 비용을 들여서까지 생성자를 호출할 필요가 없습니다. 정의하고 복사생성자를 통해 값을 복사하면 그만 입니다.**

*string 함수(string& s)*

*{*

*string A(s) // 복사생성자 호출*

*}*

루프 안에 사용되는 변수의 정의는 어떡할까요?

***A 방법 : 루프 바깥쪽에 정의***

*string s;*

*for(int i=0; i<n; ++i)*

*{*

*w = i에 따라 달라지는 값;*

*}*

***B 방법 : 루프 안쪽에 정의***

*for(int i=0; i<n; ++i)*

*{*

*string s(i에 따라 달라지는 값);*

*}*

A 방법 생성자 1번 + 소멸자 1번 + 대입 n번

B 방법 복사생성자 n번 + 소멸자 n번

통상 비용은 ( 대입 < (복사생성자 + 소멸자) ) 이렇습니다. 따라서 **효율 면에서 A방법이** 좋습니다. 그러나 **B방법에도 장점이** 있습니다. 변수의 유효범위가 for문 안으로 한정 된다는 점 입니다. 따라서 유지보수, 코드 이해도가 더 높아 질 수 있습니다.