**// endl 함수**

// MANIPULATORS

template<class \_Elem,

class \_Traits> inline

basic\_ostream<\_Elem, \_Traits>&

\_\_CLRCALL\_OR\_CDECL **endl**(basic\_ostream<\_Elem, \_Traits>& \_Ostr)

{ // insert newline and flush stream

\_Ostr.put(\_Ostr.widen('\n'));

\_Ostr.flush();

return (\_Ostr);

}

**//ADL(Argument-dependent lookup)**

template<class \_Traits> inline

basic\_ostream<char, \_Traits>& **operator<<**(

basic\_ostream<char, \_Traits>& \_Ostr,

const char \*\_Val)

{ // insert NTBS into char stream

typedef char \_Elem;

typedef basic\_ostream<\_Elem, \_Traits> \_Myos;

ios\_base::iostate \_State = ios\_base::goodbit;

streamsize \_Count = (streamsize)\_Traits::length(\_Val); // may overflow

streamsize \_Pad = \_Ostr.width() <= 0 || \_Ostr.width() <= \_Count

? 0 : \_Ostr.width() - \_Count;

const typename \_Myos::sentry \_Ok(\_Ostr);

**//오퍼레이터 << 연산자(함수 포인터)**

\_Myt& \_\_CLR\_OR\_THIS\_CALL **operator<<(\_Myt& (\_\_cdecl \*\_Pfn)(\_Myt&))**

{ // call basic\_ostream manipulator

\_DEBUG\_POINTER(\_Pfn);

return ((\*\_Pfn)(\*this));

}

**typedef \_Myt& (\*\_Pfn)(\_Myt&) endl;**

**//typedef 함수 포인터(타입) 함수(변수)의 명은 주소가 됨**

1) ADL을 이용하여 전역 변수 operator << 를 호출함

2) endl(함수 포인터에 인자로 넘김/주소)의 함수를 operator << 연산자(함수)에 매개변수로 전달함