**항목 42. Typename의 두 가지 의미를 제대로 파악하자.**

아래의 선언문에서 class와 typename의 차이가 뭘까?

|  |
| --- |
| template<class T> class Widget;  template<typename T> class Widget; |

결론적으로, 없다. 같은 의미로 사용된다. 하지만 typename을 쓰지 않으면 안 되는 때가 분명히 있다.

템플릿 안에서 참조할 수 있는 이름에는 두 가지 종류가 있다. 하나는 템플릿 매개변수에 종속되는 **중첩 의존 타입 이름**(nested dependent type name)이고 다른 하나는 int와 같은 템플릿 매개변수와는 상관 없는 **비 의존 이름**(non-dependent name)이다.

|  |
| --- |
| template<typename C>  void print2nd(const C& container)  {  if(container.size() >= 2) {  C::const\_iterator iter(container.begin());  ++iter;  int value = \*iter;  std::cout <<value;  }  } |

여기서 iter타입은 C::const\_iterator인데, C에 따라 달라지는 타입 이것을 **의존 이름**이라 한다. 의존 이름이 어떤 클래스 안에 중첩되어 있는 경우가 있는데, 이 경우를 **중첩 의존 이름**이라고 한다. 그리고 타입을 참조하는 중첩 의존 이름을 **중첩 의존 타입** 이름이라고 한다.

여기서 value는 int타입인데 템플릿 매개변수가 어떻든 상관 없는 타입 이름으로 **비 의존 이름** 이라고 한다.

|  |
| --- |
| template<typename C>  void func(const C& container)  {  C::const\_iterator \* x;  } |

위 코드에서 C::const\_iterator가 타입이 아니라 C의 멤버 데이터 라면 어쩔 것인가? 그리고 x가 전역 변수라면 저건 두 변수의 곱인가? 말도 안되는 상상이라고 생각할 수 있지만 충분히 가능한 일이다. C++는 이런 모호성을 해결하기 위해 규칙을 하나 정했다. **템플릿 안에 중첩 의존 이름을 만나면 프로그래머가 타입이라고 알려주지 않는 이상 무조건 타입이 아닌 것으로 취급 한다**. 그러니 이 난국을 바로 잡을 방법은 하나 밖에 없다. 앞에 다가 typename 키워드를 써주는 것 이다.

|  |
| --- |
| template<typename C>  void func(const C& container)  {  typename C::const\_iterator \* x;  } |

중첩 의존 타입 이름 앞에 typename을 붙이면 안되는 **예외**가 하나 있다. **기본 클래스의 리스트에 있거나 멤버 초기화 리스트 내의 기본 클래스 식별자로 있는 경우**에는 typename을 붙여 주면 안 된다. 아래 코드를 보자

|  |
| --- |
| template<typename T>  class Derived : public Base<T>::Nested { // 상속되는 기본 클래스 리스트  public:  explicit Derived(int x)  : Base<T>::Nested(x) // 멤버 초기화 리스트에 있는 기본 클래스 식별자  {  typename Base<T>::Nested temp; // typename 꼭 붙여야 함  }  }; |

현업에서 자주 사용하는 코드 중 **반복자를 매개변수로 받는 어떤 함수 템플릿이 있는데, 매개변수로 넘어온 반복자가 가리키는 객체의 사본을 만드는 경우**가 있다. 객체로부터 그 Type을 추출해 내는 과정으로 C#, Java의 Reflect를 사용하는 것과 비슷한 경우이다. 이럴 경우 typename이 여러 번 쓰인다면 무지막지하게 길게 되어서 typedef를 사용하는 것이 효과적인 경우가 많다. 보통 그럴 경우 그 멤버 이름과 똑 같은 이름으로 짓는 것이 관례로 되어 있다.

|  |
| --- |
| template<typename T>  void workWithIterator<T iter> {  typedef typename std::iterator\_traits<T>::value\_type value\_type;  value\_type temp(\*iter);  } |

**요약**

- 템플릿 매개변수를 선언할 때, class 및 typename은 서로 바꾸어 써도 무방합니다.

- 중첩 의존 타입 이름을 식별하는 용도에는 반드시 typename을 사용합니다. 단, 중첩 의존 이름이 기본 클래스 리스트에 있거나 멤버 초기화 리스트 내의 기본 클래스 식별자로 있는 경우에는 예외입니다.