**항목 46. 타입 변환이 바람직할 경우에는 비 멤버 함수를 클래스 템플릿 안에 정의해 두자**

모든 매개 변수에 대해 암시적 타입 변환이 되도록 만들기 위해서는 비 멤버 함수 밖에 방법이 없다. 이것을 템플릿에 적용되게 하면 에러가 발생한다.

|  |
| --- |
| Template<typename T>  Class Rational  {  Public:  Rational (const T& numerator = 0, const T& denominator = 1);  Const T numerator() const;  Const T denominator() const;  }  Template<typename T>  Const Rational<T> operator\* (const Rational<T>& left, const Rational<T>& right)  { . . . }  Rational<int> oneHalf(1, 2);  Rational<int> result = oneHalf \* 2; |

위의 코드는 에러가 발생한다. operator\*를 호출하기 위해서 대 관절 T가 무엇인지 알아야 한다. oneHalf 쪽은 공략이 쉽다.

Rational<int> 타입이라서 T가 int라는 것을 바로 알 수 있다. 문제는 2에 있다. 2는 int형이다. Rational<int>에는 explicit로 선언되지 않은 생성자가 있으니깐 Rational<int>로 변환이 될 것이라고 예상 되겠지만, 컴파일러는 그렇게 동작하지 못한다. 왜냐면 **템플릿 인자 추론(template argument deduction) 과정에서는 암시적 타입 변환이 고려 되지 않기 때문이다.** 절대로 안된다. 이처럼 힘든 처지에서 템플릿 인자 추론을 해야 하는 수고로부터 컴파일러를 해방시킬 수 잇는 방법이 있다. 클래스 템플릿 안에 프렌드 함수를 넣어 두면 함수 템플릿으로서의 성격을 주지 않고 특정 함수 하나를 나타낼 수 있다는 사실을 이용하면 된다.

|  |
| --- |
| Template<typename T>  Class Rational  {  Public:  Rational (const T& numerator = 0, const T& denominator = 1);  Const T numerator() const;  Const T denominator() const;  **Friend const Rational operator\* (const Rational& left, const Rational& right); // 새로 선언**  }  Template<typename T>  **Const Rational<T> operator\* (const Rational<T>& left, const Rational<T>& right)**  **{ . . . } // 위에 선언된 friend 함수에 대한 정의** |

하지만 이 코드는 컴파일은 되지만, 링크가 안 된다. 간단하게 해결하려면 함수의 본문 (정의) 부분을 선언부와 붙이면 된다.

|  |
| --- |
| Template<typename T>  Class Rational  {  Public:  Rational (const T& numerator = 0, const T& denominator = 1);  Const T numerator() const;  Const T denominator() const;  **Friend const Rational operator\* (const Rational& left, const Rational& right)**  **{**  **Return Rational(left.numerator() \* right.nemerator(), left.denominator() \* right.denominator());**  **}**  } |

이 경우는 함수가 무척 짧지만 만약 긴 경우는 어떻게 해야 할까? 이 경우는 **“프렌드 함수는 도우미만 호출하게 만들기”** 방법을 이용하면 된다.

|  |
| --- |
| Template<typename T>  Class Rational  {  Public:  Rational (const T& numerator = 0, const T& denominator = 1);  Const T numerator() const;  Const T denominator() const;  **Friend const Rational operator\* (const Rational& left, const Rational& right)**  **{**  **Return Multiply(left, right);**  **}**  };  **Template<typename T>**  **Const Rational<T> Multiply(const Rational<T>& left, const Rational<T>& right)**  **{**  **Return Rational(left.numerator() \* right.nemerator(), left.denominator() \* right.denominator());**  **}** |

**요약.**

- 모든 매개 변수에 대해서 암시적 타입 변환을 지원하는 템플릿과 관계가 있는 함수를 제공하는 클래스 템플릿을 만들려고 한다면, 이런 함수는 클래스 템플릿 안에 프렌드 함수로서 정의합니다.