Lecture 22
Template III

Member Templates

Prof. Hyeong-Seok Ko
Seoul National University
Graphics & Media Lab

Contents

- Member Templates (16.4.5)
- Template + Operator Overloading

Error can occur in the following...

```
template<typename T> class Vec2 {
public :
   Vec2() {}

   void set(const Vec2<T>&);
   void set(const T& a, const T& b);

   T val[2];
};

template<typename T>
void Vec2<T>::set(const Vec2<T>& v) { val[0] = v.val[0]; val[1] = v.val[1]; }

template<typename T>
void Vec2<T>::set(const T& a, const T& b) { val[0] = a; val[1] = b; }

void main() {
   Vec2<float> v2_f;
   Vec2<double> v2_d;

   v2_f.set(1.0f, 2.0f);

   v2_d.set(v2_f);


}
```

Error can occur in the following…

- Automatic `Vec2<float>` to `Vec2<double>` conversion does not occur.

```
template<typename T> class Vec2 {
public :
   Vec2() {}

   void set(const Vec2<T>&);
   void set(const T& a, const T& b);

   T val[2];
};

template<typename T>
void Vec2<T>::set(const Vec2<T>& v) { val[0] = v.val[0]; val[1] = v.val[1]; }

template<typename T>
void Vec2<T>::set(const T& a, const T& b) { val[0] = a; val[1] = b; }

void main() {
   Vec2<float> v2_f;
   Vec2<double> v2_d;

   v2_f.set(1.0f, 2.0f);

   v2_d.set(v2_f);             // Compilation Error !

                               // T := float (?) or double (?)
}
```

Solution: Use Member Template Parameter List

- A class template can have a member that is itself a template.
- The member template can be defined inside or outside of its enclosing class template.
- When we define a member outside the class template, we must include both template parameter lists, namely the class template parameter list and the member template parameter list, in the following way:

```
template<typename T> class Vec2 {
public :
   Vec2() {}

   template<typename S> void set(const Vec2<S>&);

   // ......
};

template<typename T> template<typename S>
void Vec2<T>::set(const Vec2<S>& v) { val[0] = v.val[0]; val[1] = v.val[1]; }

void main() {
   Vec2<float> v2_f;
   Vec2<double> v2_d;

   v2_f.set(1.0f, 2.0f);
   v2_d.set(v2_f);             // It works !
                               // T := double , S := float
}
```

Comparison

```
template<typename T, typename S> class Vec2 {
public :
   Vec2() {}

   void set(const Vec2<S>&);

   // ......
};

template<typename T, typename S>
void Vec2<T>::set(const Vec2<S>& v) { val[0] = v.val[0]; val[1] = v.val[1]; }

void main() {
   Vec2<double, float> v2_f;
   Vec2<double, float> v2_d;

   v2_f.set(1.0f, 2.0f);
   v2_d.set(v2_f);      <------------- It works. But

}
```

Operator Overloading for Class Template

```
template<typename T>
class Vec2 {
public :
   Vec2() {}
   Vec2(const T& a, const T& b) { val[0] = a; val[1] = b; }

   T& operator[](std::size_t i) const;

   T val[2];
};

template<typename T>
T& Vec2<T>::operator[](std::size_t i) const {
   return val[i];    // member operator overloading
}

template<typename T>
Vec2<T> operator+(const Vec2<T>& a, const Vec2<T>& b) {
   return Vec2<T>(a.val[0] + b.val[0], a.val[1] + b.val[1]);
}                    // non-member operator overloading
```

The Complete Vec2 Template Class

```
template<typename T>
class Vec2 {
public :
   Vec2() {}
   Vec2(const T& a, const T& b) { val[0] = a; val[1] = b; }

   template<typename S> void set(const Vec2<S>&);
   void set(const T& a, const T& b);

   const T& operator[](std::size_t i) const;

   T val[2];
};

template<typename T> template<typename S>
void Vec2<T>::set(const Vec2<S>& v) { val[0] = v.val[0]; val[1] = v.val[1]; }

template<typename T>
void Vec2<T>::set(const T& a, const T& b) { val[0] = a; val[1] = b; }

template<typename T>
T& Vec2<T>::operator[](std::size_t i) const { return val[i]; }

template<typename T>
Vec2<T> operator+(const Vec2<T>& a, const Vec2<T>& b) {
   return Vec2<T>(a.val[0] + b.val[0], a.val[1] + b.val[1]);
}
```