Object-Oriented Programming (OOP)
Lecture No. 37

Resolution Order

```
template< typename T >
class Vector { … };

template< typename T >
class Vector< T* > { … };

template< >
class Vector< char* > { … };
```

Resolution Order

- Compiler searches a complete specialization whose type matches exactly with that of declaration
- If it fails then it searches for some partial specialization
- In the end it searches for some general template

…Example – Resolution Order

```
int main() {
  Vector< char* > strVector;
     // Vector< char* > instantiated

  Vector< int* > iPtrVector;
     // Vector< T* > instantiated

  Vector< int > intVector;
     // Vector< T > instantiated
  return 0;
}
```

Function Template Overloading

```
template< typename T >
void sort( T );

template< typename T >
void sort( Vector< T > & );

template< >
void sort< Vector<char*> >(
                Vector< char* > & );

void sort( char* );
```

Resolution Order

- Compiler searches target of a function call in the following order
  - Ordinary Function
  - Complete Specialization
  - Partial Specialization
  - Generic Template

Example – Resolution Order

```
int main() {
  char* str = "Hello World!";
  sort(str); // sort( char* )

  Vector<char*> v1 = {"ab", "cd", … };
  sort(v1); //sort( Vector<char*> & )
```

…Example – Resolution Order

```
  Vector<int> v2 = { 5, 10, 15, 20 };
  sort(v2); // sort( Vector<T> &)

  int iArray[] = { 5, 2, 6 , 70 };
  sort(iArray); // sort( T )

  return 0;
}
```

Templates and Inheritance

- We can use inheritance comfortably with templates or their specializations
- But we must follow one rule:

"Derived class must take at least as many template parameters as the base class requires for an instantiation"

Derivations of a Template

- A class template may inherit from another class template

```
template< class T >
class A
{ … };

template< class T >
class B : public A< T >
{ … };
```

...Derivations of a Template

```
int main() {
  A< int > obj1;
  B< int > obj2;
  return 0;
}
```

...Derivations of a Template

- A partial specialization may inherit from a class template

```
template< class T >
class B< T* > : public A< T >
{ … };
```

...Derivations of a Template

```
int main() {
  A< int > obj1;
  B< int* > obj2;
  return 0;
}
```

...Derivations of a Template

- Complete specialization or ordinary class cannot inherit from a class template

```
template< >
class B< char* > : public A< T >
{ … }; // Error: 'T' undefined

class B : public A< T >
{ … }; // Error: 'T' undefined
```

Derivations of a Partial Sp.

- A class template may inherit from a partial specialization

```
template< class T >
class A
{ … };

template< class T >
class A< T* >
{ … };
```

...Derivations of a Partial Sp.

```
template< class T >
class B : public A< T* >
{ … }

int main() {
  A< int* > obj1;
  B< int > obj2;
  return 0;
}
```

...Derivations of a Partial Sp.

- A partial specialization may inherit from a partial specialization

```
template< class T >
class B< T* > : public A< T* >
{ … };
```

...Derivations of a Partial Sp.

```
int main() {
  A< int* > obj1;
  B< int* > obj2;
  return 0;
}
```

...Derivations of a Partial Sp.

- Complete specialization or ordinary class cannot inherit from a partial specialization

```
template< >
class B< int* > : public A< T* >
{ … }  // Error: Undefined 'T'

class B : public A< T* >
{ … }  // Error: Undefined 'T'
```

Derivations of a Complete Sp.

- A class template may inherit from a complete specialization

```
template< class T > class A
{ … };

template< >
class A< float* >
{ … };
```

...Derivations of a Complete Sp.

```
template< class T >
class B : public A< float* >
{ … };

int main() {
  A< float* > obj1;
  B< int > obj2;
  return 0;
}
```

...Derivations of a Complete Sp.

- A partial specialization may inherit from a complete specialization

```
template< class T >
class B< T* > : public A< float* >
{ … };
```

...Derivations of a Complete Sp.

```
int main() {
  A< float* > obj1;
  B< int* > obj2;
  return 0;
}
```

... Derivations of a Complete Sp.

- A complete specialization may inherit from a complete specialization

```
template< >
class B< double* > :
                    public A< float* >
{ … };
```

… Derivations of a Complete Sp.

```
int main() {
  A< float* > obj1;
  B< double* > obj2;
  return 0;
}
```

… Derivations of a Complete Sp.

- An ordinary class may inherit from a complete specialization

```
class B : public A< float* >
{ … };
```

… Derivations of a Complete Sp.

```
int main() {
  A< float* > obj1;
  B obj2;
  return 0;
}
```

Derivations of Ordinary Class

- A class template may inherit from an ordinary class

```
class A
{ … };

template< class T >
class B : public A
{ … };
```

...Derivations of Ordinary Class

```
int main() {
  A obj1;
  B< int > obj2;
  return 0;
}
```

...Derivations of Ordinary Class

- A partial specialization may inherit from an ordinary class

```
template< class T >
class B< T* > : public A
{ … };
```

...Derivations of Ordinary Class

```
int main() {
  A obj1;
  B< int* > obj2;
  return 0;
}
```

...Derivations of Ordinary Class

- A complete specialization may inherit from an ordinary class

```
template< >
class B< char* > : public A
{ … };
```

...Derivations of Ordinary Class

```
int main() {
  A obj1;
  B< char* > obj2;
  return 0;
}
```