Template Functions and Classes (1A)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Template Functions

```
template <class T>
void exchange(T& a, T& b) {
      T     t;

      t = a;
      a = b;
      b = t;
}
```

```
void main () {
      int         a=10, b=20;
      float       x=1.3, y=1,4;

      exchange <int> (a, b);
      exchange <float> (x, y);
}
```

template instances

```
// T = int
void exchange(int& a, int& b) {
      int   t;

      t = a;
      a = b;
      b = t;
}
```

```
// T = float
void exchange(float& a, float& b) {
      float      t;

      t = a;
      a = b;
      b = t;
}
```

Function Template Instantiation

```
template <class T>
void exchange(T& a, T& b) {
      T     t;

      t = a;
      a = b;
      b = t;
}
```

```
void main () {
      int         a=10, b=20;
      float       x=1.3, y=1,4;

      exchange <int> (a, b);
      exchange <float> (x, y);
}
```

Implicit Instantiation

usually template function is instantiated when it is called for the first time

```
template void exchange<int> (int, int);

...

template void exchange<float> (float, float);

...
```

Explicit Instantiation

explicit instantiation overrides any implicit ones

explicitly forces the compiler to generate codes for the template function whether it is used or not

An explicit specialization shall be declared in the namespace of which the template is a member, or, for member templates, in the namespace of which the enclosing class or enclosing class template is a member.

Template Argument Deduction

```
template <class T>
void exchange(T& a, T& b) {
      T     t;

      t = a;
      a = b;
      b = t;
}
```

Not every template argument has to be specified

➡ **Template Argument Deduction**

```
void main () {
      int         a=10, b=20;
      float       x=1.3, y=1,4;

      exchange  (a, b);
      exchange  (x, y);
}
```

exchange <int> (a, b);
exchange <float> (x, y);

```
void (*f1) (int&, int&);
void (*f2) (float&, float&);

f1 = &exchange;
f2 = &exchange;
f1 (a, b);
f2 (x, y)
```

exchange <int> (a, b);
exchange <float> (x, y);

Template and Overloaded Function Calls

```
template <class T>
void exchange(T& a, T& b) {
      T     t;

      t = a;
      a = b;
      b = t;
}
```

```
void main () {
      int         a=10, b=20;
      float       x=1.3, y=1,4;

      exchange <int> (a, b);
      exchange <float> (x, y);

      exchange (a, b);
      exchange (x, y);

}
```

Non-template overloaded functions

Overloaded functions can have variable numbers of arguments.

Overload Resolution

Non-template functions are first resolved over template functions

```
void exchange(int& a, int& b) {
      int  t;

      t = a;
      a = b;
      b = t;
}
```

```
void exchange(float& a, float& b) {
      float      t;

      t = a;
      a = b;
      b = t;
}
```

Template Function Specialization

```
template <class T>
void func(T a) {
   cout << "func <class T> is called\n";
   cout << a << endl;
}



template <>
void func <int> (int a) {
  cout << "func <int> is called\n";
  cout << a << endl;
}
```

```
template void func <int> (int);

int main (void) {

  int a = 3;
  double x = 3.14;

  func(a);

  func(x);

}
```

Class Structure

References

[1] W Savitch, “Absolute C++”
[2] P.S. Wang, “Standard C++ with objected-oriented programming”
[3] http://www.cplusplus.com
[4] http://www.cppreference.com