面向对象方法与C++程序设计

第7章 模板

大连理工大学
主讲人-宗林林

理解模板



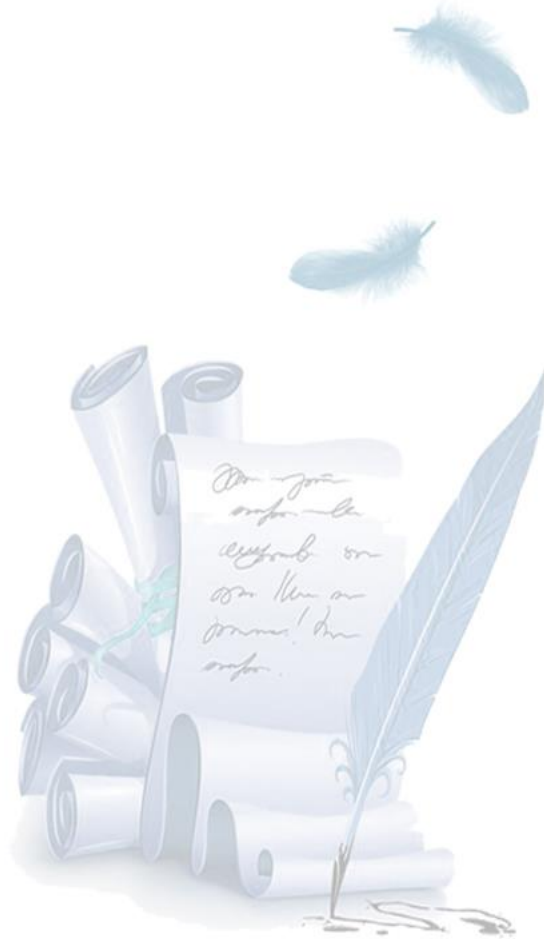
- 不具体指定操作对象的数据类型
- 减少重复代码的编写，实现代码可重用性
- 模板
  - 函数模板
  - 类模板

本节要点



函数模板

- 函数模板定义
- 函数模板实例化
- 函数模板特化



函数模板



问题：求两参数之中最小值：min (a , b)
对 a , b 的不同类型，都有相同的处理形式：
return (a < b) ? a : b ;

```
int min ( int a , int b )
{ return a < b ? a : b ; }
```

```
double min ( double a , double b )
{ return a < b ? a : b ; }
```

```
Clock c1, c2;
min(c1,c2);
```

函数模板



函数模板定义由模板参数说明和函数定义组成，语法如下：

template <class 类型参数名1 ,class 类型参数 2，…>

函数返回值类型 函数名(形式参数表)

{ 函数体 }

- 函数形式参数表中可以使用模板类型参数，也可以使用一般类型参数.

函数模板



```
template < class  T >
T  min ( T  a , T  b ){
  return  a < b ? a : b ;
}
```

```
double  min ( double a , double b )
{ return  a < b ? a : b ; }
```

```
char  min ( char  a , char b )
{ return  a < b ? a : b ; }
```

```
int  min ( int  a , int  b )
{ return  a < b ? a : b ; }
```

程序执行时
匹配不同的版本

编译器生成的
模板函数

```
void  main ( ){
  cout << " min ( 3 , 5 )  is  " << min ( 3 , 5 ) << endl ;
  cout << " min ( 'y' , 'e' )  is  " << min ( 'y' , 'e' ) << endl ;
  cout << " min ( 9.3 , 0.5 )  is  " << min ( 9.3 , 0.5 ) << endl ;
}
```

函数模板



编译器如何处理函数模板?

- 当函数模板被调用时编译器才产生代码。
- 根据实参的类型，替换函数模板中的类型参数。
- 这个过程叫做函数模板的实例化。
- 函数模板实例化后称为模板函数。

模板参数说明的每个类型参数必须在函数定义形参表中至少出现一次

函数模板



- 关键字class也可以使用关键字typename

```
template < class  T >
T  min ( T  a , T  b )
{  return  a < b ? a : b ; }
```

```
template < typename  T >
T  min ( T  a , T  b )
{  return  a < b ? a : b ; }
```

- 函数模板允许使用多个类型参数，但在template定义部分的每个形参前必须有关键字typename或class

```
template < class  T1 , class T2>
void  f ( T1  a , T2  b )
{  …  }
```

- 在template语句与函数模板定义语句<返回类型>之间不允许有别的语句

```
template < class  T >
int x = 3;   // error
T  min ( T  a , T  b )
{  return  a < b ? a : b ; }
```

函数模板



函数模板的特化：

函数模板可能不适合某种特殊类型数据的处理，这时就需要为函数模板实例化提供特化的定义。

函数模板特化的定义语法如下：

```
template<>
返回类型 函数模板名（参数列表）
{
    函数体
}
```

函数模板



```
template <class T>
T min(T a, T b){
        return (a<b)? a: b;
}
```

函数模板

```
template<> const char * min( const char * s1, const char * s2 ){
        return (strcmp(s1,s2)<0 ? s1 :s2);
}
```

函数模板的特化

```
int main(){
        int m_iv = 5, m_iu = 10;
        double m_dv = 5.5, m_du = 10.5;
        cout << min(m_iv,m_iu) << endl;
        cout << min(m_dv,m_du) << endl;
        cout << min("hello","world") << endl;
        return 0;
}
```

函数模板



程序中既定义了函数模板又定义了特化函数时，调用的匹配规则

- 如果参数类型以及返回类型完全匹配，则先选择普通函数，再选择模板显式特化函数作为调用的函数实例。
- 否则，如果模板函数能够推导出一个参数类型以及返回类型完全匹配的函数实例，则选择函数模板。
- 否则，如果调用函数的实参以及返回类型能够进行隐式转换成与普通函数或者模板显式特化函数的类型匹配，则选择普通函数或者模板显式特化函数。
- 如果以上三条都不能匹配，则函数匹配失败，发生编译错误。

面向对象方法与C++程序设计

第7章 模板

大连理工大学
主讲人-宗林林