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

本节要点



类模板

- 类模板定义
- 类模板实例化

类模板



- 类模板实际上是函数模板的推广

  类模板用于实现类所需数据的类型参数化

- 类模板主要用于数据存储（容器）类

  表示和算法不受所包含的元素类型的影响

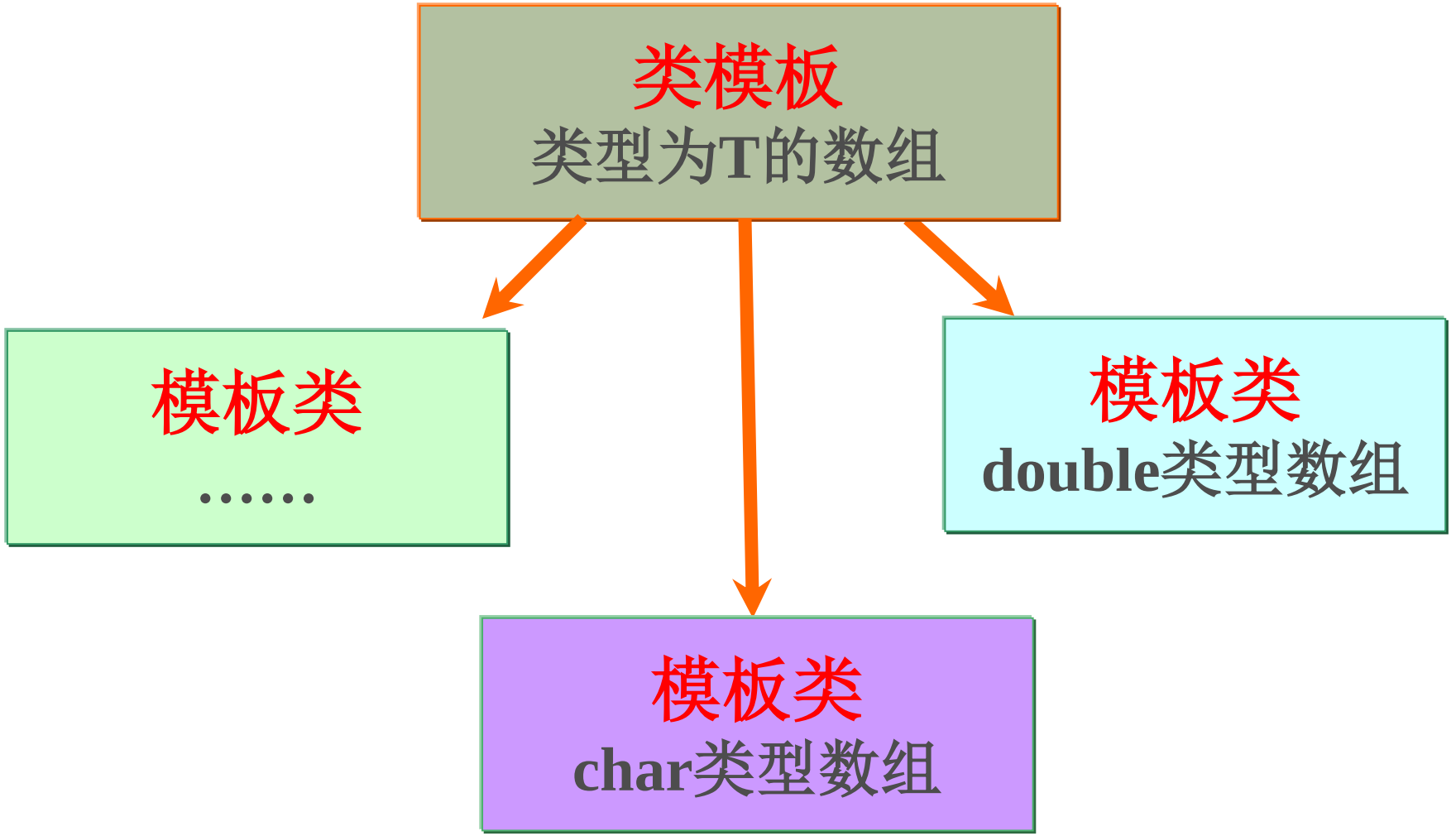
类模板



```
class  Array {    //整形数组类
 public :
        Array ( int s ) ;
        ~ Array ( ) ;
        int get( int index ) const ;
        void set( int index, const int & value ) ;
 protected :
        int size ;
        int * element ;
} ;
```

类模板





类模板



类模板由模板参数说明和类说明构成

```
template   <class 类型参数名1, class 类型参数 2，…>
class 类模板名
{ 成员函数和成员数据 };
```

例如

```
template< typename Type >
class TClass
{  // TClass的成员函数
   private :
      Type DateMember ;
   //…
};
```

类型参数必须至少

在类说明中出现一次

类模板



```
class  Array {
  public :
            Array ( int s ) ;
             ~ Array ( ) ;
             int get( int index ) const ;
             void set( int index, const int & value ) ;
protected :
            int size ;
          int * element ;
} ;
```

```
template<typename T>
class  Array {
  public :
            Array ( int s ) ;
             ~ Array ( ) ;
            T get( int index ) const ;
            void set( int index, const T & value ) ;
protected :
            int size ;
            T * element ;
} ;
```

一个数组类模板

类模板



```
template  <类型形参表>
<返回类型> <类模板名> <类型参数名表>::<成员函数1>（形参表）
{    //成员函数定义体 }

……

template  <类型形参表>
<返回类型> <类模板名> <类型参数名表>::<成员函数n>（形参表）
{    //成员函数定义体 }
```

类模板



```
template<typename T>
Array <T>::Array (int s){
        size = (s > 1 ) ? s : 1 ;
        element = new T [ size ] ;
}
template < typename T >
Array < T > :: ~Array(){
        delete [ ] element ;
}
```

```
template < typename T >
const T& Array < T > :: get ( int index ) const{
        return  element [ index ] ;
}
template < typename T >
 void Array < T > :: set(int index, const T& value){
        element [ index ] = value ;
}
```

类模板



```
template< typename T >
class  Array{
  public :
  Array ( int s ) ;
   ~ Array () ;
    const T& get( int index ) const ;
    void set( int index, const T & value ) ;
 protected :
    int size ;
    T * element ;
} ;
```

```
#include <iostream.h>
#include "Array.h"
void main(){
   Array <int> I( 5 ) ;
   for (int i = 0; i < 5; i ++ )
            I.set ( i, i ) ;
   for ( i = 0; i < 5; i ++ )
            cout << I.get(i) << '\t' ;
   Array <double> D ( 5 ) ;
   for ( i = 0; i < 5; i ++ )
            D.set ( i, (i+1)*0.35 ) ;
   for ( i = 0; i < 5; i ++ )
            cout << D.get(i) << '\t' ;
}
```

```
class  Array     //模板类
{ public :
      ……
      int set( int index ) const ;
      void get( int index, const int & value ) ;
  protected :
      int size ;   int * element ;
} ;
```

```
class  Array     //模板类
{ public :
      ……
      double get( int index ) const ;
      void set( int index, const double & value ) ;
  protected :
      int size ; double * element ;
} ;
```

类模板



使用类模板的方法

- 定义类模板。

template <类型形参表>
class <类名>{ …. };

- 创建一个类模板的模板类，同时创建模板类对象。

类名 <类型实参表> 对象表；

```
template< typename T >
class  Array{…}
void main()
{  Array <int> I ( 5 ) ; }
```

- 使用对象调用成员函数。
  其实参类型与模板类规定的类型一致。

```
void set( int index, const T & value ) ;
```

```
I.set ( 1, 2 ) ;
```

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