華中科技大學

面向对象程序设计

第十章 模板

许向阳

xuxy@hust.edu.cn

大纲

10.1 模板的概念

10.2 函数模板

10.3 类模板

10.1 模板的概念


```
int     add(int   a, int   b)
   {   int  x;    x=a+b;   return  x; }

double  add(double a, double b)
   {   double x;  x=a+b;   return  x; }

short   add(short    a, short    b)
    {  short  x;  x=a+b;   return  x; }
```

参数化多态性：

- 将一段程序所处理的对象类型进行参数化
- 使一段程序代码可以用于处理多种不同类型的对象

10.1 模板的概念

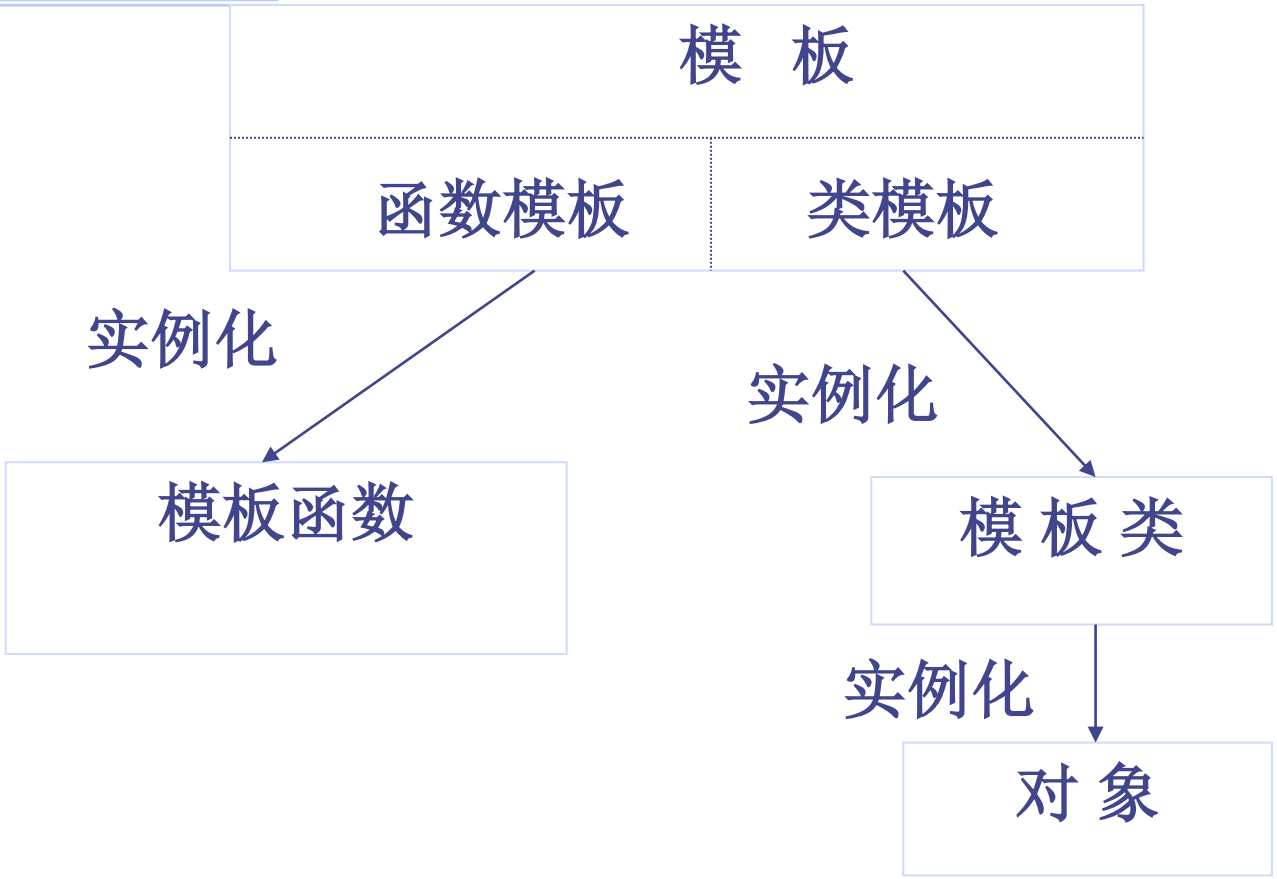
- 模板是对具有相同特性的函数或类的再抽象
- 模板是一种参数化的多态性工具；

采用模板编程，可以为各种逻辑功能相同而数据类型不同的程序提供一种代码共享的机制。

10.1 模板的概念

- 模板并非通常意义上可直接使用的函数或类
- 它是对一族函数或类的描述，是参数化的函数和类。
- 模板是一种使用无类型参数来产生一族函数或类的机制，它由通用代码构成。
- 称其为参数化是因为模板不是以数据为参数，而是以它所使用的数据类型为参数

10.1 模板的概念



模板实例化示意图

10.2 函数模板


```
int  add(int a, int b) {
      int   x;
      x= a+b;
      return  x;
}
```

```
double  add(double a, double b) {
      double   x;
      x= a+b;
      return  x;
}
```

T 换成 int

T 换成 double

```
T  add(T   a, T   b) {
      T   x;
      x= a+b;
      return  x;
}
```

10.2 函数模板


```
template  <模板形参表>
返回类型   函数名  (参数表)
{
        函数体
}
```

```
template <class T>
T  add(T   a, T   b) {
      T   x;
      x= a+b;
      return  x;
}
```

调用时：

```
int        x, y, z;
double u,v,w;
z =  add(x,  y);
w = add(u, v);
```

模板函数名：add

参数化的类型：T

10.2 函数模板

在机器代码中，有add<int> 和add<double>两个函数体

```
; 17   : z=add(x,y);

        mov     eax, DWORD PTR _y$[ebp]
        push    eax
        mov     ecx, DWORD PTR _x$[ebp]
        push    ecx
        call    ??$add@H@@YAHHH@Z
                        ; add<int>
```

10.2 函数模板


```
; 21   : w=add(u,v);
        sub      esp, 8
        movsd xmm0, QWORD PTR _v$[ebp]
        movsd QWORD PTR [esp], xmm0
        sub      esp, 8
        movsd xmm0, QWORD PTR _u$[ebp]
        movsd QWORD PTR [esp], xmm0
        call     ??$add@N@@YANNN@Z
                          ;  add<double>

        call     ??$add@H@@YAHHH@Z
                              ; add<int>
```

两个函数的名称不相同

10.2 函数模板

- <模板形参表>可以包含一个或多个用逗号分开的参数
- 每一项均有关键字class或typename引导一个标识符
- 此标识符为模板参数，表示一种数据类型
- 类型可以是基本数据类型，也可以是类类型
- 参数表中至少有一个参数说明
- 参数在函数体中至少应用一次

10.2 函数模板

- 函数模板只是一种说明，并不是一个具体函数
- 它是一组函数的模板，在定义中使用了参数化类型
- 编译系统对函数模板不产生任何执行代码
- 只有在遇到具体函数调用的时候，根据调用处的具体参数类型，在参数实例化以后才产生相应的代码，此代码称为模板函数。

10.3 类模板

- 定义一个堆栈类 STACK
- 可以将数据元素压栈
- 可以从栈中弹出数据元素
- 数据元素是整数类型时 STACK_INT
- 数据元素是double类型时 STACK_DOUBLE
- 数据元素是某一类型时 STACK_?

10.3 类模板

- 类模板是参数化的类，即用于实现数据类型参数化的类；
- 应用类模板可以使类中的数据成员、成员函数的参数及成员函数的返回值能根据模板参数匹配情况取任意数据类型；
- 类型既可以是C++预定义的数据类型，也可以是用户自定义的数据类型。

10.3 类模板


```
template  <模板形参表>
class 类名
{
    <类体说明>
}
```

- <模板形参表>中包含一个或多个用逗号分开的参数项，每一参数至少应在类的说明中出现一次；
- 参数项可以包含基本数据类型，也可以包含类类型；
- 若为类类型，使用前缀class。
- 形参类型用于说明数据成员和成员函数的类型。

10.3 类模板


```
template <class T>
class VECTOR
{
  private:
        T  *data;
        int  size;
  public:
        VECTOR(int);
        ~VECTOR();
        T  &  operator[](int);
};
```

10.3 类模板


```
template <class T>
VECTOR<T>::VECTOR(int n)
{       data = new T[size=n];  }

template<class T>
VECTOR<T>::~VECTOR()
{       delete []data;  }

template<class T>
T & VECTOR<T>::operator[](int i)
{       return data[i];  }
```

10.3 类模板

```
void main()
{
        VECTOR<int>    LI(20);
        VECTOR<double>   LD(30);
}
```

模板<模板参数表> 对象名1, 对象名2,…;

10.3 类模板

- 类模板自身并不产生代码
- 它指定类的一个家族
- 当引用时，才产生代码，生成模板类

10.3 类模板

类模板与继承

① 类模板可以从类模板派生

```
template <class T>
class VECTOR
{
  private:
        T *data;
        int size;
  public:
        VECTOR(int);
        ~VECTOR();
        T & operator[](int);
        int getsize() {return size;}
};
```

10.3 类模板

类模板与继承

① 类模板可以从类模板派生

```
template <class T>
class STACK : public VECTOR<T>
{
private:
        int  top;
public:
        int  full() { return  top==getsize();}
        int  empty() {return top==0;}
        int  push(T t);
        int  pop(T &t);
        STACK(int s):VECTOR<T>(s) { top=0;}
        ~STACK() { }
};
```

10.3 类模板

类模板与继承

① 类模板可以从类模板派生

```
template <class T>
int STACK<T>::push(T t)
{
        if (full()) return 0;
        (*this)[top++]=t;
        return 1;
}
```

类模板与继承

① 类模板可以从类模板派生

```
void main()
{
        STACK<int>   SI(20);
        STACK<double>   SD(30);

        SI.push(10);
        SI.push(20);
        SI.push(30);
}
```

② 非模板类可以从模板类派生

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

class derive:public base<int>
{
……
};
```

类模板与继承

- 在派生中，作为非模板类的基类，必须是类模板实例化后的模板类
- 在定义派生类前不需要模板声明语句：
  template<class T>

定义对象：
derive obj1(…),…

10.3 类模板

类模板与继承

③类模板可以从非模板类派生

② 非模板类可以从类模板派生

① 类模板可以从类模板派生

总结

模板的概念

函数模板的定义、使用

类模板的定义、使用