C++程序设计第十五节课官方笔记

目录

一、 课件下载及重播方法

二、 本章/教材结构图

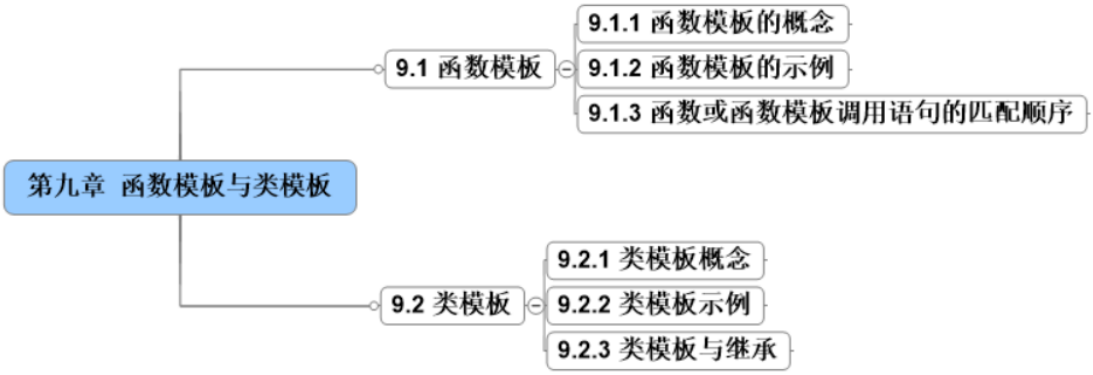
三、 本章知识点及考频总结

四、 配套练习题

五、 其余课程安排

一、课件下载及重播方法

二、教材节构图



三、本章知识点及考频总结

（一）选择题（共 7 道）

1.为了提高效率，实现代码复用，C++提供了一种处理机制，即使用**函数模板**。函数在设计时并不使用实际的类型，而是使用虚拟的类型参数。这样可以不必为每种不同的类型都编写代码段。当用实际的类型来实例化这种函数时，将函数模板与某个具体数据类型连用。编译器将以函数模板为样板，生成一个函数，即产生了模板函数，这个过程称为**函数模板实例化**。函数模板实例化的过程由编译器完成。程序设计时并不给出相应数据的类型，编译时，由编译器根据实际的类型进行实例化。

用实际的类型来实例化这种函数时，就好像按照模板来制造新的函数一样，所以称这种函数为**模板函数**。有了模板，编译器就能在需要的时候，根据模板自动生成程序的代码。从同一个模板自动生成的代码，形式几乎是一样的。

2.定义函数模板的一般格式如下：

```
template <模板参数表>
返回类型名 函数模板名(参数表)
{
    函数体的定义
}
```

函数模板的定义以**关键字 template** 开头，该关键字之后是使用尖括号<>括起来的“模板参数表”。模板参数表的写法和函数形参列表的写法是很相似的，由用逗号分隔的模板参数构成，形式是“**类型 参数名,类型 参数名,……**”。类型也称为占位符，参数名可以是 C++类型，也可以是具体的值，如数字、指针等。如果是一个类型，则需要使用 typename 或 class 关键字来表示参数的类型；如果参数是一个值，则需要写这个值的类型。“类型 参数名”可以是以下几种。

1)**class(或 typename) 标识符**，指明函数模板中可以接收的类型参数，类型参数由标识符表示。这些类型参数代表的是类型，可以是内部类型或自定义类型，应在“返回类型名”“参数表”或“函数体”中使用。

2)**类型说明符 标识符**，指明函数模板中可以接收一个由“类型说明符”所规定类型的常量作为参数。

函数模板中函数体的定义方式与定义普通函数时类似。

3.虽然**函数模板**的使用形式与**函数**类似，但二者有本质的**区别**，主要表现在以下 3 个方面。

1)函数模板本身在编译时不会生成任何目标代码，只有当通过模板生成具体的函数实例时才会生成目标代码。

2)被多个源文件引用的函数模板，应当连同函数体一同放在头文件中，而不能像普通函数那样只将声明放在头文件中。

3)函数指针也只能指向模板的实例，而不能指向模板本身。

4.在函数和函数模板名字相同的情况下，一条函数调用语句到底应该被匹配成对哪个函数或哪个模板的调用呢？C++编译器遵循以下先后顺序：

1)先找参数完全匹配的普通函数（不是由模板实例化得到的模板函数）。

2)再找参数完全匹配的模板函数。

3)然后找实参经过自动类型转换后能够匹配的普通函数。

4)如果上面的都找不到，则报错。

5.通过类模板，可以实例化一个个的类。继承机制也是在一系列的类之间建立某种联系，这两种涉及多个类的机制是有很大差异的。类是相同类型事物的抽象，有继承关系的类可以具有不同的操作。而模板是不同类型的事物具有相同的操作，实例化后的类之间没有联系，相互独立。

6.类模板声明本身并不是一个类，它说明了类的一个家族。只有当被其他代码引用时，模板才根据引用的需要生成具体的类。

不能使用类模板来直接生成对象，因为类型参数是不确定的，必须先为模板参数指定“实参”，即模板要“实例化”后，才可以创建对象。也就是说，当使用类模板创建对象时，要随类模板名给出对应于类型形参或普通形参的具体实参，格式如下：

```
类模板名 <模板参数表> 对象名 1,…,对象名 n;
```

或是

```
类模板名 <模板参数表> 对象名 1(构造函数实参),…,对象名(构造函数实参);
```

编译器由类模板生成类的过程称为类模板的实例化。由类模板实例化得到的类称为模板类。

要注意的是，与类型形参相对应的实参是类型名。

7.类之间允许继承，类模板之间也允许继承。具体来说，类模板和类模板之间、类模板和类之间可以互相继承，它们之间的**常见派生关系**有以下 4 种情况。

1)普通类继承模板类。

2)类模板继承普通类。

3)类模板继承类模板。

4)类模板继承模板类。

根据类模板实例化的类即是**模板类**。

(二) 主观题 (共·1 道)

类模板继承普通类：

```
#include<iostream>
#include<string>
using namespace std;
class TBase
{
    int k;
public:
    void print()
    {
        cout<<"TBase::"<<k<<endl;
    }
};
template<class T>
class TDerived:public TBase
{
    T data;
public:
    void setData(T x)
    {
        data=x;
    }
    void print()
    {
        TBase::print();
        cout<<"TDerived::"<<data<<endl;
    }
};
int main()
{
    TDerived<string>d;
    d.setData("2019");
    d.print();
}
```

四、配套练习题

1. 函数模板 template<typename T> void Func(T，T)可具有下列哪种实例化形式（　　）

A:void Func(float，int)

B:void Func(char，char)

C:void Func(int，double)

D:void Func(bool，float)

2.设有函数 T Sum(T x，T y){return x+y；}，其中 T 为模板类型，则下列语句中对该函数错误的使用是（　　）

A:Sum(1，2)；

B:Sum(3.0，2.2)；

C:Sum（‘A’，‘C’）；

D:Sum("A"，"C")；

3.有关函数模版和模版函数说法错误的是（　　）

A:函数模版只是对函数的描述，编译器不为其产生任何执行代码，所以它不是一个实实在在的函数

B:模版函数是实实在在的函数，它由编译系统在遇到具体函数调用时所生成，并调用执行

C:函数模版需要实例化为模版函数后才能执行

D:当函数模版和一般函数同名时，系统先去匹配函数模版，将其实例化后进行调用

[参考答案] BDD

五、其余课程安排