**面向对象程序设计作业-第1章 绪论**

|  |  |  |
| --- | --- | --- |
| 姓名：朱文灿 | | |
| 本章教学内容 | | |
| 教学内容 | 课内学时 | MOOC学习内容 |
| 第1章 绪论  1.1课程简介  1.2函数模板【重难点】 | 1  1 | 相应教学内容 |
| 作业及要求 | | |
| 1.结合课本第7章课后练习及MOOC相关内容复习本周教学内容；  2.需提交作业：  （1）基本概念题  （2）简答题  （3）选择题  （4）程序设计题  3.作业提交的相关说明：  （1）按作业模板撰写作业文档；  （2）作业提交方式：请班长/学委收齐后发至邮箱zuozejun@mapgis.com；  （3）作业提交时间：周二（2月19日）晚10:00前；  （4）每次作业根据提交的数量和质量评定成绩。 | | |

## 一、基本概念题

1.C++最重要的特性之一就是代码重用，为了实现代码重用，代码必须具有 通用性 。

通用代码需要不受数据 类型 的影响，并且可以自动适应数据类型的变化。这种程序设计类型称为 参数化 程序设计。模板是C++支持参数化程序设计的工具，通过它可以实现参数化 多态 性。

2.函数模板的定义形式是template<模板参数表> 返回类型 函数名（形式参数表）{…}。

其中，<模板参数表>中参数可以有 多 个，用逗号分开。模板参数主要是 模板类型参数。

它代表一种类型，由关键字 typename 或 class 后加一个标识符构成，标识符代表一个潜在的内置或用户定义的类型参数。类型参数又可以是任意合法标识符。C++规定参数名必须在函数定义中至少出现一次。

3.编译器通过如下匹配规则确定调用哪一个函数：首先，寻找最符合 函数名 和 参数类型 的一般函数，若找到则调用该函数；否则寻找一个 函数模板 ，将其实例化成一个 模板函数 ，看是否匹配，如果匹配，就调用该 模板函数 ；

再则，通过 类型转换 规则进行参数的匹配。若还没有找到匹配的函数则调用错误。若有多于一个函数匹配，则调用产生 二义性 ，也将产生错误。

4.类模板使用户可以为类声明一种模式，使得类中的某些数据成员、某些成员函数的参数、某些成员函数的返回值能取 任意类型 （包括 系统预定类型 和 用户自定义 的类型）。类是对一组对象的公共性质的抽象，而类模板则是对不同类的 数据类型的抽象，因此类模板是属于更高层次的抽象。由于类模板需要一种或多种 类型 参数，所以类模板也常常称为 参数化类 。

## 二、简答题

1.简述函数模板生成函数的过程。

当编译器调用函数时，针对于实参的数据类型，它会首先寻找可以匹配的一般函数，否则将根据已有的可用的函数模板生成与数据类型相匹配的函数代码。从而形成可用的函数。

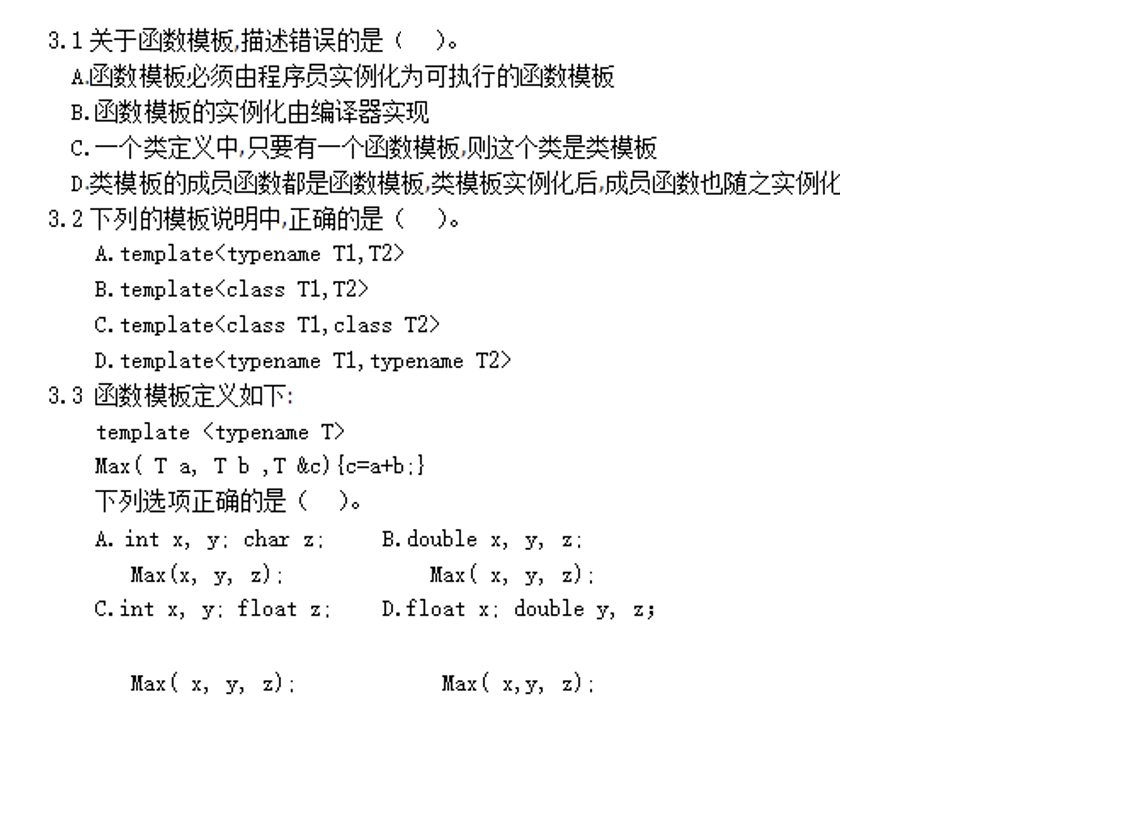
1. 简述类模板生成对象的过程。

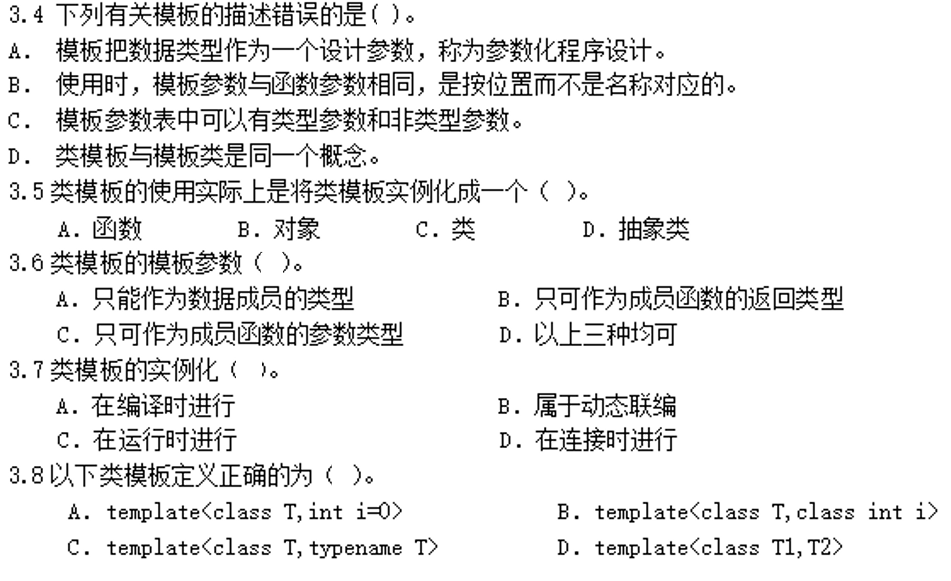
定义实参的数据类型之后，根据类型实参的不同，类模板会生成一个数据成员，成员函数的参数，成员函数的返回值均为类型实参的模板类。

1. 简述函数模板与模板函数、类模板与模板类的区别。

函数模板和类模板把不能确定数据类型的数据成员，函数参数或返回值的函数或类实例化不同的函数和类，它们代表了一类的函数或者类。模板函数和模板类分别是它们实例化后的产物。

## 三、选择题





**A D B D C D A A**

## 程序设计题

1. 设计一个函数模板，其中包括数据成员T a[n]以及对其进行排序的成员函数sort( )，模板参数T可实例化成字符串。

#include<iostream>

using namespace std;

template<typename T>

void sort(int n,T \*a) {

for (int i = 1; i < n; ++i) {

int value = a[i];

int j = 0;

for (j = i - 1; j >= 0; j--) {

if (a[j] > value) {

a[j + 1] = a[j];

}

else {

break;

}

}

a[j + 1] = value;

}

}

int main() {

char greeting[] = "hellomyarray";

sort(13,greeting);

for (int i = 0; i < 13; i++) {

cout << greeting[i];//aaehllmorryy

}

}

1. 设计一个类模板，其中包括数据成员T a[n]以及在其中进行查找数据元素的函数int search(T)模板参数 T可实例化成字符串。

#include<iostream>

using namespace std;

template<typename T,int n>

class Show {

public:

Show();

~Show();

int Search(T);

void NumberGiven(int i,const T& num);

private:

T\* a;

int len;

};

template<typename T, int n>Show<T, n>::Show() {

len = n;

a = new T[len];

}

template<typename T, int n>Show<T, n>::~Show() {

delete[]a;

}

template<typename T, int n>int Show<T, n>::Search(T k) {

for (int i = 0; i < len; i++) {

if (a[i] == k) {

cout << "find it\n";

break;

}

else {

cout << "no results\n";

}

}

return 0;

}

template<typename T, int n>void Show<T, n>::NumberGiven(int i,const T &num) {

a[i] = num;

}

int main() {

Show<string, 10> Test1;

for (int i = 0; i < 10; i++) {

Test1.NumberGiven(i,"a");

}

Show<int, 6> Test2;

for (int i = 0; i < 6; i++) {

Test2.NumberGiven(i,20);

}

Test1.Search("a");

Test2.Search(20);

return 0;

}

## 本章学习小结

学习了模板的方法来使一个函数或模板能够一次处理多种数据类型。数据成员，函数返回值类型，参数类型，局部变量都可以成为泛型。

## 本章学习待答疑问题

程序设计题2中template<typename T, int n>void Show<T, n>::NumberGiven(int i,const T &num) {

第二个参数类型为什么一定要加const呢，不加就无法通过编译。