## 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泛型程序设计。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>中参数可以有任意个，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class或typename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形参类型 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，通过自动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，则调用产生二义性错误，也将产生错误。

1. 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 基本数据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。

编译器在编译的过程中利用用户提供的实参来推断模板参数的类型，并生成一个特定类型的函数。

1. 简述类模板生成对象的过程。

编译器利用用户提供的模板实参信息来推断模板参数的类型并创建一个实例类，继而生成对象。

3.简述函数模板与模板函数、类模板与模板类的区别。

函数模板通过实例化生成一个特定的函数，称为模板函数。

类模板通过实例化创建一个实例类，成为模板类。

1. 选择题

3.1 A

3.2 C,D

3.3 B

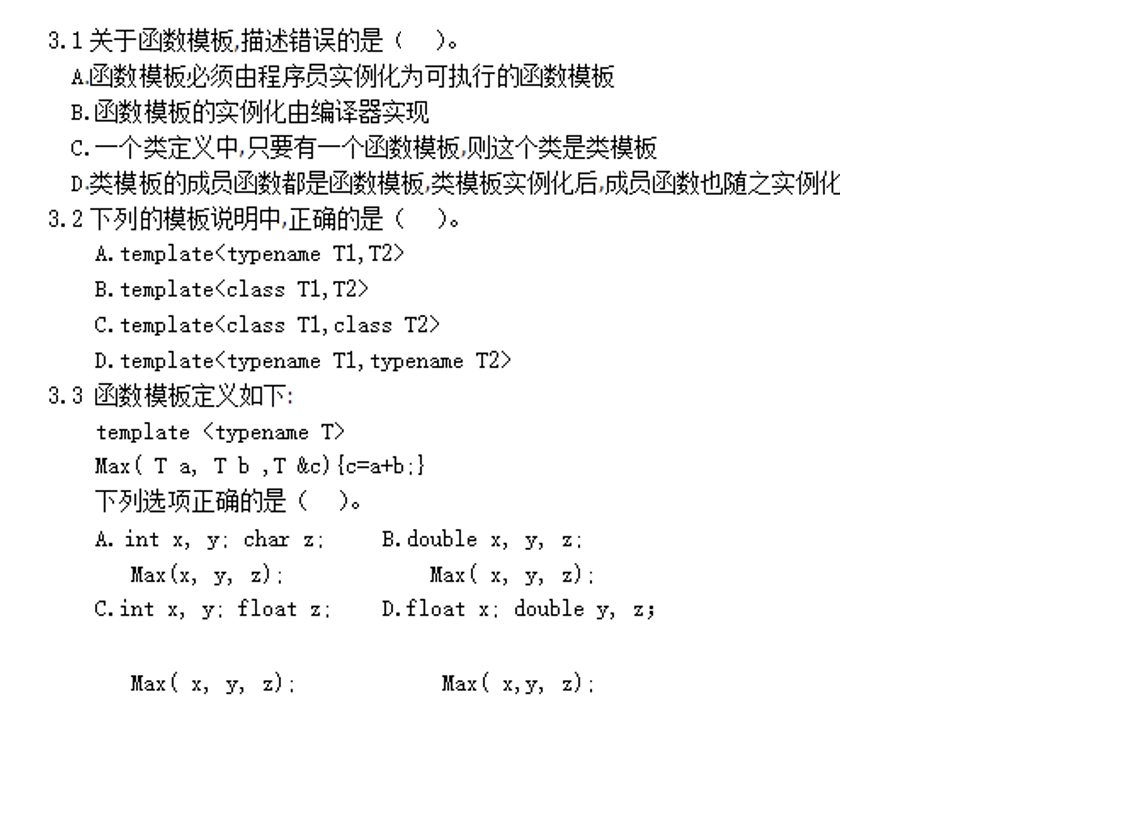
3.4 D

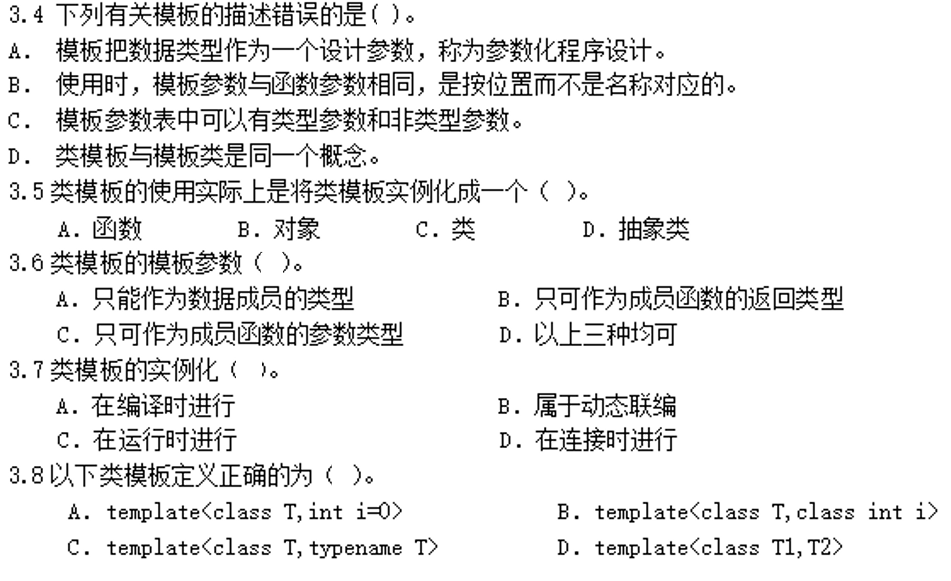
3.5 C

3.6 D

3.7 A

3.8 A





## 四、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(T \*a, int n)

{

for (int i = 0; i < n-1; i++)

{

for (int j = 0; j < n - 1 - i; j++)

{

if (a[j] < a[j+1])

{

T temp;

temp = a[j];

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

a[j+1] = temp;

}

}

}

}

int main() {

string str1[5] = { "c","g","a","b","q" };

sort<string>(str1, 5);

for (auto i : str1)

cout << i << ",";

cout << endl;

return 0;

}

2．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 M {

T a[n];

public:

M() {}

M(const std::initializer\_list<T>&);

int search(T);

void print();

};

template<typename T, int n>

M<T, n>::M(const std::initializer\_list<T>& l) :a{ T() }

{

int m = l.size() < n ? l.size() : n;

for (int i = 0; i < m; i++)

a[i] = \*(l.begin() + i);

}

template<typename T, int n>

int M<T, n>::search(T temp)

{

for (int i = 0; i < n; i++)

if (temp == a[i])

return i;

return -1;

}

template<typename T, int n>

void M<T, n>::print() {

T t;

cout << "请输入要查找的元素:";

cin >> t;

int m = search(t);

if (m == -1)

cout << "无此元素！" << endl;

else

cout << "要查找的元素下标为:" << m << endl;

}

int main() {

M<string, 5>str1 = { "a","ci","wen","q","ll" };

str1.print();

return 0;

}