1.重写（覆盖）和重载的理解

2.是否可将类的每个成员函数都声明为虚函数，为什么?

3.构造函数为什么不能是虚函数

4.析构函数为什么应该设置成虚函数

5.构造函数析构函数调用顺序

6.内联函数可以是虚函数吗

7.对象访问普通函数快还是虚函数更快

8.虚函数表是在什么阶段生成的，存放在哪里

9.什么是抽象类，抽象类的作用

10.虚函数表指针(VPTR)被编译器初始化的过程，你是如何理解的

11.静态成员可以是虚函数吗

多态

当类之间通过继承关联的时候，会用到多态，调用成员函数的时候，会根据调用函数的对象的类型来执行不同的函数。具体体现在编译和运行两个方面，在编译的时候多态性体现在函数和运算符的重载上，程序运行时的多态通过继承和虚函数来实现。要触发动态多态或者动态绑定，需要的条件是：有指定为虚函数的成员函数；通过父类的引用或指针进行的函数调用。

虚函数

虚函数　是在基类中使用关键字　virtual　声明的函数。当我们使用基类的引用或指针调用基类中定义的一个函数时，我们并不知道该函数真正作用的对象是什么类型，因为它可能是一个基类的对象也可能是一个派生类的对象。如果该函数时虚函数，在运行时会根据这个引用或指针所绑定的对象的真实类型来决定到底执行哪个版本。

虚函数允许派生类重新定义成员函数，而派生类重新定义基类的做法称为覆盖，或者称为重写。（重写的话有两种，直接重写成员函数和重写虚函数，只有

重写了虚函数的才能算作体现了c++多态性，否则叫做重定义）。

纯虚函数

如果想要在基类中定义虚函数，但是又不能对虚函数给出有意义的实现，就可以定义成纯虚函数，可以再派生类中根据需要对它定义。纯虚函数不能实例化。

虚函数表

虚函数的地址存放于虚函数表中，运行期的多态是通过虚函数表实现的。同一个类的所有对象都使用同一个虚表，这个表内是一个类的虚函数的地址。当用父类指针来操作一个子类的时候，这张虚函数表会指明实际应该调用哪个虚函数。

1.重写（覆盖）和重载的理解

重载(overload)：指在同一个作用域中允许有多个同名函数，这些函数参数列表不同，包括参数个数不同，参数类型不同，将同名的函数的名称进行修饰，从而生成一些不同名称的预处理函数，来实现同名函数调用时的重载问题，但这并没有体现c++的多态性。

重写（override)和重定义都是在不同的作用域，也就是派生类和基类。重写指派生类重写了父类的虚函数，要求参数相同，返回值相同，重写体现了c++的多态性。

重定义也叫同名隐藏，指派生类的函数屏蔽了与其同名的基类函数。如果派生类的函数与基类的函数同名，但参数不同，此时不论有无virtual关键字，基类的函数将被隐藏。如果派生类的函数与基类的函数同名，参数也相同，但是基类函数没有virtual关键字，此时基类的函数被隐藏。

2.是否可将类的每个成员函数都声明为虚函数，为什么?

没有必要。通过虚函数表指针vptr调用重写函数是在程序运行时进行的，因此需要通过寻址操作才能确定真正调用的函数。而普通成员函数在编译时就确定了要调用的函数。在效率上，虚函数的效率要低很多，所以没有必要将所有的成员函数能申明为虚函数。

3.构造函数为什么不能是虚函数

从存储空间角度：

虚函数的实现是对应一个虚函数表vtable，这个表是存储在对象的内存空间的。如果构造函数是虚的，就需要通过vtable来调用，但是对象还没有实例化，也就是内存空间还没有，无法找到表，所以构造函数不能是虚函数。

从使用角度：

构造函数是在创建对象时自动调用的，不可能通过父类的指针或者引用去调用，所以规定构造函数不能是虚函数。

从实现上看：

**vbtl在构造函数调用后才建立，因而构造函数不可能成为虚函数。

4.析构函数为什么应该设置成虚函数

假设没有设置成虚函数，如果有基类指针指向派生类，那么用基类指针delete时，如果不定义成虚函数，派生类中派生的部分无法析构，造成内存泄漏

5.构造函数析构函数调用顺序

1. 基类首先被创建
2. 派生类构造函数应通过成员初始化列表（调用顺序和继承顺序相关）将基类信息传递给基类构造函数
3. 派生类构造函数应初始化派生类新增的数据成员
4. 派生类对象析构时，程序首先调用派生类析构函数，再调用基类析构函数

6.内联函数可以是虚函数吗

不能，因为内联函数没有地址，无法把地址放到虚函数表中

7.对象访问普通函数快还是虚函数更快

如果是普通对象，是一样快的。如果是指针对象或者是引用对象，则调用普通函数快，因为构成多态，运行时调用虚函数需要到虚函数表。

8.虚函数表是在什么阶段生成的，存放在哪里

虚函数表是在编译阶段就生成的，一般情况下存在静态区。

9.什么是抽象类，抽象类的作用

抽象类是包含纯虚函数的类，抽象类强制重写了虚函数，另外抽象类体现了接口继承关系。

10.虚函数表指针(VPTR)被编译器初始化的过程，你是如何理解的

当类中声明虚函数时，编译器会在类中生成一个虚函数表。虚函数表是一个存储类成员指针的数据结构，由编译器自动生成与维护。virtua成员函数会被编译器放入虚函数表中，存在虚函数时，每个对象中都有一个指向虚函数表的指针。(vptr指针）

11.静态成员可以是虚函数吗

不能。因为静态成员函数没有this指针，使用类型：：成员函数的调用方式无法访问虚函数表，所以静态成员函数无法放进虚函数表。