高级程序设计期末复习

一、作业

1. 数据抽象和封装是什么？有哪些不同？
   a) 数据抽象：对外界隐藏数据的具体属性，数据抽象是数据的使用者只需要知道对数据所能实施的操作以及这些操作之间的关系，而不必知道数据的具体表示形式。
   b) 数据封装：数据的具体表示对使用者是不可见的（封装），对数据的访问（使用）只能通过封装体所提供的对外接口（操作）来完成。用户只通过函数对数据进行操作，对于数据的直接操作将被视为非法的。
   c) 区别：数据抽象强调外部行为，数据封装强调内部实现。类的抽象是把这个类的使用和这个类的具体实现分离开。类的封装是把具体实现封装起来，并对用户隐藏。
2. 请列举不同功能的构造函数以及析构函数。
   a) 构造函数：
      i. 默认构造函数：在定义类的对象但是没有初始化时调用。默认构造函数没有参数或者参数都有默认值。
      ii. 非默认构造函数（带参数构造函数）：初始化类采用两种方式，即初始化列表，或构造函数内部实现。在创建对象时，通过定义对象时显式调用发挥作用。
      iii. 拷贝构造函数：当用一个对象初始化同类型对象时、对象作为值传递参数时、对象作为函数返回值时调用。将一个对象的信息初始化另一个同类型对象。拷贝构造函数的参数必须是本类对象的引用。
      iv. 复制构造函数。将一个对象的信息赋值给另一个已被初始化的对象，可以通过重载=运算符实现
      v. 隐式拷贝构造函数：如程序未提供拷贝构造函数，编译器会为其生成一个隐式拷贝构造函数，将成员拷贝初始化。
   b) 析构函数：
      i. 自动调用的析构函数。在对象消亡时自动被调用，归还对象申请的资源。
      ii. 显式调用的析构函数。显式调用析构函数只会归还对象申请的资源，对象仍可以使用。
      iii. 没有返回值，没有参数，不能被重载，一个类仅一个析构函数；
      iv. 析构函数除了释放工作，还可以做一些用户希望它做的一些工作，比如输出一些信息。
3. C++ 中为什么要使用模板？
   a) 在程序设计中，经常需要用到一些功能完全相同的程序实体，但它们所涉及的数据的类型不同。通过把通用逻辑设计为模板，摆脱了类型的限制，极大地提升了代码的可重用性。
4. C++ 中的模板有哪两种形式，有什么区别？

a) 函数模板和类模板。区别：函数模板的实例化是由编译程序在处理函数调用时自动完成的，而类模板的实例化必须由程序员在程序中显式地指定。即函数模板允许隐式调用和显式调用，而类模板只能显示调用。

5. 为什么需要继承机制？
   a) 答：【软件/代码复用】在开发一个新软件时，我们常常需要进行软件复用。从面向对象程序设计的角度出发，类是主要的构成单元，因此进行软件复用的主体通常是在类的复用上。目前，不加修改的直接对已有软件进行复用显然是难以解决现实中复杂的需求的，同时也是困难的。在面向对象的程序设计中，类的继承机制支持将已有程序中的一个或多个类的功能全部包含进来，并可以根据现实需求对于新功能进行重新定义或者对已有类的功能进行重定义。这种复用方式可以降低开发的成本，缩短开发周期，极大地提升开发效率。
   b) 【补充点】此外，类的继承机制可以按照现实世界更加合理的刻画事物的层次、对概念进行组合、以及提供对软件增量开发的支持。
6. 描述什么是继承与封装之间的矛盾，以及对继承方式作用的理解。
   a) 答：【封装与继承之间的矛盾】在继承的实现中，派生类中定义新的成员函数时，往往需要直接访问基类的一些 private 成员（特别是 private 数据成员），否则新的功能就无法实现；从数据封装的角度出发，类的 private 成员是不允许外界使用的。这便是继承与封装的矛盾。
   b) 【继承方式作用的理解】通过设置 private、public、protected 三种继承方式，来控制派生类以及派生类的使用者对于基类成员的访问限制。这种方式一方面可以缓解与封装之间的矛盾，比如 protected 继承；另一方面，提供了多种多样的访问限制方法，可以在更多的场景中进行适配和设计，以满足现实开发的需求。
7. 假如我们写一个程序，程序在运行时会通过输出流来输出日志的形式来记录运行状态，但不幸的是，某个时刻程序崩溃了，此时我们是否一定能看到程序崩溃时最新输出的日志吗？如果能，为什么？如果不能，为什么？有哪些方法可以应对这一情况？思考一下这对精准调试大型复杂程序的意义。
   a) 如果某个时刻程序崩溃了，我们并不一定能看到程序崩溃时最新输出的日志
      i. 这里假设程序是通过 ofstream 的输出流来输出日志，若不能正常执行 close() 函数，倘若缓冲区未满，则日志不能输出。
   b) 应对这个情况，可以有 3 种方法来解决
      i. 捕获程序异常
         1. 倘若能捕获程序异常并对其进行处理，可以起到一定作用；但是这种方法的难度很大，很难涵盖所有异常。对于硬件异常无能为力。
      ii. 用 flush()函数
         1. 用 flush() 函数可以强制将缓冲区的内容写入磁盘，因此若每次向输出流中写数据后都调用 flush() 函数，则可以将崩溃前的内容写入日志文件中。不过这种方法在程序崩溃后，并不能得到关于崩溃的日志信息。

iii. IPC(Inter-process_communication)

1. 可以用一个监控程序在另一个进程上监控目标程序的运行，通过进程间通信的方法来得到目标程序的输出，并用输出流输出日志。这样即使是程序发生了异常，监控程序也可以得到与崩溃相关的信息。

c) 若程序通过输出流来输出日志的形式来记录运行状态，则通过查看日志可以定位异常出现的地方，并且由于有着输出流，可以通过输出的信息去推断异常发生前程序的状态，变量的内容，甚至于寄存器的状态等等，如果不是并发的程序，那么理论上同样的输入可以得到输入的结果。由于大型复杂程序的运行可能需要耗费许多硬件资源和时间，因此使用日志来精准调试可以快速定位程序的错误并且大大降低成本。

8. 一般多个程序打开同一个文件时，需要调用 clear 函数，如果不调用这个函数，会出现什么问题？

a) clear 在这里的作用是：清除 stream 对象的 error state，即：将所有的 eofbit, failbit, badbit 变为 goodbit

b) 如果重复使用同一个 stream 对象绑定到不同的文件，但之前一次文件读写过程中产生了一些 error state，仅仅调用 close 并不能实现 error state 的清除，这就使得下一次读写时这些 error state 仍然保留，这就导致下一次读写时 stream 对象的状态处于某种错误状态，从而使得文件没有办法正常读写。因此当绑定到第二个文件时需要进行 clear 操作，使得 stream 对象状态为 goodbit，从而能正常读写。