STL

STL (Standard Template Library)

- 标准模版库
- “容器”的集合
  - 序列式容器
    - vector，list，deque
  - 适配器容器
    - stack，queue，priority_queue
  - 关联式容器
    - set，multiset，map，multimap
- 算法的集合

http://www.cplusplus.com/reference/stl/

STL

STL (Standard Template Library)

- 标准模版库
- “容器”的集合
- 算法的集合
  - 比较
  - 交换
  - 查找
  - 遍历
  - 反转
  - 排序

http://www.cplusplus.com/reference/algorithm/

STL

STL (Standard Template Library)

- 标准模版库
- “容器”的集合
- 算法的集合
- 迭代器
  - 将算法和容器联系起来
  - 在算法中，通过迭代器，遍历容器

http://www.cplusplus.com/reference/iterator/

C++语言程序设计

回顾与总结

宋霜

哈尔滨工业大学（深圳）

机电工程与自动化学院

邮箱: songshuang@hit.edu.cn

类

对象

- 万物皆对象
- 静态特征：属性
- 动态特征：行为
- 外界交互：消息

类

面向对象的程序设计

- 复杂的事物总是由许多部分组成
- 研究各部分之间的功能与联系

类

面向对象的程序设计

- 复杂的事物总是由许多部分组成
- 研究各部分之间的功能与联系

- 确定系统由哪些对象组成
- 确定每个对象的属性与行为
- 研究对象之间的联系

类

面向对象

- 抽象
- 封装
- 继承
- 多态

类

构造函数

- 对象初始化操作

析构函数

- 对象销毁时的操作

类

成员变量

- 属性，名词

成员函数

- 行为，动词，接口

public，private

运算符重载

- 运算符重载使C++具有更好的扩充性和适应性

函数类型 operator 运算符名称(参数表)

类

成员函数or友元函数

- =，[]，()，->必须为成员函数
- 流插入<<和流提取运算符>>，类型转换运算符必须为友元函数
- 一般将单目和复合运算符重载为成员函数
- 一般将双目运算符重载为友元函数

继承与派生

- 派生类是基类的具体化
- 基类是派生类的抽象

```
class 派生类名：[继承方式]基类名
{
    派生类新增加的成员
}
```

类

访问属性

基类中的访问属性	继承方式	派生类中的访问属性
私有	公用	不可访问
私有	私有	不可访问
私有	保护	不可访问
公用	公用	公用
公用	私有	私有
公用	保护	保护
保护	公用	保护
保护	私有	私有
保护	保护	保护

基类与派生类的转换

- 只有公有派生类才是基类真正的子类，完整地继承了基类的功能。
- 派生类对象可以向基类对象赋值
- 派生类对象可以替代基类对象向基类对象的引用进行赋值或初始化
- 如果函数的参数是基类或基类对象的引用，相应的实参可以用子类对象
- 派生类对象的地址可以赋给指向基类对象的指针变量

多态性

- 向不同的对象发送同一个消息，不同的对象在接收时会产生不同的行为。
- C++中的表现形式：具有不同功能的函数可以用同一个函数名，实现调用不用内容的函数。
- 静态多态性
  - 函数重载，编译时决定
- 动态多态性
  - 运行时决定，通过虚函数实现

类

纯虚函数

- virtual 函数类型 函数名（参数表）=0；

抽象类

- 定义了纯虚函数的类
- 目的时用来作为基类去建立派生类
- 为一个类族提供一个公共接口

类

使公有继承体现 "是一个" 的含义

通过分层来体现 "有一个" 或 "用...来实现"

区分接口继承和实现继承

- 定义纯虚函数的目的在于，使派生类仅仅只是继承函数的接口。
- 声明简单虚函数的目的在于，使派生类继承函数的接口和缺省实现。
- 声明非虚函数的目的在于，使派生类继承函数的接口和强制性实现。

决不要重新定义继承而来的非虚函数

模块化程序设计

概念

- 模块化程序设计是指在进行程序设计时将一个大程序按照功能划分为若干小程序模块
- 每个小程序模块完成一个确定的功能
- 在这些模块之间建立必要的联系
- 通过模块的互相协作完成整个功能的程序设计方法。

模块化程序设计

思想

- 程序的编写不是一开始就逐条录入计算机语句和指令
- 而是首先用主程序、子程序、子过程等框架把软件的主要结构和流程描述出来
- 并定义和调试好各个框架之间的输入、输出链接关系
- 逐步求精得到一系列以功能块为单位的算法描述
- 以功能块为单位进行程序设计，实现其求解算法的方法称为模块化。
- 模块化的目的是为了降低程序复杂度，使程序设计、调试和维护等操作简单化。

模块化程序设计

步骤

- 分析问题，明确需要解决的任务；
- 对任务进行逐步分解和细化，分成若干个子任务，每个子任务只完成部分完整功能，并且可以通过函数来实现；
- 确定模块（函数）之间的调用关系；
- 优化模块之间的调用关系；
- 在主函数中进行调用实现。

模块化程序设计

原则

- 1．模块独立
  - 模块的独立性原则表现在模块完成独立的功能，与其他模块的联系应该尽可能得简单，各个模块具有相对的独立性。
- 2．模块的规模要适当
  - 模块的规模不能太大，也不能太小。如果模块的功能太强，可读性就会较差，若模块的功能太弱，就会有很多的接口。读者需要通过较多的程序设计来进行经验的积累。
- 3．分解模块时要注意层次
  - 在进行多层次任务分解时，要注意对问题进行抽象化。在分解初期，可以只考虑大的模块，在中期，再逐步进行细化，分解成较小的模块进行设计