**Class and Object**

(Encapsulation、Inheritance、Polymorphism) Communication

**概述**

内置类型，如int, 其分量是bit(private’s property), 数学运算符和位操作(public’s method), 该数据类型存储的值(public’s property)，其实例叫变量。

自定义类型class: Data&Operation，默认权限private，其实例叫对象。

世界观：程序的运行，是对象通过类来表现和沟通

函数签名(函数匹配的依据)：函数名，参数表的类型及个数，调用方式

类，当作一个命名空间，作用域。指针引用，需注明类域

对象的非静态成员，都附有this指针。

程序的执行入口，函数main()

 Unicode字符集对所有字符进行统一的双字节编码。ASCII码字符集国标码字符集都是它的子集。文本文件是基于字符编码的文件，二进制文件是基于值编码的文件。

**经验之谈**

类的命名, 大写字母开头

Public: 开头, 尽可能向外展示接口（API），向内隐藏实现细节

将一个完整的类，放在头文件中，形成独立的编译单位。注意头文件保护

const 保护, 一个防止出错的习惯。

自定义类型在函数中的参数传递，应定义为引用&，避免拷贝造成的开销

virtual、Static 、Friend 等属于前置关键字，对其身份的修饰形容，不算入函数签名。

sizeof(data type) 运算符，获取所占字节数

**预处理：**

**编译预处理**：以符号“#”引出，一条指令也需在一行内写完。

**封装**

静态∈类，非静态∈对象，但非静态可以访问静态。

**Const：只读属性**

**非静态成员：Member（Property、Appropriate）**

只是一个类模型的声明，未实例化。

**引用：Reference**

对象的地址只读变量的封装，对引用变量操作就是对其指向的**直接操作,** 必须初始化，一旦绑定对象，永不变更。

**外部访问权限：Access Permission**

Public: 对外开放的接口，可通过 点操作符（.）进行访问

Protected：可继承的Private部分，供子类访问（递推）

Private：不对外开放

|  |  |  |
| --- | --- | --- |
| 外部访问 | 通过子类访问 | 通过对象/类名访问 |
| Public | √ | √ |
| Protected | √ |  |
| Private |  |  |

**构造与析构函数** 对象建立时，自动调用，无显式返回值

（相当于赋值初始化）

Constructor function在对象建立时调用, 可重载，禁止virtual属性

Destructor function在对象销毁时调用, 可为virtual属性，禁止重载

Attention:

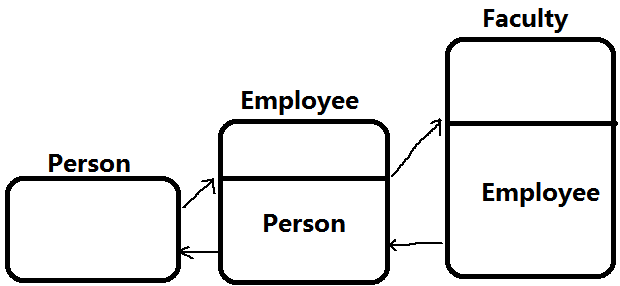
shallow copy 、 deep copy

复制构造函数，需用同类的引用，否则会无限递归

若存在自定义构造函数，则不生成默认的构造函数

析构函数为虚函数，为防止不完全释放

**Explicit** 构造函数的前置修饰，防止隐式类型转换



**初始化列表：Initialization list**

生成对象结构时，初始化操作。 父类的创建—> 初始化列表—> 构造函数

**静态成员：static member**

命名空间是类**，**只执行一次的作用域受限的全局变量**, 必须在类外初始化**

**提供同一类的所有对象之间信息交换的捷径，可被非静态成员访问，却不能访问非静态成员。**

**∴可以在类中定义 同类型的静态成员。Class A{static class A;}**

**因为static只有一份，执行一次。不会造成死循环**

**[必须类外初始化，int A::value = 3;]**

**外部声明extern**

表明此成员在其他文件中定义，不为其分配空间

（类的方法的默认属性）

**友元：friend**

敞开心扉做朋友，开放一切访问权限，友元权限不可继承

使用.（点操作符）访问类的成员.

but：这一定程度上破坏了类的封装性，需要谨慎。

**运算符重载：Operator Overload**

优先级和结合性不变，运算法则不变，只处理自定义类型（class），避免歧义。成员操作符，总是捆绑一个参数（对象的引用）

 C= a + b 即 c = a.operator+(b); 注意：只开重载一元和二元操作符

具体的运算规则及返回值，可参照sring类

**=，[]，()，->这四个操作符只能通过成员函数进行重载**

**继承**

**继承与派生：inheritance & derivation**

吸收了带有访问权限和作用域的模型，并适当改造；使其不断地 功能细化 和 内容扩大化。

**子类是功能扩展的父类**

单一继承（single-inheritance） ：一个派生类只有一个直接基类。

多重继承（multiple-inheritance） ：一个派生类可以同时有多个基类。

1除构造与析构外，吸收基类的成员

2 成员的个性化设计：同名覆盖基类成员，异名为新增成员

3 重写构造与析构函数

继承方式（物理上都继承，只是逻辑上设置特定的访问权限）：

公有继承为主流：保持public、protected不变，而private->不可访问（inaccessible） （私用 \ 保护继承： public、protected都成了 private \ protected）

【系统先执行基类的constructor再执行子类的constructor】

**虚继承：**

对基类地址或引用的继承

**多态**

定义了对父类的操作方法（函数），可适用于其子类中。

**在该类中查找函数，如果找不到，就在父类中查找**，如此循着类的继承链往上找，直到找到该函数或者查找完最后一个的类。**如果不能在类或其相关基类中找到该名字，则调用是错误的**。 //IMPORTANT

在同一个类中：重载、覆盖、隐藏：

函数名相同，形参不同： Overload （重载）

函数名相同，形参也相同：Redefine，不同命名空间的相同名字 （隐藏）

（Virtual） Override，虚函数表的特定元素的值（地址）被取代 （覆盖）

**多态：Polymorphism**

反映对象的个性，实现以共同的方法，但因个体差异而采用不同的策略。

编译时多态（静态绑定）表现形式->函数重载 Staic Binding

运行时多态（动态绑定）表现形式🡪虚函数 Dynamic Binding

**虚函数：virtual function （多态的实现方式）**

使用场合：该函数在继承后，需要根据个性重写, 虚属性对子类生效，并默认继承

C++只有标识为virtual的函数才会产生运行时多态的效果，**只能借助指针或者引用**来达到多态的效果。核心理念就是通过基类访问派生类定义的函数。

**【父类的引用 不会在子类中寻找调用， 而virtual相当于把子类的 同名函数的地址 覆盖基类中的那个，相当于函数的引用】**

纯虚函数的存在，造成了抽象类，不可实例化。作用：规范接口，实现留给子类.

（virtual void f() = 0; //规定必须是零）

**模板：template** GenericProgramming

以类型为参数的一个模型，一般 与容器（数据结构）相关时使用

**命名空间Namespace**

定义一个名字作用域，避免名字冲突，如：类、静态 等

using namespace X; //就是当前语句块中的下文指定默认的X命名空间

std::test::variable //使用全名

::value //在默认的命名空间中

//类似于Java的import，指明位置

**C++多文件组织**

头文件只用于声明，而不是定义。（const成员则需要定义）

对应的.cpp是实现，生成.obj

多文件连接

**函数对象FunctionObject**

定义operator()函数，之后对象名就像函数名一样使用。

**回调函数CallBack**

获取某函数的地址，需要时调用

**类型转换Casting：**

static\_cast 强制类型转换，总是成功。

dynamic\_cast 对存在多态继承关系的类的地址或引用，尝试类型转换。

‘ Success, return pointer or reference

‘ Failed, return NULL （类型检查，与虚函数配套使用）

Format : Dynamic\_cast<Derived\*>(pDerived)

**异常处理Exception Handling** C++中不要使用

当if 难以处理 和 错误信息不方便返回 时，异常处理机制起作用了。

Try //消耗更多的时间和空间

{

Throw runtime\_error(“fuck”);

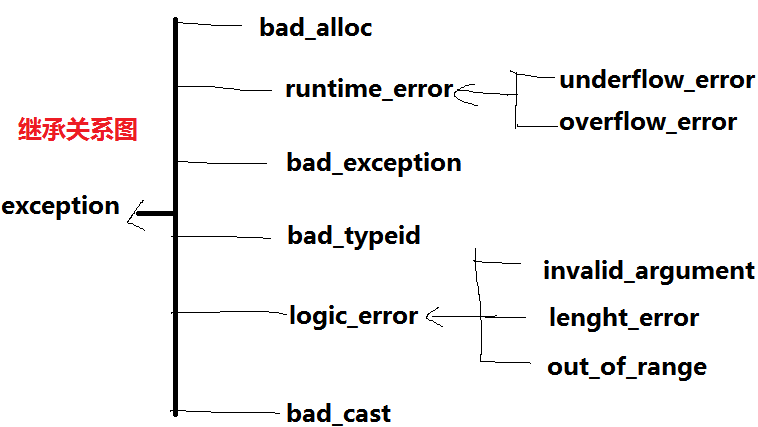
}

Catch

{

//code

}



#include <Stdexcept>

特定类型：exception，方便识别处理, +what()