阅读C++ prime plus 第13 章并总结关键内容

1. 类继承

类库由类声明和实现构成。

类继承能够从已有的类派生出新的类，而派生类继承了原有类（称为基类）的特征，包括方法。

1. 一个简单的基类
2. 从一个类派生出另一个类时，原始类称为基类，继承类称为派生类。

派生类有着特殊的声名头，例如：

class RatedPlayer:public TableTennisPlayer

{

...

};//冒号指出RatedPlayer类的基类是TableTennisplayer类。

Ratedplayer对象将具有以下特征： 派生类对象存储了基类的数据成员，即派生类继承了基类的实现； 派生类对象可以使用基类的方法，即派生类继承了基类的接口。

此外，派生类需要自己的构造函数。

1. 构造函数：访问权限的考虑

派生类必须通过基类方法访问基类的私有成员，而不能直接访问。

派生类构造函数举例：

RatedPlayer::RatedPlayer(unsigned int r,const string & fn,const string & ln,bool,ht):TableTennisPlayer(fn,ln,ht)

{

rating=r;

}

其中:TableTennisPlayer(fn,ln,ht)是成员初始化列表,是可执行的代码，调用TableTennisPlayer构造函数,这样派生类RatedPlayer就可以访问基类中的值。

派生类构造函数的要点： 首先创建基类对象；派生类构造函数应通过成员初始化列表将基类信息传递给基类构造函数；派生类构造函数应初始化派生类新增的数据成员。

1. 继承：is-a关系

C++有3种继承方式：公有继承、保护继承和私有继承。其中最常用的方式是公有继承，它建立一种is-a(is-a-kind-of)关系：派生类对象也是一个基类对象，可对基类对象执行的任何操作，也可对派生类对象执行。

1. 多态公有继承

多态：具有多种形态，即同一个方法的行为随上下文而异。

用于实现多态公有继承的机制：在派生类中重新定义基类的方法，或使用虚方法。

注意正确使用作用域解析运算符（::）。

使用虚析构函数，以确保正确的析构函数序列被调用。

1. 静态联编和动态联编

函数名联编：将源代码中的函数调用解释为执行特定的函数代码块。

静态联编：在编译过程中进行联编，又称为早期联编。

动态联编：编译器生成能够在程序运行时选择正确的虚方法的代码，又称为晚期联编。

向上强制转换：将派生类引用或指针转换为基类引用或指针，这使公有继承不需要进行显式类型转换。

向下强制转换：将基类指针或引用转换为派生类指针或引用。

1. 访问控制：protected

关键字protected与private相似，在类外只能用公有类成员来访问protected中的类成员。然而，派生类的成员可以直接访问基类的保 护成员，却不能直接访问基类的私有成员。

1. 抽象基类（abstract base class，ABC）

在原型中使用=0指出类是一个抽象基类，在类中可以不定义该函数。

比如：

class BaseEllispe

{

private:

...

public:

BaseEllispe(double x0=0,double y0=0):x(x0),y(y0){}

...

virtual double Area() const=0;

...

}

定义纯虚方法不是必须的。包含纯虚成员的类不能使用它来创建对象。纯虚方法用于定义派生类的通用接口。

1. 复习类方法

注意：使用const来确保方法不修改参数，例如

Star::Star(const char \* s){...}

还可以使用const来确保方法不修改调用它的对象，例如

void Star::show()const{...}