FFMPEPEG 音频开发 c语言开发

C++学习不仅要学习语法，更多地是学习框架

Cin与Cout是类对象，而C语言中的printf与scanf是函数

Cin<<a<<b<<c; 等价于 Cin<<a; cin<<b; cin<<c; 含义就是键盘把数据放到a、b、c中

>> 表示流输入运算符 << 表示流输出运算符

C++ 中string类型大部分时候使取代char \*的

函数重载，会出现重名的函数，重名的函数会根据语境来决定调用，运算符重载也是一种函数重载。C++利用倾轧技术(name mangling)来改变函数名，区分不同的同名函数。Name mangling 只发生在两个阶段，.cpp编译阶段和.h的声明阶段。

重载的规则：

1. 函数名相同，参数的类型，个数，顺序不同，均可以构成重载
2. 函数返回值的类型不能作为构成重载的标志
3. Ambiguous 二义性（模糊不清的）

重载函数的匹配规则：

1. 严格匹配，匹配成功，则调用
2. 严格匹配不满足，寻求是否有隐式转换。隐式转换成功，则调用

命名倾轧是依据函数声明来进行倾轧的，若声明被倾轧，则调用为倾轧版本，若声明为非倾轧版本，调用也为非倾轧版本。

C++编译都是以文件为单位进行编译的。C++默认所有函数倾轧。

C++完全兼容C，如果C++引用C的库，希望不受命名倾轧的困扰，那么可以在main.c或者其他引用到这个库文件的地方加extern “C”，或者将头文件和.c文件全部用extern “C”{}包含起来。

C++默认参数获取：

规则：①默认参数赋值顺序从右往左，不能跳跃

1. 实参传递的时候顺序是从左往右的，不能跳跃
2. 实参的个数加上有默认值的形参的个数要大于等于形参的个数
3. 声明和实现在一起的时候(就是在同一个CPP文件中，而不分开)，默认；若声明（.h）和实现(.c)不在一起，默认参数应该出现在声明(.h)中
4. 默认值可以是常数、全局变量、或者是一个函数。
5. 当实现同一功能，可以使用含有默认参数的函数，又可以使用重载函数的时候，推荐使用含有默认参数的函数。

默认参数有时候会与重载冲突，当实现同一个功能的时候，既可以用默认参数，又可以用重载，推荐使用默认参数(构造器可以重载也可以默认参数，优先选用默认参数的方案)

变量名本身是一段内存的引用，即这段内存的别名

& 引用是一种声明关系，不开辟新的内存空间，必须要初始化，不能单独存在。与被别名的变量拥有相同的数据类型。声明关系一旦确定，那么不可变更。可以对引用，再次引用（但是不能建立引用的引用），也就是对一个变量，可以建立多个引用，引用之间是一种平行关系。

& 取地址、按位与、引用 ，能辨别作用即可

传引用，等价于传作用域，把一个变量以引用的方式传到另一个作用域，就等价于扩展了该变量的作用域。

引用的本质（引用必须初始化，是个常指针，类型是int \* const的指针），是对指针的包装，避免了使用裸露的指针，对于引用的指针类型，C++避免对引用再次解封。解封：int & \*(指针)rpa(引用) = &ra; 封装：int & ra(引用) = a;

int a; int \*p = &a; //将放有a内容的内存地址赋值给p

int \*(首先是指针) &pr = p ;//将指针p进行引用(指针的引用)，int \*是指针类型 &代表引用 pr为p的别名，pr就是对指针p的引用

//对引用的解封，本质代表了引用的指针是不合法的。

这就是指针的引用 有； 引用的指针 无

引用的数组：int & pArray[] = [ra,rb,rc]; //本质上pArray是ra(引用)的指针，所以不合法。

任意指针类型的变量/常量可以赋值给void\*类型的变量

临时对象：不可取地址的对象

引用的本质是常指针，必须初始化，一经声明，不可改变。

Semantic : 语义

C++ 是否所有的表达式都可以被赋值？

宏函数的优点：内嵌，避免函数调用的开销 缺点：替换、代码体积大

函数的优点：抽象，实现一次书写，n次调用，类型检查 缺点：调用开销、

使用Inline的时候，函数声明与函数源码，最好都要加(短小而频繁使用的函数才有可能会使用，可能被编译器优化掉)

Const\_cast只作用于指针与引用，去const化

Const\_cast理解为，const semantic的补充

非const对象—>const引用/指针-->去const->原来非const 对象修改

//const 对象可不可以去对象化？

Static\_const<要转换类型>() ：①双向隐式转换可以使用 ②单项隐式可以使用

Reinterpret\_cast<要转换类型>() ：①双向不能隐式转换

C++中的命名空间namespace，就是将无名的全局命名空间再次划分的结果。

同名命名空间自动合并

String 关键字 表示字符串类型

C++申请堆上空间顺序是从外到内，释放空间的时候是从内到外

入/压栈(PUSH)（栈顶指针）始终指向一个待压入的指针，出栈(POP)指针始终指向一个待出栈指针的上一个地址

假设有0-1023 （入栈的指针push top = 0 ：栈底） （top = 1024 : 栈顶）

入栈首先将数据push，然后再将指针向上移动，出栈先将指针向下移动，然后在将数据取出。

C++封装，包含两部分，一部分数据。一部分行为（接口）

C++封装 对外提供接口，隐藏数据，对内，数据开放

C语言struct 默认全部是public class类名 默认全是private

class类名 本质也是一个命名空间 (FILO先进后出 LIFO 后进先出)

在(class)类外实现成员函数，为了实现xxx.h,xxx.cpp 不能定义为全局函数，要进行类名限定如 类名:: 进行限定

句柄：FILE\* sqlite3\*

链表的操作句柄 Node \* head (头节点)

操作链表的本质是操作其句柄

构造器 constructor：1、与类名相同，在对象生成的时候，自动调用 2、无返回值 可以有参数 3、可以有默认参数，可以被重载 4、系统提供默认无参空体的构造器，一经自实现，系统提供的默认不复存在 5、无论重载还是默认参数，应该把无参空体构造器包含进来

生成无参的对象，是一种比较常见的对象(对象数组)

初始化列表完成初始化 是 初始化 （数据成员的）

初始化列表初始化的顺序，跟列表中的顺序无关，和private属性的声明顺序有关

析构器 ：1、与类名同，无参，无返回值，前面加一个~ 2、在对象被销毁前被调用，用来处理清理工作（堆对象） 3、在对象被销毁前，适用于栈上的对象和对象的对象 4、系统提供一个空体析构器

析构器是不是必须的？

在没有堆空间对象中，默认的就好，在对象有堆空间的情形下，必须要用。

面向过程: 满是函数 面向对象：满是类(class)

Int a =0; //初始化 a = 5；//赋值

由普通数值，做参数完成的构造，constructor

由同类对象，做参数完成构造，copy constructor

Copy constructor由同类对象，完成参数构造；1、系统提供了默认的拷贝构造器，格式固定，一经实现，不复存在 2、此拷贝构造器不是空体的，提供了一个等位拷贝机制 3、系统提供的默认拷贝构造器，是一种浅拷贝：如果对象中不含有堆上的空间（指针指向的空间），此时浅拷贝可以满足需求，不需要自实现。浅拷贝会造成重析构，double free。 4、深拷贝：如果对象中含有堆上的空间，此时浅拷贝不能满足需求，需要自实现。

同类对象方法中，进行传参，可以访问其私有成员，其他则不可以

Copy constructor主要用于 传参 与 返回

This指针作为函数参数隐式传入，不占用对象体积大小，避免形参与数据成员重名，可以实现链式表达，串联a=b=c

系统提供默认的赋值运算符重载，默认是一种浅赋值行为；如果对象中，不存在由\*，构成的堆空间，此时默认也是可以满足需求的；格式比较固定，A& operator=(const A& another)，自实现的话，默认即不存在；如果对象当中存在，\*构成的堆空间，

一个对象所占的空间的大小只取决于该对象中的数据成员所占的空间，而与成员函数无关。

类的成员函数只有一份，在对象的调用过程中，传入对象的地址，具体可以表现为this指针。

对象数组中，有100个对象，就会发生100次构造，

优化是gcc g++的参数，如何避免？

This指针发生在对象创建之后

Const 修饰，类数据成员，必须要初始化

Const初始化，一种情况 ①在类内 ②在初始化列表 initial list

初始化列表，一方面可以提高效率，为了一些新扩展的功能，提供解决办法

Const修饰的数据成员，可以在非const函数中使用，到那时不可以更改

Const修饰，类成员函数，不可以修饰全局函数

Const修饰函数以后，承诺，在本函数不改变，在本函数不会发生，改变数据成员的行为

只能调用const成员函数

Const构成的重载函数，非const对象，优先调用非const版本

Const对象，只能调用const版本，很多库常见提供两个版本

Const修饰的对象，其内可以有非const数据成员，但不可修改。只能调用const成员函数。

针对const有可能修饰对象，往往提供两个版本，构成重载。

Static 修饰数据成员，需要初始化，不可以类内初始化

类外初始化，需要类名空间，且不需要加static

类的声明与实现分开的时候，初始化，在.h,.cpp? 在.cpp中

大小，static声明的数据成员，不占用类对象的大小，储存在data段的rw段

命名空间，即类名，将声明与定义联系在一起

访问，既可以通过对象进行访问，也可以不通过对象，直接通过类名访问。

Static修饰成员函数，主要用于管理静态变量

类内外，声明与定义

静态成员函数，只能访问静态的成员(数据成员和函数成员)，不能访问非静态的成员

是因为静态函数成员中没有this指针

单例模式主要用于实现共享

前向声明，只能定义指针和引用，做参数与返回值，仅用在函数声明中。

友元类(friend class)，实际工作中，一般都是用友元类

运算符重载的本质就是函数重载，函数名就是operator XXX

友元重载(即全局函数重载)：但凡涉及到自定义类型，想要跟普通类型一样的操作体验，就需要重载

成员函数重载：

Protect在外部访问的时候等价于private，在public子类继承中，可见

默认的继承方式是private，

继承方式不影响派生类成员的访问方式，影响了父类成员，在子类的内部和对象的访问权限

父类 Public继承 public protected private

子类 public protected 不可见

标配：①空题无参 ②重载 ③ 默认参数

在子类构造过程中(先内嵌子对象，然后子类)，会调用父类的构造器，(在无参以及默认参数的时候会被自动调用)，标配会被调用，没有标配，必须显示地调用，或隐式调用(必须要有标配)，或显式调用

调用内嵌子对象的构造器，如有标配，则调用，若没有，则必须显示调用。

三角转四角的问题：1.提取各父类成员中相同的成员，构成祖父类 2.让各父类继承祖父类 3.虚继承是一种继承的扩展(加virtual) 4.虚继承也是一种设计的结果

虚基类，被虚继承的类就是虚基类

多态实现的前提——赋值兼容

纯虚函数，virtual type xxx() = 0;

含有纯虚函数的累被称为抽象基类，抽象基类以及抽象基类的派生类析构的时候在析构函数前面同意加virtual