

目 录

2024秋季 c++ 辜希武	9
第二章	9
第一个出题点	9
经典错误:	9
列表初始化{} 和括号初始化	9
const constexpr和volatile	9
常量表达式	10
inline 内联符号 只局限于当前代码	10
常数的指针 和 指针是常数	10
泛型指针	10
引用	10
左右值	11
怎么读&这个符号呢? &+变量: 变量是一个引用, 这个引用指向一个xxx后面和运算顺序一样	12
引用初始化	12
而const引用则是万金油, 可以用类型相同(如类型不同, 看编译器)的左值表达式和右值表达式来初始化。	12
c++ 里面一个变量的地址是右值 没有持久的储存地址 你就考虑整体可不可以放在左边!!!	13
自学的引用指针 const翻译成只读的 接后面那个词	13
引用初始化	13
nt &可以赋值给const int &, const int &不能赋值给int &	13
右值引用	13
【】数组优先级最高	14
枚举	14
数组	15
constexpr	15
第三章	15
switch	15
inline	15
函数体可以先声明再定义	16
静态变量和函数也只有当前程序文件可以使用	16
main函数的参数	16
省略参数...	16
函数默认参数	16
重载函数	17
生命期	17

第四章 类	17
构造函数的定义和数据成员	19
构造函数和析构函数	19
构造函数和析构函数的使用	19
那如何使用构造函数呢?	19
析构函数的使用细节	20
析构函数在变量（对象）的生命期结束时被自动调用一次，通过new产生的对 象需要用delete手动释放（自动调用析构）在heap里面	20
析构函数 是倒着析构的	20
上面这个代码有几个地方需要注意！首先是析构函数里面的那个s 那个s是在 类里面的变量 所以使用的时候直接就可以用了的 还有一个MYSTRING ("Constant\n")这个没有定义一个变量 所以其实是一个常量 记住类是重定 义数据类型	21
变量？对象?	21
如何使用定义的类的成员呢?	21
this	21
复习课堂上讲过，int * = const int *是不成立的，换成char类型是一样的	21
退出（不重要 ppt都隐藏了）.....	21
接受和删除自动生成的函数：delete和default	22
成员类型	22
怎么理解这个“访问”？其实很简单，就是有没有出现	22
类的内联	22
什么是函数局部类？就是在函数里面定义类	22
new和delete	23
怎么用?	23
free, malloc和delete, new的区别	23
new还可以重新构造已经析构的对象 节约栈的空间?	24
this指针	24
最重要的this指针是非静态函数成员，包括析构和构造的隐含的第一个参数，类 型是函数成员所属对象的指针常量：A*const this	24
小tip函数签名const	24
初始化	24
1. 如果未定义或生成构造函数，且类的成员都是公有的，则可以用{}的形式初始 化（和 C 的结构初始化一样）.....	25
示例:	26
2. 联合仅需初始化第一个成员	26
示例:	26
3. 对象数组的每个元素都必须初始化，默认采用无参构造函数初始化	26

示例:	27
4. 单个参数的构造函数能自动转换单个实参值成为对象	27
示例:	27
5. 如果类未自定义构造函数, 编译器会尝试自动生成构造函数	27
示例:	28
6. 一旦自定义构造函数, 将不能接受编译生成的构造函数, 除非用 =default 接受	28
示例:	28
7. 用常量对象做实参, 总是优先调用参数为 && 类型的构造函数; 用变量等做实参, 总是优先调用参数为 & 类型的构造函数 (结合法则4来看)	28
示例:	29
默认构造函数	29
1) 如果为数据成员提供了类内初始值, 则工作成功	29
2) 如果没有为数据成员提供类内初始值: 和int之类的一样 类的对象在全局会为0 在内部会随机值报错	29
A) 非 const、非引用内置类型数据成员:	30
B) 包含 const 或引用成员: 引用和const都是常量类型 必须定义及时初始化 这里也和int一样写了const不初始化直接报错	30
C) 包含其他类类型成员且该类型没有默认构造函数:	31
对于const、引用成员, 没有默认构造函数的对象成员, 只能就地初始化和在成员初始化列表里初始化, 不能在构造函数体内被赋值。.....	31
类的构造实际上和int的一模一样, 只不过有隐形转化, 把一些常量从int变成了类的类型, 你看起来就像是一个int给类里面的一个int赋值, 实际上不是的, 实际上是一个int变成了类的类型, 再赋值给整个类 构造函数的意思 实际上描述的就是如何构造这个int变成一个类的方法 当我们用等号去初始化的时候 就是隐形的构造 当我们不用等号而是用花括号和中括号的时候 实际上也是像上面的理解 把数字转化为对应的类的类型去赋值 只不过等号有一个明显的类型转化罢了 需要注意的是 直接用等号的构造函数只能用于单参数构造函数有效 原因很简单 多参数怎么使用等号?	31
explicit 定义构造函数类型 抑制类型转化的	32
移动构造函数的noexcept声明只对容器扩容时产生影响	32
注意 虽然等号模式的隐形构造方法只能用于一个参数的构造 但是要是只有一个参数没有默认化 也是可以的	33
坑坑的	33
拷贝构造函数	33
用一个已经构造好对象去构造另外一个对象时会调用拷贝构造函数	34
拷贝构造函数一般在传递值参和返回值参各自一次 分别解释为: 1把对象作为实参传递给非引用形参 2返回类型为非引用类型的函数返回一个对象 拷贝给返回处 引用类型的时候则都是一个绑定的关系 就是他本身 所以不用拷贝构造 看下面的例子:	34

拷贝构造函数的内容	34
浅拷贝：按成员依次拷贝 函数参数为值参而把实参传给值参 不是引用	35
深拷贝：在传递实参给形参的时候，把形参对象的指针成员分配新空间，将实参成员的指针所指向的单元拷贝过去 参数必须是类的引用，推荐用const A& 而且要自己写这个函数	35
与等号的重载联动：	35
为什么拷贝构造函数一定类型是引用？	36
移动构造和移动赋值	36
先来个四种函数的模板	36
这个例子也说明了移动构造函数存在的意义若没有移动构造，	38
根据上面的代码看看下面的题：	38
移动构造移动赋值和vector的关系	39
如果右值没有移动构造？	39
类里面有默认函数：	40
sum up：对于类型为A且内部有指针的类，应自定义A()、A(A&&) noexcept 、A(const A&)、A& operator=(const A&)、A& operator=(A&&) noexcept以及~A()函数。	41
当A&&作为参数的时候 他是左值 放在寄存器里面，有自己的地址	41
第六章 继承和构造 访问继承的类取决于继承类的访问权限 和基类无关	41
单继承类：只能获取一个基类的属性和行为	42
区分	42
多继承	43
在控制派生类的时候：	43
什么是实例函数成员？就是控制和修改类的非静态数据，然后只能用类的实例访问	44
特点：	44
示例：	44
默认类型类继承	44
什么是合理的类继承？	44
如何强制修改访问权限	44
恢复访问权限	45
继承其他注意的点：	45
为了访问基类的友元化：	46
派生类友元化：	46
派生函数友元化：	46
构造和析构	47
构造：	47
必须有派生类自己定义的构造函数：	47
析构是构造倒着来	48

父类和子类	48
子类不能指向父类直接 子类指向父类要强制转化而且父类的要是实例的	50
派生类对象内部基类指针	50
第八章 虚函数 多态（这一章最容易错的：派生类构造先要基类构造）.....	50
虚函数：即用virtual定义的实例成员函数 基类对象指针(引用)指向(引用)不同类型 派生类对象时，通过虚函数到基类或派生类中同名函数的映射实现(动态)多态。虚函 数内联是失败的	50
多态：每个子类对象都是父类的实例，如 <code>Person *p = new Teacher();</code> 这是基础	50
下面这个例子：说明了和原型的虚函数定义一样的时候会自动变虚函数	51
注意点	52
ppt的巨大错误！虚函数的默认是继承的结果，不是父类子类的结果，一个类能用虚函 数访问，一定是最基类的指针，不是父类！！！！但是，只有父类不用类型转化	52
特殊：如果基类的析构函数是虚析构函数 则派生类的析构函数就会自动成为虚析 构函数（即使原型不同，即函数名不一样） 动态内存一定要虚析构函数	54
类的引用问题：.....	54
详细分析下面的代码：.....	54
小tips：传入形参的临时变量在函数结束后才析构	55
抽象类：含有纯虚函数的类，没有对象或者类实例	55
一旦派生类继承了抽象类的纯虚函数但是没有重定义虚函数的函数内容，或者定义了 新的纯虚函数，那么这就是新的抽象类 请看下面代码的注释：.....	55
友元没有传递性	56
第九章 异常断言（不重要）.....	56
throw语句：.....	56
具体处理	57
异常处理	58
处理顺序	58
断言	59
第五章 成员和成员指针	59
实例成员指针：.....	59
运算符	59
const只读、volatile易变和mutable机动（只需要在声明的时候带上）.....	60
强制类型改变	61
mutable 等级高于const 只要有mutable就可以修改	61
全军复诵！！！！.....	62
2. 函数的引用参数在调用时初始化	62
3. 非const左值引用变量和参数必须用同类型的左值表达式初始化	62

例子：函数参数的左值引用与传递的实参类型必须匹配	63
static	63
函数里面的局部类不能定义静态数据成员	64
静态函数成员没有this参数，而构造函数和析构函数都有this指针（在类里面，我们 无法用this指向构造函数和构造函数），虚函数，纯虚函数也有this，都不能是静态 函数	65
静态成员指针	65
静态成员指针（不管指向静态数据成员还是指向静态函数成员）就是普通指针， 因此如果申明指向类的静态成员的指针，就用普通指针。.....	65
普通函数成员指针	65
第七章 可访问性	65
作用域：： 优先级最高	65
单目：.....	65
具体意思	66
示例：同名标识符和类名冲突的情况	66
1. 类名与局部变量重名	66
2. 使用 class 限定符来指定类名	66
3. 使用 struct 或 union 限定符	67
双目：.....	67
作用域	68
名字空间	68
三种访问形式	69
1. 直接访问成员	69
2. 引用成员	69
3. 引用整个命名空间	70
4. 多个命名空间成员同名时的处理	70
using 关键字	71
using + 名词空间变量名	71
using+空间	71
避免冲突	72
嵌套名词空间	72
匿名名词空间	72
成员友元 友元关系不可以传递也不可以交换	73
全部函数均为友元 友元类	73
第十一章	73
运算符重载	73
不能为全局函数和静态函数重载的原因：.....	74
为什么赋值运算符不能作为全局函数重载？	74

为什么其他运算符不能作为静态重载?	74
重载运算符参数的位置:	75
1. 二元运算符 (两个操作数)	75
例子: 加法运算符重载 (operator+)	75
区别:	75
2. 赋值运算符 (operator=) 左值操作符, 最好返回非只读左值, 不能用const去 声明一个左值参数	76
3. 前置递增运算符 (++)	76
4. 下标运算符 ([])	76
重载运算符的类型	77
1. 重载运算符函数可以声明为类的友元 (Friend Functions)	77
2. 重载运算符的普通成员函数也可定义为虚函数	77
3. 重载运算符的非成员函数被视为普通函数	78
重载运算符一般不能却省参数, () 表示省略参数, 转化为	79
参数表	79
前后置++和--	79
后置 (返回值) 双目	79
前置单目	79
例子:	79
重载->	80
下面这个例子说明->重载之后做操作数不是对象指针, 右操作数不是对象成 员	80
下面的例子 是多个重载运算符造成的重名和使用方法	80
为什么 friend 函数不是成员函数?	81
() 的重载	81
赋值 调用	81
为了防止内存泄漏	82
实例	82
运算符重载强制类型	83
单参数构造函数	83
重载new和delete	83
模板与内存回收	84
例子:	84
全军背诵!	85
函数模板可省略性	86
特化: 其实就是定义函数 可以加多一句template<>而已	88
类模板	88
类模板中的多个类型形参的顺序变化对类模板的定义没有影响	89

实例化类模板	90
特化类： 样式和模板实例一样	90
特化函数成员	92
实例函数可以成为类的友元函数	92
定义派生类B时，自动生成实例类A作为B的父类 但是要在构造B的时候手动构造构造父类的构造函数 其实这里的规矩和前面上面父类子类上面时候要子类显示构造一样 如果是默认的无参构造函数不要手动 可以自动	92
实例成员指针	92
区分	93
定义用两个栈模拟一个队列的类模板	93
第十章 多继承类和虚基类	94
代理模式实现多继承	94
定义方式	95
虚基类	95
若虚基类的构造函数都有参数，必须在派生类构造函数的初始化列表中列出虚基类的构造实参的值（注意：使用虚基类继承的时候要声明基类达到virtual 在派生类的构造函数里面，不加virtual ）	96
为什么 LandVehicle 和 WaterVehicle 中调用 Engine(p) 无效?	97
非虚类派生	98
虚基类	98
最终派生类：在实例化的时候截止的到的类 都是最终派生类 在一个最终派生类所在的继承链上只有他初始化虚基类 之前的类都没有初始化	98
同名:	98
内存布局	100
构造顺序	101
这个构造函数体是什么意思?	101
解释和什么是派生树?	101
1. 派生类、基类和虚基类构成的派生树	101
2. 虚继承和虚基类	102
3. 对象成员成为新派生树的根	102
例子:	102
分析:	102
ppt例子	103
好好琢磨下面的例子!!!	104