一、this指针

1、概念：类非静态成员函数的第一个隐藏的参数，该参数使用指向调用当前函数的对象

2、特性：

a、this指针类型：T* const

普通类型成员函数：T* const 可以修改对象的内容，可以调用普通和const类型的成员函数

const类型成员函数：const T* const：this指向不能修改并且指向对象中的内容也不能修改,只能

调用const类型的成员函数

b、this是非静态成员函数的第一个隐藏参数，隐藏：用户在编写函数时不用给出this的参数，该参

数是编译器自己维护，调用该函数也不需要手动传递，this指针的传参也是编译器自己进行

c、this指针只存在于正在运行的成员函数中，this指针不会存在于对象中，不会影响类对象的大小

d、静态成员函数：没有this指针

e、this指针的传递：一般情况exc寄存器（this_call:调用约定),也可能通过参数压栈的方式进行传

递：push 对象地址(比如:类中如果包含不定参数的成员函数)

面试题：你都知道有哪些调用约定 _cdecl this_call

3、this指针是否可以为NULL

a、如果成员函数是通过对象的方式进行调用，this指针一定不会为NULL：T t; t.TestFun();

b、如果成员函数是通过类类型的指针方式进行调用,this指针可能会为NULL:

T t; T* pt=&t; pt->TestFun(); pt=nullptr;pt->TestFun();

pt=nullptr;pt->TestFun();代码可以通过编译，但是在运行期间可能会出现错误：如果在成员函

数中没有访问非静态的成员变量，一定不会奔溃，如果访问任何的非静态成员变量，代码一定

会崩溃

4、this指针位置：栈

类的编译过程：

1、识别类名

2、识别类中的成员

3、识别类中的成员函数，并对成员函数进行改写

>>给非静态的成员函数增加隐藏的this指针参数

>>对成员函数中每个访问非静态成员变量前增加this

面试题：

```
class A                                  1、没有任何语法问题，编译通过
{                                        2、运行时：代码会崩溃
public:
    ~A()     对象在销毁时，要调用析构函数
    {
        delete this;
    }                        delete：1、调用析构函数，清理对象中的资源
};                                       2、释放指针所指向的堆空间
      无限递归，造成栈溢出而导致代码崩溃
```

二、类中六个默认的成员函数

1、什么是默认的成员函数：如果用户没有显示定义该成员函数，编译器将会自动生成一个

2、默认的成员函数都有那些?

a、构造函数

1、概念：是一个特殊的成员函数,函数名必须与类名相同,在创建对象期间由编译器自动来调用,已

完成对象的初始化工作,并且在对象的整个生命周期内只调用一次。(因此初始化只需要初始化一

次)

2、特性

1>函数名必须与类名相同

2>没有返回值

3>由编译器自动调用

4>构造函数可以重载（用户根据自己的实际情况）

5>如果用户没有显示定义,编译器将会生成一个默认的构造函数(注意:默认的构造函数是无参的)

6>将无参的构造函数和全缺省的构造函数统称为缺省的构造函数，并且他们不能同时存在

7>构造函数不能使用const进行修饰，构造函数是要修改类的成员变量

8>构造函数不能是静态的成员函数

9>构造函数不能使用virtual来进行修饰

10>构造函数具有初始化列表

a、语法

b、注意

>>只有构造函数具有初始化列表

>>初始化列表中只能初始化类中的非静态成员变量

>>成员变量在初始化列表中不能重复出现，只能初始化一次

>>成员变量在初始化列表中的初始化次序：初始化次序与成员变量在初始化列表中出现

的次序无关，初始化次序与其在类中声明次序一致

>>尽量避免采用成员来初始化成员

c、必须要在初始化列表中初始化的成员

>>const类型的成员变量

>>引用类型的成员

>>如果包含类类型的对象，并且该对象所对应的类包含非缺省的构造函数

如果继承位置的知识点掌握没有问题：也可以将构造函数在继承中的一些规则讲一下

3、编译器生成的默认构造函数

a、用户没有显示定义构造函数的前提下,编译器才会生成,只要用户定义(不论构造函数是什么形

式的),编译器不在生成

b、编译器生成的默认构造函数一定是无参的

4、调用场景

在创建新对象时(注意：不是用类类型对象创建对象)，编译器会调用构造函数

b、拷贝构造函数

1、概念：是一个特殊的构造函数，一定是单参，并且该参数一定是类类型对象的引用，在用类

类型对象构造新对象时由编译器自动进行调用。

2、特性

1>单参的，参数必须要是类类型对象的引用

否则就会造成编译失败，为什么编译失败：可能会造成无限递归

2>如果用户没有演示定义，编译器会生成一个默认的拷贝构造函数

注意：编译器生成的默认拷贝构造函数是浅拷贝

如果类中没有包含资源管理，比如Date类，用户是否给出拷贝构造函数都可以

如果类中涉及到资源管理，比如：string类，用户必须显示的给出拷贝构造函数的定义----

一般都是按照深拷贝的方式---99%，例外：比如智能指针

3、默认拷贝构造函数

4、调用场景：用类类型对象创建新对象时,A a1; A a2(a1);以值的方式传参||以值的方式作为函数

的返回值

c、赋值运算符重载

1、如何定义

```
T& operator=(const T& t)
{
    if(this !=&t)
    {赋值操作}
    return *this;
}
```

考点：

1、有没有返回值，为什么？如果有该怎么返回？

2、参数的类型？注意：一般情况都需要加const，个别情况例外：auto_ptr

3、有没有检测是否是自己给自己赋值

4、返回值的内容 注意：最好返回值*this 原因：为了和连续赋值的规则保持一致

2、关于编译器生成的默认赋值运算符的重载

>>如果用户没有定义，编译器将会生成一份默认的赋值运算符重载

>>编译器生成的赋值运算符重载是：浅拷贝

如果类中涉及到对象资源的管理时,需要用户自己显式定义赋值运算符重载--深拷贝

比如：string

3、调用场景

a1=a2;

运算符重载

1、注意函数重载进行区分---运算符重载是一个函数，而函数重载是一些在相同作用域具有相同

名称但参数列表不同的函数

概念：运算符重载是具有特殊函数名的函数，也具有其返回值类 型，函数名字以及参数列表，其

返回值类型与参数列表与普通的函数类似

2、作用:可以提高代码的可读性

3、注意：

>>那些运算符可以重载？那些不可以重载？-----选择题方式

>>不能重载一些其他的运算符，比如：@

>>重载的运算符：一定要符合该运算符的含义

>>运算符重载可以是类的成员函数，如果重载成类的成员函数，参数个数会少一个，因为有

默认this指针

>>常见运算符的重载：

前置++/后置++（前置--/后置--）;cout的重载；[]；

operator*()/operator->()：比如智能指针、迭代器；operator()(参数列表)----->仿函数

d、析构函数

1、概念：一个特殊的成员函数,在对象销毁时,由编译器自动进行调用，完成对象中资源的清理工作

注意：只是清理对象中管理的资源，并不会回收对象的空间

对象如果是堆上的，对象的空间有delete或者delete[]来进行回收

如果是栈上的对象

2、特性：

>>析构函数的名字：类名前添加~

>>用户如果没有显式定义，将由编译器自动生成

编译器生成的析构函数是不会自动释放对象中资源，因此如果类中涉及到资源管理，需要

用户显式给出析构函数的定义

>>析构函数一定是无参，没有返回值：不能重载，一个类最多只能有一个析构函数

3、编译器生成的默认的析构函数

4、调用场景：在对象销毁时---->函数结束时调用delete和delete[]

e、operator&()和operator&()const

三、const关键字*****

1、在C语言中：const只能修饰变量（局部变量、函数形参、函数的返回值等),将其称作为一个不能

被修改的变量

1、不能修改：直接赋值

2、是否为变量：const int count=10；int array[count];

2、在C++中

a、使用const修饰变量

非类成员变量：局部变量，全局变量，函数参数，函数的返回值----const修饰的变量，不能

被修改，该变量已经是一个常量，还具有宏替换的属性

>>是一个常量

>>具有宏替换的属性----该替换发生在编译期间：该常量就会参与类型检测,

提高代码的安全性

类成员变量：const类型的成员变量

>>初始化：只能在构造函数||拷贝构造函数初始化列表中初始化

b、使用const修饰类非静态的成员函数：const修饰成员函数，其出现位置一定在函数原型之后

实际：const本质修饰的是该成员函数的this指针，表明在该成员函数中不能修改类中成员变

量（例外：如果类中某个成员变量采用mutable修饰）

问题：

const成员函数中：不可以调用普通类型的成员函数，可以调用const类型的成员函数

普通成员函数中：可以调用普通类型的成员函数，可以调用const类型的成员函数

volatile关键字

四、static关键字*****

1、static在C语言中的作用

a、static修饰变量：

初始化

static修饰变量的存储位置

修饰函数中的局部变量：延长该变量的生命周期、具有记忆功能：会记录上次函数运行结束

之后的内容

修饰全局变量：改变该变量的链接属性----只能在当前文件中进行访问

b、static修饰函数：表明该函数只能在当前文件中进行访问

2、static在C++中的作用

a、static修饰变量：

普通、全局与C语言类似

类成员变量-----静态成员变量&普通成员变量的区别

特性：

1>静态成员变量是类的属性，不属于某个具体的对象

2>存储位置：静态区---不会存在于对象中，不会影响sizeof的结果

3>访问：类名::静态成员变量名字||对象.静态成员变量名字

4>初始化：不能放在构造函数初始化列表中初始化，在类中只是声明，必须在类外进行初始化

b、static修饰函数：普通函数----与C语言中作用相似

类成员变量-----静态成员变量&普通成员变量的区别

特性：

1>没有this指针

2>不能直接访问非静态成员函数

3>不能直接调用非静态成员函数

4>不能用const修饰、virtual

5>调用方式：类名::静态成员函数名字||对象.静态成员函数名字

extern

五、C++中动态内存管理：

1、C/C++程序运行时：内存分布情况---必须是能够画出图解****

a、 栈又叫堆栈，非静态局部变量/函数参数/返回值等等，栈是向下增长的。

b、内存映射段是高效的I/O映射方式，用于装载一个共享的动态内存库。用户可使用系统接口创建共享共 享内存，做进程间通信。（Linux课程如果没学到这块，现在只需要了解一下）

c、. 堆用于程序运行时动态内存分配，堆是可以上增长的。

d、数据段--存储全局数据和静态数据。

e、代码段--可执行的代码/只读常量。

2、malloc/calloc/realloc三个函数之间的区别　尤其喜欢问malloc****

3、malloc/free和new/delete的区别*****

malloc/free和new/delete的共同点是：都是从堆上申请空间，并且需要用户手动释放。

不同的地方是：

1. malloc和free是函数，new和delete是操作符
2. malloc申请的空间不会初始化，new可以初始化

3. malloc申请空间时，需要手动计算空间大小并传递，new只需在其后跟上空间的类型即可

4. malloc的返回值为void*, 在使用时必须强转，new不需要，因为new后跟的是空间的类型

5. malloc申请空间失败时，返回的是NULL，因此使用时必须判空，new不需要，但是new需要捕获异常

6. 申请自定义类型对象时，malloc/free只会开辟空间，不会调用构造函数与析构函数，而new在申请空间 后会调用构造函数完成对象的初始化，delete在释放空间前会调用析构函数完成空间中资源的清理

4、什么是内存泄露？如何检测程序中是否存在内存泄露？预计方式***

什么是内存泄漏：内存泄漏指因为疏忽或错误造成程序未能释放已经不再使用的内存的情况。内存泄漏并不 是指内存在物理上的消失，而是应用程序分配某段内存后，因为设计错误，失去了对该段内存的控制，因而 造成了内存的浪费。

内存泄漏的危害：长期运行的程序出现内存泄漏，影响很大，如操作系统、后台服务等等，出现内存泄漏会 导致响应越来越慢，最终卡死。

预计方式：1、事前预防型。如智能指针等。2、事后查错型。如泄漏检测工 具。