C，C++部分：

Static：函数内部的static生命周期长久，值不会随着函数调用的结束而消失，下次调用还会保留上一次的值

const 指针常量和常量指针: (1)指针常量：const int \*p=&a; p可以改变，\*p不可以改变  
(2)常量指针：int \* const p ,p不可以改变，\*p可以改变  
(3)常量指针：int const \*p,与（1）相同，只与const的位置有关

++I i++区别:++i是先进行+操作，再进行别的操作，i++是先进行别的操作，再+操作

返回局部变量?原因

指针数组？数组指针：数组指针是指向数组地址的指针，本质是指针。指针数组是数组元素都是指针的数组，本质是数组。

指针函数，函数指针：int \* (\* func\_ptr) (int, float \* )是函数指针，int\*是函数返回值类型。func\_ptr是函数指针的名字，float\*是参数类型是指针，函数指针指向函数入口的地址

ifndef/define/endif作用？define是宏定义，#define PI 3.14159 PI将代替3.14

ifndef检测名称是否被定义了。

c和C++是如何定义常量的，区别？

%一个特点？结果是整数，没有余数

函数传参数方式(值，指针,引用)传值：形参是实参的拷贝，修改形参并不改变实参的值。传引用：形参是实参的别名，修改形参，实参也改变。传指针：形参与实参指向同一变量，形参通过解引用，实参改变，双向联系。

构造函数作用，特征 声明对象时，自动调用构造函数。函数名与类名一样，没有返回值，可以有参数

数组和指针的区别数组是连续的存储空间，指针是一个变量，来存储地址

内存分配？有动态内存分配（堆和栈）和静态内存分配（全局变量、静态全局变量、静态局部变量）

空指针，野指针？空指针是指向一块空的内存，野指针是指向一块不可用的垃圾内存，

<>和“”区别？<>一般是系统头文件“”一般是自定义的头文件

虚函数？在继承时，当父类指针后者引用指向子类对象时，把父类的成员函数声明为虚函数，当对象调用函数时就能知道调用正确的版本，而与该对象的指针或者引用无关。

构造函数 virtual?继承时，子类要先实现父类成员变量的构造函数，再实现自己的

引用和指针的区别:指针是指向一块内存，存得时内存的地址，引用是内存的别名。

1.指针是一个实体，而引用仅是个别名；  
2. 引用使用时无需解引用(\*)，指针需要解引用；  
3. 引用只能在定义时被初始化一次，之后不可变；指针可变；  
4. 引用没有 const，指针有 const；  
5. 引用不能为空，指针可以为空；  
6. “sizeof 引用”得到的是所指向的变量(对象)的大小，而“sizeof 指针”得到的是指针本身(所指向的变量或对象的地址)的大小；  
7. 指针和引用的自增(++)运算意义不一样；  
8.从[内存分配](https://www.baidu.com/s?wd=%E5%86%85%E5%AD%98%E5%88%86%E9%85%8D&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWPjRLrHubPvn3n1DvmW010AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtzrjDzn1RLrj6" \t "_blank)上看：程序为指针变量[分配内存](https://www.baidu.com/s?wd=%E5%88%86%E9%85%8D%E5%86%85%E5%AD%98&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWPjRLrHubPvn3n1DvmW010AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtzrjDzn1RLrj6" \t "_blank)区域，而引用不需要[分配内存](https://www.baidu.com/s?wd=%E5%88%86%E9%85%8D%E5%86%85%E5%AD%98&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWPjRLrHubPvn3n1DvmW010AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtzrjDzn1RLrj6" \t "_blank)区域。

析构函数 virtual?原因继承时，父类的析构函数必须写成虚析构，因为在父类指针指向子类的这种情况下，如果父类不是虚析构，就会只删除来自父类的成员变量，而不会删除子类自己的那部分，就会造成内存泄露。

菱形继承问题？A、B都继承父类R，而C又同时继承AB。父类AB和爷爷类R之间的继承用虚继承，孙子类C 的构造函数要调用ABR3个类的构造函数，再扩展自己类的

构造函数可以virtual?不能。一个对象在创建时已经知道了它使什么类的，将调用哪个类的构造方法。

Extern？引入其他文件的全局变量

全局变量和局部变量的区别？全局变量在整个类里面可以直接用（除了静态全局变量），局部变量只能在函数内部使用，出了函数就释放了。

Main函数之前会执行什么代码？全局对象的构造函数

拷贝构造函数？用一个已经存在的对象去初始化另外一个对象

结构体和联合的内存区别？结构体是多个不同类型变量的组合，其占用的内存大小是所有变量大小的总和联合体是多个不同类型变量的抽取，其占用内存的大小是联合体中最大的那个变量的大小

类型转化方式？

动态绑定？

重载重写？重载是在同一个类里面函数名相同，参数个数或者类型不同，重写是子类覆盖父类的方法，要求方法名和参数都相同

空类占用空间？1

Inline？在类内部实现的方法都是内联函数，在类外部实现的函数想要内联的话就在函数声明时声明为Inline这样在外部就不用写域名了。内联函数效率高

Malloc/free new/delete?

纯虚函数？在父类里的虚函数声明时=0，只有函数体不用实现，在子类里就必须实现

抽象基类？有纯虚函数的类是抽象类，被子类继承就变为了抽象基类

继承的3种方式公有。私有。保护。不管哪一种，父类中的私有成员是不能直接访问的，只能通过父类的成员函数访问。公有继承：父类中除了私有成员，其他的都被原样集继承到子类。保护性继承：除了私有成员，父类中的其它成员都被集成为protect属性，内部、派生类可以直接访问，外部不行。私有继承：被继承来的成员全部变为private私有，派生类不能

继承的3种方式？

C++默认有哪些函数？默认构造函数，拷贝构造函数，默认赋值函数以及析构函数

类中含有一个静态成员空间？

类中含有一个virtual函数空间？

累中有一个virtual函数和char占用空间

stl?Vector list区别 map set区别 mutilemap map区别

函数模版？类模版？

浅拷贝和深拷贝的区别

多态？

冒泡思想

快排思想

选择排序

插入排序

Cocos2d-x部分：

内存机制(OC)

创建精灵的方式?

锚点？

Schedule？

如何理解导演，场景，层，精灵

触摸优先级

适配策略

不同精灵能否同时执行一个动作

节点引用计数什么时候会增加

实现代理

spawn

sequence

searchPath

nullptr 和 NULL区别

runWithScene?

replaceScene?

pushScene?

popScene?

内存管理机制

texture2D

spriteFrameCache

场景跳转的内存变化，应如何避免

异步加载 addAsyncImage(“ImagePath”,”CallBack”)

分析创建的数组为何往往需要retain下

实现一个精灵围绕场景中的某个点转动（该点不是精灵锚点）

Lua部分：

类型(8)

闭合函数

非全局函数

尾调用，好处

协同程序参数问题

元表 元方法

\_\_index

\_\_newindex

rawget

rawest

lua实现面向对象

lua继承

说出以下三种函数中分别可以抛出什么异常

void f();

void f() throw()

int func( int x ) throw( int, Error\_message)

运行以下程序会有什么样的结果：

int main()

{

　char \*str = new char[100];

　strcpy( str, "hello" );

　delete []str;

　if(str != NULL)

{

strcpy(str,”world”);

printf(“%s\n”,str);

}

return 0;

}

void GetMemory( char \*\*p, int num )

{

　\*p = new char[num];

}

int main( )

{

　char \*str = NULL;

　GetMemory( &str, 100 );

　strcpy( str, "hello" );

　cout<<str<<endl;

return 0;

}

运行以下程序会有什么样的结果：

char \*GetMemory( void )

{

　char p[] = "hello world";

　return p;

}

int main( )

{

　char \*str = NULL;

　str = GetMemory();

　cout<<str<<endl;

return 0;

}

运行以下程序会有什么样的结果：

void GetMemory( char \*p )

{

　p = new char[100];

}

int main( )

{

　char \*str = NULL;

　GetMemory( str );

　strcpy( str, "hello world" );

　cout<<str<<endl;

return 0;

}

union Person{

short num;

double num1;

int name;

}

sizeof(Person) =

Struct A{

char s[10];

int num;

short num1;

}

sizeof(A) =

Struct B{

Int s;

int num;

short num1;

}

sizeof(B) =

struct C{

}

sizeof(C)

class AA{

}

sizeof(AA)

class BB{

int num1;

}

sizeof(BB)

class CC{

static int num;

}

sizeof(CC)

class DD{

virtual ~DD(){}

}

sizeof(DD)

class EE{

char n;

virtual ~DD(){}

}

sizeof(EE)