[**C++ 语言基础**](http://www.cnblogs.com/answeryi/archive/2012/07/13/2586913.html)

（1）static

函数内的static变量，内存中分配一次

模块内的静态全局变量和函数只能在模块内访问

类：静态成员，静态函数：整个类共有，一份复制

（2）const

修饰只读变量，定义后不能修改；必须初始化；

[编译器通常不为普通const分配内存，直接符号表，效率高]

（3）#define

没有类型，简单替换，不能调试，预编译阶段

#define MIN(A,B) (A)<(B)?(A):(B) 括号括起来

#define LONG\_NUM （60\*69\*6777777）UL

（4）数组与指针

数组初始化后地址不能改变，指针可以更改；

sizeof 可以计算数组容量，但是得不到指针所指内容容量；

（5）运算符优先级

括号、成员 --> 单 --> 双 --> 三 --> 赋值 --> 逗号 （注：逗号表达式取最右边的值）

不能重载的运算符：\* . :: ?: sizeof typeid

重载前置++和后置++

type operator++（）

type operator++（int）

=======================================================

（6）引用和指针

引用必须初始化，指针不必；引用初始化不能变，指针可以；引用不能为空，指针可以；

引用不空，故更安全，指针须检验，效率稍低。

（7）void

指向不属于任何类型对象，与空指针不同。

（8）指针数组，数组指针，函数指针

指针数组：int \*p[10]

数组指针：int (\*p)[10]

函数指针：int (\*a)(int) 参数，返回值均为int

函数指针数组：int(\*s[10])(int)

返回值为函数指针的函数指针：int (\*(\*f)(int))(int)

(9) 指针常量，常量指针

指针常量：char\* const p;

常量指针：const char\* p (char const \* p)

=========================================================

(10) 堆 和 栈

都是一块物理内存；

堆，动态分配，较大得数据，内存中由小到大；

栈，主要存储函数参数，返回值，局部变量，由高到低；

堆，动态分配，手动释放；栈静态分配，无需手动释放。

（11）内存分配方式

静态存储区，堆，栈

C|C++程序编译时内存分 5 大存储区:堆区、栈区、全局区、文字常量区、程序代码区

【补充】\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**##变量的声明和定义有什么区别**

为变量分配地址和存储空间的称为定义,不分配地址的称为声明。一个变量可以在多个地方声明,但是只在一个地方定义。加入 extern 修饰的是变量的声明,说明此变量将在文件以外或在文件后面部分定义。

说明:很多时候一个变量,只是声明不分配内存空间,直到具体使用时才初始化,分配内存空间, 如外部变量。

##sizeof 和 strlen 的区别：

sizeof 是一个操作符,strlen 是库函数。

sizeof 的参数可以是数据的类型,也可以是变量,而 strlen 只能以结尾为‘\0‘的字符串作参数。 编译器在编译时就计算出了 sizeof 的结果。而 strlen 函数必须在运行时才能计算出来。并且 sizeof计算的是数据类型占内存的大小,而 strlen 计算的是字符串实际的长度。

数组做 sizeof 的参数不退化,传递给 strlen 就退化为指针了。

##C 语言的关键字 static 和 C++ 的关键字 static 有什么区别

在 C 中 static 用来修饰局部静态变量和外部静态变量、函数。而 C++中除了上述功能外,还用来定义类的成员变量和函数。即静态成员和静态成员函数。

##一个指针可以是 volatile 吗 ？？？？？？？？？？？

可以,因为指针和普通变量一样,有时也有变化程序的不可控性。

（12）面向对象的三大特征：封装性、继承性和多态性:

封装性:将客观事物抽象成类,每个类对自身的数据和方法实行 protection(private, protected, public)。

继承性:广义的继承有三种实现形式:实现继承(使用基类的属性和方法而无需额外编码的能力)、可视继承(子窗体使用父窗体的外观和实现代码)、接口继承(仅使用属性和方法,实现滞后到子类实现)。

多态性:在面向对象语言中，接口的多种不同的实现方式即为多态。父类对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。

（13）C++的空类有哪些成员函数

缺省构造函数。

缺省拷贝构造函数。

缺省析构函数。

缺省赋值运算符。

缺省取址运算符。

缺省取址运算符 const。

（14）覆盖(overwrite)和重载（override） ，隐藏

范围的区别:被重写的和重写的函数在两个类中,而重载和被重载的函数在同一个类中。

参数的区别:被重写函数和重写函数的参数列表一定相同,而被重载函数和重载函数的参数列表一 定不同。

virtual 的区别:重写的基类中被重写的函数必须要有 virtual 修饰,而重载函数和被重载函数可以被virtual 修饰,也可以没有。

与重载的范围不同:和重写一样,隐藏函数和被隐藏函数不在同一个类中。

参数的区别:隐藏函数和被隐藏的函数的参数列表可以相同,也可不同,但是函数名肯定要相同。

当参数不相同时,无论基类中的参数是否被 virtual 修饰,基类的函数都是被隐藏,而不是被重写。

说明:虽然重载和覆盖都是实现多态的基础,但是两者实现的技术完全不相同,达到的目的也是完全不同的,覆盖是动态态绑定的多态,而重载是静态绑定的多态。

(15)简述多态实现的原理

编译器发现一个类中有虚函数,便会立即为此类生成虚函数表 vtable。虚函数表的各表项为指向对应虚函数的指针。

编译器还会在此类中隐含插入一个指针 vptr(对 vc 编译器来说,它插在类的第一个位置上)指向虚函数表。调用此类的构造函数时,在类的构造函数中,编译器会隐含执行 vptr 与 vtable 的关联代码,将 vptr 指向对应的 vtable,将类与此类的 vtable 联系了起来。另外在调用类的构造函数时,指向基础类的指针此时已经变成指向具体的类的 this 指针,这样依靠此 this 指针即可得到正确的 vtable。如此才能真正与函数体进行连接,这就是动态联编,实现多态的基本原理。

注意:一定要区分虚函数,纯虚函数、虚拟继承的关系和区别。牢记虚函数实现原理,因为多态C++面试的重要考点之一,而虚函数是实现多态的基础。

（16）explicit

禁止将构造函数作为转换函数

（17）实现内联函数的两种方法

1. 使用inline

2. 定义成员函数时直接写出函数体

（18）mutable

在方法末尾使用const关键字时，表示该方法不能修改对象信息；

但是用mutable修饰的成员除外。

（19）杂

多个构造函数和一个析构函数；

两个类互为成员时，有一个预先声明；

类的友元函数可以访问private成员；

多态的作用：一是隐藏实现细节，代码模块化；而是代码重用，实现通用功能。

（20）继承优缺点

优点：代码重用，灵活的改变父类方法；

缺点：层次过多，维护困难；修改父类，影响之类；无法在运行时改变继承关系

（21）继承方式

public，protected，private 权限不会升高

（22）访问被隐藏的父类成员

((Father)Child).fathersFunc

(23) 多态的两个必要条件

父类定义虚函数，子类改写该函数；

定义基类指针，调用子类构造函数。

（24） 虚继承

防止多继承时存在重复父类

**【补充】===================================**

1 char\* s 一定要分配空间 new char[n];

char s[] = "abcdefg";

char\* 字符串 ‘\0’结束，char s[] 默认添加；

2 char 范围 -128 ~ 127

3 class 和 struct 区别

默认的继承访问权限。struct是public的，class是private的

4 C++中定义类的对象：用new和不用new有何区别？

[复制代码](javascript:void(0);)

1 class Point {   
2 private:   
3 int x; int y;   
4 public:  
5 void Set(int a,int b) { x=a; y=b; }   
6 void Print() {   
7 cout<<"("<<x<<","<<y<<")"<<endl;  
8 }  
9 };   
10 void main()  
11 {   
12 Point p1;   
13 Point \*p2=new Point();   
14 p1.Set(1,2); p2->Set(4,5);   
15 p1.Print(); p2->Print(); delete p2;  
16 }

[复制代码](javascript:void(0);)

p1有系统创建并释放，你不要担心会出现内存泄露，但是生命期只有在本区域的大括号内，出了大括号就没用了。  
P2是指针，要自己释放，用不好很危险，用好了功能强大，因为他可以赋值给全局的变量，一下子从局部变量变成全局变量，还能把对象作为函数返回值。

5 类与对象

* 不能再类声明中给数据成员赋初值；
* 构造函数可用初始化表对成员初始化，但放在堆中或数组的成员在构造函数内部初始化；
* 没有定义构造函数，可以用初始化表进行初始化。

[复制代码](javascript:void(0);)

1 class A {  
2 int i;  
3 char c;  
4 float f;  
5 char name[25];  
6 public:  
7 A(int ii, char cc, float ff, char N[]):  
8 i(ii), c(cc), f(ff)  
9 {  
10 strcpy(name,N);  
11 }   
12 };

[复制代码](javascript:void(0);)

若没有定义构造函数： 可以这样初始化 A a = { 1, 'c', 1.23, "name"};

* 拷贝构造函数 ClassA (const ClassA & A) : 带入或者赋值调用

ClassA a2(a1) 或者ClassA a3 = a1;

* 向函数传递对象参数时，是传值；若想改变对象使用对象指针或者引用
* 静态数据成员不能在类内初始化，因为不分配空间，在内外定义。
* 友元函数，友元成员可以访问私有数据，对封装机制的补充，提高效率。

5 继承与多态

多重继承的父类如果同时继承同一个父类，则数据成员会同时存在多个拷贝； 解决方法：虚继承

同一层次同时存在虚基类和非虚基类，先调用虚基类的构造函数

多态：不同对象收到相同消息，产生不同的动作。

* 编译多态（重载） 和 运行多态（虚函数）

虚函数，纯虚函数（抽象类）

* 声明为指向基类的指针，可以指向其公有派生的对象，不可以指向私有派生的对象

6 模板类中友元函数重载输入>>和输出<<

若使用友元在类内声明，在类外实现，那么连接时将会报错.

若使用友元，在类内实现；在类外实现，不能使用friend关键字，想访问私有变量可以通过公用方法

7

编程规范可总结为:程序的可行性、可读性、可移植性以及可测试性。

面 向对象可以理解成对待每一个问题,都是首先要确定这个问题由几个部分组成,而每一个部分其实就是一个对象。然后再分别设计这些对象,最后得到整个程序。传 统的程序设计多是基于功能的思想来进行考虑和设计的,而面向对象的程序设计则是基于对象的角度来考虑问题。这样做能够使得程序更加的简洁清晰。