### 黄色是要注意的

### 红色是有疑问的

### 3. 对象都具有的两方面特征是什么？分别是什么含义？

答：对象都具有的特征是：静态特征和动态特征。

静态特征是指能描述对象的一些属性（成员变量），动态特征是指对象表现出来的行为（成员函数）

### 4. 在头文件中进行类的声明，在对应的实现文件中进行类的定义有什么意义？

答：这样可以提高编译效率，因为分开的话只需要编译一次生成对应的.obj文件后，再次应用该类的地方，这个类就不会被再次编译，从而大大的提高了编译效率。

### 5. 在类的内部定义成员函数的函数体，这种函数会具备那种属性？

答：这种函数会自动为内联函数，这种函数在函数调用的地方在编译阶段都会进行代码替换。

### 6. 成员函数通过什么来区分不同对象的成员数据？为什么它能够区分？

答：通过this指针指向对象的首地址来区分的。

### 7. C++编译器自动为类产生的四个缺省函数是什么？

答：默认构造函数，拷贝构造函数，析构函数，赋值函数。

### 8. 拷贝构造函数在哪几种情况下会被调用？

答：

1.当类的一个对象去初始化该类的另一个对象时；

2.如果函数的形参是类的对象，调用函数进行形参和实参结合时；

3.如果函数的返回值是类对象，函数调用完成返回时。

### 10.  什么时候必须重写拷贝构造函数？

答：当构造函数涉及到动态存储分配空间时，要自己写拷贝构造函数，并且要深拷贝。

### 11.  构造函数的调用顺序是什么？

答：

1.先调用基类构造函数

2.按声明顺序初始化数据成员

3.最后调用自己的构造函数。

### ****12.  哪几种情况必须用到初始化成员列表？****

答：类的成员是常量成员初始化；

类的成员是对象成员初始化，而该对象没有无参构造函数。

类的成员为引用时。

### 13.  什么是常对象？

答：常对象是指在任何场合都不能对其成员的值进行修改的对象。

### 14.  静态函数存在的意义？

答：静态私有成员在类外不能被访问，可通过类的静态成员函数来访问；

当类的构造函数是私有的时，不像普通类那样实例化自己，只能通过静态成员函数来调用构造函数。

### 15.  在类外有什么办法可以访问类的非公有成员？

答：友元，继承，公有成员函数。

### ****16.  什么叫抽象类？****

答：不用来定义对象而只作为一种基本类型用作继承的类。

### 17.  运算符重载的意义？

答：为了对用户自定义数据类型的数据的操作与内定义的数据类型的数据的操作形式一致。

### 18.  不允许重载的5个运算符是哪些？

答：

1. .\*（成员指针访问运算符号）

2. ：：域运算符

3. Sizeof 长度运算符号

4. ？：条件运算符号

5.  .（成员访问符）

### 19.  运算符重载的三种方式？

答：普通函数，友元函数，类成员函数。

### 20.  流运算符为什么不能通过类的成员函数重载？一般怎么解决？

答：因为通过类的成员函数重载必须是运算符的第一个是自己，而对流运算的重载要求第一个参数是流对象。所以一般通过友元来解决。

### 21.  赋值运算符和拷贝构造函数的区别与联系？

答：相同点：都是将一个对象copy到另一个中去。

不同点：拷贝构造函数涉及到要新建立一个对象。

### 22.  在哪种情况下要调用该类的析构函数？

答：对象生命周期结束时。

### 23.  对象间是怎样实现数据的共享的？

答：通过类的静态成员变量来实现对象间的数据共享。静态成员变量占有自己独立的空间不为某个对象所私有。

### 24.  友元关系有什么特性？

答：单向的，非传递的，不能继承的。

### 25.  对对象成员进行初始化的次序是什么？

答：它的次序完全不受它们在初始化表中次序的影响，只有成员对象在类中声明的次序来决定的。

### 26.  类和对象之间的关系是什么？

答：类是对象的抽象，对象是类的实例。

### 27.  对类的成员的访问属性有什么？

答：public，protected，private。

### 28．const char \*p和char \* const p; 的区别

答：

如果const位于星号的左侧，则const就是用来修饰指针所指向的变量，即指针指向为常量；

如果const位于星号的右侧，const就是修饰指针本身，即指针本身是常量。

### 29.  是不是一个父类写了一个virtual 函数，如果子类覆盖它的函数不加virtual ,也能实现多态?

答：

virtual修饰符会被隐形继承的。

virtual可加可不加,子类覆盖它的函数不加virtual ,也能实现多态。

### 30.  函数重载是什么意思？它与虚函数的概念有什么区别？

答：函数重载是一个同名函数完成不同的功能，编译系统在编译阶段通过函数**参数个数、参数类型不同，函数的返回值**来区分该调用哪一个函数，即实现的是**静态的多态性**。

但是记住：不能仅仅通过函数返回值不同来实现函数重载。而虚函数实现的是在基类中通过使用关键字virtual来申明一个函数为虚函数，含义就是该函数的功能可能在将来的派生类中定义或者在基类的基础之上进行扩展，系统只能在运行阶段才能动态决定该调用哪一个函数，所以实现的是**动态的多态性**。它体现的是一个纵向的概念，也即在基类和派生类间实现。

### 31.  构造函数和析构函数是否可以被重载,为什么?

答：构造函数可以被重载，析构函数不可以被重载。因为构造函数可以有多个且可以带参数，而析构函数只能有一个，且不能带参数。

32.  如何定义和实现一个类的成员函数为回调函数？

答：

所谓的回调函数，就是预先在系统的对函数进行注册，让系统知道这个函数的存在，以后，当某个事件发生时，再调用这个函数对事件进行响应。

定义一个类的成员函数时在该函数前加CALLBACK即将其定义为回调函数，函数的实现和普通成员函数没有区别

### 33.  虚函数是怎么实现的？

答：简单说来使用了虚函数表.

### 34.  抽象类不会产生实例，所以不需要有构造函数。 错

### 35.  从一个模板类可以派生新的模板类，也可以派生非模板类。 对

### 36.  main 函数执行以前，还会执行什么代码？

答案：全局对象的构造函数会在main 函数之前执行。

### 37.  当一个类A 中没有生命任何成员变量与成员函数,这时sizeof(A)的值是多少，如果不是零，请解释一下编译器为什么没有让它为零。（Autodesk）

答案：肯定不是零。举个反例，如果是零的话，声明一个class A[10]对象数组，而每一个对象占用的空间是零，这时就没办法区分A[0],A[1]…了。

38.  delete与 delete []区别：

答：delete只会调用一次析构函数，而delete[]会调用每一个成员的析构函数。

39．子类析构时要调用父类的析构函数吗？

答：会调用。析构函数调用的次序是先派生类的析构后基类的析构，也就是说在基类的的析构调用的时候,派生类的信息已经全部销毁了

40.  继承的优缺点。

1、类继承是在编译时刻静态定义的，且可直接使用，

2、类继承可以较方便地改变父类的实现。

缺点：

1、因为继承在编译时刻就定义了，所以无法在运行时刻改变从父类继承的实现

2、父类通常至少定义了子类的部分行为，父类的任何改变都可能影响子类的行为

3、如果继承下来的实现不适合解决新的问题，则父类必须重写或被其他更适合的类替换。这种依赖关系限制了灵活性并最终限制了复用性。

### 41.  解释堆和栈的区别。

答：栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。

堆（heap）一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收 。

### 42.  一个类的构造函数和析构函数什么时候被调用,是否需要手工调用?

答：构造函数在创建类对象的时候被自动调用，析构函数在类对象生命期结束时，由系统自动调用。

### 43.  何时需要预编译：

答：总是使用不经常改动的大型代码体。

程序由多个模块组成，所有模块都使用一组标准的包含文件和相同的编译选项。在这种情况下，可以将所有包含文件预编译为一个预编译头。

### 44.  多态的作用？

答：主要是两个：

1. 隐藏实现细节，使得代码能够模块化；扩展代码模块，实现代码重用；

2. 接口重用：为了类在继承和派生的时候，保证使用家族中任一类的实例的某一属性时的正确调用

45.  虚拟函数与普通成员函数的区别？内联函数和构造函数能否为虚拟函数？

答案：区别：虚拟函数有virtual关键字，有虚拟指针和虚函数表，虚拟指针就是虚拟函数的接口，而普通成员函数没有。内联函数和构造函数不能为虚拟函数。

46.  构造函数和析构函数的调用顺序? 析构函数为什么要虚拟?

答案：构造函数的调用顺序：基类构造函数—对象成员构造函数—派生类构造函数；析构函数的调用顺序与构造函数相反。析构函数虚拟是为了防止析构不彻底，造成内存的泄漏。

47. C++中类型为private的成员变量可以由哪些函数访问?

答：只可以由本类中的成员函数和友元函数访问

48.  请说出类中private，protect，public三种访问限制类型的区别

答：private是私有类型，只有本类中的成员函数访问;protect是保护型的，本类和继承类可以访问;public是公有类型，任何类都可以访问.

### 49.  类中成员变量怎么进行初始化？

答：可以通过构造函数的初始化列表或构造函数的函数体实现。

### 50.  在什么时候需要使用“常引用”？

答：如果既要利用引用提高程序的效率，又要保护传递给函数的数据不在函数中被改变，就应使用常引用。

### 51.  引用与指针有什么区别？

答：

1) 引用必须被初始化，指针不必。

2) 引用初始化以后不能被改变，指针可以改变所指的对象。

3) 不存在指向空值的引用，但是存在指向空值的指针。

52.  描述实时系统的基本特性

答 、在特定时间内完成特定的任务，实时性与可靠性。

54.  全局变量和局部变量在内存中是否有区别？如果有，是什么区别？

答 、全局变量储存在静态数据区，局部变量在堆栈中。

55.  堆栈溢出一般是由什么原因导致的？

答 、没有回收垃圾资源

56.  什么函数不能声明为虚函数？

答 构造函数（constructor）

57.  IP地址的编码分为哪俩部分？

答 IP地址由两部分组成，网络号和主机号。

58.  不能做switch()的参数类型是：

答 、switch的参数不能为实型。

### 59.  如何引用一个已经定义过的全局变量？

答 、可以用引用头文件的方式，也可以用extern关键字，如果用引用头文件方式来引用某个在头文件中声明的全局变理，假定你将那个变写错了，那么在编译期间会报错，如果你用extern方式引用时，假定你犯了同样的错误，那么在编译期间不会报错，而在连接期间报错

### 60.  对于一个频繁使用的短小函数,在C语言中应用什么实现,在C++中应用什么实现?

答 、c用宏定义，c++用inline

### 61.  C++是不是类型安全的？

答案：不是。两个不同类型的指针之间可以强制转换（用reinterpret cast)

### 62.  当一个类A 中没有生命任何成员变量与成员函数,这时sizeof(A)的值是多少，请解释一下编译器为什么没有让它为零。

答案：为1。举个反例，如果是零的话，声明一个class A[10]对象数组，而每一个对象占用的空间是零，这时就没办法区分A[0],A[1]…了。

### 63.  简述数组与指针的区别？

答：数组要么在静态存储区被创建（如全局数组），要么在栈上被创建。指针可以随时指向任意类型的内存块。

(1)修改内容上的区别

char a[] = “hello”;

a[0] = ‘X’;

char \*p = “world”; // 注意p 指向常量字符串

p[0] = ‘X’; // 编译器不能发现该错误，运行时错误

(2) 用运算符sizeof 可以计算出数组的容量（字节数）。sizeof(p),p 为指针得到的是一个指针变量的字节数，而不是p 所指的内存容量。

### 64.  C++函数中值的传递方式

答：有三种方式：值传递、指针传递、引用传递

### 65.  内存的分配方式

答：分配方式有三种，

1、 静态存储区，是在程序编译时就已经分配好的，在整个运行期间都存在，如全局变量、常量。

2、 栈上分配，函数内的局部变量就是从这分配的，但分配的内存容易有限。

3、 堆上分配，也称动态分配，如我们用new,malloc分配内存，用delete,free来释放的内存。

66.  extern“C”有什么作用？

答：Extern “C”是由Ｃ＋＋提供的一个连接交换指定符号，用于告诉Ｃ＋＋这段代码是Ｃ函数。这是因为C++编译后库中函数名会变得很长，与C生成的不一致，造成Ｃ＋＋不能直接调用C函数，加上extren “c”后，C++就能直接调用C函数了。

Extern “C”主要使用正规DLL函数的引用和导出 和 在C++包含C函数或C头文件时使用。使用时在前面加上extern “c” 关键字即可。可以用一句话概括extern “C”这个声明的真实目的：实现C++与C及其它语言的混合编程。

### 73. 在定义一个宏的时候要注意什么？

答：定义部分的每个形参和整个表达式都必须用括号括起来，以避免不可预料的错误发生

### 74.  数组在做函数实参的时候会转变为什么类型？

答：数组在做实参时会变成指针类型。

75.  系统会自动打开和关闭的3个标准的文件是？

(1)标准输入—-键盘—stdin

(2)标准输出—-显示器—stdout

(3)标准出错输出—-显示器—stderr

76.  .在Win32下 char, int, float, double各占多少位？

(1)Char 占用8位

(2)Int占用32位

(3)Float占用32位

(4)Double占用64位

### 77.  strcpy()和memcpy()的区别？

答：strcpy()和memcpy()都可以用来拷贝字符串，strcpy()拷贝以’\0’结束，但memcpy()必须指定拷贝的长度。

78.  说明define和const在语法和含义上有什么不同？

答：

(1)#define是C语法中定义符号变量的方法，符号常量只是用来表达一个值，在编译阶段符号就被值替换了，它没有类型；

(2)Const是C++语法中定义常变量的方法，常变量具有变量特性，它具有类型，内存中存在以它命名的存储单元，可以用sizeof测出长度。

79.  说出字符常量和字符串常量的区别，并使用运算符sizeof计算有什么不用？

答：字符常量是指单个字符，字符串常量以‘\0’结束，使用运算符sizeof计算多占一字节的存储空间。

81.  总结static的应用和作用？

答：

（1）函数体内static变量的作用范围为该函数体，不同于auto变量，该变量的内存只被分配一次，因此其值在下次调用时仍维持上次的值；

（2）在模块内的static全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问；

（3）在模块内的static函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明它的模块内；

（4）在类中的static成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；

（5）在类中的static成员函数属于整个类所拥有，这个函数不接收this指针，因而只能访问类的static成员变量。

82.  总结const的应用和作用？

答：

（1）欲阻止一个变量被改变，可以使用const关键字。在定义该const变量时，通常需要对它进行初始化，因为以后就没有机会再去改变它了；

（2）对指针来说，可以指定指针本身为const，也可以指定指针所指的数据为const，或二者同时指定为const；

（3）在一个函数声明中，const可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值；

（4）对于类的成员函数，若指定其为const类型，则表明其是一个常函数，不能修改类的成员变量；

（5）对于类的成员函数，有时候必须指定其返回值为const类型，以使得其返回值不为“左值”。

83.  什么是指针？谈谈你对指针的理解？

答：指针是一个变量，该变量专门存放内存地址；

指针变量的类型取决于其指向的数据类型，在所指数据类型前加\*

指针变量的特点是它可以访问所指向的内存。

84.  什么是常指针，什么是指向常变量的指针？

答：常指针的含义是该指针所指向的地址不能变，但该地址所指向的内容可以变化，使用常指针可以保证我们的指针不能指向其它的变量，

指向常变量的指针是指该指针的变量本身的地址可以变化，可以指向其它的变量，但是它所指的内容不可以被修改。指向长变量的指针定义，

85.  函数指针和指针函数的区别？

答：函数指针是指向一个函数入口的指针；

指针函数是函数的返回值是一个指针类型。

88.  指针的几种典型应用情况？

答：

int \*p[n];—–指针数组，每个元素均为指向整型数据的指针。

int (\*)p[n];—p为指向一维数组的指针，这个一维数组有n个整型数据。

int \*p();——函数带回指针，指针指向返回的值。

int (\*)p();—-p为指向函数的指针。

89.  static函数与普通函数有什么区别?

答：static函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝

90.  struct(结构) 和 union(联合)的区别？

答：

1. 结构和联合都是由多个不同的数据类型成员组成, 但在任何同一时刻, 联合中只存放了一个被选中的成员（所有成员共用一块地址空间）, 而结构的所有成员都存在（不同成员的存放地址不同）。

2. 对于联合的不同成员赋值, 将会对其它成员重写,  原来成员的值就不存在了, 而对于结构的不同成员赋值是互不影响的。

91.  class 和 struct 的区别？

答：struct 的成员默认是公有的，而类的成员默认是私有的。

92.  简述枚举类型？

答：枚举方便一次定义一组常量，使用起来很方便；

93.  assert()的作用？

答：ASSERT()是一个调试程序时经常使用的宏，在程序运行时它计算括号内的表达式，如果表达式为FALSE (0), 程序将报告错误，并终止执行。如果表达式不为0，则继续执行后面的语句。这个宏通常原来判断程序中是否出现了明显非法的数据，如果出现了终止程序以免导致严重后果，同时也便于查找错误。

94.  局部变量和全局变量是否可以同名？

答：能。局部会屏蔽全局。要用全局变量，需要使用”::”(域运算符)。

95.  程序的局部变量存在于（堆栈）中，全局变量存在于（静态区 ）中，动态申请数据存在于（ 堆）中。

96.  在什么时候使用常引用？

答：如果既要利用引用提高程序的效率，又要保护传递给函数的数据不在函数中被改变，就应使用常引用。

### 97.  类的声明和实现的分开的好处？

答：1. 起保护作用；

2. 提高编译的效率。

100. 什么是UDP和TCP的区别是什么？

答：TCP的全称为传输控制协议。这种协议可以提供面向连接的、可靠的、点到点的通信。

UDP全称为用户报文协议，它可以提供非连接的不可靠的点到多点的通信。用TCP还是UDP，那要看你的程序注重哪一个方面？可靠还是快速？

101. winsock建立连接的主要实现步骤?

答：

服务器端：socket()建立套接字，绑定（bind）并监听（listen），用accept（）等待客户端连接, accept（）发现有客户端连接，建立一个新的套接字，自身重新开始等待连接。该新产生的套接字使用send()和recv（）写读数据，直至数据交换完毕，closesocket()关闭套接字。

客户端：socket()建立套接字，连接（connect）服务器，连接上后使用send()和recv（），在套接字上写读数据，直至数据交换完毕，closesocket()关闭套接字。

102. 进程间主要的通讯方式？

答：信号量，管道，消息，共享内存

110. 简述Visual C++ 、Win32 API和MFC之间的关系？

答：(1)Visual C+是一个以C++程序设计语言为基础的、集成的、可视化的编程环境；

(2)Win32 API是32位Windows操作系以C/C++形式提供的一组应用程序接口；

(3)MFC是对Win32 API的封装，简化了开发过程。

111.怎样消除多重继承中的二义性？

答： 1．成员限定符  2．虚基类

112什么叫静态关联，什么叫动态关联

答：在多态中，如果程序在编译阶段就能确定实际执行动作，则称静态关联，

如果等到程序运行才能确定叫动态关联。

113多态的两个必要条件

答：1.一个基类的指针或引用指向一个派生类对象  2.虚函数

114.什么叫智能指针？

答：当一个类中，存在一个指向另一个类对象的指针时，对指针运算符进行重载，那么当前类对象可以通过指针像调用自身成员一样调用另一个类的成员。

115.什么时候需要用虚析构函数？

答：当基类指针指向用new运算符生成的派生类对象时，delete基类指针时，派生类部分没有释放掉而造成释放不彻底现象，需要虚析构函数。 补充：虚函数就是让派生类调用基类的虚函数。

117.什么是平衡二叉树？

答：左右子树都是平衡二叉树，而且左右子树的深度差值的约对值不大于1。

118.语句for( ；1 ；)有什么问题？它是什么意思？

答：无限循环，和while(1)相同。

### 119．派生新类的过程要经历三个步骤

答：1.吸收基类成员 2.改造基类成员 3.添加新成员

### 122. memset ,memcpy 的区别

答：memset用来对一段内存空间全部设置为某个字符，一般用在对定义的字符串进行初始化为’\0′。

memcpy用来做内存拷贝，你可以拿它拷贝任何数据类型的对象，可以指定拷贝的数据长度；

### 124怎样定义一个纯虚函数？含有纯虚函数的类称为什么？

答：在虚函数的后面加=0，含有虚函数的类称为抽象类。

125．已知strcpy函数的原型是：

char \* strcpy(char \* strDest,const char \* strSrc);不调用库函数，实现strcpy函数。其中，strSrc是原字符串，strDest是目标字符串 。

答案：

char \*strcpy(char \*strDest, const char \*strSrc)

{

if ( strDest == NULL || strSrc == NULL)

return NULL ;

if ( strDest == strSrc)

return strDest ;

char \*tempptr = strDest ;  //指针tempptr指向strDest的地址；

while( (\*strDest++ = \*strSrc++) != ‘\\0’)  //注意：别忘了转义符；

;

return tempptr ;  //返回指针向的地址；

}

126．已知类String 的原型为：

class String

{

public:

String(const char \*str = NULL); // 普通构造函数

String(const String &other);  // 拷贝构造函数

~ String(void); // 析构函数

String & operate =(const String &other); // 赋值函数

private:

char \*m\_data;  // 用于保存字符串

};

请编写String 的上述4 个函数。

答案：

// 普通构造函数

String::String(const char \*str)

{

if ( str == NULL )//strlen在参数为NULL时会抛异常才会有这步判断

{

m\_data = new char[1] ;

m\_data[0] = ” ;

}

else

{

m\_data = new char[strlen(str) + 1];

strcpy(m\_data,str);

}

}

//拷贝构造函数

String::String(const String &other)

{

m\_data = new char[strlen(other.m\_data) + 1];

strcpy(m\_data,other.m\_data);

}

//赋值函数(重载运算符)

String & String::operator =(const String &other)

{

if ( this == &other)

return \*this ;

delete []m\_data;

m\_data = new char[strlen(other.m\_data) + 1];

strcpy(m\_data,other.m\_data);

return \*this ;

}

//析构函数

String::~ String(void)

{

delete []m\_data ;

}

### 127．类成员函数的重载、覆盖和隐藏的区别

答案：

成员函数被重载的特征：

（1）相同的范围（在同一个类中）；

（2）函数名字相同；

（3）参数不同；

（4）virtual 关键字可有可无。

覆盖是指派生类函数覆盖基类函数，特征是：

（1）不同的范围（分别位于派生类与基类）；

（2）函数名字相同；

（3）参数相同；

（4）基类函数必须有virtual 关键字。

“隐藏”是指派生类的函数屏蔽了与其同名的基类函数，规则如下：

（1）如果派生类的函数与基类的函数同名，但是参数不同。此时，不论有无virtual关键字，基类的函数将被隐藏（注意别与重载混淆）。

（2）如果派生类的函数与基类的函数同名，并且参数也相同，但是基类函数没有virtual 关键字。此时，基类的函数被隐藏（注意别与覆盖混淆）

### 133 将“引用”作为函数参数有哪些特点？

（1）传递引用给函数与传递指针的效果是一样的。这时，被调函数的形参就成为原来主调函数中的实参变量或对象的一个别名来使用，所以在被调函数中对形参变量的操作就是对其相应的目标对象（在主调函数中）的操作。

（2）使用引用传递函数的参数，在内存中并没有产生实参的副本，它是直接对实参操作；而使用一般变量传递函数的参数，当发生函数调用时，需要给形参分配存储单元，形参变量是实参变量的副本；如果传递的是对象，还将调用拷贝构造函数。因此，当参数传递的数据较大时，用引用比用一般变量传递参数的效率和所占空间都好。

（3）使用指针作为函数的参数虽然也能达到与使用引用的效果，但是，在被调函数中同样要给形参分配存储单元，且需要重复使用”\*指针变量名”的形式进行运算，这很容易产生错误且程序的阅读性较差；另一方面，在主调函数的调用点处，必须用变量的地址作为实参。而引用更容易使用，更清晰。

### 134. 什么时候需要“引用”？

流操作符（<<、>>）和赋值操作符（=）的返回值、拷贝构造函数的参数、赋值操作符的参数、其它情况都推荐使用引用。

### 135.面向对象的三个基本特征，并简单叙述之？

1. 封装：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)

2. 继承：广义的继承有三种实现形式：实现继承（指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）。前两种（类继承）和后一种（对象组合=>接口继承以及纯虚函数）构成了功能复用的两种方式。

3. 多态：是将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父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。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。

136.求下面函数的返回值（微软）

int func(x)

{

int countx = 0;

while(x)

{

countx ++;

x = x&(x-1);

}

return countx;

}

假定x = 9999。 答案：8

思路：将x转化为2进制，看含有的1的个数。

137、写出下列代码的输出内容

＃i nclude<stdio.h>

int inc(int a)

{

return(++a);

}

int multi(int\*a,int\*b,int\*c)

{

return(\*c=\*a\*\*b);

}

typedef int(FUNC1)(int in);

typedef int(FUNC2) (int\*,int\*,int\*);

void show(FUNC2 fun,int arg1, int\*arg2)

{

INCp=&inc;

int temp =p(arg1);

fun(&temp,&arg1, arg2);

printf(“%d\n”,\*arg2);

}

main()

{

int a;

show(multi,10,&a);

return 0;

}

答：110

140.一个单向链表，不知道头节点,一个指针指向其中的一个节点，问如何删除这个指针指向的节点？

将这个指针指向的next节点值copy到本节点，将next指向next->next,并随后删除原next指向的节点。

141、用指针的方法，将字符串“ABCD1234efgh”前后对调显示

＃i nclude <stdio.h>

＃i nclude <string.h>

＃i nclude <dos.h>

int main()

{

char str[] = “ABCD1234efgh”;

int length = strlen(str);

char \* p1 = str;

char \* p2 = str + length – 1;

while(p1 < p2)

{

char c = \*p1;

\*p1 = \*p2;

\*p2 = c;

++p1;

–p2;

}

printf(“str now is %s\n”,str);

system(“pause”);

return 0;

}

142、有一分数序列：1/2,1/4,1/6,1/8……，用函数调用的方法，求此数列前20项的和

＃i nclude <stdio.h>

double getValue()

{

double result = 0;

int i = 2;

while(i < 42)

{

result += 1.0 / i;//一定要使用1.0做除数，不能用1，否则结果将自动转化成整数，即0.000000

i += 2;

}

return result;

}

int main()

{

printf(“result is %f\n”, getValue());

system(“pause”);

return 0;

}

143、有一个数组a[1000]存放0–1000;要求每隔二个数删掉一个数，到末尾时循环至开头继续进行，求最后一个被删掉的数的原始下标位置。

以7个数为例：

{0,1,2,3,4,5,6,7} 0–>1–>2（删除）–>3–>4–>5(删除)–>6–>7–>0（删除），如此循环直到最后一个数被删除。

方法1：数组

＃include <iostream>

using namespace std;

#define null 1000

int main()

{

int arr[1000];

for (int i=0;i<1000;++i)

arr[i]=i;

int j=0;

int count=0;

while(count<999)

{

while(arr[j%1000]==null)

j=(++j)%1000;

j=(++j)%1000;

while(arr[j%1000]==null)

j=(++j)%1000;

j=(++j)%1000;

while(arr[j%1000]==null)

j=(++j)%1000;

arr[j]=null;

++count;

}

while(arr[j]==null)

j=(++j)%1000;

cout<<j<<endl;

return 0;

}

方法2：链表

＃i nclude<iostream>

using namespace std;

#define null 0

struct node

{

int data;

node\* next;

};

int main()

{

node\* head=new node;

head->data=0;

head->next=null;

node\* p=head;

for(int i=1;i<1000;i++)

{

node\* tmp=new node;

tmp->data=i;

tmp->next=null;

head->next=tmp;

head=head->next;

}

head->next=p;

while(p!=p->next)

{

p->next->next=p->next->next->next;

p=p->next->next;

}

cout<<p->data;

return 0;

}

方法3：通用算法

＃i nclude <stdio.h>

#define MAXLINE 1000//元素个数

/\*

MAXLINE元素个数

a[] 元素数组

R[] 指针场

suffix 下标

index  返回最后的下标序号

values 返回最后的下标对应的值

start  从第几个开始

K间隔

\*/

int find\_n(int a[],int R[],int K,int& index,int& values,int s=0) {

int suffix;

int front\_node,current\_node;

suffix=0;

if(s==0) {

current\_node=0;

front\_node=MAXLINE-1;

}

else {

current\_node=s;

front\_node=s-1;

}

while(R[front\_node]!=front\_node) {

printf(“%d\n”,a[current\_node]);

R[front\_node]=R[current\_node];

if(K==1) {

current\_node=R[front\_node];

continue;

}

for(int i=0;i<K;i++){

front\_node=R[front\_node];

}

current\_node=R[front\_node];

}

index=front\_node;

values=a[front\_node];

return 0;

}

int main(void) {

int a[MAXLINE],R[MAXLINE],suffix,index,values,start,i,K;

suffix=index=values=start=0;

K=2;

for(i=0;i<MAXLINE;i++) {

a[i]=i;

R[i]=i+1;

}

R[i-1]=0;

find\_n(a,R,K,index,values,2);

printf(“the value is %d,%d\n”,index,values);

return 0;

}

144、指出下列程序有什么错误：

void test2()

{

char string[10], str1[10];

int i;

for(i=0; i<10; i++)

{

str1[i] = ‘a’;

}

strcpy( string, str1 );

}

解答：对试题2，如果面试者指出字符数组str1不能在数组内结束可以给3分；如果面试者指出strcpy(string, str1)调用使得从str1内存起复制到string内存起所复制的字节数具有不确定性可以给7分，在此基础上指出库函数strcpy工作方式的给10分；

str1不能在数组内结束:因为str1的存储为：{a,a,a,a,a,a,a,a,a,a},没有’\0′(字符串结束符)，所以不能结束

strcpy( char \*s1,char \*s2)他的工作原理是，扫描s2指向的内存，逐个字符付到s1所指向的内存，直到碰到’\0′,因为str1结尾没有’\0′，所以具有不确定性，不知道他后面还会付什么东东。

正确应如下

void test2()

{

char string[10], str1[10];

int i;

for(i=0; i<9; i++)

{

str1[i] = ‘a’+i; //把abcdefghi赋值给字符数组

}

str[i]=’\0′;//加上结束符

strcpy( string, str1 );

}

145、实现strcmp

int StrCmp(const char \*str1, const char \*str2)

{

assert(str1 && srt2);

while(\*str1 && \*str1++ = = \*str2++);

return \*str1-\*str2;

}

146.符串A和B,输出A和B中的最大公共子串。

比如A=”aocdfe” B=”pmcdfa” 则输出”cdf”

\*/

//Author: azhen

＃i nclude<stdio.h>

＃i nclude<stdlib.h>

＃i nclude<string.h>

char \*commanstring(char shortstring[], char longstring[])

{

int i, j;

char \*substring=malloc(256);

if(strstr(longstring, shortstring)!=NULL)  //如果……，那么返回shortstring

return shortstring;

for(i=strlen(shortstring)-1;i>0; i–)  //否则，开始循环计算

{

for(j=0; j<=strlen(shortstring)-i; j++){

memcpy(substring, &shortstring[j], i);

substring[i]=’\0′;

if(strstr(longstring, substring)!=NULL)

return substring;

}

}

return NULL;

}

main()

{

char \*str1=malloc(256);

char \*str2=malloc(256);

char \*comman=NULL;

gets(str1);

gets(str2);

if(strlen(str1)>strlen(str2)) //将短的字符串放前面

comman=commanstring(str2, str1);

else

comman=commanstring(str1, str2);

printf(“the longest comman string is: %s\n”, comman);

}

147、写一个函数比较两个字符串str1和str2的大小，若相等返回0，若str1大于

str2返回1，若str1小于str2返回－1

int strcmp ( const char \* src,const char \* dst)

{

int ret = 0 ;

while( ! (ret = \*(unsigned char \*)src – \*(unsigned char \*)dst) && \*dst)

{

++src;

++dst;

}

if ( ret < 0 )

ret = -1 ;

else if ( ret > 0 )

ret = 1 ;

return( ret );

}

148、判断一个字符串是不是回文

int IsReverseStr(char \*aStr)

{

int i,j;

int found=1;

if(aStr==NULL)

return -1;

j=strlen(aStr);

for(i=0;i<j/2;i++)

if(\*(aStr+i)!=\*(aStr+j-i-1))

{

found=0;

break;

}

return found;

149 #include main()

{

int c[3][3]={1,2,3,4,5,6,7,8,9};

for(int i=0;i<3;i++)

for(int j=0;j<3;j++)

printf(“%ld\n”,&c[j]);

printf(“————————-\n”);

printf(“%ld\n”,(c+1));

printf(“%ld\n”,(\*c+1));

printf(“%ld\n”,&c[0][0]);

printf(“%ld\n”,\*\*c);

printf(“%ld\n”,\*c[0]);

if(int(c)==int(\*c)) printf(“equl”);

}

为什么c，\*c的值相等，（c+1），（\*c+1）的值不等 c，\*c，\*\*c，代表什么意思？

参考答案：

c是第一个元素的地址，\*c是第一行元素的首地址，其实第一行元素的地址就是第一个元素的地址，这容易理解。\*\*c是提领第一个元素。

为什么c，\*c的值相等？

int c因为直接用c表示数组c[0][0] printf(“%ld\n”,\*c[0]);语句已将指针移到数组头。 int(\*c)表示c0的值为1，所以相等。 数组c的存放空间示意如下：（机器中是行优先存放的） c[0][0] c[0][1] c[0][2] c[1][0] c[1][1] c[1][2] c[2][0] c[2][1] c[2][2] c是一个二维数组名，实际上它是一个指针常量，不能进行自加、自减运算，即：c++、c–、++c、–c 都是不允许的；

c： 数组名；是一个二维指针，它的值就是数组的首地址，也即第一行元素的首地址（等于 \*c），也 等于第一行第一个元素的地址（ & c[0][0]）；可以说成是二维数组的行指针。 \*c： 第一行元素的首地址；是一个一维指针，可以说成是二维数组的列指针。 \*\*c：二维数组中的第一个元素的值；即：c[0][0] 所以： c 和 \*c的值是相等的，但他们两者不能相互赋值，（类型不同）； （c + 1） ：c是行指针，（c + 1）是在c的基础上加上二维数组一行的地址长度，即从&c[0][0] 变到了&c[1][0]； （\*c + 1）：\*c是列指针，（\*c + 1）是在\*c的基础上加上二数组一个元素的所占的长度，即从 &c[0][0]变到了&c[0][1] 从而（c + 1）和（\*c + 1）的值就不相等了

150、定义 int \*\*a[3][4], 则变量占有的内存空间为：\_\_32\_\_\_ 参考答案：

int \*\*p; /\*16位下sizeof(p)=2, 32位下sizeof(p)=4\*/

总共 3\*4\*sizeof(p)

151、写出判断ABCD四个表达式的是否正确, 若正确, 写出经过表达式中 a的值

int a = 4;

(A)a += (a++); (B) a += (++a) ;(C) (a++) += a;(D) (++a) += (a++);

a = ?

答：C错误，左侧不是一个有效变量，不能赋值，可改为(++a) += a;

改后答案依次为9,10,10,11

152、某32位系统下, C++程序，请计算sizeof 的值

char str[] = “http://www.ibegroup.com/”

char \*p = str ;

int n = 10;

请计算

（1）sizeof (str ) = ？

（2）sizeof ( p ) = ？

（3）sizeof ( n ) = ？

void Foo ( char str[100]){

请计算

sizeof( str ) = ？（4）

}

void \*p = malloc( 100 );

请计算

sizeof ( p ) = ？（5）

答：（1）25 （2）4 （3） 4 （4）4 （5）4

153、 回答下面的问题

(1).Void GetMemory(char \*\*p, int num){

\*p = (char \*)malloc(num);

}

void Test(void){

char \*str = NULL;

GetMemory(&str, 100);

strcpy(str, “hello”);

printf(str);

}

请问运行Test 函数会有什么样的结果？

答：输出“hello”

154、 void Test(void)

{

char \*str = (char \*) malloc(100);

strcpy(str, “hello”);

free(str);

if(str != NULL){

strcpy(str, “world”);

printf(str);

}

}

请问运行Test 函数会有什么样的结果？

答：输出“world”

155、 char \*GetMemory(void){

char p[] = “hello world”;

return p;

}

void Test(void){

char \*str = NULL;

str = GetMemory();

printf(str);

}

请问运行Test 函数会有什么样的结果？

答：无效的指针，输出不确定

160、函数模板与类模板有什么区别？

答：函数模板的实例化是由编译程序在处理函数调用时自动完成的，而类模板的实例化必须由程序员在程序中显式地指定。

### 161、 int i=10, j=10, k=3; k\*=i+j; k最后的值是？

答：60，此题考察优先级，实际写成： k\*=(i+j);，赋值运算符优先级最低

### 163、请写出下列代码的输出内容

#i nclude

main()

{

int a,b,c,d;

a=10;

b=a++;

c=++a;

d=10\*a++;

printf(“b，c，d：%d，%d，%d”，b，c，d）;

return 0;

}

答 、10，12，120

### 166 以下代码中的两个sizeof用法有问题吗？

void UpperCase( char str[] ) // 将 str 中的小写字母转换成大写字母

{

for( size\_t i=0; i<sizeof(str)/sizeof(str[0]); ++i )

if( ‘a’<=str[i] && str[i]<=’z’ )

str[i] -= (‘a’-'A’ );

}

char str[] = “aBcDe”;

cout << “str字符长度为: ” << sizeof(str)/sizeof(str[0]) << endl;

UpperCase( str );

cout << str << endl;

答：函数内的sizeof有问题。根据语法，sizeof如用于数组，只能测出静态数组的大小，无法检测动态分配的或外部数组大小。函数外的str是一个静态定义的数组，因此其大小为6，函数内的str实际只是一个指向字符串的指针，没有任何额外的与数组相关的信息，因此sizeof作用于上只将其当指针看，一个指针为4个字节，因此返回4。

### 167 非C++内建型别 A 和 B，在哪几种情况下B能隐式转化为A？

答：

a. class B : public A { ……} // B公有继承自A，可以是间接继承的

b. class B { operator A( ); } // B实现了隐式转化为A的转化

c. class A { A( const B& ); } // A实现了non-explicit的参数为B（可以有其他带默认值的参数）构造函数

d. A& operator= ( const A& ); // 赋值操作，虽不是正宗的隐式类型转换，但也可以勉强算一个

### 168.以下代码有什么问题？

struct Test

{

Test( int ) {}

Test() {}

void fun() {}

};

void main( void )

{

Test a(1);

a.fun();

Test b();

b.fun();

}

答：变量b定义出错。按默认构造函数定义对象，不需要加括号。

型。

### 172 C++中的空类，默认产生哪些类成员函数？

答：

class Empty

{

public:

Empty(); // 缺省构造函数

Empty( const Empty& ); // 拷贝构造函数

~Empty(); // 析构函数

Empty& operator=( const Empty& ); // 赋值运算符

Empty\* operator&(); // 取址运算符

const Empty\* operator&() const; // 取址运算符 const

};

173 以下两条输出语句分别输出什么？

float a =1.0f;

cout << (int)a << endl;

cout << (int&)a << endl;

cout << boolalpha << ( (int)a == (int&)a ) << endl; // 输出什么？

float b =0.0f;

cout << (int)b << endl;

cout << (int&)b << endl;

cout << boolalpha << ( (int)b == (int&)b ) << endl; // 输出什么？

答：分别输出false和true。注意转换的应用。(int)a实际上是以浮点数a为参数构造了一个整型数，该整数的值是1，(int&)a则是告诉编译器将a当作整数看（并没有做任何实质上的转换）。因为1以整数形式存放和以浮点形式存放其内存数据是不一样的，因此两者不等。对b的两种转换意义同上，但是0的整数形式和浮点形式其内存数据是一样的，因此在这种特殊情形下，两者相等（仅仅在数值意义上）。

注意，程序的输出会显示(int&)a=1065353216，这个值是怎么来的呢？前面已经说了，1以浮点数形式存放在内存中，按ieee754规定，其内容为0x0000803F（已考虑字节反序）。这也就是a这个变量所占据的内存单元的值。当(int&)a出现时，它相当于告诉它的上下文：“把这块地址当做整数看待！不要管它原来是什么。”这样，内容0x0000803F按整数解释，其值正好就是1065353216（十进制数）。

通过查看汇编代码可以证实“(int)a相当于重新构造了一个值等于a的整型数”之说，而(int&)的作用则仅仅是表达了一个类型信息，意义在于为cout<<及==选择正确的重载版

174、请简述以下两个for循环的优缺点（5分）

for (i=0; i<N; i++)

{

if (condition)

DoSomething();

else

DoOtherthing();

}

if (condition)

{

for (i=0; i<N; i++)

DoSomething();

}

else

{

for (i=0; i<N; i++)

DoOtherthing();

}

优点：程序简洁

缺点：多执行了N-1次逻辑判断，并且打断了循环“流水线”作业，使得编译器不能对循环进行优化处理，降低了效率。

优点：循环的效率高

缺点：程序不简洁

### 175 请问运行Test函数会有什么样的结果?

void GetMemory(char \*p)

{

p = (char \*)malloc(100);

}

void Test(void)

{

char \*str = NULL;

GetMemory(str);

strcpy(str, “hello world”);

printf(str);

}

请问运行Test函数会有什么样的结果？

答：程序崩溃。

因为GetMemory并不能传递动态内存，

Test函数中的 str一直都是 NULL。

strcpy(str, “hello world”);将使程序崩溃。

### 176 请问运行Test函数会有什么样的结果？

char \*GetMemory(void)

{

char p[] = “hello world”;

return p;

}

void Test(void)

{

char \*str = NULL;

str = GetMemory();

printf(str);

}

请问运行Test函数会有什么样的结果？

答：可能是乱码。

因为GetMemory返回的是指向“栈内存”的指针，该指针的地址不是 NULL，但其原现的内容已经被清除，新内容不可知。

### 177 请问运行Test函数会有什么样的结果？

void GetMemory2(char \*\*p, int num)

{

\*p = (char \*)malloc(num);

}

void Test(void)

{

char \*str = NULL;

GetMemory(&str, 100);

strcpy(str, “hello”);

printf(str);

}

请问运行Test函数会有什么样的结果？

答：

（1）能够输出hello

（2）内存泄漏

### 178 请问运行Test函数会有什么样的结果？

void Test(void)

{

char \*str = (char \*) malloc(100);

strcpy(str, “hello”);

free(str);

if(str != NULL)

{

strcpy(str, “world”);

printf(str);

}

}

请问运行Test函数会有什么样的结果？

答：篡改动态内存区的内容，后果难以预料，非常危险。

因为free(str);之后，str成为野指针，

if(str != NULL)语句不起作用。

### 179、请阅读以下一段程序，并给出答案。

class A

{

public:

A(){ doSth(); }

virtual void doSth(){printf(“I am A”);}

};

class B:public A

{

public:

virtual void doSth(){ printf(“I am B”);}

};

B b;

执行结果是什么？为什么？

答：执行结果是I am A

因为b对象构造时调用基类A的构造函数A()，得此结果。

**181．完成下列程序**

**\***

**\*.\*.**

**\*..\*..\*..**

**\*…\*…\*…\*…**

**\*….\*….\*….\*….\*….**

**\*…..\*…..\*…..\*…..\*…..\*…..**

**\*……\*……\*……\*……\*……\*……\*……**

**\*…….\*…….\*…….\*…….\*…….\*…….\*…….\*…….**

＃i nclude<iostream>

using namespace std;

const int n = 8;

main()

{

int i;

int j;

int k;

for(i = n; i >= 1; i–)

{

for(j = 0; j < n-i+1; j++)

{

cout<<”\*”;

for(k=1; k < n-i+1; k++)

{

cout<<”.”;

}

}

cout<<endl;

}

system(“pause”);

}

### 182完成程序，实现对数组的降序排序

＃include <iostream>

using namespace std;

void sort(int\* arr, int n);

int main()

{

int array[]={45,56,76,234,1,34,23,2,3};

sort(array, 9);

for(int i = 0; i <= 8; i++)//曾经在这儿出界

cout<<array[i]<<” “;

cout<<endl;

system(“pause”);

}

void sort(int\* arr, int n)

{

int temp;

for(int i = 1; i < 9; i++)

{

for(int k = 0; k < 9 – i; k++)//曾经在这儿出界

{

if(arr[k] < arr[k + 1])

{

temp = arr[k];

arr[k] = arr[k + 1];

arr[k + 1] = temp;

}

}

}

}

195.101个硬币100真、1假，真假区别在于重量。请用无砝码天平称两次给出真币重还是假币重的结论。

答:

101个先取出2堆,

33,33

第一次称,如果不相等,说明有一堆重或轻

那么把重的那堆拿下来,再放另外35个中的33

如果相等,说明假的重,如果不相等,新放上去的还是重的话,说明假的轻(不可能新放上去的轻)

第一次称,如果相等的话，这66个肯定都是真的,从这66个中取出35个来,与剩下的没称过的35个比

下面就不用说了

方法二:

第3题也可以拿A(50),B(50)比一下，一样的话拿剩下的一个和真的比一下。

如果不一样，就拿其中的一堆。比如A(50)再分成两堆25比一下，一样的话就在

B(50)中，不一样就在A(50)中，结合第一次的结果就知道了。

### 198、 sizeof应用在结构上的情况

请看下面的结构：

struct MyStruct

{

double dda1;

char dda;

int type

};

对结构MyStruct采用sizeof会出现什么结果呢？sizeof(MyStruct)为多少呢？也许你会这样求：

sizeof(MyStruct)=sizeof(double)+sizeof(char)+sizeof(int)=13

但是当在VC中测试上面结构的大小时，你会发现sizeof(MyStruct)为16。你知道为什么在VC中会得出这样一个结果吗？

其实，这是VC对变量存储的一个特殊处理。为了提高CPU的存储速度，VC对一些变量的起始地址做了”对齐”处理。在默认情况下，VC规定各成员变量存放的起始地址相对于结构的起始地址的偏移量必须为该变量的类型所占用的字节数的倍数。下面列出常用类型的对齐方式(vc6.0,32位系统)。

类型

对齐方式（变量存放的起始地址相对于结构的起始地址的偏移量）

Char

偏移量必须为sizeof(char)即1的倍数

int

偏移量必须为sizeof(int)即4的倍数

float

偏移量必须为sizeof(float)即4的倍数

double

偏移量必须为sizeof(double)即8的倍数

Short

偏移量必须为sizeof(short)即2的倍数

各成员变量在存放的时候根据在结构中出现的顺序依次申请空间，同时按照上面的对齐方式调整位置，空缺的字节VC会自动填充。同时VC为了确保结构的大小为结构的字节边界数（即该结构中占用最大空间的类型所占用的字节数）的倍?

### 205.写出运行结果:

{

char str[] = “world”; cout << sizeof(str) << “: “;

char \*p = str;  cout << sizeof(p) << “: “;

char i  = 10;cout << sizeof(i) << “: “;

void \*pp= malloc(10); cout << sizeof(p) << endl;

}

6：4：1：4

### 212输出下面程序结果。

#include <iostream.h>

class A

{

public:

virtual void print(void)

{

cout << "A::print()" << endl;

}

};

class B :public A

{

public:

virtual void print(void)

{

cout << "B::print()" << endl;

};

};

class C :public A

{

public:

virtual void print(void)

{

cout << "C::print()" << endl;

}

};

void print(A a) //只有通过基类引用或指针调用派生类才会产生多态

{

a.print();

}

int main(void)

{

A a, \*pa, \*pb, \*pc;

B b;

C c;

pa = &a;

pb = &b;

pc = &c;

a.print();//A

b.print();//B

c.print();//C

pa->print();//A

pb->print();//B

pc->print();//C

print(a); //A

print(b); //A

print(c); //A

system("pause");

return 0;

}

220一个函数的函数体就是一条\_\_\_\_复合\_\_\_\_\_语句。

221. 当执行cin语句时，从键盘上输入每个数据后必须接着输入一个\_\_\_空白\_\_\_\_\_符，然后才能继续输入下一个数据。

243. signed char类型的值域范围是\_\_-128\_\_至\_\_\_+127 \_\_之间的整数。

244. int和float类型的数据分别占用\_\_\_ 4\_\_\_和\_\_\_\_ 4\_\_\_个字节。

245. float和double类型的数据分别占用\_\_\_\_ 4\_\_\_和\_\_\_\_\_8\_\_\_个字节。

246. bool和char类型的数据分别占用\_\_\_\_\_1\_\_\_\_和\_\_\_\_1\_\_\_个字节。

252. 假定x和y为整型，其值分别为16和5，则x%y和x/y的值分别为\_\_\_1\_\_\_\_\_\_\_和\_\_\_\_3\_\_\_\_。

253. 假定x和y为整型，其值分别为16和5，则x/y和double(x)/y的值分别为\_\_\_\_3\_\_\_\_和\_\_\_3.2\_\_\_\_。

284. 存储字符’a’和字符串”a”分别需要占用\_\_\_\_\_1\_\_\_和\_\_\_\_2 \_\_\_个字节。

288. C++支持两种多态性：\_\_\_编译\_\_\_\_时的多态性和\_\_\_\_运行\_\_时的多态性。

289. 在C++中，编译时的多态性是通过\_\_\_重载\_\_\_实现的，而运行时的多态性则是通过\_\_\_虚函数\_\_\_\_实现的。

290. 对于类中定义的任何成员，其隐含访问权限为\_\_\_ private（或私有）\_\_。

291. 对于结构提中定义的任何成员，其隐含访问权限为\_\_ public(或公有)\_。

292. 若在类的定义体中给出了一个成员函数的完整定义，则该函数属于\_\_内联\_\_函数。

293. 为了避免在调用成员函数时修改对象中的任何数据成员，则应在定义该成员函数时，在函数头的后面加上\_\_ const \_\_关键字。

294. 若只需要通过一个成员函数读取数据成员的值，而不需要修改它，则应在函数头的后面加上\_\_ const \_\_关键字。

### 296．.写出判断ABCD四个表达式的是否正确, 若正确, 写出经过表达式中 a的值(3分)

int a = 4;

(A)a += (a++); (B) a += (++a) ;(C) (a++) += a;(D) (++a) += (a++);

a = ?

答：C错误，左侧不是一个有效变量，不能赋值，可改为(++a) += a;

改后答案依次为9,10,10,11

### 300．设有以下说明和定义：

typedef union {long i; int k[5]; char c;} DATE;

struct data { int cat; DATE cow; double dog;} too;

DATE max;

则语句 printf(“%d”,sizeof(struct date)+sizeof(max));的执行结果是？

答 、结果是：\_\_\_52\_\_\_\_。DATE是一个union, 变量公用空间. 里面最大的变量类型是int[5], 占用20个字节. 所以它的大小是20

data是一个struct, 每个变量分开占用空间. 依次为int4 + DATE20 + double8 = 32.

所以结果是 20 + 32 = 52.

当然…在某些16位编辑器下, int可能是2字节,那么结果是 int2 + DATE10 + double8 = 20

302．以下代码能够编译通过吗，为什么？

unsigned int const size1 = 2;

char str1[ size1 ];

unsigned int temp = 0;

cin >> temp;

unsigned int const size2 = temp;

char str2[ size2 ];

答：str2定义出错，size2非编译器期间常量，而数组定义要求长度必须为编译期常量。

### 303．以下反向遍历array数组的方法有什么错误？

vector array;

array.push\_back( 1 );

array.push\_back( 2 );

array.push\_back( 3 );

for( vector::size\_type i=array.size()-1; i>=0; –i ) // 反向遍历array数组

{

cout << array[i] << endl;

}

答：首先数组定义有误，应加上类型参数：vector<int> array。其次vector::size\_type被定义为unsigned int，即无符号数，这样做为循环变量的i为0时再减1就会变成最大的整数，导致循环失去控制。

### 304. 以下代码中的输出语句输出0吗，为什么？

struct CLS

{

int m\_i;

CLS( int i ) : m\_i(i) {}

CLS()

{

CLS(0);

cout<<m\_i;//同理这里输出也不为0

}

};

CLS obj;

cout << obj.m\_i << endl;

答：不能。在默认构造函数内部再调用带参的构造函数属用户行为而非编译器行为，亦即仅执行函数调用，而不会执行其后的初始化表达式。只有在生成对象时，初始化表达式才会随相应的构造函数一起调用。

