最新版：
http://docs.google.com/Doc?id=dc23x53c_27gp6f3kgq
需要登录 google 帐号。

/****************
* GPL v3
* 答案非官方
*如果想要修改权限，联系颜开

* 未标注内容来自 PPT
****************/

1 写出 C 和 C++的关系，并说明应注意的问题。10%
答：
C 和 C++的关系：
C++ 支持 C 所支持的全部编程技巧 ；
任何 C 程序都能被 C++ 用基本相同的方法编写，并具备相同的运行效率和空间；
完全包含 C 语言成分（C 的超集）；
C++ 由 4 个部分有机组成：
C 语言；OOP 面向对象（添加了 OOP 的完全支持）；STL 标准模板库；Inside-model 内置模型。\\xjj
C 和 C++混合编程:
name mangling：extern “C” { … }。
Static Initialization。静态初始化
Dynamic Memory Allocation（分配）
new / delete
malloc / free。
Compatibility（适合） of data structure。

2 给给出影响表达式求值的因素。10%
答：
操作数、操作符和标点符号组成的序列，表示一个计算过程
优先级 a+b*c 解决相邻两个运算符的运算顺序
结合性 a+b-c
求值次序 (a+b)*(a-b) 解决不相邻的两个运算符的运算顺序 与编译系统有关
类型转换 int x=10; float y=2.0; x*y
在求值次序上：操作符的副作用：改变操作数的值 （X+1）*(++X)

3 请举例说明 C++多态的几种表现方式。10%
答：
多态的一般含义是：某一论域中的一个元素存在多种解释。

虚函数：
只有类的成员函数才可以是虚函数
静态成员函数不可以是虚函数
构造函数不可以是虚函数
析构函数可以（往往）是虚函数
当用指针或者引用访问时采用动态绑定；或者子类对象访问时采用动态绑定
函数达到可对基类函数进行覆盖的目的
一名多用：

函数重载:
在同一个作用域中，相同的标识符可以用于定义不同的函数，但要求这些函数应拥有不同的参数（参数类型或个数）
函数重载主要用于定义多个功能相同而参数不同的函数
在 C++中，对重载函数的绑定采用静态绑定，由编译系统根据实参与形参的匹配实现

操作符重载:
动机
语言提供的操作符只定义了针对基本数据类型操作的语义
操作符重载机制提供了对自定义数据类型进行操作的语义描述手段
作用
提高程序的可读性
提高语言的灵活性、可扩充性
类属性：
模板:
源代码复用机制，实际上为提供了一系列的重载函数
参数化模块
对程序模块（如：类、函数）加上类型参数
对不同类型的数据实施相同的操作
多态的一种形式

C++
类属函数
类属类
在面向对象程序设计中，还有下面的多态：
消息的多态。一个公共消息集可以有多种解释。
对象类型的多态，子类对象既属于子类，也属于父类。
对象标识的多态，父类的应用或者指针可以引用或指向子类对象。

多态性可以严格的分为四类：重载多态，强制多态，包含多态，和参数多态，前面两种统称为专用多态，而后面两种也称为通用多态。//xjj

4 何为引用 ？其主要作用是什么？何时需要将一个函数的返回值类型定义为引用类型？

如果随意将函数的返回值类型定义为引用类型会造成什么危害？10%
答：
引用定义： C++提供了引用类型，通过引用类型可以定义一个变量，它与另一个变量占用相同的内存空间；或为一个变量取一个别名。
引用作用：引用主要用于函数的形式参数和动态变量名。//C++ Primer
返回引用：当函数返回引用类型时，没有复制返回值。相反,返回对象本身。 //C++ Primer
如果函数返回值的类型是引用或指针类型，则函数不应该把局部量或局部量的地址作为返回值返回

5 如何利用析构函数防止内存漏洞？举例说明。10%
析构函数:
~?类名？ 无参数和无返回类型的成员函数
对象消亡时，在系统收回他所占的存储空间之前，系统将自动调用析构函数
主要作用：一般情况下不需要定义析构函数，但是如果对象在创建后申请了一些资源并且没有归还这些资源，则应定义析构函数来在对象消亡时归还对象申请的资源
在对象创建时，系统会为对象分配一块存储空间来存储对象的数据成员，但对于指针类型的数据成员来说，系统只分配了存储该指针所需要的空间，而没有分配指针所指向的空间，对象自己需要申请（作为资源）。同样，在对象消亡时，系统收回的只是指针成员本身的存储空间，而指针所指向的空间需要对象自己归还（作为资源）。

```
class String
{          char *str;
   public:
          String()        {   str = NULL; }
          String(char *p)
      { str = new char[strlen(p)+1];   strcpy(str,p); }
          ~String()
          {   delete []str; }
}
```

当 String 对象创建初始化后，会有 char* str 对象在堆中。如果没有析构函数，但对象会撤销，char* str 仍在堆中，造成内存泄露。而用析构函数，可以在对象撤销时删除 str，防止内存泄露。//自己写的。

6 请说明 C++设计类依附于什么原则将你所定义的成员函数定义为纯虚函数、虚函数或非虚函数？5%
纯虚函数：只给出函数声明而没有给出函数实现的虚成员函数。
只有函数接口会被继承
子类必须继承函数接口
（必须）提供实现代码

一般虚函数
函数的接口及缺省实现代码都会被继承
子类必须继承函数接口
可以继承缺省实现代码

非虚函数
函数的接口和其实现代码都会被继承
必须同时继承接口和实现代码

7 请给出你认为最有价值的 C++程序设计应该遵守的 5 条原则，并简明分析其意义所在。10%

Use const whenever possible:
Guard against potential ambiguity
Never treat arrays polymorphically
Make non-leaf classes abstract
Strive for exception-safe code
Use destructor to prevent resource leaks

8 编写函数 int count_word(const char *text,const char*word)来统计一个英语文本（由参数 text 指向）中的某个单词（由 word 指向）出现的次数。例如：函数调用 count——word（“the theater is showing the film Gone With The Wind ”,"the"）返回值为 3. 10%

```
// Author : yankai
bool beginWith(const char *text, const char* word) {
	while (*word != '\0') {
		if (*word != *text) {
			return false;
		}
		++text;
		++word;
	}
	return true;
}

int count_word(const char *text, const char* word) {
	int count = 0;
	while (*text != '\0') {
		if (beginWith(text, word))
			++count;
		++text;
	}
	return count;
}
```

9 定义一个时间类 CTime，它表示时分秒，并能实现以下程序段所需的功能。25%

```
CTime t1;//t1 表示的时间为 0 时 0 分 0 秒
CTime t2;//t2 表示的时间为 18 时 10 分 40 秒
int s;cin??s;
t1=t2+s//把 t1 的时间设为 t2 表示的时间加 S 秒
```

```
    cout??t1??endl;//输出时间格式时分秒
    cout??t1-t2??endl;//输出时间差


//CTime.h
 #ifndef CTIME_H_
 #define CTIME_H_

#include ?iostream?
 using namespace std;

class CTime {
 private:
       long time;

public:
       CTime();

      CTime(long);

      CTime(long,long,long);

      CTime& operator+=(const CTime&);

      CTime& operator-=(const CTime&);

      friend CTime operator+(const CTime& cTime, const CTime& buffer);

      friend CTime operator-(const CTime& cTime, const CTime& buffer);

      friend std::ostream& operator??(std::ostream&, const CTime&);

};

#endif /* CTIME_H_ */



//CTime.cpp
 #include "CTime.h"
 #include ?iostream?

CTime::CTime() :
       time(0) {
```

```
 }

CTime::CTime(long l) :
        time(l) {
 }

CTime::CTime(long h, long m, long s) {
        m += 60 * h;
        s += 60 * m;
        time = s;
 }

CTime& CTime::operator+=(const CTime& buffer) {
        time += buffer.time;
        return *this;
 }

CTime& CTime::operator-=(const CTime& buffer) {
        time -= buffer.time;
        return *this;
 }

CTime operator+(const CTime& cTime, const CTime& buffer) {
        CTime newOne(cTime);
        newOne += buffer;
        return newOne;
 }

CTime operator-(const CTime& cTime, const CTime& buffer) {
        CTime newOne(cTime);
        newOne -= buffer;
        return newOne;
 }

std::ostream& operator??(std::ostream& out, const CTime& cTime) {
        long m = cTime.time / 60;
        long h = m / 60;
        m %= 60;
        long s = cTime.time % 60;
        out ?? h ?? ":" ?? m ?? ":" ?? s;
        return out;
 }

int main(int argc, char **argv) {
```

```
    CTime t1;//t1 表示的时间为 0 时 0 分 0 秒
    CTime t2(18, 10, 40);//t2 表示的时间为 18 时 10 分 40 秒
    cout ?? t1 ?? endl;
    cout ?? t2 ?? endl;
    int s;
    cin ?? s;
    t1 = t2 + s;//把 t1 的时间设为 t2 表示的时间加 S 秒
    cout ?? t1 ?? endl;//输出时间格式时分秒
    cout ?? t1 - t2 ?? endl;//输出时间差
}
```

引用类型与指针类型相比，其优势在哪里？
答：引用类型与指针类型都可以实现通过一个变量访问另一个变量，但访问的语法形式不同：引用是
采用直接访问形式，指针则采用间接访问形式。
在作为函数参数类型时，引用类型参数的实参是一个变量，而指针类型参数的实参是一个变量的
地址。
除了在定义时指定的被引用变量外，引用类型变量不能再引用其他变量；而指针变量定义后可以
指向其他同类型的变量。因此，引用类型比指针类型要安全。
大多数编译程序往往把引用类型作为指针类型来实现，它对使用者而言是透明的。(igroo.com 更新中)