***注：请不要在本试题上作答，答案填写在另外的答题纸上。***

**分布式数据库研发中心 C++ 试题**

***注意事项：***

1. ***本试题为闭卷测试，请勿使用互联网或书籍。***
2. ***答题者请基于32位元操作系统的特性回答所有题目***
3. ***答题者碰到题目的答案是输出部分结果后崩溃,必须回答该部分结果的内容***
4. ***答题者若认为答案跟编译器设计有相依性,请注明是基于哪个编译器作答***
5. **在 C++中，struct 和 class 在定义或操作特性上有什么不同？**
6. 默认的继承访问权限不同，struct是public，class是private。
7. struct可以认为是数据结构的实现体，它的默认数据访问控制是public.而class作为对象的实现体，他默认成员访问控制是private。
8. class 这个关键字还用于定义模板参数，就像“typename”,但struct不用于定义模板参数。
9. **当一个类A 中没有声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，请试著解释为什么编译器没有让它为零？**

答：为1，这个是标准规定的，标准禁止对象大小为0，因为不同的对象需要不同的地址表示

1. **运行下图中代码，输出的结果是什么？这段代码有什么问题？**

**#include <iostream>**

**class A {**

**public:**

**A() {**

**std::cout << "A is created." << std::endl;**

**Print();**

**}**

**~A() {**

**std::cout << "A is deleted." << std::endl;**

**}**

**virtual void Print() {**

**std::cout << "A::Print called." << std::endl;**

**}**

**};**

**class B : public A {**

**public:**

**B() {**

**std::cout << "B is created." << std::endl;**

**Print();**

**}**

**~B() {**

**std::cout << "B is deleted." << std::endl;**

**}**

**virtual void Print() {**

**std::cout << "B::Print called." << std::endl;**

**}**

**};**

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

**A\* pA = new B();**

**delete pA;**

**return 0;**

**}**

**答：A is created**

**A ::Print Called**

**B is created**

**B::Print called**

**A is deleted**

问题：B的析构函数没有执行，因为A定义虚构函数没有加virtual,会存在内存泄漏

1. **下面的C++代码，能否正常编译执行？如果能，运行结果是什么？如果不能，请说明原因。**

**class A {**

**private:**

**int m\_value;**

**public:**

**A(int value) {**

**m\_value = value;**

**}**

**void Print1() {**

**printf("hello world, print1");**

**}**

**virtual void Print2() {**

**printf("hello world, print2");**

**}**

**void Print3() {**

**printf("%d", m\_value);**

**}**

**};**

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

**A\* pA = NULL;**

**pA->Print1();**

**pA->Print2();**

**pA->Print3();**

**return 0;**

**}**

**答：不能，**

**1，没有头文件#include<stdio.h>**

**2, pA->Print2()出错，因为虚函数是动态绑定的，而A\*pA=NULL没有绑定。**

1. **如下的C 代码，能否正常运行？如果能，输出是什么？如果不能，请说明原因。**

**#include <iostream>**

**class A {**

**private:**

**int n1;**

**int n2;**

**public:**

**A(): n2(0), n1(n2 + 2) {}**

**void Print() {**

**std::cout << "n1: " << n1 << ", n2: " << n2 << std::endl;**

**}**

**};**

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

**A a;**

**a.Print(); return 0;**

**}**

答：可以正常运行

结果：n1:2，n2:0

1. **如下的C 代码，能否正常运行？如果能，输出是什么？如果不能，请说明原因。**

**int SizeOf(char pString[]) {**

**return sizeof(pString);**

**}**

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

**char\* pString1 = "BONC\_TEST";**

**int size1 = sizeof(pString1);**

**int size2 = sizeof(\*pString1);**

**char pString2[100] = "BONC\_TEST";**

**int size3 = sizeof(pString2);**

**int size4 = SizeOf(pString2);**

**printf("%d, %d, %d, %d", size1, size2, size3, size4); return 0;**

**}**

**答：4,1,100,4**

1. **说明一下线程的基本概念，多线程同步和互斥有何异同以及在什么情况下分别使用他们？举例说明。**

**线程的基本概念**：线程是程序执行的最小单元，由线程ID，当前指令指针，堆栈，寄存器组合，

也是进程中的一个实体，拥有就绪，阻塞，运行三种状态

**线程同步**是指线程之间所具有的一种制约关系，一个线程的执行依赖另一个线程的消息，

当它没有得到另一个线程的消息时应等待，直到消息到达时才被唤醒。

**线程互斥**是指对于共享的进程系统资源，在各单个线程访问时的排它性。当有若干个线程都要使用某一共享资源时，任何时刻最多只允许一个线程去使用，其它要使用该资源的线程必须等待，直到占用资源者释放该资源。线程互斥可以看成是一种特殊的线程同步。

线程互斥是保护临界资源，线程同步是协调线程，例如判断缓冲区是否满。

1. **下列C++代码，能否正常编译运行？如果能，结果是什么？如果不能，为什么？**

**struct Point3D {**

**int x;**

**int y;**

**int z;**

**};**

**int \_tmain(int argc, \_TCHAR\* argv[]) {**

**Point3D\* pPoint = NULL;**

**int offset = (int)(&(pPoint)->z);**

**printf("%d", offset);**

**return 0;**

**}**

**答：不能，&(pPoint)->z 为int\*不能转成int**

1. **以下代码能否正常编译执行？如果能，请描述其功能，并给出示例；如果不能，请说明原因。**

**int main() {**

**srand(time(0));**

**vector<int> intList(4, 0);**

**intList.reserve(4);**

**vector<int>::iterator ivBegin = intList.begin();**

**vector<int>::iterator ivEnd = intList.end();**

**for (vector<int>::iterator iter = ivBegin;**

**iter != ivEnd; iter++) {**

**\*iter = 100 + (rand()%10);**

**}**

**for (vector<int>::iterator iter = ivBegin;**

**iter != ivEnd; iter++) {**

**if (\*iter == 102) {**

**intList.insert(iter, 121);**

**} else if (\*iter == 100) {**

**intList.erase(iter);**

**}**

**}**

**for (int i = 0; i < intList.size(); i++) {**

**cout << "value\t" << i << "\t= " << intList[i] << endl;**

**}**

**return 0;**

**}**

**答：**

value 0 = 121

value 1 = 102

value 2 = 106

value 3 = 103

value 4 = 102

1. **实现一个二分查找函数**

**(1)函数原型定义**

**int bin\_search(int \*array, int array\_size, int key)**

**(2)函数入参说明**

**参数 array 是按从小到大排序的数组**

**参数 array\_size 是array的有效长度**

**参数 key 是要查找的数值**

**(3)函数返回值说明：**

**在array中找出与key相等的值，并返回它的下标**

|  |
| --- |
| int bin\_search(int \*array, int array\_size, int key)  {  // 二查找，数据已经排好序。  // 1，先比较中间如果相等，找到  // 2，如果小于，在左边  // 3，如果大于，在在右边  // 4，重复1,2,3,直到找到  if (array == nullptr || array\_size == 0) return -1;  int min = 0, max = array\_size - 1;  int mid = (max + min) / 2;  while (max >= min)  {  if (key == array[mid]) return mid;  else if (key > array[mid])  {  min = mid + 1;  }  else  {  max = mid - 1;  }  mid = (max + min) / 2;  //std::cout << "max: " << max << " min: " << min << std::endl;  }  return -1;  }  int main()  {  int array\_size = 7;  int array[7] = {1, 4, 6, 8, 9, 16, 24};  int key = 16;  int index = bin\_search(array, array\_size, key);  cout << key<<"'s index is : " << index << endl;  } |

1. **以下两题选择一题进行作答。**

**1）XXX 公司可以在同一时刻生产 Model A，Model B， Model C 三种型号的汽 车。**

**其中 Model A 为汽油车，行驶 100 公里需消耗 10±2L 汽油；Model B 为柴油车，**

**行驶 100 公里需消耗 8±1L 柴油；Model C 为电动汽车，行驶 100 公里需消耗**

**25±2 度电。注：同一辆车，不同的 100 公里消耗量也不相同。 每一辆电动汽车**

**都给出一个唯一的编号，且三种型号编号规则一致，不允许通过编号规则分辨出具**

**体车型。每种电动汽车都有各自的售价、最快速度等信息。 请设计相关的类，并模**

**拟生产 10 台车型随机的汽车，并行驶 1000 公里，统计总能源消耗量，总售价。**

1. **某种银行卡有持有人、卡号、余额、VIP 等级信息，可以多人同时进行查询余额、**

**存款、消费、取现操作。要求银行卡的持有人信息不能变更；卡号由系统自动生成；**

**VIP等级可以由银行方面进行人工变更；余额在银行卡进行转账、消费等操作时，自动**

**变更，且余额不能为负数。请实现该银行卡类，要求支持银行卡开卡、频繁使用、销毁**

**等场景。**

|  |
| --- |
| #include<string>  #include<iostream>  #include <cstdlib>  #include<windows.h>  #include <ctime>  using namespace std;  HANDLE hMutex; //定义互斥对象句柄  class Card  {  private:  /\* 信息 \*/  double money;  string name;  int cardNums;  string VIPLevel;  public:  Card(string name,string VIPLevel,double money)  {  this->money = money;  this->name = name;  this->VIPLevel = VIPLevel;  srand((int)time(0));  cardNums = rand()%100000;  }  // 取款  double getMoney(int m)  {  if (m<0)  {  return m;  }  if (this->money<=0)  {  return 0;  }  WaitForSingleObject(hMutex, INFINITE);  this->money = this->money-m;  ReleaseMutex(hMutex);  return m;    }  // 查询余额  double queryMoney()  {  return this->money;  }  // 存款  void saveMoney(int m )  {  if(m<=0)  {  return;  }  WaitForSingleObject(hMutex, INFINITE);  this->money = this->money+m;  ReleaseMutex(hMutex);  }  // 销毁  // 设置VIPlevel  void setVIPLevel(string VIPlevel)  {  this->VIPLevel = VIPlevel;  }  ~Card();  }; |

1. **集合操作算法**

**数据源定义**

**S1 : { a, b, c, d, d, a, b }**

**S2 : { b, d, e, f, b }**

**…**

**…**

**Sn : { … … }**

**EnterS 函数**

**Talking a set as input and storing the information in the data structure that you are going to define.**

**PrintS 函数**

**Taking one of { UA, U, I } as input, visiting the data structure that you defined and …**

**-UA ( Union All 全并 ) : printing out all elements in all sets that have been entered.**

**-U ( Union 并集 ) : printing out all unique elements in all sets that have been entered.**

**-I ( Intersect 交集 ) : printing out the intersection of the sets.**

**For example**

**EnterS( S1 ) -> enter the info into the data structure that you defined.**

**EnterS( S2 ) -> enter the info into the data structure that you defined.**

**PrintS( UA ) : { a, b, c, d, d, a, b, b, d, e, f, b }**

**PrintS( U ) : { a, b, c, d, e, f }**

**PrintS( I ) : { b, d }**

**Note :**

**a)a, b, c, d, e, f 等是集合的元素，不代表英文字母 ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’，它们可能代表任意的数据结构。**

**b)集合的操作无须考虑顺序，打印不需依赖顺序。**

**试题：**

**试定义出 EnterS 及 PrintS 所需要的核心数据结构与操作算法，需考虑大数据量时的完整性、合理性与可执行性。**

|  |
| --- |
| // **a, b, c, d, e, f 等是集合的元素，不代表英文字母 ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’，它们可能代表任意的数据结构 ,则数据结构定义为struct** Item，并重载比较运算符。  // 调用库函数iterator set\_intersection和set\_difference 计算并集  #include "stdafx.h"  #include<vector>  #include<algorithm>  #include<iostream>  #include <string>  #include <iterator>  using namespace std;  struct Item {  string group;  string md5;  Item(const string &g, const string &m) {  group = g;  md5 = m;  }  bool operator<(const Item &\_I) const {  if (group == \_I.group) {  return md5 < \_I.md5;  }  return group < \_I.group;  }  };  #EnterS  void EnterS()  {  }  void PrintS(vector<Item> mysets,string info){  cout << info<< endl;  for (auto i : mysets)  cout << "group:" << i.group << " md5:" << i.md5 << endl;  }  int main()  {  Item i1("1", "111");  Item i2("1", "222");  Item i3("1", "333");  Item i4("2", "110");  Item i5("2", "220");  vector<Item> list;  list.push\_back(i4);  list.push\_back(i1);  list.push\_back(i2);  list.push\_back(i5);  list.push\_back(i3);  Item i6("3", "330");  Item i7("4", "440");  vector<Item> list1 = { i2, i4, i6, i7 };  sort(list.begin(), list.end());  sort(list1.begin(), list1.end());  PrintS(list, "list1:###");  PrintS(list1, "list2:###");  vector<Item> item\_intersection;  set\_intersection(list.begin(), list.end(), list1.begin(), list1.end(), back\_inserter(item\_intersection));  PrintS(item\_intersection, "list1 和 list2 的交集: ###");  vector<Item> item\_difference;  set\_difference(list.begin(), list.end(), item\_intersection.begin(), item\_intersection.end(), back\_inserter(item\_difference));  PrintS(item\_difference, "list1中有，而list2中没有的元素：###");  return 0;  } |