1.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?  
virtual修饰符会被隐形继承的。private 也被集成，只事派生类没有访问权限而已。virtual可加可不加。子类的空间里有父类的所有变量(static除外)。同一个函数只存在一个实体(inline除外)。子类覆盖它的函数不加virtual ,也能实现多态。在子类的空间里，有父类的私有变量。私有变量不能直接访问。  
————————————————————————–  
2.输入一个字符串，将其逆序后输出。（使用C++，不建议用伪码）  
#include <iostream>  
using namespace std;  
void main()  
{  
char a[50];memset(a,0,sizeof(a));  
int i=0,j;  
char t;  
cin.getline(a,50,’\n’);  
for(i=0,j=strlen(a)-1;i<strlen(a)/2;i++,j–)  
{  
t=a[i];  
a[i]=a[j];  
a[j]=t;  
}  
cout<<a<<endl;  
}  
//第二种  
string str;  
cin>>str;  
str.replace;  
cout<<str;  
————————————————————————–  
3.请简单描述Windows内存管理的方法。  
内存管理是操作系统中的重要部分，两三句话恐怕谁也说不清楚吧～～  
我先说个大概，希望能够抛砖引玉吧  
当程序运行时需要从内存中读出这段程序的代码。代码的位置必须在物理内存中才能被运行，由于现在的操作系统中有非常多的程序运行着，内存中不能够完全放下，所以引出了虚拟内存的概念。把哪些不常用的程序片断就放入虚拟内存，当需要用到它的时候在load入主存（物理内存）中。这个就是内存管理所要做的事。内存管理还有另外一件事需要做：计算程序片段在主存中的物理位置，以便CPU调度。  
内存管理有块式管理，页式管理，段式和段页式管理。现在常用段页式管理  
块式管理：把主存分为一大块、一大块的，当所需的程序片断不在主存时就分配一块主存空间，把程 序片断load入主存，就算所需的程序片度只有几个字节也只能把这一块分配给它。这样会造成很大的浪费，平均浪费了50％的内存空间，但时易于管理。  
页式管理：把主存分为一页一页的，每一页的空间要比一块一块的空间小很多，显然这种方法的空间利用率要比块式管理高很多。  
段式管理：把主存分为一段一段的，每一段的空间又要比一页一页的空间小很多，这种方法在空间利用率上又比页式管理高很多，但是也有另外一个缺点。一个程序片断可能会被分为几十段，这样很多时间就会被浪费在计算每一段的物理地址上（计算机最耗时间的大家都知道是I/O吧）。  
段页式管理：结合了段式管理和页式管理的优点。把主存分为若干页，每一页又分为若干段。好处就很明显，不用我多说了吧。  
各种内存管理都有它自己的方法来计算出程序片断在主存中的物理地址，其实都很相似。  
这只是一个大概而已，不足以说明内存管理的皮毛。无论哪一本操作系统书上都有详细的讲解  
————————————————————————–  
4.  
#include “stdafx.h”  
#define SQR(X) X\*X  
int main(int argc, char\* argv[])  
{  
int a = 10;  
int k = 2;  
int m = 1;  
a /= SQR(k+m)/SQR(k+m);  
printf(“%d\n”,a);  
return 0;  
}  
这道题目的结果是什么啊?  
define 只是定义而已，在编择时只是简单代换X\*X而已，并不经过算术法则的  
a /= (k+m)\*(k+m)/(k+m)\*(k+m);  
=>a /= (k+m)\*1\*(k+m);  
=>a = a/9;  
=>a = 1;  
————————————————————————–  
5.  
const 符号常量；  
(1)const char \*p  
(2)char const \*p  
(3)char \* const p  
说明上面三种描述的区别；  
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。  
(1)const char \*p  
一个指向char类型的const对象指针，p不是常量,我们可以修改p的值，使其指向不同的char，但是不能改变它指向非char对象，如：  
const char \*p;  
char c1=’a';  
char c2=’b';  
p=&c1;//ok  
p=&c2;//ok  
\*p=c1;//error  
(3)char \* const p  
此时\*p可以修改，而p不能修改。  
(4)const char \* const p  
这种是地址及指向对象都不能修改。  
————————————————————————–  
6.下面是C语言中两种if语句判断方式。请问哪种写法更好？为什么？  
int n;  
if (n == 10) // 第一种判断方式  
if (10 == n) // 第二种判断方式  
如果少了个=号,编译时就会报错,减少了出错的可能行,可以检测出是否少了=  
————————————————————————–  
7.下面的代码有什么问题？  
void DoSomeThing(…)  
{  
char\* p;  
…  
p = malloc(1024); // 分配1K的空间  
if (NULL == p)  
return;  
…  
p = realloc(p, 2048); // 空间不够，重新分配到2K  
if (NULL == p)  
return;  
…  
}  
A:  
p = malloc(1024);     应该写成： p = (char \*) malloc(1024);  
没有释放p的空间，造成内存泄漏。  
————————————————————————–  
8.下面的代码有什么问题？并请给出正确的写法。  
void DoSomeThing(char\* p)  
{  
char str[16];  
int n;  
assert(NULL != p);  
sscanf(p, “%s%d”, str, n);  
if (0 == strcmp(str, “something”))  
{  
…  
}  
}  
A:  
sscanf(p, “%s%d”, str, n);   这句该写成： sscanf(p, “%s%d”, str, &n);  
————————————————————————–  
9.下面代码有什么错误?  
Void test1()  
{  
char string[10];  
char \*str1=”0123456789″;  
strcpy(string, str1);  
}  
数组越界  
————————————————————————–  
10.下面代码有什么问题?  
Void test2()  
{  
char string[10], str1[10];  
for(i=0; i<10;i++)  
{  
str1[i] =’a';  
}  
strcpy(string, str1);  
}  
数组越界

11.下面代码有什么问题?  
Void test3(char\* str1)  
{  
char string[10];  
if(strlen(str1)<=10)  
{  
strcpy(string, str1);  
}  
}  
==数组越界  
==strcpy拷贝的结束标志是查找字符串中的\0 因此如果字符串中没有遇到\0的话 会一直复制，直到遇到\0,上面的123都因此产生越界的情况  
建议使用 strncpy 和 memcpy  
————————————————————————–  
12.下面代码有什么问题?  
#define MAX\_SRM 256  
DSN get\_SRM\_no()  
{  
static int SRM\_no; //是不是这里没赋初值？  
int I;  
for(I=0;I<MAX\_SRM;I++,SRM\_no++)  
{  
SRM\_no %= MAX\_SRM;  
if(MY\_SRM.state==IDLE)  
{  
break;  
}  
}  
if(I>=MAX\_SRM)  
return (NULL\_SRM);  
else  
return SRM\_no;  
}  
系统会初始化static int变量为0,但该值会一直保存,所谓的不可重入…  
————————————————————————–  
13.写出运行结果:  
{// test1  
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  
————————————————————————–  
14.写出运行结果:  
{// test2  
union V {  
struct X {  
unsigned char s1:2;  
unsigned char s2:3;  
unsigned char s3:3;  
} x;  
unsigned char c;  
} v;  
v.c = 100;  
printf(“%d”, v.x.s3);  
}  
3  
————————————————————————–  
15.用C++写个程序，如何判断一个操作系统是16位还是32位的？不能用sizeof()函数  
A1:  
16位的系统下，  
int i = 65536;  
cout << i; // 输出0；  
int i = 65535;  
cout << i; // 输出-1；  
32位的系统下，  
int i = 65536;  
cout << i; // 输出65536；  
int i = 65535;  
cout << i; // 输出65535；  
A2:  
int a = ~0;  
if( a>65536 )  
{  
cout<<”32 bit”<<endl;  
}  
else  
{  
cout<<”16 bit”<<endl;  
}  
————————————————————————–  
16.C和C++有什么不同?  
从机制上：c是面向过程的（但c也可以编写面向对象的程序）；c++是面向对象的，提供了类。但是，  
c++编写面向对象的程序比c容易  
从适用的方向：c适合要求代码体积小的，效率高的场合，如嵌入式；c++适合更上层的，复杂的； llinux核心大部分是c写的，因为它是系统软件，效率要求极高。  
从名称上也可以看出，c++比c多了+，说明c++是c的超集；那为什么不叫c+而叫c++呢，是因为c++比  
c来说扩充的东西太多了，所以就在c后面放上两个+；于是就成了c++  
C语言是结构化编程语言，C++是面向对象编程语言。  
C++侧重于对象而不是过程，侧重于类的设计而不是逻辑的设计。  
————————————————————————–  
17.在不用第三方参数的情况下，交换两个参数的值  
#include <stdio.h>  
void main()  
{  
int i=60;  
int j=50;  
i=i+j;  
j=i-j;  
i=i-j;  
printf(“i=%d\n”,i);  
printf(“j=%d\n”,j);  
}  
方法二：  
i^=j;  
j^=i;  
i^=j;  
方法三：  
// 用加减实现，而且不会溢出  
a = a+b-(b=a)  
————————————————————————–  
18.有关位域的面试题（为什么输出的是一个奇怪的字符）  
a.t = ‘b’;效果相当于 a.t= ‘b’ & 0xf;  
‘b’ –> 01100010  
‘b’ & 0xf –>>00000010  
所以输出Ascii码为2的特殊字符  
char t:4;就是4bit的字符变量，同样  
unsigned short i:8;就是8bit的无符号短整形变量  
————————————————————————–  
19.int i=10, j=10, k=3; k\*=i+j; k最后的值是?  
60  
————————————————————————–  
20.进程间通信的方式有?  
进程间通信的方式有 共享内存， 管道 ，Socket ，消息队列 , DDE等

21.  
struct A  
{  
char t:4;  
char k:4;  
unsigned short i:8;  
unsigned long m;  
}  
sizeof(A)=?（不考虑边界对齐）  
7  
struct CELL             // Declare CELL bit field  
{  
unsigned character : 8; // 00000000 ????????  
unsigned foreground : 3; // 00000??? 00000000  
unsigned intensity : 1; // 0000?000 00000000  
unsigned background : 3; // 0???0000 00000000  
unsigned blink      : 1; // ?0000000 00000000  
} screen[25][80];       // Array of bit fields  
二、位结构  
位结构是一种特殊的结构, 在需按位访问一个字节或字的多个位时, 位结构  
比按位运算符更加方便。  
位结构定义的一般形式为:  
struct位结构名{  
数据类型 变量名: 整型常数;  
数据类型 变量名: 整型常数;  
} 位结构变量;  
其中: 数据类型必须是int(unsigned或signed)。 整型常数必须是非负的整  
数, 范围是0~15, 表示二进制位的个数, 即表示有多少位。  
变量名是选择项, 可以不命名, 这样规定是为了排列需要。  
例如: 下面定义了一个位结构。  
struct{  
unsigned incon: 8; /\*incon占用低字节的0~7共8位\*/  
unsigned txcolor: 4;/\*txcolor占用高字节的0~3位共4位\*/  
unsigned bgcolor: 3;/\*bgcolor占用高字节的4~6位共3位\*/  
unsigned blink: 1; /\*blink占用高字节的第7位\*/  
}ch;  
位结构成员的访问与结构成员的访问相同。  
例如: 访问上例位结构中的bgcolor成员可写成:  
ch.bgcolor  
注意:  
1. 位结构中的成员可以定义为unsigned, 也可定义为signed, 但当成员长  
度为1时, 会被认为是unsigned类型。因为单个位不可能具有符号。  
2. 位结构中的成员不能使用数组和指针, 但位结构变量可以是数组和指针,  
如果是指针, 其成员访问方式同结构指针。  
3. 位结构总长度(位数), 是各个位成员定义的位数之和, 可以超过两个字  
节。  
4. 位结构成员可以与其它结构成员一起使用。  
例如:  
struct info{  
char name[8];  
int age;  
struct addr address;  
float pay;  
unsigned state: 1;  
unsigned pay: 1;  
}workers;  
上例的结构定义了关于一个工人的信息。其中有两个位结构成员, 每个位结  
构成员只有一位, 因此只占一个字节但保存了两个信息, 该字节中第一位表示工  
人的状态, 第二位表示工资是否已发放。由此可见使用位结构可以节省存贮空间。  
————————————————————————–  
22.下面的函数实现在一个固定的数上加上一个数，有什么错误，改正  
int add\_n(int n)  
{  
static int i=100;  
i+=n;  
return i;  
}  
答:  
因为static使得i的值会保留上次的值。  
去掉static就可了  
————————————————————————–  
23.下面的代码有什么问题?  
class A  
{  
public:  
A() { p=this; }  
~A() { if(p!=NULL) { delete p; p=NULL; } }  
A\* p;  
};  
答:  
会引起无限递归  
————————————————————————–  
24.  
union a {  
int a\_int1;  
double a\_double;  
int a\_int2;  
};  
typedef struct  
{  
a a1;  
char y;  
} b;  
class c  
{  
double c\_double;  
b b1;  
a a2;  
};  
输出cout<<sizeof(c)<<endl;的结果?  
答:  
VC6环境下得出的结果是32  
另:  
我(sun)在VC6.0+win2k下做过试验:  
short – 2  
int-4  
float-4  
double-8  
指针-4  
sizeof(union)，以结构里面size最大的为union的size  
———————————————————————————-  
25.i最后等于多少?  
int i = 1;  
int j = i++;  
if((i>j++) && (i++ == j)) i+=j;  
答:  
i = 5  
————————————————————————–  
26.  
unsigned short array[]={1,2,3,4,5,6,7};  
int i = 3;  
\*(array + i) = ?  
答:  
4  
————————————————————————–  
27.  
class A  
{  
virtual void func1()；  
void func2();  
}  
Class B: class A  
{  
void func1(){cout << “fun1 in class B” << endl;}  
virtual void func2(){cout << “fun2 in class B” << endl;}  
}  
A, A中的func1和B中的func2都是虚函数.  
B, A中的func1和B中的func2都不是虚函数.  
C, A中的func2是虚函数.，B中的func1不是虚函数.  
D, A中的func2不是虚函数，B中的func1是虚函数.  
答:  
A  
————————————————————————–  
28.  
数据库：抽出部门，平均工资，要求按部门的字符串顺序排序，不能含有”human resource”部门，  
employee结构如下：employee\_id, employee\_name, depart\_id,depart\_name,wage  
答:  
select depart\_name, avg(wage)  
from employee  
where depart\_name <> ‘human resource’  
group by depart\_name  
order by depart\_name  
————————————————————————–  
29.  
给定如下SQL数据库：Test(num INT(4)) 请用一条SQL语句返回num的最小值，但不许使用统计功能，如MIN，MAX等  
答:  
select top 1 num  
from Test  
order by num desc  
————————————————————————–  
30.  
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 B  
{  
public:  
virtual void print(void)  
{  
cout<<”C::print()”<<endl;  
}  
};  
void print(A a)  
{  
a.print();  
}  
void main(void)  
{  
A a, \*pa,\*pb,\*pc;  
B b;  
C c;

pa=&a;  
pb=&b;  
pc=&c;

a.print();  
b.print();  
c.print();

pa->print();  
pb->print();  
pc->print();

print(a);  
print(b);  
print(c);  
}  
A:  
A::print()  
B::print()  
C::print()  
A::print()  
B::print()  
C::print()  
A::print()  
A::print()  
A::print()