C 语言面试题大汇总之华为面试题

1、局部变量能否和全局变量重名？

答：能，局部会屏蔽全局。要用全局变量，需要使用"::" ;局部变量可以与全局变量同名，在函数内引用这个变量时，会用到同名的局部变量，而不会用到全局变量。对于有些编译器而言，在同一个函数内可以定义多个同名的局部变量，比如在两个循环体内都定义一个同名的局部变量，而那个局部变量的作用域就在那个循环体内。

2、如何引用一个已经定义过的全局变量？

答：**extern** 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，假定你将那个编写错了，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。

3、全局变量可不可以定义在可被多个.C 文件包含的头文件中？为什么？

答：可以，在不同的 C 文件中以 **static** 形式来声明同名全局变量。 可以在不同的 C 文件中声明同名的全局变量，前提是其中只能有一个 C 文件中对此变量赋初值，此时连接不会出错.

4、请写出下列代码的输出内容

```
#include  <stdio.h>
int main(void)
{
    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

5、static 全局变量与普通的全局变量有什么区别？static 局部变量和普通局部变量有什么区别？static 函数与普通函数有什么区别？

答: 1) 全局变量(外部变量)的说明之前再冠以 **static** 就构成了静态的全局变量。全局变量本身就是静态存储方式， 静态全局变量当然也是静态存储方式。 这两者在存储方式上并无不同。这两者的区别在于非静态全局变量的作用域是整个源程序， 当一个源程序由多个源文件组成时，非静态的全局变量在各个源文件中都是有效的。 而静态全局变量则限制了其作用域， 即只在定义该变量的源文件内有效， 在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用， 因此可以避免在其它源文件中引起错误。

2) 从以上分析可以看出， 把局部变量改变为静态变量后是改变了它的存储方式即改变了它的生存期。把全局变量改变为静态变量后是改变了它的作用域，限制了它的使用范围。 3) **static** 函数与普通函数作用域不同,仅在本文件。只在当前源文件中使用的函数应该说明为内部函数(static)，内部函数应该在当前源文件中说明和定义。对于可在当前源文件以外使用的函数，应该在一个头文件中说明，要使用这些函数的源文件要包含这个头文件

综上所述:

static 全局变量与普通的全局变量有什么区别：

static 全局变量只初使化一次，防止在其他文件单元中被引用;

static 局部变量和普通局部变量有什么区别：

static 局部变量只被初始化一次，下一次依据上一次结果值；

static 函数与普通函数有什么区别：

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

6、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。

7、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 data)+sizeof(max));的执行结果是：___52____

考点:区别 struct 与 union.(一般假定在 32 位机器上)

答：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

8、队列和栈有什么区别？

队列先进先出，栈后进先出

9、写出下列代码的输出内容

```
#include <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)
  {
FUNC1 p=&inc;
  int temp =p(arg1);
  fun(&temp,&arg1, arg2);
  printf("%dn",*arg2);
}
main()
{
  int a;            //局部变量 a 为 0;
  show(multi,10,&a);
```

```
  return 0;
    }
```
答：110

10、请找出下面代码中的所有错误 (题目不错,值得一看)

说明：以下代码是把一个字符串倒序，如“abcd”倒序后变为“dcba”

```
#include"string.h"
main()
{
char*src="hello,world";
char* dest=NULL;
int len=strlen(src);
dest=(char*)malloc(len);
char* d=dest;
char* s=src[len];
while(len--!=0)
d++=s--;
printf("%s",dest);
return 0;
}
```

答：

方法 1：一共有 4 个错误;

```
int main()
{
  char* src = "hello,world";
  int len = strlen(src);
  char* dest = (char*)malloc(len+1);//要为分配一个空间        char* d = dest;
  char* s = &src[len-1];          //指向最后一个字符
  while( len-- != 0 )
   *d++=*s--;
  *d = 0;          //尾部要加'\0'
  printf("%sn",dest);
  free(dest);       //  使用完，应当释放空间，以免造成内存汇泄露
  dest = NULL;       //防止产生野指针
  return 0;
  }
```

方法 2： (方法一需要额外的存储空间,效率不高.) 不错的想法

```
#include <stdio.h>
#include <string.h>
main()
{
char str[]="hello,world";
int len=strlen(str);
char t;
for(int i=0; i<len/2; i++)
{
t=str[i];
str[i]=str[len-i-1]; //小心一点
```

```
str[len-i-1]=t;
}
printf("%s",str);
return 0;
}
```

11.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?

c 用宏定义，c++用 inline

12.直接链接两个信令点的一组链路称作什么?

PPP 点到点连接

13.接入网用的是什么接口?

V5 接口

14.voip 都用了那些协议?

H.323 协议簇、SIP 协议、Skype 协议、H.248 和 MGCP 协议

15.软件测试都有那些种类?

黑盒：针对系统功能的测试

白盒：测试函数功能，各函数接口

16.确定模块的功能和模块的接口是在软件设计的那个队段完成的?

概要设计阶段

17.

```
unsigned char *p1;
unsigned long *p2;
p1=(unsigned char *)0x801000;
p2=(unsigned long *)0x810000;
```

请问 p1+5= ;

p2+5= ;

答案:0x801005(相当于加上 5 位) 0x810014(相当于加上 20 位);

选择题:

21.Ethternet 链接到 Internet 用到以下那个协议? D

A.HDLC;B.ARP;C.UDP;D.TCP;E.ID

22.属于网络层协议的是:(B C)

A.TCP;B.IP;C.ICMP;D.X.25

23.Windows 消息调度机制是:(C)

A.指令队列;B.指令堆栈;C.消息队列;D.消息堆栈;

找错题:

25.请问下面程序有什么错误?

```
int a[60][250][1000],i,j,k;
for(k=0;kMax_GT_Length)
{          return GT_Length_ERROR;
  }        ..... }
```

答: 死循环//

问答题:

29.IP Phone 的原理是什么?

IP 电话（又称 IP PHONE 或 VoIP）是建立在 IP 技术上的分组化、数字化传输技术,其基本原理是：通过语音压缩算法对语音数据进行压缩编码处理,然后把这些语音数据按 IP 等相关协议进行打包,经过 IP 网络把数据包传输到接收地,再把这些语音数据包串起来,经过解码解压处理后,恢复成原来的语音信号,从而达到由 IP 网络传送语音的目的。

30.TCP/IP 通信建立的过程怎样，端口有什么作用？

三次握手，确定是哪个应用程序使用该协议

31.1 号信令和 7 号信令有什么区别，我国某前广泛使用的是那一种？

1 号信令接续慢，但是稳定，可靠。

7 号信令的特点是：信令速度快，具有提供大量信令的潜力，具有改变和增加信令的灵活性，便于开放新业务，在通话时可以随意处理信令，成本低。目前得到广泛应用。

32.列举 5 种以上的电话新业务

如“闹钟服务”、“免干扰服务”、“热线服务”、“转移呼叫”、“遇忙回叫”、“缺席用户服务”、“追查恶意呼叫”、“三方通话”、“会议电话”、“呼出限制”、“来电显示”、“虚拟网电话”等

四.找错题:

1.请问下面程序有什么错误?

```
int a[60][250][1000],i,j,k;
for(k=0;k<=1000;k++)
for(j=0;j<250;j++)
for(i=0;i<60;i++)
a[i][j][k]=0;
```

答: 把循环语句内外换一下

2.

```
#define Max_CB 500
 void LmiQueryCSmd(Struct MSgCB * pmsg)
 {
  unsigned char ucCmdNum;
 ……
   for(ucCmdNum=0;ucCmdNum<Max_CB;ucCmdNum++)
  {
  ……;
  }
```

答: 死循环,unsigned int 的取值范围是 0~255

3.以下是求一个数的平方的程序,请找出错误:

```
 #define SQUARE(a)((a)*(a))
 int a=5;
 int b;
 b=SQUARE(a++);
```

答:结果与编译器相关,得到的可能不是平方值.

微软亚洲技术中心的面试题！！！

1. 进程和线程的差别。

答:线程是指进程内的一个执行单元,也是进程内的可调度实体.

与进程的区别:

(1)调度：线程作为调度和分配的基本单位，进程作为拥有资源的基本单位

(2)并发性：不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行

(3)拥有资源：进程是拥有资源的一个独立单位，线程不拥有系统资源，但可以访问隶属于进程的资源.

(4)系统开销：在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。

2.测试方法

答:人工测试：个人复查、抽查和会审

机器测试：黑盒测试和白盒测试

3. Heap 与 stack 的差别。

答:Heap 是堆，stack 是栈。

Stack 的空间由操作系统自动分配/释放，Heap 上的空间手动分配/释放。
Stack 空间有限，Heap 是很大的自由存储区
C 中的 malloc 函数分配的内存空间即在堆上,C++中对应的是 new 操作符。
程序在编译期对变量和函数分配内存都在栈上进行,且程序运行过程中函数调用时参数的传递也在栈上进行

4．Windows 下的内存是如何管理的？
分页管理
8．谈谈 IA32 下的分页机制
小页(4K)两级分页模式，大页(4M)一级
9．给两个变量，如何找出一个带环单链表中是什么地方出现环的？
一个递增一，一个递增二，他们指向同一个接点时就是环出现的地方
10．在 IA32 中一共有多少种办法从用户态跳到内核态？
通过调用门，从 ring3 到 ring0，中断从 ring3 到 ring0，进入 vm86 等等
11．如果只想让程序有一个实例运行，不能运行两个。像 winamp 一样，只能开一个窗口，怎样实现？
用内存映射或全局原子（互斥变量）、查找窗口句柄..
FindWindow，互斥，写标志到文件或注册表,共享内存。.
12．如何截取键盘的响应，让所有的'a'变成'b'？
答:键盘钩子 SetWindowsHookEx
14．存储过程是什么？有什么用？有什么优点？
答:我的理解就是一堆 sql 的集合，可以建立非常复杂的查询，编译运行，所以运行一次后，以后再运行速度比单独执行 SQL 快很多

15．Template 有什么特点？什么时候用？
: Template 可以独立于任何特定的类型编写代码,是泛型编程的基础.
当我们编写的类和函数能够多态的用于跨越编译时不相关的类型时,用 Template.
模板主要用于 STL 中的容器,算法,迭代器等以及模板元编程.
(C++的 template 是实现在库设计和嵌入式设计中的关键。
template 能实现抽象和效率的结合；同时 template 还能有效地防止代码膨胀)
16．谈谈 Windows DNA 结构的特点和优点。
答:Windows Distributed interNet Application Architecture（Windows 分布式应用结构，简称 Windows DNA）是微软创建新一代高适应性商业解决方案的框架，它使公司能够充分地挖掘数字神经系统的优点。Windows DNA 是第一个将 Internet、客户/服务器、和用于计算的 PC 模型结合并集成在一起的为新一类分布式计算方案而设计的应用软件体系结构
17．网络编程中设计并发服务器，使用多进程与多线程，请问有什么区别？
答:1)进程：子进程是父进程的复制品。子进程获得父进程数据空间、堆和栈的复制品。
2)线程：相对与进程而言，线程是一个更加接近与执行体的概念，它可以与同进程的其他线程共享数据，但拥有自己的栈空间，拥有独立的执行序列。
两者都可以提高程序的并发度，提高程序运行效率和响应时间。
线程和进程在使用上各有优缺点：线程执行开销小，但不利于资源管理和保护；而进程正相反。同时，线程适合于在 SMP 机器上运行，而进程则可以跨机器迁移。

思科

1. 用宏定义写出 swap（x，y）
答#define swap(x, y)
x = x + y;
y = x - y;
x = x - y;
2.数组 a[N]，存放了 1 至 N-1 个数，其中某个数重复一次。写一个函数，找出被重复的数字.时间复杂度必须为 o（N）函数原型：

```
int do_dup(int a[],int N)
```
答:
```
int do_dup(int a[],int N)    //未经调试
{
    int sun = 0;
    int sum2;
    for(int i=0;i<N;++i)
    {
     Sum+=a[i];
    }
    Sum2 = (1+N-1)*N/2;
    Return (sum-sum2);
}
```
3 一语句实现 x 是否为 2 的若干次幂的判断

答:方法 1)
```
int i = 512;
cout << boolalpha << ((i & (i - 1)) ? false : true) << endl; //位与为 0,则表示是 2 的若干次幂
```
2)
```
return  (x>>N==1);
```
4.unsigned int intvert(unsigned int x,int p,int n)实现对 x 的进行转换,p 为起始转化位,n 为需要转换的长度,假设起始点在右边.如 x=0b0001 0001,p=4,n=3 转换后 x=0b0110 0001

答:
```
unsigned int intvert(unsigned int x,int p,int n) //假定 p=4,n=3
{
  unsigned int _t = 0;
  unsigned int _a = 1;
   for(int i = 0; i < n; ++i)//循环的目的主要是-t
   {
    _t |= _a;       //位或
    _a = _a << 1;
    }
   _t = _t << p;   //转换后_t 变为 1110000
   x ^= _t;       /异或,将原来的位取反
   return x;
}
```
慧通：

1. 什么是预编译，何时需要预编译：

答: 就是指程序执行前的一些预处理工作,主要指#表示的.

何时需要预编译?

1)、总是使用不经常改动的大型代码体。

2)、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。

2. 下述三个有什么区别?
```
char * const p;
char const * p
const char *p
```
解答：
```
char * const p; //常量指针，p 的值不可以修改
char const * p；//指向常量的指针，指向的常量值不可以改
const char *p；  //和 char const *p
```
3. 解释下列输出结果
```
char str1[] = "abc";
```

```
char str2[] = "abc";
const char str3[] = "abc";
const char str4[] = "abc";
const char *str5 = "abc";
const char *str6 = "abc";
char *str7 = "abc";
char *str8 = "abc";
cout << ( str1 == str2 ) << endl;
cout << ( str3 == str4 ) << endl;
cout << ( str5 == str6 ) << endl;
cout << ( str7 == str8 ) << endl;
```

结果是：0 0 1 1

解答：str1,str2,str3,str4 是数组变量，它们有各自的内存空间；

而 str5,str6,str7,str8 是指针，它们指向相同的常量区域。

4. 以下代码中的两个 sizeof 用法有问题吗？[C 易]

```
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。

注意:数组名作为函数参数时,退化为指针.

数组名作为 sizeof()参数时,数组名不退化,因为 sizeof 不是函数.

4. 一个 32 位的机器,该机器的指针是多少位

指针是多少位只要看地址总线的位数就行了。80386 以后的机子都是 32 的数据总线。所以指针的位数就是 4 个字节了。

5. 指出下面代码的输出，并解释为什么。(不错,对地址掌握的深入挖潜)

```
main()
{
int a[5]={1,2,3,4,5};
int *ptr=(int *)(&a+1);
printf("%d,%d",*(a+1),*(ptr-1));
}
```

输出：2,5

(a+1）就是 a[1]，(ptr-1)就是 a[4],执行结果是 2，5

&a+1 不是首地址+1，系统会认为加一个 a 数组的偏移，是偏移了一个数组的大小（本例是 5 个 int）

```
int *ptr=(int *)(&a+1);
```

则 ptr 实际是&(a[5]),也就是 a+5

原因如下：
&a 是数组指针，其类型为 int (*)[5];
而指针加 1 要根据指针类型加上一定的值，
不同类型的指针+1 之后增加的大小不同
a 是长度为 5 的 int 数组指针，所以要加 5*sizeof(int)
所以 ptr 实际是 a[5]
但是 prt 与(&a+1)类型是不一样的(这点很重要)
所以 prt-1 只会减去 sizeof(int*)
a,&a 的地址是一样的，但意思不一样，a 是数组首地址，也就是 a[0]的地址，&a 是对象（数组）首地址，a+1 是数组下一元素的地址，即 a[1],&a+1 是下一个对象的地址，即 a[5].

6.请问以下代码有什么问题：

```
1).
int main()
{
char a;
char *str=&a;
strcpy(str,"hello");
printf(str);
return 0;
}
```

答;没有为 str 分配内存空间，将会发生异常
问题出在将一个字符串复制进一个字符变量指针所指地址。虽然可以正确输出结果，但因为越界进行内在读写而导致程序崩溃。
Strcpy 的在库函数 string.h 中.程序的主要错误在于越界进行内存读写导致程序崩溃//

```
2).
char* s="AAA";
printf("%s",s);
s[0]='B';
printf("%s",s);
```

有什么错?
答: "AAA"是字符串常量。s 是指针，指向这个字符串常量，所以声明 s 的时候就有问题。

```
cosnt char* s="AAA";
```

然后又因为是常量，所以对是 s[0]的赋值操作是不合法的。

1、写一个“标准”宏，这个宏输入两个参数并返回较小的一个。
答:#define Min(X, Y) ((X)>(Y)?(Y):(X)) //结尾没有;
2、嵌入式系统中经常要用到无限循环，你怎么用 C 编写死循环。
答:while(1){}或者 for(;;) //前面那个较好
3、关键字 static 的作用是什么?
答:1)定义静态局部变量,作用域从函数开始到结束.
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;
3) 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
4、关键字 const 有什么含意?
答 :1)表示常量不可以修改的变量。
2)可以修饰参数,作为输入参数.
3)修饰函数,防止以外的改动.
4)修饰类的成员函数,不改变类中的数据成员.

5、关键字 volatile 有什么含意？并举出三个不同的例子？

答: 提示编译器对象的值可能在编译器未监测到的情况下改变。

例子: 硬件时钟;多线程中被多个任务共享的变量等

6. int (*s[10])(int) 表示的是什么啊

int (*s[10])(int) 函数指针数组，每个指针指向一个 int func(int param)的函数。

1.有以下表达式：

int a=248; b=4;int const c=21;const int *d=&a;

int *const e=&b;int const *f const =&a;

请问下列表达式哪些会被编译器禁止？为什么？

答:*c=32;d=&b;*d=43;e=34;e=&a;f=0x321f;

*c 这是个什么东东，禁止

*d 说了是 const， 禁止

e = &a 说了是 const 禁止

const *f const =&a; 禁止

2.交换两个变量的值，不使用第三个变量。即 a=3,b=5,交换之后 a=5,b=3;

答:有两种解法，一种用算术算法，一种用^(异或)

```
a = a + b;
b = a - b;
a = a - b;
```

or

```
a = a^b;//  只能对 int,char..
b = a^b;
a = a^b;
```

or

```
a ^= b ^= a;
```

3.c 和 c++中的 struct 有什么不同？

: c和c++中struct的主要区别是c中的struct不可以含有成员函数，而c++中的struct可以。c++中struct和 class 的主要区别在于默认的存取权限不同，struct 默认为 public，而 class 默认为 private.

4.

```
#include <stdio.h>
#include <stdlib.h>
void getmemory(char *p)
{
p=(char *) malloc(100);
}
int main( )
{
char *str=NULL;
getmemory(str);
strcpy(p,"hello world");
printf("%s/n",str);
free(str);
return 0;
}
```

答: 程序崩溃，getmemory 中的 malloc 不能返回动态内存， free（）对 str 操作很危险

5.char szstr[10];

strcpy(szstr,"0123456789");
产生什么结果？为什么？
答;正常输出,长度不一样，会造成非法的 OS,覆盖别的内容.
6.列举几种进程的同步机制，并比较其优缺点。
答:原子操作
信号量机制
自旋锁
管程，会合，分布式系统
7.进程之间通信的途径
答 共享存储系统
消息传递系统
管道：以文件系统为基础

silver6 | 02 一月, 2007 11:41

Author：Vince
————即使你是个编程高手，你在面试前也应该要看看这套题，她也许会给你带来好运，否则你有可能后悔当初为什么没有看而跳楼自杀，这样我会很内疚的。这套题看似简单，但你未必能得高分，即使你看不懂也要把她背下来！
1 编程基础
1.1 基本概念
1. const 的理解：const char*, char const*, char*const 的区别问题几乎是 C++面试中每次 都会有的题目。 事实上这个概念谁都有只是三种声明方式非常相似很容易记混。 Bjarne 在他的 The C++ Programming Language 里面给出过一个助记的方法： 把一个声明从右向左读。
char * const cp; (* 读成 pointer to)
cp is a const pointer to char
const char * p;
p is a pointer to const char;
char const * p;
同上因为 C++里面没有 const*的运算符，所以 const 只能属于前面的类型。

2. c 指针
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。

3. 数组越界问题 (这个题目还是有点小险的)
下面这个程序执行后会有什么错误或者效果:
```
#define MAX 255
int main()
{
unsigned char A[MAX],i;
for (i=0;i<=MAX;i++)
A[i]=i;
}
```
解答：MAX=255,数组 A 的下标范围为:0..MAX-1,这是其一,其二 当 i 循环到 255 时,循环内执行:
A[255]=255;这句本身没有问题，但是返回 for (i=0;i<=MAX;i++)语句时,由于 unsigned char 的取值范围在(0..255),i++以后 i 又为 0 了..无限循环下去.

注：char 类型为一个字节，取值范围是[-128，127]，unsigned char [0 ,255]

4. C++:memset ,memcpy 和 strcpy 的根本区别？

答:#include "memory.h"

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' '或''；例:char a[100];memset(a, '', sizeof(a));

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；例：char a[100],b[50]; memcpy(b, a, sizeof(b));注意如用 sizeof(a)，会造成 b 的内存地址溢出。

strcpy 就只能拷贝字符串了，它遇到'\0'就结束拷贝；例：char a[100],b[50];strcpy(a,b);如用 strcpy(b,a)，要注意 a 中的字符串长度（第一个'\0'之前）是否超过 50 位，如超过，则会造成 b 的内存地址溢出。

strcpy

原型：extern char *strcpy(char *dest,char *src);

```
{
  ASSERT((dest!=NULL)&&(src!=NULL));
  Char *address = dest;
  While((*dest++=*src++)!='\0')
    Continue;
  Return dest;
}
```

用法：#include <string.h>

功能：把 src 所指由 NULL 结束的字符串复制到 dest 所指的数组中。

说明：src 和 dest 所指内存区域不可以重叠且 dest 必须有足够的空间来容纳 src 的字符串。

返回指向 dest 的指针。

memcpy

原型：extern void *memcpy(void *dest, void *src, unsigned int count);

```
{
 ASSERT((dest!=NULL)&&(src!=NULL));
 ASSERT((dest>src+count)||(src>dest+count));//防止内存重叠,也可以用 restrict 修饰指针
 Byte* bdest = (Byte*)dest;
 Byte* bsrc = (Byte*) src;
 While(count-->0)
  *bdest++ = **bsrc++;
 Return dest;
}
```

用法：#include <memory.h>

功能：由 src 所指内存区域复制 count 个字节到 dest 所指内存区域。

说明：src 和 dest 所指内存区域不能重叠，函数返回指向 dest 的指针。

Memset

原型：extern void *memset(void *buffer, char c, int count);

用法：#include

功能：把 buffer 所指内存区域的前 count 个字节设置成字符 c。

说明：返回指向 buffer 的指针。

5. ASSERT()是干什么用的

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。例如，变量 n 在程序中不应该为 0，如果为 0 可能导致错误，你可以这样写程序：

……
ASSERT(n != 0);
k = 10/ n;
……
ASSERT 只有在 Debug 版本中才有效，如果编译为 Release 版本则被忽略。
assert()的功能类似，它是 ANSI C 标准中规定的函数，它与 ASSERT 的一个重要区别是可以用在 Release 版本中。

6. system（"pause");作用?
答:系统的暂停程序，按任意键继续，屏幕会打印，"按任意键继续。。。。。" 省去了使用 getchar（）；
7. 请问 C++的类和 C 里面的 struct 有什么区别?
答:c++中的类具有成员保护功能，并且具有继承，多态这类 oo 特点，而 c 里的 struct 没有
c 里面的 struct 没有成员函数,不能继承,派生等等.
8. 请讲一讲析构函数和虚函数的用法和作用?
答:析构函数也是特殊的类成员函数，它没有返回类型，没有参数，不能随意调用，也没有重载。只是在类对象生命期结束的时候，由系统自动调用释放在构造函数中分配的资源。这种在运行时，能依据其类型确认调用那个函数的能力称为多态性，或称迟后联编。另： 析构函数一般在对象撤消前做收尾工作，比如回收内存等工作，
虚拟函数的功能是使子类可以用同名的函数对父类函数进行覆盖，并且在调用时自动调用子类覆盖函数，如果是纯虚函数，则纯粹是为了在子类覆盖时有个统一的命名而已。
注意:子类重新定义父类的虚函数的做法叫覆盖,override,而不是 overload(重载),重载的概念不属于面向对象编程,重载指的是存在多个同名函数,这些函数的参数表不同..重载是在编译期间就决定了的,是静态的,因此,重载与多态无关.与面向对象编程无关.
含有纯虚函数的类称为抽象类,不能实例化对象,主要用作接口类//

9. 全局变量和局部变量有什么区别？是怎么实现的？操作系统和编译器是怎么知道的？
答;全局变量的生命周期是整个程序运行的时间，而局部变量的生命周期则是局部函数或过程调用的时间段。其实现是由编译器在编译时采用不同内存分配方法。
全局变量在 main 函数调用后，就开始分配，
静态变量则是在 main 函数前就已经初始化了。
局部变量则是在用户栈中动态分配的（还是建议看编译原理中的活动记录这一块）
10. 8086 是多少位的系统？在数据总线上是怎么实现的？
答:8086 系统是 16 位系统，其数据总线是 20 位。

12 程序设计
1. 编写用 C 语言实现的求 n 阶阶乘问题的递归算法：
答:long int fact(int n)
{
If(n==0||n==1)
Return 1;
Else
Return n*fact(n-1);
}
2. 二分查找算法：
1) 递归方法实现：
int BSearch(elemtype a[],elemtype x,int low,int high)
/*在下届为 low，上界为 high 的数组 a 中折半查找数据元素 x*/
{

```
int mid;
if(low>high) return -1;
mid=(low+high)/2;
if(x==a[mid]) return mid;
if(x<a[mid]) return(BSearch(a,x,low,mid-1));
else return(BSearch(a,x,mid+1,high));
}
```

2) 非递归方法实现：

```
int BSearch(elemtype a[],keytype key,int n)
{
int low,high,mid;
low=0;high=n-1;
while(low<=high)
{
mid=(low+high)/2;
if(a[mid].key==key) return mid;
else if(a[mid].key<key) low=mid+1;
else high=mid-1;
}
return -1;
}
```

3. 递归计算如下递归函数的值（斐波拉契）：

```
f(1)=1
f(2)=1
f(n)=f(n-1)+f(n-2) n>2
```

解：非递归算法:

```
int f(int n)
{
int i,s,s1,s2;
s1=1;/*s1用于保存f(n-1)的值*/
s2=1;/*s2用于保存f(n-2)的值*/
s=1;
for(i=3;i<=n;i++)
{
s=s1+s2;
s2=s1;
s1=s;
}
return(s);
}
```

递归算法:

```
Int f(int n)
{
 If(n==1||n==2)
   Rerurn 1;
 Else
   Rerutn f(n-1)+f(n-2);
}
```

4. 交换两个数，不用第三块儿内存：

答:
```
int a = ……;
  int b = ……;
  a = a + b;
  b = a - b;
  a = a - b;
```

5. 冒泡排序：

答:
```
void BubbleSort(elemtype x[],int n)    //时间复杂度为 0(n*n);
 {
  int i,j;
  elemtype temp;
  for(i=1;i<n;i++)
   for(j=0;j<n-i;j++)
  {
   if(x[j].key>x[j+1].key)
     {
     temp=x[j];
     x[j]=x[j+1];
     x[j+1]=temp;
     }
  }
}
//补充一个改进的冒泡算法:
 void BubbleSort(elemtype x[],int n)
{
 Int i,j;
 BOOL exchange;  //记录交换标志
 for(i=1;i<n;++i)     //最多做 n-1 趟排序
 {
     Exchange = false;
     For(j=n-1;j>=i;--j)
     {
      If(x[j]>x[j+1])
      {
        x[0] = x[j];
        X[j] = x[j+1];
        X[j+1] = x[0];
        Exchange = true;    //发生了交换,设置标志为真.
      }
      }
      if (!Exchange )       //为发生替换,提前终止算法
        return;

 }
}
```

6. c 语言 文件读写

```
#include "stdio.h"
main()
```

```
{
FILE *fp;
char ch,filename[10];
scanf("%s",filename);
if((fp=fopen(filename,"w")==NULL)
{
 printf("cann't open filen");
 exit(0);
}
ch=getchar();
while(ch!='#')
{
fputc(ch,fp);
putchar(ch);
ch=getchar();
}
fclose(fp);
}
```

7. winsocket 编程　//这个不错

```
// 服务器代码
#include <Winsock2.h>
#include <stdio.h>
void main()
{
WORD wVersionRequested; //版本号
WSADATA  wsaData;       //数据
int err;
wVersionRequested = MAKEWORD(1,1);
err = WSAStartup(wVersionRequested,&wsaData);
if( err != 0)
{
return;
}
if(LOBYTE( wsaData.wVersion ) != 1||
HIBYTE( wsaData.wVersion) != 1)
{
WSACleanup();
return;
}
SOCKET sockSrv=socket(AF_INET,SOCK_STREAM,0); //建立套接字
SOCKADDR_IN addrSrv;
addrSrv.sin_addr.S_un.S_addr=htonl(INADDR_ANY);
addrSrv.sin_family=AF_INET;
addrSrv.sin_port=htons(6000);
bind(sockSrv,(SOCKADDR*)&addrSrv,sizeof(SOCKADDR));  //绑定端口
listen(sockSrv,5);   //转换 socket 套接子为侦听套接子
SOCKADDR_IN addrClient;
```

```
int len=sizeof(SOCKADDR);
while(1)    //无限循环
{
SOCKET sockConn=accept(sockSrv,(SOCKADDR*)&addrClient,&len);
char sendBuf[100];
sprint(sendBuf,"Welcome %s to http://www.sunxin.org/",
inet_ntoa(addrClient.sin_addr));
send(sockConn,sendBuf,strlen(sendBuf)+1,0);
char recvBuf[100];
recv(sockConn,recvBuf);
printf("%sn",recvBuf);
closesocket(sockConn);
WSACleanup();
}
}
```

注：这是 Server 端；File->New->Win32 Console Application，工程名：TcpSrv；然后，File->New->C++ Source File，文件名：TcpSrv；在该工程的 Setting 的 Link 的 Object/library modules 项要加入 ws2_32.lib

```
#include <Winsock2.h>
#include <stdio.h>
void main()
{
WORD    wVersionRequested;
WSADATA    wsaData;
int err;
wVersionRequested = MAKEWORD(1,1);
err = WSAStartup(wVersionRequested,&wsaData); //启动 winsock Dll
if( err != 0)
{
return;
}
if(LOBYTE( wsaData.wVersion ) != 1||
HIBYTE( wsaData.wVersion) != 1)
{
WSACleanup();
return;
}
SOCKET sockClient=socket(AF_INET,SOCK_STREAM,0);
SOCKADDR_IN addrSrv;
addrSrv.sin_addr.S_un.S_addr=inet_addr("127.0.0.1");
addrSrv.sin_family=AF_INET;
addrSrv.sin_port = htons(6000);
connect(sockClient,(SOCKADDR*)&addrSrv,sizeof(SOCKADDR));
char recvBuf[100];
recv(sockClient,recvBuf,100,0);
printf("%sn",recvBuf);
send(sockClient,"This is zhangsan",strlen("This is zhangsan")+1,0);
closesocket(sockClient);
```

```
WSACleanup();
}
```
注：这是 Client 端；File->New->Win32 Console Application，工程名：TcpClient；然后，File->New->C++ Source File，文件名：TcpClient；同理，在该工程的 Setting 的 Link 的 Object/library modules 项要加入 ws2_32.lib

8. 类的知识 (非常不错的一道题目)..

C++

```
#include <iostream.h>
class human
{
public:
   human(){ human_num++;}; //默认构造函数
   static int human_num;     //静态成员
   ~human()
   {
     human_num--;
     print();
   }
   void print()          //
   {
    cout<<"human num is: "<<human_num<<endl;
   }
 protected:
 private:
};
int human::human_num = 0;   //类中静态数据成员在外部定义,仅定义一次
human f1(human x)
{
   x.print();
   return x;
}
int main(int argc, char* argv[])
{
 human h1; //调用默认构造函数,human_num 变为 1
 h1.print(); //  打印 Human_man:1
 human h2 = f1(h1); //先调用函数 f1(),输出 human_num:1,而后输出 human_num 为 0,
 h2.print();//打印输出:human_num:0
 return 0;
} //依次调用两个析构函数:输出:human_num:-1,human_num:-2;
```

调用f1()后, return时会调用析构函数,故会变为0

输出：

```
1
1
0
0
－1
－2
```

－－－－－－－－－－－－－－－－－－－－－－－－－－－－

分析：

human h1; //调用构造函数,---hum_num = 1;

h1.print(); //输出:"human is 1"

human h2 = f1(h1); //再调用 f1(h1)的过程中,由于函数参数是按值传递对象,调用默认的复制构造函数,h2 并没有调用定义的构造函数.

C/C++ 程序设计员应聘常见面试试题深入剖析

silver6 | 25 九月, 2006 09:07

本文的写作目的并不在于提供 C/C++程序员求职面试指导，而旨在从技术上分析面试题的内涵。文中的大多数面试题来自各大论坛，部分试题解答也参考了网友的意见。

许多面试题看似简单，却需要深厚的基本功才能给出完美的解答。企业要求面试者写一个最简单的 strcpy 函数都可看出面试者在技术上究竟达到了怎样的程度，我们能真正写好一个 strcpy 函数吗？我们都觉得自己能，可是我们写出的 strcpy 很可能只能拿到 10 分中的 2 分。读者可从本文看到 strcpy 函数从 2 分到 10 分解答的例子，看看自己属于什么样的层次。此外，还有一些面试题考查面试者敏捷的思维能力。

分析这些面试题，本身包含很强的趣味性；而作为一名研发人员，通过对这些面试题的深入剖析则可进一步增强自身的内功。

2.找错题

试题 1：

```
void test1()
{
  char string[10];
  char* str1 = "0123456789";
  strcpy( string, str1 );
}
```

试题 2：

```
void test2()
{
  char string[10], str1[10];
  int i;
  for(i=0; i<10; i++)
  {
    str1[i] = 'a';
  }
  strcpy( string, str1 );
}
```

试题 3：

```
void test3(char* str1)
{
  char string[10];
  if( strlen( str1 ) <= 10 )
  {
    strcpy( string, str1 );
  }
```

```
}
```

解答：

试题 1 字符串 str1 需要 11 个字节才能存放下（包括末尾的''），而 string 只有 10 个字节的空间，strcpy 会导致数组越界；

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；

对试题 3，if(strlen(str1) <= 10)应改为 if(strlen(str1) < 10)，因为 strlen 的结果未统计''所占用的 1 个字节。

剖析：

考查对基本功的掌握：

(1)字符串以'\0'结尾；

(2)对数组越界把握的敏感度；

(3)库函数 strcpy 的工作方式，如果编写一个标准 strcpy 函数的总分值为 10，下面给出几个不同得分的答案：

2 分

```
void strcpy( char *strDest, char *strSrc )
{
   while( (*strDest++ = * strSrc++) != '' );
}
```

4 分

```
void strcpy( char *strDest, const char *strSrc )
//将源字符串加 const，表明其为输入参数，加 2 分
{
   while( (*strDest++ = * strSrc++) != '' );
}
```

7 分

```
void strcpy(char *strDest, const char *strSrc)
{
  //对源地址和目的地址加非 0 断言，加 3 分
  assert( (strDest != NULL) && (strSrc != NULL) );
  while( (*strDest++ = * strSrc++) != '' );
}
```

10 分

```
//为了实现链式操作，将目的地址返回，加 3 分！

char * strcpy( char *strDest, const char *strSrc )
{
  assert( (strDest != NULL) && (strSrc != NULL) );
  char *address = strDest;
```

```
  while( (*strDest++ = * strSrc++) != '' );
    return address;
}
```

从 2 分到 10 分的几个答案我们可以清楚的看到，小小的 strcpy 竟然暗藏着这么多玄机，真不是盖的！需要多么扎实的基本功才能写一个完美的 strcpy 啊！

(4)对 strlen 的掌握，它没有包括字符串末尾的''。

读者看了不同分值的 strcpy 版本，应该也可以写出一个 10 分的 strlen 函数了，完美的版本为： int strlen(const char *str) //输入参数 const

```
{
  assert( strt != NULL ); //断言字符串地址非 0
  int len;
  while( (*str++) != '' )
  {
    len++;
  }
  return len;
}
```

试题 4：

```
void GetMemory( char *p )
{
  p = (char *) malloc( 100 );
}

void Test( void )
{
  char *str = NULL;
  GetMemory( str );
  strcpy( str, "hello world" );
  printf( str );
}
```

试题 5：

```
char *GetMemory( void )
{
  char p[] = "hello world";
  return p;
}

void Test( void )
{
  char *str = NULL;
  str = GetMemory();
  printf( str );
```

```
}
```

试题 6：

```
void GetMemory( char **p, int num )
{
  *p = (char *) malloc( num );
}

void Test( void )
{
  char *str = NULL;
  GetMemory( &str, 100 );
  strcpy( str, "hello" );
  printf( str );
}
```

试题 7：

```
void Test( void )
{
  char *str = (char *) malloc( 100 );
  strcpy( str, "hello" );
  free( str );
  ... //省略的其它语句
}
```

解答：

试题 4 传入中 GetMemory(char *p)函数的形参为字符串指针，在函数内部修改形参并不能真正的改变传入形参的值，执行完

```
char *str = NULL;
GetMemory( str );
```

后的 str 仍然为 NULL；

试题 5 中

```
char p[] = "hello world";
return p;
```

的 p[]数组为函数内的局部自动变量，在函数返回后，内存已经被释放。这是许多程序员常犯的错误，其根源在于不理解变量的生存期。

试题 6 的 GetMemory 避免了试题 4 的问题，传入 GetMemory 的参数为字符串指针的指针，但是在 GetMemory 中执行申请内存及赋值语句

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

后未判断内存是否申请成功，应加上：

```
if ( *p == NULL )
```

```
{
  ...//进行申请内存失败处理
}
```

试题 7 存在与试题 6 同样的问题，在执行

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

后未进行内存是否申请成功的判断；另外，在 free(str)后未置 str 为空，导致可能变成一个“野”指针，应加上：

```
str = NULL;
```

试题 6 的 Test 函数中也未对 malloc 的内存进行释放。

剖析：

试题 4～7 考查面试者对内存操作的理解程度，基本功扎实的面试者一般都能正确的回答其中 50～60 的错误。但是要完全解答正确，却也绝非易事。

对内存操作的考查主要集中在：

（1）指针的理解；

（2）变量的生存期及作用范围；

（3）良好的动态内存申请和释放习惯。

再看看下面的一段程序有什么错误：

```
swap( int* p1,int* p2 )
{
  int *p;
  *p = *p1;
  *p1 = *p2;
  *p2 = *p;
}
```

在 swap 函数中，p 是一个“野”指针，有可能指向系统区，导致程序运行的崩溃。在 VC++中 DEBUG 运行时提示错误“Access Violation”。该程序应该改为：

```
swap( int* p1,int* p2 )
{
  int p;
  p = *p1;
  *p1 = *p2;
  *p2 = p;
}
```

3.内功题

试题 1：分别给出 BOOL，int，float，指针变量 与“零值”比较的 if 语句（假设变量名为 var）

解答：

BOOL 型变量：if(!var)

int 型变量： if(var==0)

float 型变量：

const float EPSINON = 0.00001;

if ((x >= - EPSINON) && (x <= EPSINON)

指针变量： if(var==NULL)

剖析：

考查对 0 值判断的“内功”，BOOL 型变量的 0 判断完全可以写成 if(var==0)，而 int 型变量也可以写成 if(!var)，指针变量的判断也可以写成 if(!var)，上述写法虽然程序都能正确运行，但是未能清晰地表达程序的意思。

一般的，如果想让 if 判断一个变量的“真”、“假”，应直接使用 if(var)、if(!var)，表明其为“逻辑”判断；如果用 if 判断一个数值型变量(short、int、long 等)，应该用 if(var==0)，表明是与 0 进行“数值”上的比较；而判断指针则适宜用 if(var==NULL)，这是一种很好的编程习惯。

浮点型变量并不精确，所以不可将 float 变量用“==”或“！=”与数字比较，应该设法转化成“>=”或“<=”形式。如果写成 if (x == 0.0)，则判为错，得 0 分。

试题 2：以下为 Windows NT 下的 32 位 C++程序，请计算 sizeof 的值

```
void Func ( char str[100] )
{
  sizeof( str ) = ?
}

void *p = malloc( 100 );
sizeof ( p ) = ?
```

解答：

```
sizeof( str ) = 4
sizeof ( p ) = 4
```

剖析：

Func (char str[100])函数中数组名作为函数形参时，在函数体内，数组名失去了本身的内涵，仅仅只是一个指针；在失去其内涵的同时，它还失去了其常量特性，可以作自增、自减等操作，可以被修改。

数组名的本质如下：

（1）数组名指代一种数据结构，这种数据结构就是数组；

例如：

```
char str[10];
cout << sizeof(str) << endl;
```

输出结果为 10，str 指代数据结构 char[10]。

（2）数组名可以转换为指向其指代实体的指针，而且是一个指针常量，不能作自增、自减等操作，不能被修改；

```
char str[10];
str++; //编译出错，提示 str 不是左值
```

（3）数组名作为函数形参时，沦为普通指针。

Windows NT 32 位平台下，指针的长度（占用内存的大小）为 4 字节，故 sizeof(str) 、sizeof (p) 都为 4。

试题 3：写一个“标准”宏 MIN，这个宏输入两个参数并返回较小的一个。另外，当你写下面的代码时会发生什么事？

```
least = MIN(*p++, b);
```

解答：

```
#define MIN(A,B) ((A) <= (B) ? (A) : (B))
```

MIN(*p++, b)会产生宏的副作用

剖析：

这个面试题主要考查面试者对宏定义的使用，宏定义可以实现类似于函数的功能，但是它终归不是函数，而宏定义中括弧中的“参数”也不是真的参数，在宏展开的时候对“参数”进行的是一对一的替换。

程序员对宏定义的使用要非常小心，特别要注意两个问题：

（1）谨慎地将宏定义中的“参数”和整个宏用用括弧括起来。所以，严格地讲，下述解答：

```
#define MIN(A,B) (A) <= (B) ? (A) : (B)
#define MIN(A,B) (A <= B ? A : B )
```

都应判 0 分；

（2）防止宏的副作用。

宏定义#define MIN(A,B) ((A) <= (B) ? (A) : (B))对 MIN(*p++, b)的作用结果是：

((*p++) <= (b) ? (*p++) : (*p++))

这个表达式会产生副作用，指针 p 会作三次++自增操作。

除此之外，另一个应该判 0 分的解答是：

```
#define MIN(A,B) ((A) <= (B) ? (A) : (B));
```

这个解答在宏定义的后面加“;”，显示编写者对宏的概念模糊不清，只能被无情地判 0 分并被面试官淘汰。

试题 4：为什么标准头文件都有类似以下的结构？

```
#ifndef __INCvxWorksh
#define __INCvxWorksh
#ifdef __cplusplus

extern "C" {
#endif
/*...*/
```

```
#ifdef __cplusplus
}

#endif
#endif /* __INCvxWorksh */
```

解答：
头文件中的编译宏

```
#ifndef  __INCvxWorksh
#define  __INCvxWorksh
#endif
```

的作用是防止被重复引用。

作为一种面向对象的语言，C++支持函数重载，而过程式语言 C 则不支持。函数被 C++编译后在 symbol 库中的名字与 C 语言的不同。例如，假设某个函数的原型为：

```
void foo(int x, int y);
```

该函数被 C 编译器编译后在 symbol 库中的名字为_foo，而 C++编译器则会产生像_foo_int_int 之类的名字。_foo_int_int 这样的名字包含了函数名和函数参数数量及类型信息，C++就是考这种机制来实现函数重载的。

为了实现 C 和 C++的混合编程，C++提供了 C 连接交换指定符号 extern "C"来解决名字匹配问题，函数声明前加上 extern "C"后，则编译器就会按照 C 语言的方式将该函数编译为_foo，这样 C 语言中就可以调用 C++的函数了。

试题 5：编写一个函数，作用是把一个 char 组成的字符串循环右移 n 个。比如原来是“abcdefghi”如果 n=2，移位后应该是“hiabcdefgh”

函数头是这样的：

```
//pStr 是指向以''结尾的字符串的指针
//steps 是要求移动的 n

void LoopMove ( char * pStr, int steps )
{
  //请填充...
}
```

解答：

正确解答 1：

```
void LoopMove ( char *pStr, int steps )
{
  int n = strlen( pStr ) - steps;
  char tmp[MAX_LEN];
  strcpy ( tmp, pStr + n );
  strcpy ( tmp + steps, pStr);
```

```
  *( tmp + strlen ( pStr ) ) = '';
  strcpy( pStr, tmp );
}
```

正确解答 2：

```
void LoopMove ( char *pStr, int steps )
{
  int n = strlen( pStr ) - steps;
  char tmp[MAX_LEN];
  memcpy( tmp, pStr + n, steps );
  memcpy(pStr + steps, pStr, n );
  memcpy(pStr, tmp, steps );
}
```

剖析：

这个试题主要考查面试者对标准库函数的熟练程度，在需要的时候引用库函数可以很大程度上简化程序编写的工作量。

最频繁被使用的库函数包括：

（1） strcpy

（2） memcpy

（3） memset

试题 6：已知 WAV 文件格式如下表，打开一个 WAV 文件，以适当的数据结构组织 WAV 文件头并解析 WAV 格式的各项信息。

WAVE 文件格式说明表

	偏移地址	字节数	数据类型	内 容
文件头	00H	4	Char	"RIFF"标志
	04H	4	int32	文件长度
	08H	4	Char	"WAVE"标志
	0CH	4	Char	"fmt"标志
	10H	4		过渡字节（不定）
	14H	2	int16	格式类别
	16H	2	int16	通道数
	18H	2	int16	采样率（每秒样本数），表示每个通道的播放速度
	1CH	4	int32	波形音频数据传送速率
	20H	2	int16	数据块的调整数（按字节算的）
	22H	2		每样本的数据位数
	24H	4	Char	数据标记符 " data "
	28H	4	int32	语音数据的长度

解答：

将 WAV 文件格式定义为结构体 WAVEFORMAT：

```
typedef struct tagWaveFormat
{
```

```
  char cRiffFlag[4];
  UIN32 nFileLen;
  char cWaveFlag[4];
  char cFmtFlag[4];
  char cTransition[4];
  UIN16 nFormatTag ;
  UIN16 nChannels;
  UIN16 nSamplesPerSec;
  UIN32 nAvgBytesperSec;
  UIN16 nBlockAlign;
  UIN16 nBitNumPerSample;
  char cDataFlag[4];
  UIN16 nAudioLength;

} WAVEFORMAT;
```

假设 **WAV** 文件内容读出后存放在指针 **buffer** 开始的内存单元内，则分析文件格式的代码很简单，为：

```
WAVEFORMAT waveFormat;
memcpy( &waveFormat, buffer,sizeof( WAVEFORMAT ) );
```

直接通过访问 **waveFormat** 的成员，就可以获得特定 **WAV** 文件的各项格式信息。

剖析：

试题 **6** 考查面试者组织数据结构的能力，有经验的程序设计者将属于一个整体的数据成员组织为一个结构体，利用指针类型转换，可以将 **memcpy**、**memset** 等函数直接用于结构体地址，进行结构体的整体操作。 透过这个题可以看出面试者的程序设计经验是否丰富。

试题 **7**：编写类 **String** 的构造函数、析构函数和赋值函数，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
};
```

解答：

```
//普通构造函数

String::String(const char *str)
{
  if(str==NULL)
  {
```

```
    m_data = new char[1]; // 得分点：对空字符串自动申请存放结束标志''的空
    //加分点：对 m_data 加 NULL 判断
    *m_data = '\0';
  }
  else
  {
    int length = strlen(str);
    m_data = new char[length+1]; // 若能加 NULL 判断则更好
    strcpy(m_data, str);
  }
}

// String 的析构函数

String::~String(void)
{
  delete [] m_data; // 或 delete m_data;
}

//拷贝构造函数

String::String(const String &other)        // 得分点：输入参数为 const 型
{
  int length = strlen(other.m_data);
  m_data = new char[length+1];          //加分点：对 m_data 加 NULL 判断
  strcpy(m_data, other.m_data);
}

//赋值函数

String & String::operate =(const String &other) // 得分点：输入参数为 const 型
{
  if(this == &other)      //得分点：检查自赋值
    return *this;
  delete [] m_data;          //得分点：释放原有的内存资源
  int length = strlen( other.m_data );
  m_data = new char[length+1];    //加分点：对 m_data 加 NULL 判断
  strcpy( m_data, other.m_data );
  return *this;                  //得分点：返回本对象的引用
}
```

剖析：

能够准确无误地编写出 **String** 类的构造函数、拷贝构造函数、赋值函数和析构函数的面试者至少已经具备了 **C++**基本功的 **60%**以上！

在这个类中包括了指针类成员变量 **m_data**，当类中包括指针类成员变量时，一定要重载其拷贝构造函数、赋值函数和析构函数，这既是对 **C++**程序员的基本要求，也是《**Effective C++**》中特别强调的条款。

仔细学习这个类，特别注意加注释的得分点和加分点的意义，这样就具备了 **60%**以上的 **C++**基本功！

试题 8：请说出 static 和 const 关键字尽可能多的作用

解答：

static 关键字至少有下列 n 个作用：

（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。

const 关键字至少有下列 n 个作用：

（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”。例如：

```
const classA operator*(const classA& a1,const classA& a2);
```

operator*的返回结果必须是一个 const 对象。如果不是，这样的变态代码也不会编译出错：

```
classA a, b, c;
(a * b) = c; // 对 a*b 的结果赋值
```

操作(a * b) = c 显然不符合编程者的初衷，也没有任何意义。

剖析：

惊讶吗？小小的 static 和 const 居然有这么多功能，我们能回答几个？如果只能回答 1～2 个，那还真得闭关再好好修炼修炼。

这个题可以考查面试者对程序设计知识的掌握程度是初级、中级还是比较深入，没有一定的知识广度和深度，不可能对这个问题给出全面的解答。大多数人只能回答出 static 和 const 关键字的部分功能。

4.技巧题

试题 1：请写一个 C 函数，若处理器是 Big_endian 的，则返回 0；若是 Little_endian 的，则返回 1

解答：

```
int checkCPU()
{
  {
    union w
    {
      int a;
      char b;
    } c;
    c.a = 1;
    return (c.b == 1);
  }
}
```

剖析：

嵌入式系统开发者应该对 Little-endian 和 Big-endian 模式非常了解。采用 Little-endian 模式的 CPU 对操作数的存放方式是从低字节到高字节，而 Big-endian 模式对操作数的存放方式是从高字节到低字节。例如，16bit 宽的数 0x1234 在 Little- endian 模式 CPU 内存中的存放方式（假设从地址 0x4000 开始存放）为：

内存地址	存放内容
0x4000	0x34
0x4001	0x12

而在 Big-endian 模式 CPU 内存中的存放方式则为：

内存地址	存放内容
0x4000	0x12
0x4001	0x34

32bit 宽的数 0x12345678 在 Little-endian 模式 CPU 内存中的存放方式（假设从地址 0x4000 开始存放）为：

内存地址	存放内容
0x4000	0x78
0x4001	0x56
0x4002	0x34
0x4003	0x12

而在 Big-endian 模式 CPU 内存中的存放方式则为：

内存地址	存放内容
0x4000	0x12
0x4001	0x34
0x4002	0x56
0x4003	0x78

联合体 union 的存放顺序是所有成员都从低地址开始存放，面试者的解答利用该特性，轻松地获得了 CPU 对内存采用 Little-endian 还是 Big-endian 模式读写。如果谁能当场给出这个解答，那简直就是一个天才的程序员。

试题 2：写一个函数返回 1+2+3+…+n 的值（假定结果不会超过长整型变量的范围）

解答：

```
int Sum( int n )
{
  return ( (long)1 + n) * n / 2;     //或 return (1l + n) * n / 2;
}
```

剖析：

对于这个题，只能说，也许最简单的答案就是最好的答案。下面的解答，或者基于下面的解答思路去优化，不管怎么“折腾”，其效率也不可能与直接 return (1 l + n) * n / 2 相比！

```
int Sum( int n )
{
```

```
  long sum = 0;
  for( int i=1; i<=n; i++ )
  {
    sum += i;
  }
  return sum;
}
```

所以程序员们需要敏感地将数学等知识用在程序设计中。

终于明白了：按值传递意味着当将一个参数传递给一个函数时，函数接收的是原始值的一个副本。因此，如果函数修改了该参数，仅改变副本，而原始值保持不变。按引用传递意味着当将一个参数传递给一个函数时，函数接收的是原始值的内存地址，而不是值的副本。因此，如果函数修改了该参数，调用代码中的原始值也随之改变。

不管是在 c/c++中还是在 java 函数调用都是传值调用，.

当参数是对象的时候，传递的是对象的引用，这个和 c/c++传递指针是一个道理，在函数中改变引用本身，不会改变引用所指向的对象.

华为面试题

4、SQL 问答题

SELECT * FROM TABLE 和 SELECT * FROM TABLE
WHERE NAME LIKE '% %' AND ADDR LIKE '% %'
AND (1_ADDR LIKE '% %' OR 2_ADDR LIKE '% %'
OR 3_ADDR LIKE '% %' OR 4_ADDR LIKE '% %')

的检索结果为何不同?

答: 前者检索全部，后者有三种情况检索不出：NAME=null 或 ADDR=null 或 1_ADDR LIKE 2_ADDR 3_ADDR 4_ADDR 其一为 null.

前者检索所有记录，后者只能检索出 NAME 和 ADDR 中非 Null 的记录。

5、SQL 问答题

表结构：

1、 表名：g_cardapply

字段(字段名/类型/长度)：

g_applyno varchar 8； //申请单号（关键字）

g_applydate bigint 8； //申请日期

g_state varchar 2； //申请状态

2、 表名：g_cardapplydetail

字段(字段名/类型/长度)：

g_applyno varchar 8； //申请单号（关键字）

g_name varchar 30； //申请人姓名

g_idcard varchar 18； //申请人身份证号

g_state varchar 2； //申请状态

其中，两个表的关联字段为申请单号。

题目：

1、 查询身份证号码为 440401430103082 的申请日期

```
select A.g_applydate
from g_cardapply A inner join g_cardapplydetail B on A.g_applyno = B.g_applyno
where B.g_idCard = '440401430103082'
```

2、 查询同一个身份证号码有两条以上记录的身份证号码及记录个数

```
select g_idCard,count(*) as Cnt from g_cardapplydetail
group by g_idcard
having count(*) > 1
```

3、 将身份证号码为 440401430103082 的记录在两个表中的申请状态均改为 07

```
update g_cardapplydetail set g_state = '07'
where g_idcard = '440401430103082'

update A set g_state = '07'
from g_cardapply A inner join g_cardapplydetail B on A.g_applyno = B.g_applyno
where B.g_idcard = '440401430103082'
```

4、 删除 g_cardapplydetail 表中所有姓李的记录

```
delete from g_cardapplydetail
where g_name like '李%'
```

3、 将身份证号码为 440401430103082 的记录在两个表中的申请状态均改为 07

```
update g_cardapplydetail set g_state = '07'
where g_idcard = '440401430103082'

update A set g_state = '07'
from g_cardapply A inner join g_cardapplydetail B on A.g_applyno = B.g_applyno
where B.g_idcard = '440401430103082'
```

5、SQL 问答题:

```
/*Select g_cardapply. g_applydate
From g_cardapply, g_cardapplydetail
Where g_cardapply. g_applyno=g_cardapplydetail. g_applyno
And g_cardapplydetail.g_idcard='440401430103082'*/

/*Select *From (select count(*) g_count , g_idcard
From g_cardapplydetail
Group by g_idcard ) a
Where a. g_count >= 2*/

/*Update g_cardapply
set g_state='07'
where g_applyno in (select distinct g_applyno
from g_cardapplydetail
where g_idcard ='440401430103082')
update g_cardapplydetail
set g_state='07'
where g_idcard='440401430103082' */

/*Delete from g_cardapplydetail
Where g_name like '李%'*/
```

通过测试

PS:偶 GF 做的，自己先汗一下

金山公司几道面试题

4. In C++, there're four type of Casting Operators, please enumerate and explain them especially the difference.

解析： C++类型转换问题

答案： reinterpret_cast,static_cast,const_cast,dynamic_cast

static_cast 数制转换

dynamic_cast 用于执行向下转换和在继承之间的转换

const_cast 去掉 const

reinterpret_cast 用于执行并不安全的 orimplmentation_dependent 类型转换

7 以下代码有什么问题，如何修改？

```
#include <iostream>
#include <vector>
using namespace std;
void print(vector<int>);
int main()
{
vector<int> array;
array.push_back(1);
array.push_back(6);
array.push_back(6);
array.push_back(3);
//删除 array 数组中所有的 6
vector<int>::iterator itor;
vector<int>::iterator itor2;
itor=array.begin();

for(itor=array.begin(); itor!=array.end(); )
{
if(6==*itor)
{
itor2=itor;
array.erase(itor2);
}
itor++;
}
print(array);
return 0;
}
void print(vector<int> v)
{
cout << "n vector size is: " << v.size() << endl;
vector<int>::iterator p = v.begin();
}
```

我的答案是，迭代器问题，只能删除第一个 6，以后迭代器就失效了，不能删除之后的元素。

但我不知道怎么改

```
void print(const vector<int>&);
int main()
{
```

```
vector<int> array;
array.push_back(1);
array.push_back(6);
array.push_back(6);
array.push_back(3);

//删除 array 数组中所有的 6
array.erase( remove( array.begin(), array.end(), 6 ) , array.end() );

print(array);
return 0;
}
void print(const vector<int>& v)
{
cout << "n vector size is: " << v.size() << endl;
copy(v.begin(), v.end(), ostream_iterator<int>(cout, " ") );
}
#include <iostream>
#include <vector>
using namespace std;
int main()
{
vector<int> array;
array.push_back(1);
array.push_back(6);
array.push_back(6);
array.push_back(6);
array.push_back(6);
array.push_back(6);
array.push_back(3);
array.push_back(9);
array.push_back(8);
array.push_back(5);
//&Eacute;&frac34;&sup3;&yacute;array&Ecirc;&yacute;×é&Ouml;&ETH;&Euml;ù&Oac
ute;&ETH;&micro;&Auml;6
vector<int>::iterator itor;
itor=array.begin();
for(itor=array.begin(); itor!=array.end();++itor )
{
if(6==*itor)
{
itor=array.erase(itor);
--itor;
}
}
cout << "vector size is: " << array.size() << endl;
for(itor=array.begin(); itor!=array.end();++itor )
{
```

```
cout<<*itor<<" ";
}
system("pause");
return 0;
}
```

答案: 执行 itor=array.erase(itor);这句话后，itor 不会移动，而只是把删除的数后面的数都往前移一位，所以删除了第一个 6 后，指针指向第 2 个 6，然后在来个 itor++，指针就指向 array.end()了，给你画个草图：

1 6 6 3 array.end() //最开始指针 itor 指向第一个 6；

1 6 3 array.end() //删除第一个 6 后，指向第二个 6

1 6 3 array.end() //itor++后，就指向 3 了，所以不能删除

2. What are three ways in which a thread can enter the waiting state?

答:CPU 调度给优先级更高的 thread，原先 thread 进入 waiting

阻塞的 thread 获得资源或者信号，进入 waiting

还有什么

面试与被面试总结

我从事技术工作，

这几年的面试与被面试总结

先说我去被面试的经验吧。

回答清楚了 2 个问题，就能顺利过关了。

1。为什么要离开上一家公司。

2。公司为什么要雇佣你。

问第一个问题的是 hr（或老板）,呵呵 ，即使你技术过关，hr 那里没有好的影响，结果是一个字，难！

如何回答呢？ hr 想推论出你在他的公司能呆多久。这个时候，你甚至可以明确告诉他，我在贵公司至少能呆 n（n>=1）年----当然，你没有把握的话，绝对不能乱说，社会上混，要讲信用的。

有一次，我就在这个问题上吃了大亏，我看公司环境还不错，就我自做主张回答 1 年，结果，hr 心目中是 m（m>=2）年，呵呵，结果可想而知了。要知道，技术面试都过关了，Hr 面试是 2 选 1，在回家的路上，我只能祈祷对手自动放弃或找到了其他更好的工作。：）

问第二个问题的是技术官。你要让他知道你已经做过哪些商业作品。一定要是商业作品。在里面负责哪方面具体工作，对于你熟悉的地方要多说。最好就是能争取笔试或上机，因为用口说的话，大家理解都不一样，误差可能很大，结果对你相当不利。在这个问题上我也吃过亏的，曾有一个我很看好的职位，认为把握很大，业务理解上也很有优势，和技术官一谈，结果是 game over。要知道，在其他公司的上机和笔试中，我都能在应聘者中取得高分。

再说我去面试别人的经验吧。

当时，我的任务是出题，给分。若你觉得题很难，那么，请千万不要放弃，显然，你的对手也觉得难。只要坚持，我会认为这人有耐心很毅力，在以后的工作中也是好的合作者。题一定要做完，表现出认真的态度，若有疑问或卡壳，还可以寻求面试官的帮助，这些不会减分，相反，会增加你和他们的接触机会，面试官会评估你的沟通能力。

有一次，有 1 个人来面试，题没有完全 ok,但很规范，态度很认真，他把他知道的都做上去了，我给了他技术类的高分。后来，顺利进入公司，再后来进步很快，成了重要角色。

若文章对你有帮助的话，请在此讨论。

祝你成功

面试题

1.链表和数组的区别在哪里？

2.编写实现链表排序的一种算法。说明为什么你会选择用这样的方法？

3.编写实现数组排序的一种算法。说明为什么你会选择用这样的方法？

4.请编写能直接实现 strstr()函数功能的代码。
5.编写反转字符串的程序，要求优化速度、优化空间。
6.在链表里如何发现循环链接？
7.给出洗牌的一个算法，并将洗好的牌存储在一个整形数组里。
8.写一个函数，检查字符是否是整数，如果是，返回其整数值。（或者：怎样只用 4 行代码 ,编写出一个从字符串到长整形的函数？）
9.给出一个函数来输出一个字符串的所有排列。
10.请编写实现 malloc()内存分配函数功能一样的代码。
11.给出一个函数来复制两个字符串 A 和 B。字符串 A 的后几个字节和字符串 B 的前几个字节重叠。
12.怎样编写一个程序，把一个有序整数数组放到二叉树中？
13.怎样从顶部开始逐层打印二叉树结点数据？请编程。
14.怎样把一个链表掉个顺序（也就是反序，注意链表的边界条件并考虑空链表）？

另外：

一、单项选择题：（共 12 题，每题 2 分，共 24 分）
1. 下面哪一个不是 C++的标准数据类型？ （ D）
A. int B. char
C. bool D. real
2. break 关键字在哪一种语法结构中不能使用？ （C）
A. for 语句 B. switch 语句
C. if 语句 D. while 语句
3. 类的继承方式有几种？ （B ）
A. 两种 B. 三种
C. 四种 D. 六种
4. extern 关键字的作用是什么？ （D）
A. 声明外部链接 B. 声明外部头文件引用
C. 声明使用扩展 C++语句 D. 声明外部成员函数、成员数据。
5. C 库函数 strstr 的功能是？ （A ）
A. 查找子串 B. 计算字符串长度
C. 字符串比较 D. 连结字符串
6. stl::deque 是一种什么数据类型？ （A ）
A. 动态数组 B. 链表
C. 堆栈 D. 树
7. STL 库里含有下面的哪一种泛型算法？ （D ）
A. KMP 查找 B. 折半查找
C. 冒泡排序 D. 快速排序
8. 现在最快且最通用的排序算法是什么？ （A ）
A. 快速排序 B. 冒泡排序
C. 选择排序 D. 外部排序
9. Win32 下的线程的哪一种优先级最高？ （ C）
A. THREAD_PRIORITY_HIGHEST 高优先级
B. THREAD_PRIORITY_IDLE 最低优先级,仅在系统空闲时执行
C. THREAD_PRIORITY_TIME_CRITICAL 最高优先级
D. THREAD_PRIORITY_ABOVE_NORMAL 高于普通优先级
10. 下面四个选项中，哪一个不是 WinMain 函数的参数？ （D ）
A. HINSTANCE B. INT
C. LPSTR D. WPARAM
11. VC++的编译器中，运算符 new 底层的实现是什么？ （B ）

A. VirtualAlloc() B. HeapAlloc()
C. GlobalAlloc() D. AllocateUserPhysicalPages()
12. 下面哪一本 C++参考书最厚？ （ C）
A. 《Think in C++》 B. 《深入浅出 MFC》
C. 《C++ Primer》 D. 《Effective C++》
13. 当调用 Windows API 函数 InvalidateRect,将会产生什么消息（A）
A. WM_PAINT B. WM_CREATE
C. WM_NCHITTEST D. WM_SETFOCUS
14. 关于 virtual void Draw()=0，下面说法正确的有几个（C）
(1)它是纯虚函数(对)
(2)它在定义它的类中不能实现(对)
(3)定义它的类不可实例化(对)
(4)如果一个类要继承一个 ADT 类，必须要实现其中的所有纯虚函数(错) //可以不实现,派生之后的类仍旧作为一个抽象类.
A. 1 B. 2
C. 3 D. 4
二、不定项选择题：（共 6 题，每题 3 分，共 18 分，多选、错选、漏选均不给分）
1. vector::iterator 重载了下面哪些运算符？ （ACD）
A. ++ B. >>
C. *（前置） D. ==
2. CreateFile( )的功能有哪几个？ （AB ）
A. 打开文件 B. 创建新文件
C. 文件改名 D. 删除文件
3. 下面哪些是句柄（HANDLE）？ （ABCD ）
A. HINSTANCE 实例句柄 B. HWND 窗口句柄
C. HDC 设备描述符号句柄 D. HFONT 字体句柄
4. 下面哪些不是 OpenGL 标准几何元素的绘制模式？ （A ）
A. GL_FOG B. GL_LINE_STRIP
C. GL_POINTS D. GL_TRIANGLE_FAN
5. 下面哪些运算符不能被重载？ （ABD ）
A. 做用域运算符“::” B. 对象成员运算符“.”
C. 指针成员运算符“->” D. 三目运算符“? :”
6. 下面哪些人曾参与了世界上第一个 C++编译器的开发？ （ ）
A. Bill Gates B. Stanley Lippman
C. Anderson Hejlsberg D. Bjarne Stroustrup
7. 以下说法正确的是？ （ ABC）
A. 头文件中的 ifndef/define/endif 是为了防止该头文件被重复引用。
B. 对于#include <filename.h> ，编译器从标准库路径开始搜索 filename.h
对于#include “filename.h”，编译器从用户的工作路径开始搜索 filename.h
C. C++语言支持函数重载，C 语言不支持函数重载。函数被 C++编译后在库中的名字与 C 语言的不同。假设某个函数的原型为： void foo(int x, int y); 该函数被 C 编译器编译后在库 中的名字为_foo，而 C++编译器则会产生像_foo_int_int 之类的名字。C++提供了 C 连接交换指定符号 extern“C”来解决名字匹配问题。
D. fopen 函数只是把文件目录信息调入内存。//错,fopen 是把整个文件读入内存
三、填空题：（共 8 题，每题 3 分，共 24 分）
1. 一个大小为 320 X 192，颜色为灰度索引色的设备相关位图有_____字节。如果此位图颜色为 24 位真彩色，则它的大小有_____字节。
2. Windows API 的中文意义是____ windows 应用程序接口___。
3. 计算反正弦的库函数是__asin()____；计算浮点数绝对值的库函数是__fabs()____；计算浮点数 n 次

方的库函数是__pow()____；将浮点数转化为字符串的库函数是___fcvt()___。
4. 如果 i 等于 5，那么(++i) --的返回值是__6__。
5. API LoadBitmap()的功能是从__指定的模块和或应用程序实例__中读取位图数据到内存。
6. new 和__delete___对应，malloc 和_free___对应，他们之间_ 不能__交叉混用。calloc 的功能是__为数组动态分配内存___，realloc 的功能是_改变原有内存区域的大小_。
7. SendMessage 和 PostMessage 都会向窗体发送一个消息，但 SendMessage__将一条消息发送到指定窗口,立即处理__而 PostMessage__将一条消息投递到指定窗口的消息队列,不需要立即处理___。
8. 输出指定圆心、半径、边数的圆上的点：

```
const int nCount = 12;
const double dOrgX = 5.0,
dOrgY = 3.0;
const double dRadius = 2.0;
for( int i = 0; i < nCount; i++ )
{
double dAngle = M_PI * 2.0 / (double)nCount * i;
cout << "第" << i << "点：X = " << _________; cout << ", Y = " << __________ << endl;
}
```

三、判断题：（共 12 题，每题 2 分，共 24 分）
1. 一个类必须要有一个不带参数的构造函数。 错
2. 你不能写一个虚的构造函数。 对
3. 类里面所有的函数都是纯虚函数时才是纯虚类。 错
4. const 成员函数对于任何本类的数据成员都不能进行写操作。 对
5. 函数中带默认值的参数必须位于不带默认值的参数之后。 对
6. char *p = "Test"; p[0] = 'R'; 错
7. cout << "Test"; 对
8. stl::list 不支持随机访问叠代器。 对
9. stl::vector 的效率比 stl::list 高。 错
10. VC 和 VC++是一回事，而 VC++是一种比 C++更难一些的语言。 错
11. 理论上，new 和 malloc 造成的内存泄露都会由操作系统回收。 错
12. 在 C++中 struct 和 class 的差别很大，所以从语法上不能混用。对
四、简述题(共 3 题，每题 5 分，共 15 分)
1. 请简述 PeekMessage 和 GetMessage 的区别。
答: Peekmessage 和 Getmessage 都是向系统的消息队列中取得消息，两个函数的不同在于取不到消息的时候,若 Getmessage（）向消息队列中取不到消息，则程序的主线程会被 OS（操作系统）挂起,等到有合适的消息时才返回;若是用 Peekmessage（）在消息队列中取不到消息,则程序会取得 OS 控制权，运行一段时间。另外,在处理消息的时候,GetMessag()会将消息从队列中删除,而 PeekMessage()可以设置最后一个参数 wRemoveMsg 来决定是否将消息保留在队列中。
2. 请列出你所知道的在 Windows SDK 平台上，实现计时功能的方法。
答:可以使用 SetTimer 函数创建一个计时器,SetTimer 的函数原型如下：

```
UINT_PTR  SetTimer( HWND hWnd, UINT_PTR nIDEvent, UINT uElapse, TIMERPROC lpT
imerFunc
```

3. 请简述你所知道的 const 的各种用法。
答: const 常量
const 修饰类的数据成员
const 修饰指针
const 应用在函数声明中
const 应用在类成员函数
五、编程题：（共 3 题，第 1 小题 7 分，第 2 小题 14 分，第 3 小题 24 分）

1. 深度遍历二叉树。

```
struct Node
{
Node *Parent;

Node *Left, *Right;

};
void Through(Node *Root)
{
}
```

2. 二分法查找。

```
int DicFind( int *Array, int Count, int Value )
{

}
```

3. 写出字符串类 String 的默认构造函数、析构函数和重载赋值运算符。

已知类 String 的原型为：

```
class String
{
public:
String( const char *pStr = NULL ); // 默认构造函数
~String( void ); // 析构函数
String &operate = ( const String &Source ); // 重载赋值运算符
private:
char *m_pData; // 指向字符串的指针
};
```

今天下午的两道面试题

1。 一人岁数的 3 次方是四位数，四次方是六位数，并知道此人岁数的 3 次方和 4 次方用遍了 0～9 十个数字。编写一程序求此人的岁数。

2。对 1，2，3， 4， 5 这五个数任意取出两个数，列出他们的所有组合。

```
public static int getAge() {
int age;
int third;
int fourth;

for (int i = 11; true; i++) {
if (i < 200) {
third = (int) Math.pow(i, 3);
fourth = (int) Math.pow(i, 4);
if (getLength(third, fourth) == 10) {
age = i;
break;
}
```

```
}
}
return age;
}

public static int getLength(int args1, int args2) {
String str1 = String.valueOf(args1);
String str2 = String.valueOf(args2);
String str = str1 + str2;
if (str.length() != 10) {
return -1;
}
int[] intarray = new int[10];
for (int i = 0; i < str.length(); i++) {
intarray[i] = Integer.parseInt(str.substring(i,i+1));
}
Arrays.sort(intarray);
if(intarray[0]!=0 && intarray[9]!=9)
return -1;

return 10;
}
```
第二题还更简单了

```
for(int i=1; i<6; i++){
for(int j=1; j<6; j++){
if(i==j){
System.out.println(j+""+j);
}else{
System.out.println(i+""+j);
System.out.println(j+""+i);
}
}
}
public class A {
// http://community.csdn.net/Expert/topic/4667/4667929.xml?temp=.57922
public static void main(String[] args) {
String t;
String[] s = new String[5];
int j = s.length;
for(int i=0; i<j; i++) {
s[i] = new Integer(i+1).toString();
}

for(int i=0; i<j; i++) {
t = s[i];
for(int a=0; a<j; a++) {
t += s[i];
```

```
System.out.println(t);
}
System.out.println();
}
}
}
```
第二题还更简单了

```
for(int i=1; i<6; i++){
for(int j=1; j<6; j++){
if(i==j){
System.out.println(j+""+j);
}else{
System.out.println(i+""+j);
System.out.println(j+""+i);
}
}
}
```
==========

楼上的没看清题目，它是让你对 1，2，3， 4， 5 这五个数任意取出两个数，列出他们的所有组合，所以重复的数字不应该算在里面。

第二题应该改为：

```
for(int i=1; i<6; i++){
for(int j=1; j<6; j++){
if(i==j){
break;
}else{
System.out.println(i+""+j);
System.out.println(j+""+i);
}
}
}
public class B {
public static void main(String[] args) {
for (int i = 1; i < 6; i++) {
int t = i;
for(int a = 0; a<5; a++) {
int c = a+1;
if(c == t) {
continue;
}else {
System.out.println(t*10+c);
}
}
System.out.println();
}
}
```

```
}
```
第二题
```
public class Test
{
public static void main(String[] args)
{
int[][] a=new int[5][];
for(int i=0;i<a.length;i++)
{
a[i]=new int[i+1];
}
for(int i=1;i<=a.length;i++)
{

for(int j=i+1;j<=a.length;j++)
{
System.out.print(i);
System.out.print(j+" ");
}
System.out.print(" ");
}

for(int i=a.length;i>0;i--)
{

for(int j=i-1;j>0;j--)
{
System.out.print(i);
System.out.print(j+" ");
}
System.out.print(" ");
}
}
}


public class Test {


public static int getDigits(String str) {

int[] intarr = new int[10];

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

intarr[i] = 0;

for (int i = 0; i < str.length(); i++) {

int j = Integer.parseInt(str.substring(i, i + 1));

intarr[j] = 1;
```

```
}
int num = 0;
for (int i = 0; i < 10; i++)
num = num + intarr[i];
return num;
}

private static int getAge() {
int age;
int third;
int fourth;
for (age = 1; age < 100; age++) {
third = (int) Math.pow(age, 3);
fourth = (int) Math.pow(age, 4);
if (third < 1000 || third >= 10000)
continue;
if (fourth < 100000 || fourth >= 1000000)
continue;
String str = String.valueOf(third) + String.valueOf(fourth);
if (getDigits(str) == 10)
return age;
}
return 0;
}
}
```

第二道题

```
class Combine
{
public static void main(String[] args)
{
for(int i=1; i<5; i++)
{
for(int j=i+1; j<6; j++)
{
System.out.println(i+""+j);
System.out.println(j+""+i);
}
}
```

```
}
public class Age
{
public static void main(String [] args)
{
String str1 = null;
String str2 = null;
String str3 = null;
String str4 = "0123456789";
for(int i=10;i<50;i++)
{
str1 = Integer.toString(i*i*i);
str2 = Integer.toString(i*i*i*i);
str3 = str1+str2;
if((str1.length() == 4) && (str2.length() ==6))
{
boolean flag = true;
for(int j=0;j<10;j++)
if(str3.indexOf(str4.charAt(j))==-1)
flag = false;
if(flag){
System.out.println(">>>"+i);
System.out.println(str3);
}
}
}

}
}
```

比赛贴~微软又一道笔试题

silver6 | 04 四月, 2006 09:48

怎样只用 **4** 行代码编写出一个从字符串到长整形的转换函数?

我的方法，不过好象比 **4** 行多 ***_#!~**

```
long atol(char *str)
{
char c = *str;
if( !isdigit(c) ) str++;
for(long value = 0; *str != ''; value = value * 10 + (*str -'0'),str++);
return c == '-' ? -value : value ;
```

```
}
void stol(const char * des, long& num)
{
for (int base = 1, i = 0; des[i] != ''; base = 10, ++i)
{
num *= base;
num += (int)(des[i] - '0');
}
}
```

num 要初始化为 0

```
void stol(const char * des, long& num)
{
for (int i=num=0; des[i] != ''; i++)
{
num *= 10;
num += (int)(des[i] - '0');
}
}
void stol(char *str, long &num)
{
while(*str != '')
{
num = num * 10 + (*str - '0');
str++;
}
}
void stol(const char * des, long& num)
{
char p = des[0];
for (int b = 1, pos = 1, base = 1; des[pos] != ''; b = 10, ++pos, base *= 10)
{
(num *= b) += (int)(des[pos] - '0');
}
p == '-' ? (num *= -1) : (num = (int)(des[0] - '0') * base + num);
}
```

改了一下

真的是微软的笔试题么？

我只用了一行。

```
#include <iostream>
using namespace std;
long str2long(char* p,long xxx=0L)
{
return *p==''?xxx:str2long(p,xxx*10+(*p+++0-'0'));
}

int main()
{
char *str="123456789",*p=str;
cout<<str2long(p);
getchar();
return 0;
}
```

用 STL，四行

```
#include <sstream>
#include <iostream>
#include <string>
using namespace std;
long ToLong(string& s)
{
long l;
istringstream iss(s);
iss>>l;
return l;
}
int main(int argc, _TCHAR* argv[])
{
string s = "-12356";
cout<<ToLong(s);
return 0;
}
```

谢谢刚才上面的帖子提醒负数的问题，我更正了，还是只用一行：

```
#include <iostream>
using namespace std;

long str2long(char* p,long xxx=0L,bool IsPositive=true)
{
return
*p==''?(IsPositive?xxx:xxx*(-1)):(*p=='-'?str2long(++p,0L,false):str2long(p,xxx
```

```
*10+*p+++0-'0',IsPositive));
}

int main()
{
char *str="-123456789",*p=str;
cout<<str2long(p);
getchar();
return 0;
}
```