

fangyukuan

永无止境的追求...追求卓越!!!

博客园 :: 首页 :: 博文 :: 闪存 :: 新随笔 :: 联系 :: 订阅  管理 ::  93 随笔 :: 0 文章 :: 98 评论 :: 0 引用

公告

昵称: fangyukuan  
园龄: 4年11个月  
粉丝: 166  
关注: 3  
[+加关注](#)

搜索

谷歌搜索

常用链接

[我的随笔](#)  
[我的评论](#)  
[我的参与](#)  
[最新评论](#)  
[我的标签](#)

我的标签

[线程](#) (19)  
[内存管理](#) (6)  
[DLL](#) (3)  
[面试](#) (2)

随笔分类(93)

[Android](#) (9)  
[Android工程经验](#) (1)  
[C/C++](#) (7)  
[C++工程经验](#) (14)  
[COM/ATL](#) (13)  
[D3D](#)  
[DDraw](#) (6)  
[json](#) (1)  
[Mac](#)  
[STL](#) (2)

C/C++ 笔试、面试题目大汇总2

见上一篇: [C/C++ 笔试、面试题目大汇总](#)


继续 ~~~~~

一.找错题

试题1:

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

试题2:



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

[Windows\(40\)](#)**随笔档案(93)**

[2012年5月 \(1\)](#)  
[2012年4月 \(1\)](#)  
[2011年8月 \(1\)](#)  
[2011年7月 \(1\)](#)  
[2011年6月 \(5\)](#)  
[2011年5月 \(4\)](#)  
[2011年3月 \(2\)](#)  
[2010年12月 \(2\)](#)  
[2010年11月 \(2\)](#)  
[2010年9月 \(28\)](#)  
[2010年8月 \(14\)](#)  
[2010年7月 \(1\)](#)  
[2010年6月 \(15\)](#)  
[2010年5月 \(5\)](#)  
[2010年4月 \(10\)](#)  
[2010年3月 \(1\)](#)

**其它博客**

[我的163博客](#)  
[我的百度博客](#)

**学习伙伴**

[colin小屋](#)  
[fangshenghui](#)

**学习资源**

[Android开发](#)

**积分与排名**

积分 - 53808  
排名 - 3258

**最新评论**

1. [Re:C/C++ 笔试、面试题目大汇总](#)  
很考验基础的

--C++专家

2. [Re:C++中类的多态与虚函数的使用](#)  
讲解的非常好。

--得路洒

3. [Re:C++中类的多态与虚函数的使用](#)

```
strcpy( string, str1 );  
}
```

**试题3:**

```
void test3(char* str1)  
{  
    charstring[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'\0'），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'\0'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函数的总分为10，下面给出几个不同得分的答案：

**试题4:**

赞一个.

--Sunstrider

4. Re:COM笔记-CoCreateInstance  
谢谢 帮到我了

--仁明的球员

5. Re:windows笔记-内存映射文件  
写的不错

--Castor

### 阅读排行榜

1. C/C++ 笔试、面试题目大汇总(115728)
2. 【STL】list基础(28528)
3. C/C++ 笔试、面试题目大汇总2(20162)
4. COM笔记-CoCreateInstance(17843)
5. COM笔记-QueryInterface函数(13695)
6. windows笔记-内存映射文件(13464)
7. Eclipse常用快捷键(10193)
8. VS2008 条件断点(9453)
9. 【Android笔记】-TextView标签(8934)
10. vs2008【断点无效】解决方法(8910)

### 评论排行榜

1. DDraw笔记-一个简单的DDraw应用程序(15)
2. C/C++ 笔试、面试题目大汇总(11)
3. VS2008 条件断点(10)
4. COM笔记-CoCreateInstance(7)
5. C/C++ 笔试、面试题目大汇总2(5)

### 推荐排行榜

1. C/C++ 笔试、面试题目大汇总(11)
2. windows笔记-内存映射文件(8)
3. VS2008 条件断点(8)
4. 【STL】list基础(5)
5. windows笔记-API实现透明窗体



```
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:

- (4)
- 6. C++中类的多态与虚函数的使用(4)
- 7. 使用ATL设计组件(vs2008)(4)
- 8. windows笔记-使用内存映射文件在进程之间共享数据(3)
- 9. windows笔记-等待定时器与用户定时器的区别(3)
- 10. vs2008 showIncludes命令的使用(3)



```
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“AccessViolation”。该程序应该改为：



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



## 二.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WindowsNT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。

函数头是这样的：

```
// pStr是指向以'\0'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 ) ) = '\0';
    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
	00H	4	Char	"RIFF"标志
	04H	4	int32	文件长度

文件头	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(constchar*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 & operator =(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(constchar*str)
{
    if(str==NULL)
    {
        m_data =newchar[1]; // 得分点: 对空字符串自动申请存放结束标志'\0'的空
        //加分点: 对m_data加NULL 判断
        *m_data ='\0';
    }
    else
    {
        int length = strlen(str);
        m_data =newchar[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deletem_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 =newchar[length+1]; //加分点: 对m_data加NULL 判断
    strcpy(m_data, other.m_data);
}
//赋值函数
```

```
String & String::operator =(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 =newchar[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。

### 三.技巧题

试题1: 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( 1l + 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。

本文地址: <http://www.cnblogs.com/fangyukuan/archive/2010/09/18/1830493.html>

分类: C/C++

标签: 面试

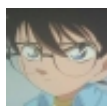
绿色通道:

好文要顶

关注我

收藏该文

与我联系



fangyukuan

关注 - 3

粉丝 - 166

+加关注

3

0

(请您对文章做出评价)

« 上一篇: C/C++ 笔试、面试题目大汇总

» 下一篇: 【STL】list基础

posted on 2010-09-18 23:43 fangyukuan 阅读(20162) 评论(5) 编辑 收藏

## 评论

#1楼 2013-02-03 22:18 大力神

// pStr是指向以'\0'结尾的字符串的指针// steps是要求移动的nvoid LoopMove ( char \* pStr, int steps ){ // 请填写...}



此题感觉有二个问题：

- 1、没有指明是左移还是在右移
- 2、移位没有考虑 steps > strlen(pstr)的情况

支持(0) 反对(0)

#### #2楼 2013-02-22 16:43 怪人Kook

@大力神

引用

```
// pStr是指向以'\0'结尾的字符串的指针// steps是要求移动的nvoid LoopMove ( char * pStr, int steps ){ // 请填写...}
```

此题感觉有二个问题：

- 1、没有指明是左移还是在右移
- 2、移位没有考虑 steps > strlen(pstr)的情况

并且这个题给的代码也有问题

支持(0) 反对(0)

#### #3楼 2013-02-22 17:50 怪人Kook

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

这里是operator，不是operate，上网一查，还有很多人用这个字  
靠

支持(0) 反对(0)

#### #4楼 2013-02-22 21:47 怪人Kook

试题8：请说出static和const关键字尽可能多的作用

对于const还可以再加一条

(6)可以减少内存占用,提高效率。因为当const修饰的变量比较简单,如int,部分编译器会进行常量折叠操作,不为该变量分配内存,而是直接将其放入符号表中。可以看看thinking in C++中关于const的介绍。

支持(0) 反对(0)

#5楼[楼主] 2013-06-12 16:27 fangyukuan

@andy song

嗯,多谢指正,已修改。这些是整理了网上的资料集中在一起,做了一些小修改而已。

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论,请 [登录](#) 或 [注册](#), [访问网站首页](#)。

**【免费课程】案例：PHP加密技术专题**

**【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库**

融云, 免费为你的App加入IM功能——让你的App“聊”起来！！



外汇黄金交易首选平台

**50美元真实资金**  
**直接存入账户**

可用于真实交易

[点此无条件领取](#)

**GCMASIA 国汇亚洲**

#### 最新IT新闻:

- 前苹果胖子工程师的励志创业：为减肥而创业并大获成功
  - 干货必看：31个当下流行的金融热词
  - Facebook推出新功能，宣称可降低自杀率
  - 世界最强电脑都是猫星人？认图大考验全认出猫咪
  - 我创业1000天以来最黑暗的一夜
- » [更多新闻...](#)



**史上最全的HTML5教程**

CSS3 • JS • jQuery • Bootstrap • Egret • creatJS

极客学院  
likegeek.com

#### 最新知识库文章:

- 影响架构决策的非功能性需求
  - 在线数据迁移经验：如何为正在飞行的飞机更换引擎
  - HHVM 是如何提升 PHP 性能的？
  - Web API设计方法论
  - Bitmap的秘密
- » [更多知识库文章...](#)

