**高质量C编程指针基础测试 C++/C 试题**

本试题仅用于考查 **C++/C** 程序员的基本编程技能。内容限于 **C++/C** 常用语法，不  
涉及数据结构、算法以及深奥的语法。考试成绩能反映出考生的编程质量以及对 **C++/C**  
的理解程度，但不能反映考生的智力和软件开发能力。  
笔试时间 **90** 分钟。请考生认真答题，切勿轻视。  
一、请填写 **BOOL , float,** 指针变量 与“零值”比较的 **if** 语句。（ **10** 分）  
提示：这里“零值”可以是 **0, 0.0 , FALSE** 或者“空指针”。例如 **int** 变量 **n** 与“零值”  
比较的 **if** 语句为：  
**if ( n == 0 )**  
**if ( n != 0 )**  
以此类推。  
请写出 **BOOL flag** 与“零值”比较的 **if** 语句：

if (false == flag)请写出 **float x** 与“零值”比较的 **if** 语句：

if (0.0 == x)

if (0.0 != x)请写出 **char \*p** 与“零值”比较的 **if** 语句：

if (NULL == p)

if (NULL != p)二、以下为 **Windows NT** 下的 **32** 位 **C++**程序，请计算 **sizeof** 的值（ **10** 分）  
**char str[] = “Hello” ;**  
**char \*p = str ;**  
**int n = 10;**  
请计算  
**sizeof (str ) =** 6  
**sizeof ( p ) =** 6  
**sizeof ( n ) =** 4

**void Func ( char str[100])**  
**{**  
请计算  
**sizeof( str ) =** 100  
**}**  
  
**void \*p = malloc( 100 );**  
请计算  
**sizeof ( p ) =** 100  
高质量 C++/C 编程指南， v 1.0  
2001 Page 94 of 101  
三、简答题（ **25** 分）  
**1**、头文件中的 **ifndef/define/endif** 干什么用？

用于防止头文件被多个文件重复包含而引起冲突。  
**2**、 **#include <filename.h>** 和 **#include “filename.h”** 有什么区别？

#include <filename.h> 是从标准库中查找filename.h，而#include “filename.h”是先从用户目录下去查找filename.h,如果查不到会再去标准库下查找filename.h。  
**3**、 **const** 有什么用途？（请至少说明两种）

1.const加在变量前则限定变量为只读变量，不可以轻易更改。一般可以用来传入只可以读取函数参数。  
**4**、在 **C++** 程序中调用被 **C** 编译器编译后的函数，为什么要加 **extern “C”**声明？

为了声明这是一个C函数，C++和C有部分区别，为了兼容。  
**5**、请简述以下两个 **for** 循环的优缺点  
// 第一个  
for (i=0; i<N; i++)  
{  
if (condition)  
DoSomething();  
else  
DoOtherthing();  
}

优点： 结构更加清晰，代码简练  
缺点： 效率不高，每次判断  
// 第二个  
if (condition)  
{  
for (i=0; i<N; i++)  
DoSomething();  
}  
else  
{  
for (i=0; i<N; i++)  
DoOtherthing();  
}  
优点： 效率高无需每次都进行判断  
缺点： 不够清晰易懂  
  
高质量 C++/C 编程指南， v 1.0  
2001 Page 95 of 101  
四、有关内存的思考题（ **20** 分）  
void GetMemory(char \*p)  
{  
p = (char \*)malloc(100);  
}  
void Test(void)  
{  
char \*str = NULL;  
GetMemory(str);  
strcpy(str, "hello world");  
printf(str);  
}  
请问运行 Test 函数会有什么样的结果？   
答：无法分配成功，内存错误

**因为 GetMemory 并不能传递动态内存，Test 函数中的 str 一直都是 NULL。strcpy(str, "hello world");将使程序崩溃。**char \*GetMemory(void)  
{  
char p[] = "hello world";  
return p;  
}  
void Test(void)  
{  
char \*str = NULL;  
str = GetMemory();  
printf(str);  
}  
请问运行 Test 函数会有什么样的结果？   
答：可能错误，函数虽然返回了值，但p变量已经释放了，所在空间可能会被重写

**GetMemory 返回的是指向“栈内存”的指针，该指针的地址不是 NULL，但其原  
现的内容已经被清除，新内容不可知。**Void GetMemory2(char \*\*p, int num)  
{  
\*p = (char \*)malloc(num);  
}  
void Test(void)  
{  
char \*str = NULL;  
GetMemory(&str, 100);  
strcpy(str, "hello");  
printf(str);  
}  
请问运行 Test 函数会有什么样的结果？   
答：hello正确执行 **内存泄漏**void Test(void)  
{  
char \*str = (char \*) malloc(100);  
strcpy(str, **“**hello**”**);  
free(str);  
if(str != NULL)  
{  
strcpy(str, **“**world**”**);  
printf(str);  
}  
**}**  
请问运行 Test 函数会有什么样的结果？   
答：报错，提前将str堆空间释放了。

**篡改动态内存区的内容，后果难以预料，非常危险。因为 free(str);之后， str 成为野指针，if(str != NULL)语句不起作用。**  
高质量 C++/C 编程指南， v 1.0  
2001 Page 96 of 101  
五、编写 **strcpy** 函数（ **10** 分）  
已知 **strcpy** 函数的原型是  
**char \*strcpy(char \*strDest, const char \*strSrc);**  
其中 **strDest** 是目的字符串， **strSrc** 是源字符串。  
（ **1**）不调用 **C++/C** 的字符串库函数，请编写函数 **strcpy**  
（ **2**） **strcpy** 能把 **strSrc** 的内容复制到 **strDest**，为什么还要 **char \*** 类型的返回值？

char \*strcpy(char \*strDest, const char \*strSrc)

{

if (sizeof(strDest) < sizeof(strSrc))

return NULL;

for (i = 0; strSrc[i] != ‘\0’; i++)

{

strDest[i] = strSrc[i];

}

strDest[i] = ‘\0’;

return 1;

}  
 char \* 错误则返回NULL空指针

**char \*strcpy(char \*strDest, const char \*strSrc);  
 {  
 assert((strDest!=NULL) && (strSrc !=NULL)); // 2分  
 char \*address = strDest; // 2分  
 while( (\*strDest++ = \* strSrc++) != ‘\0’ ) // 2分  
 NULL ;  
 return address ; // 2分  
 }**六、编写类 **String** 的构造函数、析构函数和赋值函数（ **25** 分）  
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: