附录 B ： C++/C 试题的答案与评分标准

一、请填写 BOOL , float, 指针变量 与“零值”比较的 if 语句。（10 分）

<table>
<tr><td colspan="2">请写出 BOOL flag 与“零值”比较的 if 语句。（3 分）</td></tr>
<tr><td>标准答案：
if (flag)
if (!flag)</td><td>如下写法均属不良风格，不得分。
if (flag == TRUE)
if (flag == 1)
if (flag == FALSE)
if (flag == 0)</td></tr>
<tr><td colspan="2">请写出 float x 与“零值”比较的 if 语句。（4 分）</td></tr>
<tr><td>标准答案示例：
const float EPSINON = 0.00001;
if ((x >= - EPSINON) && (x <= EPSINON)
不可将浮点变量用“==”或“！=”与数字比较，应该设法转化成“>=”或“<=”此类形式。</td><td>如下是错误的写法，不得分。
if (x == 0.0)
if (x != 0.0)</td></tr>
<tr><td colspan="2">请写出 char *p 与“零值”比较的 if 语句。（3 分）</td></tr>
<tr><td>标准答案：
if (p == NULL)
if (p != NULL)</td><td>如下写法均属不良风格，不得分。
if (p == 0)
if (p != 0)
if (p)
if (!)</td></tr>
</table>

二、以下为 Windows NT 下的 32 位 C++程序，请计算 sizeof 的值（10 分）

<table>
<tr><td rowspan="2">char str[] = “Hello” ;
char *p = str ;
int n = 10;
请计算
sizeof (str) = 6 （2 分）

sizeof (p) = 4 （2 分）

sizeof (n) = 4 （2 分）</td><td>void Func (char str[100])
{
请计算
sizeof(str) = 4 （2 分）
}</td></tr>
<tr><td>void *p = malloc(100);
请计算
sizeof (p) = 4 （2 分）</td></tr>
</table>

三、简答题（25 分）

1、头文件中的 ifndef/define/endif 干什么用？（5 分）

答：防止该头文件被重复引用。

2、#include <filename.h> 和 #include “filename.h” 有什么区别？（5 分）

答：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**

3、const 有什么用途？（请至少说明两种）（5 分）

答：（1）可以定义 **const** 常量

（2）**const** 可以修饰函数的参数、返回值，甚至函数的定义体。被 **const** 修饰的东西都受到强制保护，可以预防意外的变动，能提高程序的健壮性。

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”？ （5 分）

答：**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。

5、请简述以下两个 for 循环的优缺点（5 分）

for (i=0; i<N; i++)
{
  if (condition)
  DoSomething();
  else
  DoOtherthing();
}
	if (condition)
	{
	  for (i=0; i<N; i++)
	  DoSomething();
	}
	else
	{
	  for (i=0; i<N; i++)
	  DoOtherthing();
	}

优点：程序简洁 缺点：多执行了 N-1 次逻辑判断，并且打断了循环“流水线”作业，使得编译器不能对循环进行优化处理，降低了效率。	优点：循环的效率高 缺点：程序不简洁

四、有关内存的思考题（每小题 5 分，共 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：程序崩溃。

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：可能是乱码。

因为 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：

（1）能够输出 hello

（2）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：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。

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

```
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
}
```

（2）strcpy 能把 strSrc 的内容复制到 strDest，为什么还要 char * 类型的返回值？

答：为了实现链式表达式。 // 2 分

例如 `int length = strlen( strcpy( strDest, "hello world") );`

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:
    char    *m_data;                  // 用于保存字符串
};
```

请编写 String 的上述 4 个函数。

标准答案：

```
// String 的析构函数
String::~String(void)                 // 3 分
{
    delete [] m_data;
    // 由于 m_data 是内部数据类型，也可以写成 delete m_data;
}
```

```
// String 的普通构造函数
String::String(const char *str)       // 6 分
{
    if(str==NULL)
    {
        m_data = new char[1];    // 若能加 NULL 判断则更好
        *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);
    }
}
// 拷贝构造函数
String::String(const String &other)   // 3 分
{
    int length = strlen(other.m_data);
    m_data = new char[length+1];       // 若能加 NULL 判断则更好
    strcpy(m_data, other.m_data);
}
// 赋值函数
String & String::operate =(const String &other)    // 13 分
{
    // (1) 检查自赋值                        // 4 分
    if(this == &other)
        return *this;

    // (2) 释放原有的内存资源              // 3 分
    delete [] m_data;

    // （3）分配新的内存资源，并复制内容 // 3 分
    int length = strlen(other.m_data);
    m_data = new char[length+1];          // 若能加 NULL 判断则更好
    strcpy(m_data, other.m_data);

    // （4）返回本对象的引用              // 3 分
    return *this;
}
```