- 一，六道选择题　（可以多选）

1　char　*p　=　"hello　world";　p　　存储在（）指向　　//　堆栈

char　p[]　=　"hello　world";　p　　存储在（）指向

全局变量　　//　数据段

static　　变量　　//　数据段

分别在哪个地方？

1　数据段　2　代码段　3　堆　4　堆栈

（此题可以配合同文件夹下的　　char.cpp　）

(二、例子程序

这是一个前辈写的，非常详细

//main.cpp

int　　a　　=　　0;　　全局初始化区

char　　*p1;　　全局未初始化区

main()

{

int　　b;　　栈

char　　s[]　　=　　"abc";　　栈

char　　*p2;　　栈

char　　*p3　　=　　"123456";　　123456\0　在常量区，　p3 在栈上。

static　　int　　c　　=0　；　　全局（静态）初始化区

p1　　=　　(char　　*)malloc(10);

p2　　=　　(char　　*)malloc(20);

分配得来得　10 和 20 字节的区域就在堆区。

strcpy(p1,　　"123456");　　123456\0　放在常量区，编译器可能会将它与　　p3 所指向的 "123456" 优化成一个地方。

}　　不知道是那个高人怎么想的和我一样，我估计中间应该有错误　　）

2　%　　&　　.　　&&　<= =　　那个优先级别最高

. &　%　<=　&&　=

3

4 以下哪些通信方式是可靠的通讯方式

1　信号　2 管道　3 消息　4tcp 5udp 6　　串口　I/O

5 是（ M）？（ a++ ）：（　a--），此处的　M 等于　　我选 C

A，M==O ，B，M==1 ，C，M！=O，D，M！=1

6 是 Unix 的启动顺序排序。　(6 个选项)

二

1 是数制转换　151 转 2 进制和九进制。　10010111　　177

2 已知 0 的 ASCII 码为 0x40，那么　int　120;　　在内存中的表示形式是　　0x__　　78（0 的 ASCII 码为 0x40，应该为　0x30）

3

1、在 linux　下，查看目录大小的命令是：　　du　– sh dirname

2　、修改文件属性的命令是：　　chomd/chgrp

3　、切换为其他用户身份的命令是：　　su

4 还有一道指针交换数值　　int　i=0,j=10,int*　　p=&i,　int*　q=&j,

int　fun　(**a,*b)

{int*　temp=a;

a=10;

b=10;

a=b;

b=temp;

}最后问调用　fun(&p,q) 问 i、j、p、q 的最终值 (具体形式大概如此，但中间指针肯定记的错误)

此题主要考察指针指向一个整数，然后利用指针改变变量，最后交换指针

- 5 有道填插入排序的算法。有一个数组　　a[0]　到 a[i-1] 为从小到大排序，　a[i]　到 a[count-1] 没有排序，请您添加　3 条语句使它们按照从小到大排序

```
int  insert_sort(int   a[],int   count)
{
        for(int   i=1;i<count;++i)
        {
        int  j,t;
        t=a[i];
        (j=i-1;)
        while(j>=0&&t<a[j])
              {
              (a[j+1]=a[j];)
              j--;
              }
              (a[j+1]=t;)
        }
        return   0;
}
```

三，编程与逻辑题

1 自己写一个　strstr

（单链表判断有无环，）

```
char*   strstr(char*    buf,   char*   sub)
{
char*   bp;
char*   sp;
If(!*sub)
return     buf;
while(*buf)
{bf=buf;
sp=sub;
do{        if(!*sp)
return   buf;
}
```

```
while(*bp++==*sp++)
buf+=1;
}
return  0;
}
```

2 遍历文本找单词并删掉出现频率最少的单词, fun (char* pText)

```
 #include <stdio.h>
#include <stdarg.h>    // 定义 av_list 、 av_start 、 av_arg 等宏
```

- 3 实现一个与 printf 功能相似的函数

```
#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <stdarg.h>    // 定义 av_list 、 av_start 、 av_arg 等宏
/*********************************************************************
此函数的作用：
   实现一个参数个数可变的函数，此函数的功能与  printf  类似,
   但在格式处理上，不如  printf  丰富
   无异常，返回一个  true, 否则返回  false
format  字符串的合法情况如下：
   1."%%zyk%%zyk%%",OUTPUT:%zyk%zyk%
   2."%dzyk%fzyk%s",OUTPUT:(int)zyk(float)zyk(string)
   3."zyk",  OUTPUT:zyk
非法情况如下：
   1."%zyk%"  ERROR:  不存在 %z 格式、 %后面必须跟一个格式字符
*********************************************************************/
bool  zykPrintf(  const  char  * format,...)
{
   // 定义一个可用于指向参数的指针（实为  char * ）,
   va_list argPtr;

   // 把函数的第一个参数  format  的地址传给  argPtr
   va_start(argPtr,format);

   const  int  size = strlen(format)+1;
   char  *tmp =  new  char [size];
   memset(tmp, 0, size);
```

- while (*format != 0)

```
{
   int  i;
   for  (i=0; i<size && *format!='%' && *format!=0; i++)
   {
      tmp[i]=*format++;
   }
```

```
tmp[i] = 0;    // 在有效的字符串末尾作     0 值防护

printf("%s",tmp);

if  (*format == 0)
    return   true ;

switch  (*++format)
{
// 按指定类型读取下一个参数     ,并打印
case 'd': { printf("%d", va_arg(argPtr,         int ));  break ;}
case 's': { printf("%s", va_arg(argPtr,        char  *));  break ; }
case 'c': { printf("%c", va_arg(argPtr,        char ));  break ;}
case 'f': { printf("%f", va_arg(argPtr,        float ));  break ;}

// 对 %%的处理
case '%': { printf("%%");      break ; }

// 格式错误
default   : { printf("     Error Ocurr!Please Check the Format!");         return   false ;}
}

        ++format;

    }

    delete[] tmp;
    return   true ;
}

int  main( int   argc, char  * argv[])
{

    zykPrintf("%zyk");      //error
    zykPrintf("zyk%");      //error
    zykPrintf("%%zyk%%zyk%%");       //OUTPUT: %zyk%zyk%
    zykPrintf("\nzyk is a pretty boy! His age is %d and %s",5,"I love zyk^_^!");

                getch();
                return   0;
}
```

- 4 是一道逻辑题，有的数是   2，3，5 的倍数，在三位数中出去可整除这三个数的和

（ 5 升和 3 升桶量 4 升水）

四，改错题三道

1tozero 算法

2 比较简单

3 是高质量里的一道题

五，问答题

1VC中有哪些方法避免 C编程中的头文件重复包含：

#ifndef !!!!

#def !!!!

#endif

2 在 C++ 中 extern c 的作用

（按键转换，比如点击 p 输出 q）

作为 extern 是 C/C++语言中表明函数和全局变量作用范围（可见性）的关键字，该关键字告诉编译器，其声明的函数和变量可以在本模块或其它模块中使用。

extern "C" 是连接申明 (linkage declaration), 被 extern "C" 修饰的变量和函数是按照 C 语言方式编译和连接的

3 编程中异步 IO 和同步 IO 有什么区别？说说你可知道的几种 IO？

4 使用异步 socket 编程，通常因为网络拥塞 send 不出数据，会获得什么样的错误码 (windows 下举例)，通常如何处理这种情况？

（核心太与用户太的区别, x86 如何转换。）

5 将程序移植到不同的 32 位 cpu 中，经常出现结构字节对齐和大小端的问题，有哪能些方法避免？

（是子网源码的判断，计算, ABCDE 网络的区别, DE 网络的用途，）

6 怎样解决在 vc 中内存泄漏的问题（ release 版本）

（1）放置关键字 assert （）

（2）生成 map 文件。它并不往 exe 文件中添加任何东西，仅仅只是把编译连接时的所有函数入口地址记录在后缀为 .map 文件。程序崩溃的时候, 可以得到一个崩溃时的 EIP 地址，通过地址可以很容易的查到崩溃所在的函数。（在 vc setting 下有个 link 按钮选上 generate mapfile ）

（3） Release 版本也是可以设置断点的，在希望设置断点处加入 _asm int 3

(4) 熟悉汇编，通过编译时的汇编看出

（5）使用第三方调试器。

（6）关掉发行版中的一些优化选项，生成调试信息。

（是 p2p 软件在 nat 用户里实现数据互传的原理

开发类笔试全部是 C/C++ ，要求对底层有一定的了解开发类的笔试题目比较晕, 共五页纸, 要求两个钟头完成（我的简历没有通过筛选，我是去霸王笔的 -_-）好像考的内容都跟网上流传的差不多，题目内容大致如下 :希望对参加深信服笔试和面试的同学有所帮助 :)

1.选择题：6 题 第一题是考变量和值的存储位置 (堆/栈/代码段 /数据段等) 最后一题是 Unix 系统的启动顺序，其他几题比较简单。

2.填空题： 4/5 题 考 sizeof、指针、数制转换、排序等，看过高质量 C/C++ 应该都没有问题。

3.改错题： 3 题 有道题跟高质量 C/C++ 中一道指针题类似，不过那题中没有错，原本不需要修改，却反倒被我改错了，汗 ...另外两题比较简单。

4.编程题 : 4/5 题 判断链表有没有环 (要求用两种方法)； 实现 C 中的 printf

深圳某公司几个 **vc/mfc** 笔试题目 **(**含参考答案 **)**

1：Release 版本下如何解决 memory leak 以及非法操作的 BUG。（搞不清什么非法操作）

2: 在异步 socket 时，为什么有时 send 不出数据，会报什么错误（分 windows/linux 下），你一般怎么处理？

下面是几个编程的

3：实现 strstr 模型 （我晕，我写了个函数，只是不是 strstr,而是 strchr）

4: 实现 printf 类似的函数 , void myprintf(char *str,...)
(用 console API 吗？好像在哪书上看到可用那些 API 实现，嘿，我就写了个 std::cout<<)
5: 删除文本文件中出现频率最小的单词, （文件里以空格表示间隔一个单词）
void func(char *pTxt)

1、 strstr 的实现原型。

```
char *my_strstr(const char *str, const char *strSearch)
{
while (*str != '\0')
{
char *p = (char *)str;
char *ps = (char *)strSearch;
while ( ps && *p == *ps )
p , ps ;
if ('\0' == *ps)
return (char *)str;
str ;
}
return NULL;
}
```

2、从指定文本中删除出现频率最少的单词, 如果有多个, 则都删除。实现 void func(char* pTxt) 函数。

... 看 单词处理, 论坛中很多都有涉及

3、 printf 的实现。

```
int printf(const char *format, ...)
{
va_list arglist;
int buffing;
```

```
int retval;

va_start(arglist, format);

_ASSERTE(format != NULL);

#ifdef _MT
_lock_str2(1, stdout);
__try {
#endif /* _MT */

buffing = _stbuf(stdout);

retval = _output(stdout,format,arglist);

_ftbuf(buffing, stdout);

#ifdef _MT
}
__finally {
_unlock_str2(1, stdout);
}
#endif /* _MT */

return(retval);
}
```

4、 VC 中有哪些方法避免 C 编译头文件重复。 （除了 #ifndef/#define/#endif 外，就想不出来了）

```
#pragma once
```

5、 extern "C" 的用法。

用于 提供 C 接口， 如使用 C 命名方式 等 .

6、异步 socket 编程中， send 不出数据的错误码是什么， （举 Linux 或 Windows 为例），你是怎么处理的？

非阻塞 SOCKET， SEND 不出数据的原因有 2 个吧， TCP 下连接断开了和该 SOCKET 处在阻塞状态（也就是说在发送数据中） 。UPD 发不出只有 TCP 后面的情况。

处理的办法就是记录下该 SOCKET 的状态，当状态为阻塞的时间， 放入缓冲， 当该 SOCKET 再次可写时，发送。

7、 异步 IO 和同步 IO 有什么区别？举例说明有几种（如 read）？

异步 IO 当函数返回时不一定就完成了 IO 操作，而 同步 IO 已经完成了。所以 异步 IO 需要有一个事件，当 IO 完成时会设置此事件，调用者在事件上等待。

8、 32 位系统中，出现结构字节对齐的问题和大小端的问题的避免？

#pragma pack(4)

9、如何查出内存泄漏和非法操作的 BUG （在 Release 版本下）？

使用 map 文件

1， PostMessage只把消息放入队列，不管其他程序是否处理都返回，然后继续执行，这是个异步消息投放函数。 而 SendMessage必须等待其他程序处理消息完了之后才返回，继续执行，这是个同步消息投放函数。而且， PostMessage 的返回值表示 PostMessage函数执行是否正确； 而 SendMessage的返回值表示其他程序处理消息后的返回值。 这点大家应该都明白。

2， 如果在同一个线程内， PostMessage发送消息时，消息要先放入线程的消息队列，然后通过消息循环 Dispatch 到目标窗口。 SendMessage发送消息时，系统直接调用目标窗口的消息处理程序，并将结果返回。 SendMessage在同一线程中发送消息并不入线程消息队列。 如果在不同线程内。最好用 PostThreadMessage 代替 PostMessage,他工作的很好。 SendMessage发送消息到目标窗口所属的线程的消息队列，然后发送消息的线程等待 (事实上，他应该还在做一些监测工作， 比如监视 QS_SENDMESSAGE 状态)，直到目标窗口处理完并且结果返回，发送消息的线程才继续运行。 这是 SendMessage的一般情况，事实上，处理过程要复杂的多。比如，当发送消息的线程监测到有别的窗口 SendMessage 一个消息到来时，他直接调用窗口处理过程 (重入)，并将处理结果返回 (这个过程不需要消息循环中 GetMessage等的支持)。

3， msdn: If you send a message in the range below WM_USER to the asynchronous message functions (PostMessage, SendNotifyMessage, and SendMessageCallback), its message parameters can not include pointers. Otherwise, the operation will fail.

如果发送的消息码在 WM_USER 以下 (非自定义消息) 且消息参数中带有指针， 那么 PostMessage,SendNotifyMessage,SendMessageCallback 这些异步消息发送函数将会调用失败。 最好不要用 PostMessage 发送带有指针参数的消息。

PostMessage 和 SendMessage 的区别主要在于是否等待其他程序消息处理。 PostMessage 只是把消息放入队列， 不管其他程序是否处理都返回， 然后继续执行； 而 SendMessage必须等待其他程序处理消息后才返回，继续执行。这两个函数的返回值也不同， PostMessage 的返回值表示 PostMessage函数执行是否正确， 而 SendMessage的返回值表示其他程序处理消息后的返回值。