```
//三：省略号形参(...)：可变参数函数；
//省略号形参一般无法正确处理类类型对象；能正确处理的int,char *;

//这种省略形参式的可变参数函数，虽然参数数量不固定，但是函数的所有参数是存储在线性连续的栈空间中的，
//而且带...的可变参数函数必须至少要有一个普通参数，我们就可以通过这个普通参数来寻址后续的所有可变参数的类型以及值。
```

```
void funcTest(const char *msg, ...) //"2"
{
    va_list valist; //创建一个va_list类型的变量
    int csgs = atoi(msg); //这个就是拿到可变参数数量，这个数量我们后续要用于获取可变参数；
    va_start(valist, msg); //使valist指向起始的参数
    int paramcout = 0;
    while (paramcout < csgs)
    {
        char *p; //这块我们假定给的可变参数都是字符串；  如果是printf你就要根据%d,%s来分析可变参数类型。
        p = va_arg(valist, char *);
        printf("第%d个参数是:%s\n", paramcout, p);
    }
}
```

这里，其实可以省略，即写出（const char *msg...）含义相同

```
//int printf(char const *co
funcTest("3", "aa", "bb", "
```

```
Microsoft Visual Studio 调试控制
第0个参数是:aa
第1个参数是:bb
第2个参数是:cc
```

va_list 、va_start、 va_arg、 va_end 使用说明

转载 土戈 2018-11-07 13:55:53 8410 收藏 26 展开

在ANSI C中，这些宏的定义位于stdarg.h中：

typedef char *va_list;

va_start宏，获取可变参数列表的第一个参数的地址（list是类型为va_list的指针，param1是可变参数最左边的参数）：

#define va_start(list,param1) (list = (va_list)&param1+ sizeof(param1))

va_arg宏，获取可变参数的当前参数，返回指定类型并将指针指向下一参数（mode参数描述了当前参数的类型）：

#define va_arg(list,mode) ((mode *)(list += sizeof(mode)))[-1]

va_end宏，清空va_list可变参数列表：

#define va_end(list) (list = (va_list)0)

注：以上sizeof()只是为了说明工作原理，实际实现中，增加的字节数需保证为为int的整数倍

如：#define _INTSIZEOF(n) ((sizeof(n) + sizeof(int) - 1) & ~(sizeof(int) - 1))

```
/a)至少有一个有效的形参，形参不能是完全的...
/b)...形参只能出现在形参列表最后一个位置。void myfunc(参数列表,...);
/c)...之前的,可以省略;
/d)如果有多个普通参数，那么va_start(varlist,msg),第二个参数必须绑...之前的那个参数;
/e)一般这些可变参数类型是数值型或者字符串型还能正常处理，其他类类型一般都不能正常处理，所以...用的场合并不多。
/f)不建议大家在c++中使用...但是遇到了这种...大家要能看懂;
```