VA_LIST 是在 C 语言中解决变参问题的一组宏

他有这么几个成员：

1） va_list 型变量：

```
#ifdef   _M_ALPHA
typedef struct {
        char *a0;       /* pointer to first homed integer argument */
        int offset;     /* byte offset of next parameter */
} va_list;
#else
typedef char *   va_list;
#endif
```

2）_INTSIZEOF 宏，获取类型占用的空间长度，最小占用长度为 int 的整数倍：

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

```
高地址 |-----------------------------------|
       |----------函数返回地址--------------|
       |----------.....------------------|
       |-----------------------------------| <--va_arg 后 ap 指向
       |第 n 个参数（第一个可变参数）       |
       |-----------------------------------|<--va_start 后 ap 指向
       |第 n-1个参数（最后一个固定参数）    |
 低地址|-----------------------------------|<--&v
```

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

```
#define va_start(ap,v)   ( ap = (va_list)&v + _INTSIZEOF(v) )
```

4）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（t 参数描述了当前参数的类型）：

```
#define va_arg(ap,t)      ( *(t *)((ap += _INTSIZEOF(t)) - _INTSIZEOF(t)) )
```

5）VA_END 宏，清空 va_list 可变参数列表：

```
#define va_end(ap)        ( ap = (va_list)0 )
```

VA_LIST 的用法：

（1）首先在函数里定义一具 VA_LIST 型的变量，这个变量是指向参数的指针；

（2）然后用 VA_START 宏初始化变量刚定义的 VA_LIST 变量，使其指向第一个可变参数的地址；

（3）然后用 VA_ARG 返回可变的参数，VA_ARG 的第二个参数是你要返回的参数的类型（如果函数有多个可变参数的，依次调用 VA_ARG 获取各个参数）；

（4）最后用 VA_END 宏结束可变参数的获取。

使用 VA_LIST 应该注意的问题：

（1）可变参数的类型和个数完全由程序代码控制,它并不能智能地识别不同参数的个数和类型；

（2）如果我们不需要一一详解每个参数，只需要将可变列表拷贝至某个缓冲，可用 vsprintf 函数；

（3）因为编译器对可变参数的函数的原型检查不够严格,对编程查错不利.不利于我们写出高质量的代码；

小结：可变参数的函数原理其实很简单，而 VA 系列是以宏定义来定义的，实现跟堆栈相关。我们写一个可变参数的 C 函数时，有利也有弊,所 以在不必要的场合，我们无需用到可变参数，如果在 C++里，我们应该利用 C++多态性来实现可变参数的功能，尽量避免用 C 语言的方式来实现。

举例：

```
#include <stdio.h>
#include <stdarg.h>
/* calculate sum of a 0 terminated list */
void sum(char* msg, ...);
Int main(int argc, char* argv[])
{
    sum("The total is %d\n", 1, 2, 3, 4, 5, 0);
    return 0;
}
void sum(char* msg, ...)
{
    int total = 0;
    int arg;
    va_list ap;
    va_start(ap, msg); /* ap 指向第一个可变参数*/
    while ((arg = va_arg(ap, int)) != 0)
    {
```

```
            total += arg;
        }
        printf(msg, total);
        va_end(ap);
}
```