# vJass 系列教程 5

文本宏

**Aeris** ▸ NJU ▸ 2008-7-9

# vJass 系列教程 5

文本宏

## 教程简介

在做地图时，我们经常碰到一些很“像”的代码，比如下面的一组 ReturnBug 函数：

```
function I2H takes integer i returns handle
    return i
    return null
endfunction

function I2Unit takes integer i returns unit
    return i
    return null
endfunction

function I2Trigger takes integer i returns trigger
    return i
    return null
endfunction

function I2Effect takes integer i returns effect
    return i
    return null
endfunction
```

这些代码都差不多，但是我们却不得不一遍又一遍地重复写类似的代码，而且以后万一代码要改，那么所有重复的代码都要修改，真是麻烦啊。如果有一种机制能够减少这种重复工作，那该多好。

vJass 提供了文本宏，从某种程度上解决了这个问题，如果用文本宏的话，这段代码就像这样：

```
//! textmacro ReturnBugFunction takes NAME, TYPE
    function I2$NAME$ takes integer i returns $TYPE$
        return i
        return null
    endfunction
//! endtextmacro

//! runtextmacro ReturnBugFunction("H", "handle")
//! runtextmacro ReturnBugFunction("Unit", "unit")
//! runtextmacro ReturnBugFunction("Trigger", "trigger")
//! runtextmacro ReturnBugFunction("Effect", "effect")
//! runtextmacro ReturnBugFunction("Sound", "sound")
//! runtextmacro ReturnBugFunction("Timer", "timer")
//! runtextmacro ReturnBugFunction("Destructable", "destructable")
//! runtextmacro ReturnBugFunction("Lightning", "lightling")
// ......还有更多......
```

这样在最终地图脚本里我们就有了 8 个 ReturnBug 函数，只用 14 行代码就完成了比前面 19 行代码多两倍的工作量，是不是很方便？

## 文本宏语法

文本宏是一个比较简单的语法。它的机制类似于 C/C++中因为滥用而饱受批评的宏，为防止文本宏也被滥用，vJass 作者对宏的语法作了适当的限制，使得它的功能既强大，又受到一定的限制而不会被乱用。

### 定义

定义文本宏的基本语法是：

```
//! textmacro 宏名 [takes 预处理参数列表]
    文本宏主体
//! endtextmacro
```

文本宏的定义在一对`//! textmacro`和`//! endtextmacro`之中，注意`//!`是一种有特殊意义的预处理注释，它后面可以跟许多特定的标记，编译时可以被 vJass 预处理器识别，从而产生各种代码。和 Unix 里 Shell 脚本开头的“#! /bin/sh”作用有点像。这里，它后面跟的是 textmacro 标记，表明下面的内容是文本宏。

*宏名*：文本宏的名字，在调用文本宏时使用。文本宏的名字可以和函数、变量、库、域、结构等同名而不会引起混乱，但是不能和其他文本宏同名。文本宏的命名遵循 Jass 函数的一般命名规则。文本宏定义**不可嵌套**。（截止 vJass 0.9.B.1 版本为止）

*预处理参数列表*：文本宏调用的时候一般都会带参数，参数可以不止一个，中间用逗号隔开，就像上面的例子一样。参数列表很像函数的参数列表，但是文本宏的参数是没有类型的，因为只是“文本”而已。参数的命名遵循 Jass 函数和变量一般命名规则。不过，我个人习惯把文本宏的参数名统一大写，以避免和函数参数混淆，就像上面例子中的 NAME 和 TYPE 一样。注意如果一个文本宏没有参数，那么就不要写 takes 了，直接写“`//! textmacro 宏名`”就可以了。

*主体*：文本宏的主要内容，每次运行宏的时候，主体代码会被原样复制一份，然后作参数替换。主体可以是任意内容，除了“子”文本宏，因为其定义不可嵌套。
主体需要引用参数时，需要把参数名**放在一对“$”中**，比如：

```
function I2$NAME$ takes integer i returns $TYPE$
```

中间的**$NAME$**和**$TYPE$**就是引用了 NAME 和 TYPE 参数。在编译预处理时，参数名连同两边的“$”会被替换成实际文本。

### 调用

文本宏定义好了就可以调用了。文本宏的调用和函数十分像，但要注意文本宏**不遵循**先定义后调用的规则。调用文本宏的语法是：

```
//! runtextmacro 宏名()                                  // 无参数
//! runtextmacro 宏名("参数 1", "参数 2", ...)          // 带参数
```

调用无参数的文本宏只要在预处理注释//! runtextmacro 后面写上宏名和括号就可以了。对于有参数的文本宏，其参数要**用一对引号括起**，放在表示参数的括号中，中间用逗号隔开，比如：

```
//! runtextmacro ReturnBugFunction("Unit", "unit")
```

文本宏被调用时，其内容在执行参数替换后，完整复制一份到//! runtextmacro 处。比如使用“H”和“handle”为参数来调用文本宏时，会被替换成：

```
function I2$NAME$ takes integer i returns $TYPE$
```

```
function I2H takes integer i returns handle
```

## 文本宏的应用

文本宏在哪里用呢？下面情况可以考虑用：

- 有大量相似的重复代码时（例如几段代码结构相同，只是有几个类型或者函数名不同）
- 有复制——粘贴——替换的欲望时

记住，文本宏功能很强大，但它只是文本替换而已，善用而不要滥用。

这里就不举例了，简介中有一个例子，教程 4 中的示例也用到了文本宏，感兴趣的读者可以去看看。