【长发言：旁白】 # 为了确保内容出现【】等符号时不会被误认为指定了发言者

【函数：样例四】# 也可写作：【章节：样例四】

这是语涵编译器原型的样例文本其四，文件组织与控制流部分说明。

在之前的内容中，除了选项命令使用户可以点选选项之外，其他样例的内容都是一条直线，从头到尾。实际使用中，我们会希望能将内容分散在多个文件中，部分剧情可能会用在不止一处。这最后一个样例文本就来解释这些。

首先，除了全局生效的命令（比如声明角色、声明场景）之外，语涵编译器要求所有的内容（如发言，角色上场退场命令，场景切换命令等）都在“函数”或者说“章节”之下，这是进行内容管理的单位。上面的【函数：样例四】就定义了叫“样例四”的章节，之后所有的内容都在这个函数内。不包含在函数、章节内的内容将不会被用于生成，所以如果该文档在函数命令之前还有内容的话，这些内容都将被忽略。不过，如果整个文档通篇都没有章节、函数命令的话，语涵编译器会根据文档名创建一个函数并把所有内容移至其中。这也是之前几个样例能够正确进行代码生成的保证。

【转至函数：第二个函数】# 下面会介绍

【函数：第二个函数】

从这里开始，所有的内容都属于“第二个函数”。如果上一个函数没有“转至函数”命令，语涵编译器会为其添加返回指令。转至函数指令使得控制流能够到达该函数。

如果要手动调用函数，那么可以使用如下命令：（这里我们用一个选项命令来避免其执行）

【选项：名称=另一个无意义的选项】

* 调用函数
  + 【调用函数：第二个函数】# 后面仍可接其他内容
  + 这则内容可以正常显示，与此同时，该选项结束处还会跳转至选项出口
* 转至函数
  + 【转至函数：第二个函数】# 后面不可以接其他内容，认为该调用不会返回
* 正常继续
  + 继续

当使用“调用函数”命令时，语涵编译器假定当控制流转移至被调用的函数之后，在某个时间可能还会返回，所以该命令之后还可以续其他内容。当使用“转至函数”命令时，语涵编译器假定不会返回，所以该命令后不能再接内容。【函数返回】命令可以结束当前函数。

【调用函数：第三个函数】

【函数返回】# 也可写作【章节结束】

这句话不在任何函数中，不会出现在输出里。

【函数：第三个函数】

除了上述的函数间跳转外，函数内也可以使用基于标签的函数内跳转。使用基于标签的跳转有如下限制：1. 所有的跳转都是单向的，有去无回；2. 所有的标签名都只在函数内部有效，其他函数无法跳转。

【标签：标签1】

与函数命令相同，当内容中出现标签时，游戏运行到此处时默认会跳转到该标签所在位置。要进行跳转时可使用【转至标签】命令，比如：

【选项：名称=又一个无意义的选项】

* 转至标签
  + 【转至标签：标签1】# 后面不可以接其他内容
* 正常继续
  + 继续

由于目前暂未支持变量与表达式，语涵编译器还不支持有条件分支，所以能够改变控制流的指令就这些。

对于当前的场景、出场的角色、定义的别名状态等运行时状态，在不同的控制流路径上可能会不一样，语涵编译器会努力使其在控制流交汇点保持一致。目前如果这些信息存在分歧，语涵编译器会采用在内容上“从上到下”中第一个可以到达交汇点的状态。后续抵达的路径中多出来的内容会被去掉（角色退场、图片收起，等等）。

以上就是语涵编译器的样例四。感谢您的坚持！完结撒花！