Vim：寄存器与宏

寄存器

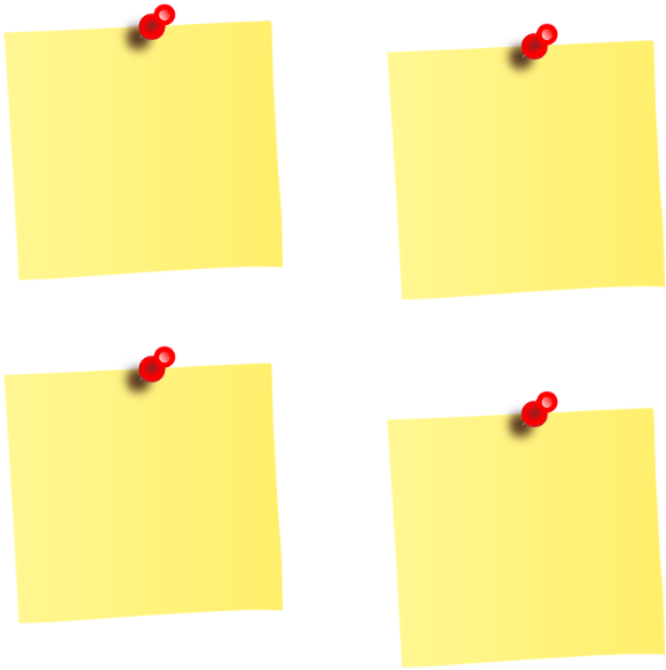
Vim 提供了许多寄存器用于存放内容，可以理解为剪贴板

一个字符对应一个寄存器（如 `a-z`，`0-9`）

特别的寄存器：

- `"`：默认寄存器，平时复制、删除的内容都放在里面
- `%`：当前文件名
- `.`：上一次插入的内容
- `:`：上一次执行的命令

通过 `:reg {register}` 查看对应寄存器中的内容



指定寄存器

在复制/删除/粘贴等操作前加上 `"{register}` 就可以指定本次操作所用的寄存器

只要涉及寄存器操作的都可以这样指定

- `"ayy`：将这一行复制到 `a` 寄存器中
- `"bdiw`：将单词删除，保存到 `b` 寄存器中
- `"cp`：将 `c` 寄存器中的内容粘贴出来

常见用途：将想要**持久保存**的文本放到特定寄存器里，随时进行粘贴，避免被覆盖

寄存器字符大写：添加（append）而非覆盖

宏

宏（Macro）：录制一系列键盘操作，并允许我们重放这些操作

操作序列存储在指定的寄存器中

- `q{register}`：开始录制宏，存在寄存器 `register` 中
- 录制过程中按 `q` 退出录制
- `@{register}`：重放寄存器 `register` 中的操作
- `@@`：重放上一次宏操作

常见用法：`q{register}` 录制一段操作，`@{register}` 重放，然后一直 `@@` 重放

> 注意：`.` 命令对宏不生效，`.` 命令只记录上一次修改，而宏可能包含多次修改

建议：让你的宏对连续重放友好

1. 让你的光标移动更加 general
2. 假设你要在多个特定的位置进行特定的操作，一个好的习惯是在宏录制的最后，跳转到下一个需要编辑的位置

   即，宏包括 【编辑动作】+【跳转到下一个需要编辑的位置】

   这样一来，连续重放就可以实现对所有需要编辑的位置进行编辑操作

通过大写的寄存器，在宏的后面添加命令

如果宏是重放友好的，允许我们使用 `[count]@{register}` 直接重放 `count` 次

小结

- 了解寄存器的概念
- 了解常用的内置寄存器
- （**重要**）掌握使用宏进行批量操作