

Vimエディタ

マクロの話

小見 拓

mail@nanasi.jp

マクロって何ですか

- 操作を記録する機能
- 記録した操作を実行する機能
- キーボードマクロとか呼ばれる

マクロを使うと、できること

- vimエディタ上でできることは、だいたい何でもできる。
- カーソル操作、テキスト入力
- コマンド実行、ウィンドウ移動
- 検索、置換処理

マクロでは、できないこと

- vimエディタの外の操作
- いったん他のアプリケーションに
移ったフォーカスの移動などは無理

Vimエディタのマクロは超強力

- そんなわけで、マクロ厨が
マクロ教の布教に参りました。

自己紹介

- 発表者 小見 拓 (おみ たく)



小文字

- 名無しの**vim**使い <http://nanasi.jp/>
- はてダ <http://d.hatena.ne.jp/taku-o/>
- twitter taku_o
- 仕事、趣味とか <http://taku-o.net/>

マクロの使い方

マクロの使い方

詳細は後で。
a-zA-Z0-9"が使える

- q{レジスタ}で記録開始
- 何か操作する
- qで記録終了
- @{レジスタ}で記録し
た操作を実行

操作例.

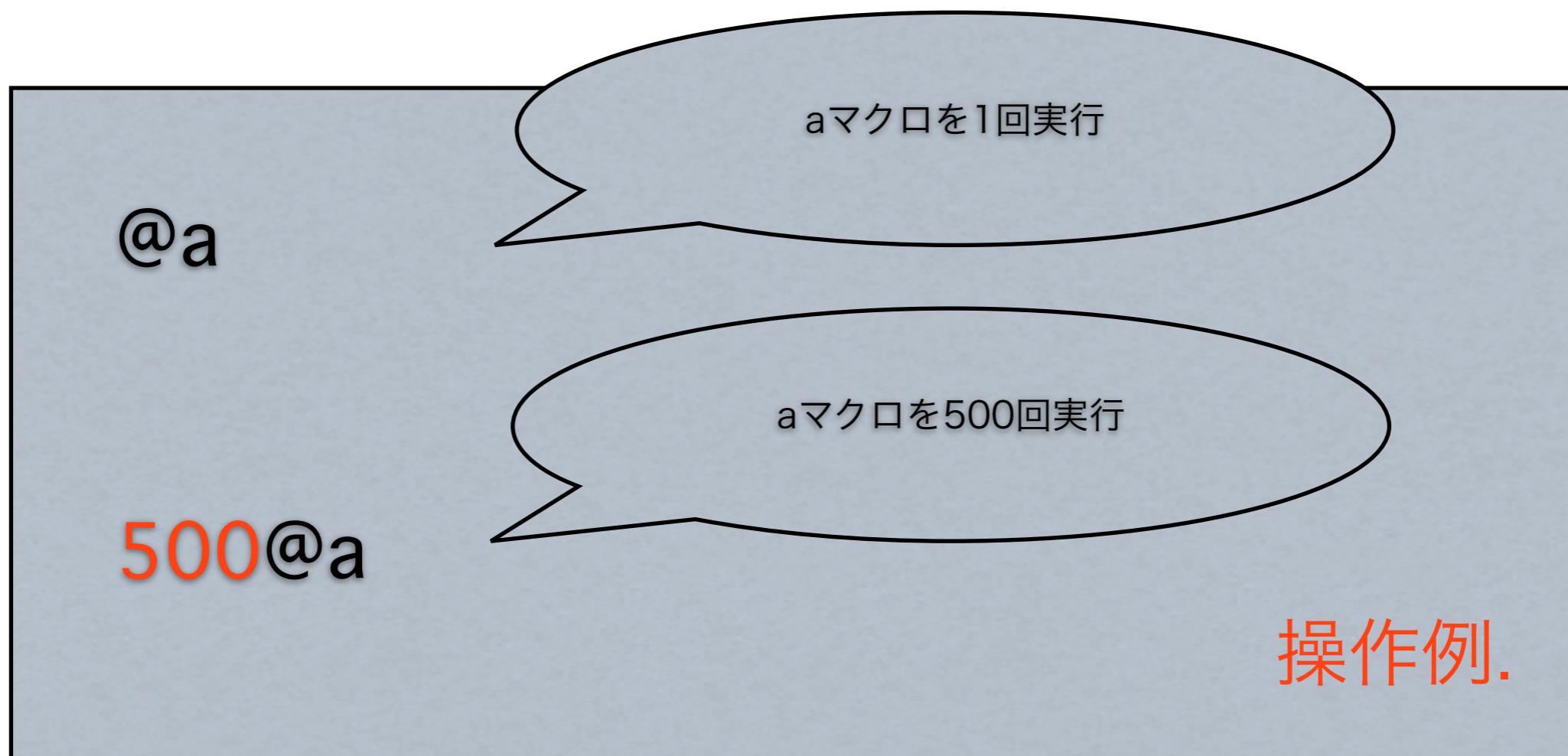
```
455 <
456 <
index.txt [utf
recording

qa          aマクロを作成
dd          マクロ作成完了
q           aマクロを実行
@a
```

簡単でしょ？

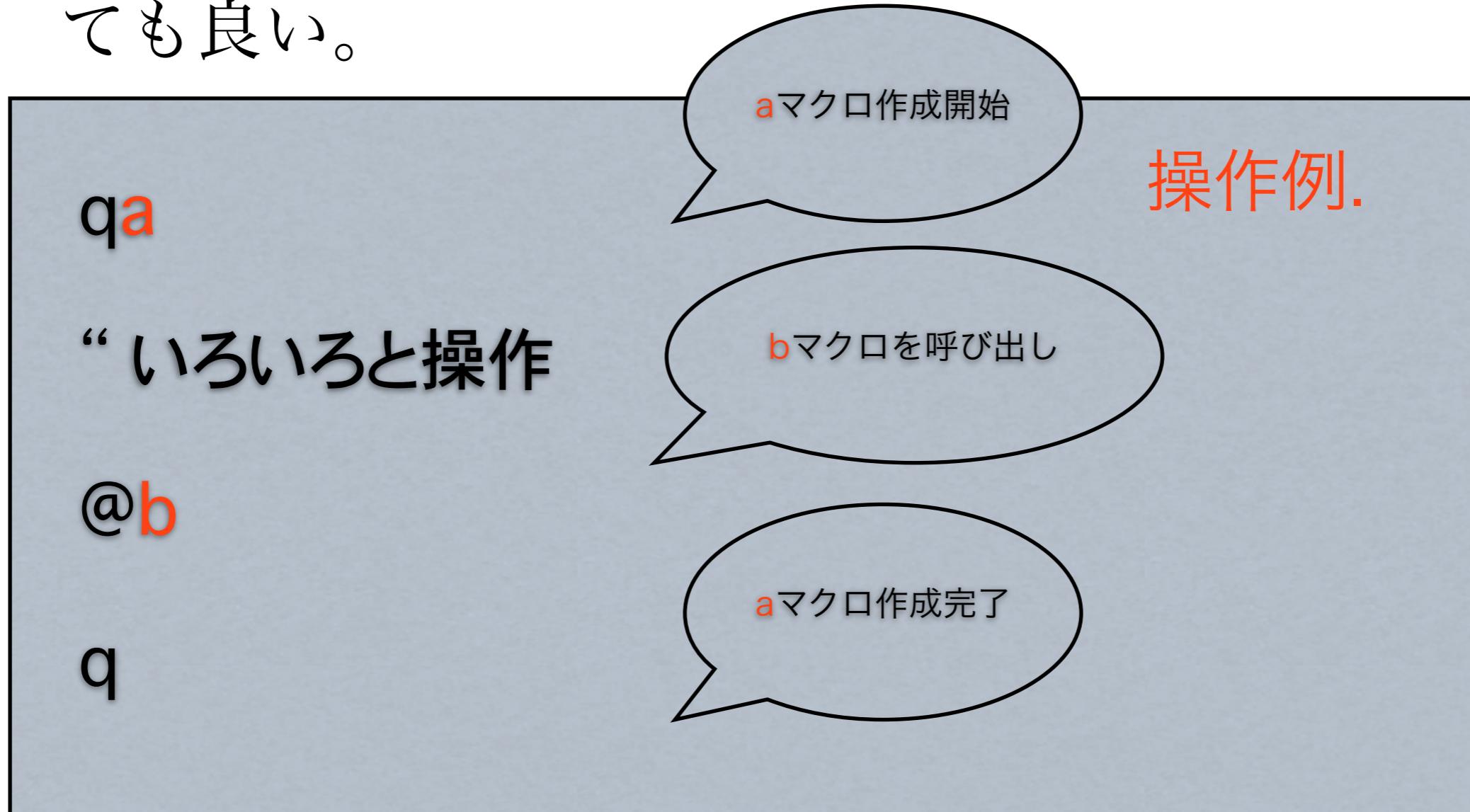
連続してマクロを実行する

- 数字を頭につけると、指定した数字の回数分だけマクロを連続して実行できる。

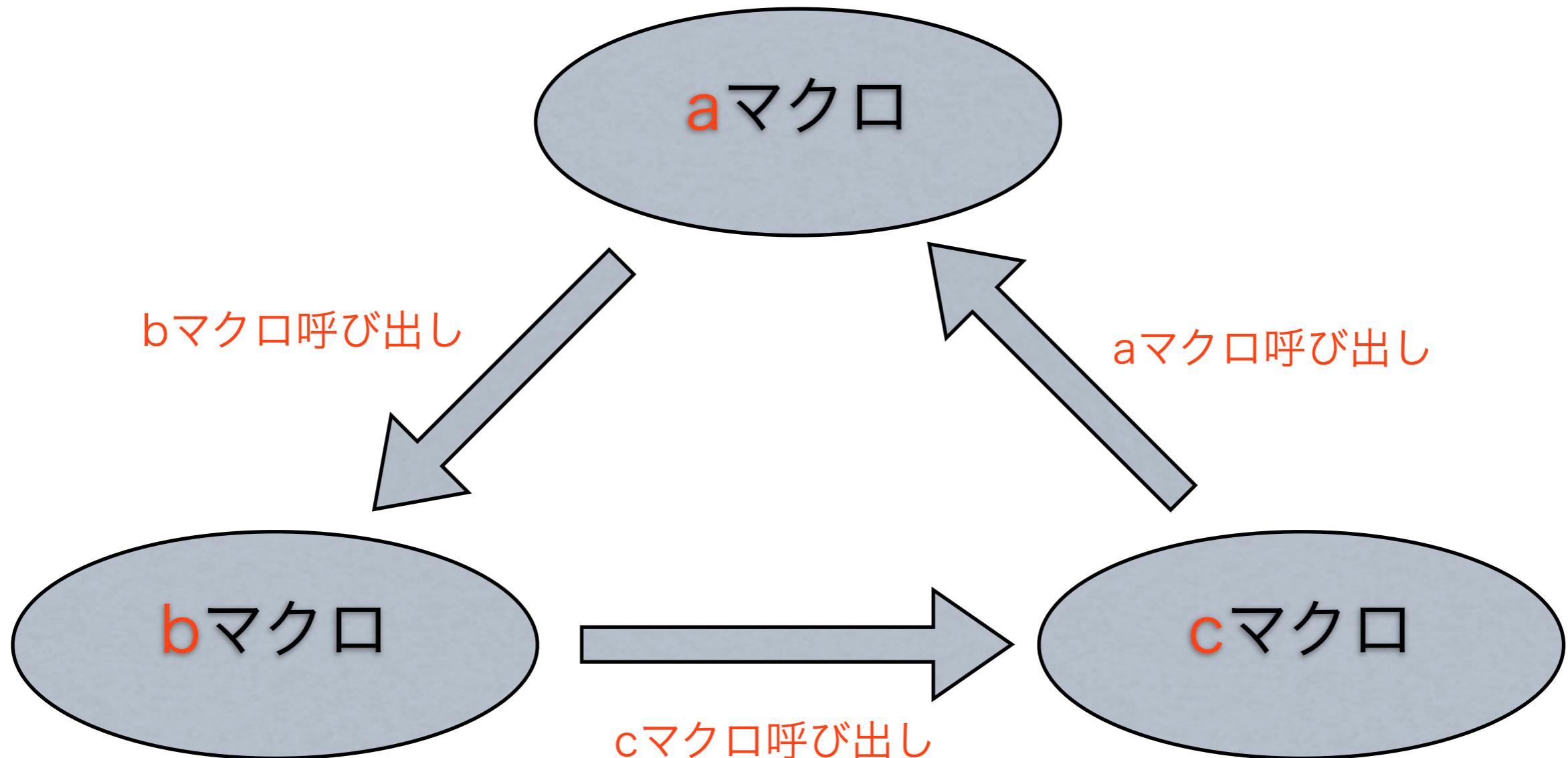


マクロから他のマクロを呼び出せる

- あるマクロの記録中に、別のマクロを呼び出しても良い。



こういうこともできる

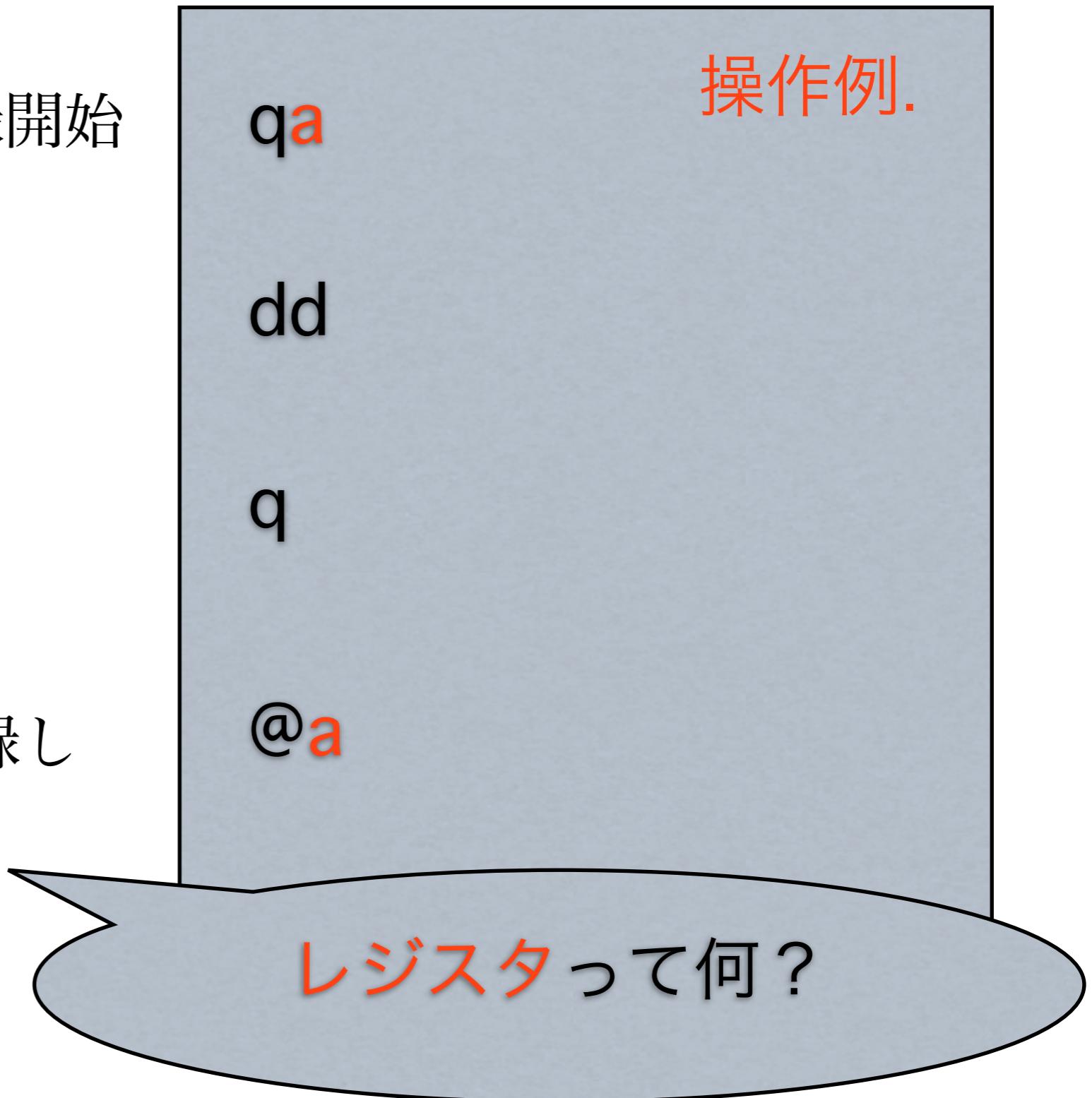


無限ループって怖いですね。
(Control-Cで止められる)

マクロの仕組み

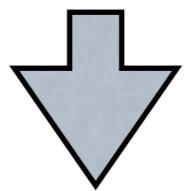
マクロの仕組み

- `q{レジスタ}`で記録開始
- 何か操作する
- `q`で記録終了
- `@{レジスタ}`で記録した操作を実行

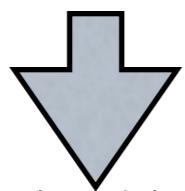


マクロの仕組み

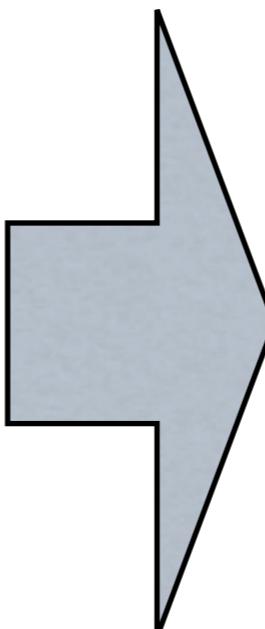
- `q{レジスタ}`で記録開始



- 何か操作する



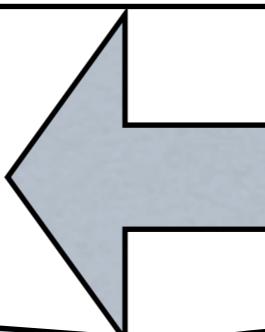
- `q`で記録終了



レコーディング機能で
操作を
レジスタに記録

- `@{レジスタ}`で記録し

た操作を実行



レジスタに記録された
操作を読み込んで実行

レジスタはデータを記録できる変数
みたいなもの

レジスタ

- Vimエディタで行われた様々な処理のデータが入る。（検索、コピー、カット、実行コマンド、開いたファイル）
- マクロの操作記録もレジスタで管理される。
- :registersか、:displayコマンドで、レジスタの中身を確認できる。

:registers

:display

操作例.

レジスタの種類

| レジスタ | | 読込 | 書込 |
|------|---|----|----|
| “ | 無名レジスタ。編集作業やレジスタの操作を行うと、書き込まれる。 | ○ | ○ |
| 0 | 無名レジスタと似ているが、別のレジスタを指定してレジスタ操作した場合には、書き込まれない。 | ○ | ○ |
| 1~9 | 番号つきレジスタ。1行以上の編集作業を行うと、書き込まれる。 | ○ | ○ |
| a~z | 名前つきレジスタ。 | ○ | ○ |
| A~Z | 名前つきレジスタ。書き込むとa~zレジスタに追記する。 | ○ | ○ |
| / | 最後に検索されたパターン | ○ | ○ |
| : | 最後に実行されたコマンド | ○ | × |
| % | カレントファイル | ○ | × |
| # | 代替ファイル | ○ | × |
| = | expressionレジスタ。式の結果を取り出せる。 | ○ | × |

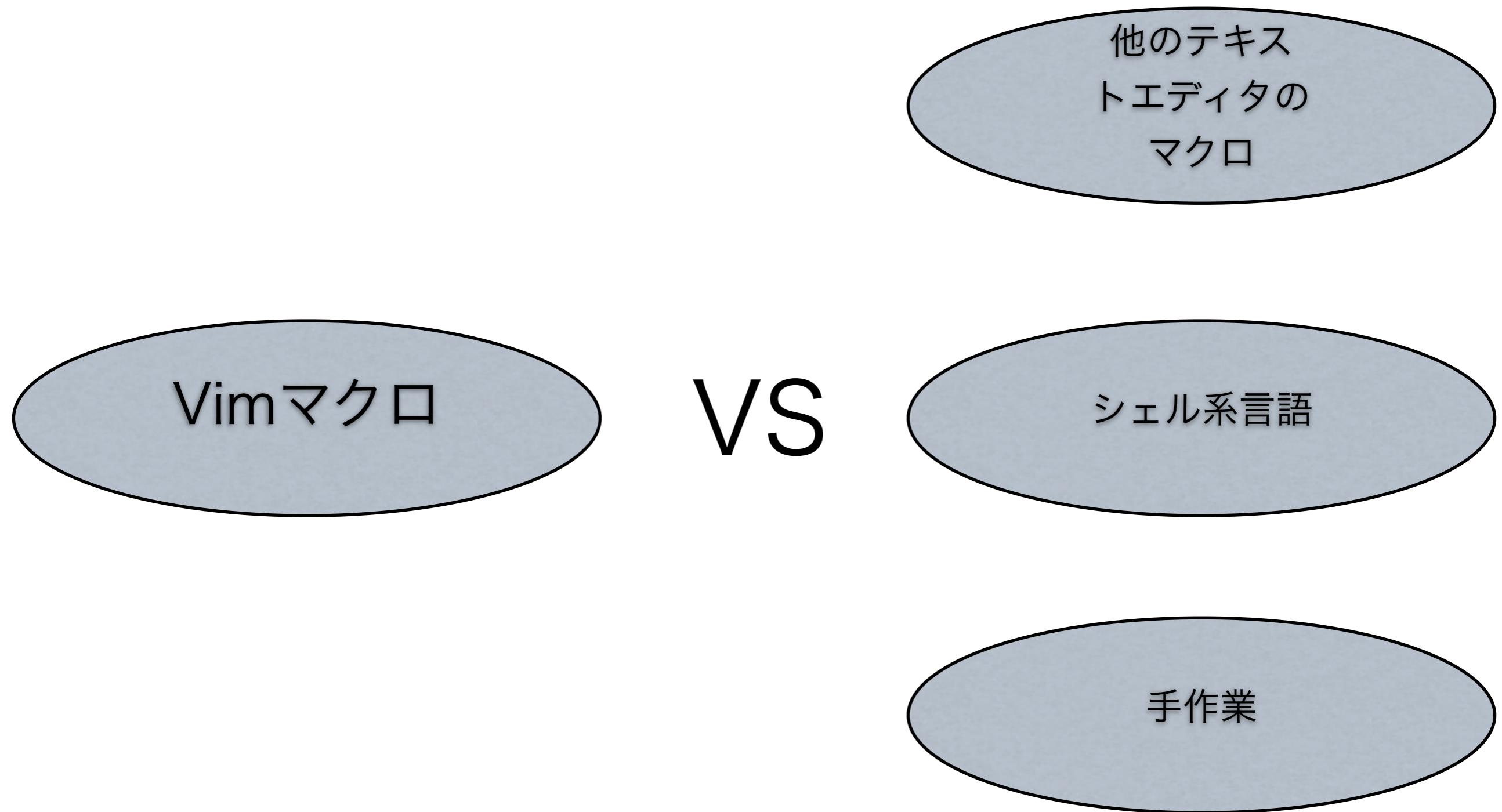
他にもレジスタはあります。

マクロで利用できるレジスタ

- マクロはa-zA-Z0-9"に記録できる
- 基本はアルファベット小文字を使用する
 - 0"のレジスタは頻繁に書き換わってしまう
 - 1~9のレジスタは番号がズれてしまう
- アルファベット大文字レジスタは追記。機会があれば使いましょう。

マクロをどういう時に使
えば良いのか

Vimエディタマクロの特徴



マクロの良いところ

- Vimのカーソル移動能力を使って、編集箇所を簡単に細かく指定できる
- 頭使わない。簡単。
- 作って、即使える。手間がかからない。
- 何回でも実行できる

マクロの駄目なところ

- マクロを登録するコストがかかる
- 登録済みマクロのミスを修正しづらい。作り直した方が早い。
- 処理速度が遅い
- 条件分岐つきマクロを作るのは面倒くさい

マクロが向いている作業

- 少しだけの編集処理
- 件数そこそこ
 - 6件～10万件くらい？
- 使い捨ての処理
- スクリプトを組もうとすると、面倒なもの

テンプレート的な修正

- 変数のget、setメソッドの作成。
- 同じ原因で起きた、複数のコンパイルエラーの修正。
- データ構造のコンバート。

マクロお題①

```
package example;

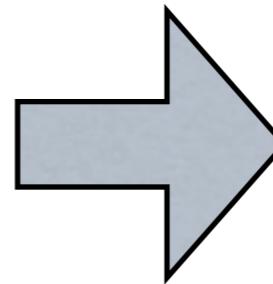
/**
 * this is dummy class.
 */
public class HelloWorld {

    public HelloWorld() {
        super();
    }

    public static void main(String[] argv) {
        // dummy
    }

    private int privateld;
    private String givenName;
    private String familyName;
    private String givenNameKana;
    private String familyNameKana;
    private String postalCode;
    private String tel;
    private String phone;
    private String cellphone;
    private String email;

}
```



このように変更

```
package example;

/**
 * this is dummy class.
 */
public class HelloWorld {

    public HelloWorld() {
        super();
    }

    public static void main(String[] argv) {
        // dummy
    }

    private int privateld;
    public int getPrivateld() {
        return this.privateld;
    }
    public void setPrivateld(int privateld) {
        this.privateld = privateld;
    }
    private String givenName;
    public String getGivenName() {
        return this.String;
    }
    public void setGivenName(String givenName) {
        this.givenName = givenName;
    }
    private String familyName;
    public String getFamilyName() {
        return this.String;
    }
    public void setFamilyName(String familyName) {
        this.familyName = familyName;
    }
    private String givenNameKana;
    public String getGivenNameKana() {
        return this.String;
    }
    public void setGivenNameKana(String givenNameKana)
```

データの抜き出し

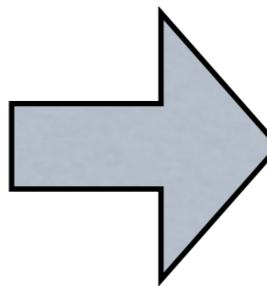
- ぐちゃぐちゃのデータ形式からのデータ抜き出し処理
- デザイン重視のExcelファイルから、データを抜き出したいとか

データの生成

- 簡易なテストデータの作成
- 繰り返しだけど、1レコードごとに、少し変えなければならない場合

マクロお題②

```
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (1,  
'TST1', '太郎1', '田中', 'タロウ1', 'タナカ',  
'tst1@example.com', now(), now(), 0);
```

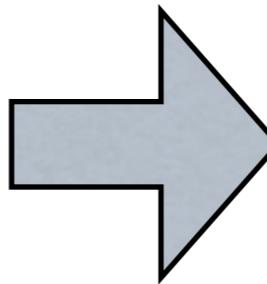


このように変更

```
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (1,  
'TST1', '太郎1', '田中', 'タロウ1', 'タナカ',  
'tst1@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (2,  
'TST2', '太郎2', '田中', 'タロウ2', 'タナカ',  
'tst2@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (3,  
'TST3', '太郎3', '田中', 'タロウ3', 'タナカ',  
'tst3@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (4,  
'TST4', '太郎4', '田中', 'タロウ4', 'タナカ',  
'tst4@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (5,  
'TST5', '太郎5', '田中', 'タロウ5', 'タナカ',  
'tst5@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (6,  
'TST6', '太郎6', '田中', 'タロウ6', 'タナカ',  
'tst6@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (7,  
'TST7', '太郎7', '田中', 'タロウ7', 'タナカ',  
'tst7@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
email, created, updated, update_counter) VALUES (8,  
'TST8', '太郎8', '田中', 'タロウ8', 'タナカ',  
'tst8@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,
```

マクロお題③

```
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
postal_code, phone, cell_phone, email, created,  
updated, update_counter) VALUES (1, 'TST000001',  
'太郎000001', '田中', 'タロウ000001', 'タナカ',  
'111-0001', '043-000-0001', '090-0000-0001',  
'tst1@example.com', now(), now(), 0);
```



このように変更

```
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
postal_code, phone, cell_phone, email, created, updated,  
update_counter) VALUES (1, 'TST000001', '太郎  
000001', '田中', 'タロウ000001', 'タナカ', '111-0001',  
'043-000-0001', '090-0000-0001',  
'tst1@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
postal_code, phone, cell_phone, email, created, updated,  
update_counter) VALUES (2, 'TST000002', '太郎  
000002', '田中', 'タロウ000002', 'タナカ', '111-0002',  
'043-000-0002', '090-0000-0002',  
'tst2@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
postal_code, phone, cell_phone, email, created, updated,  
update_counter) VALUES (3, 'TST000003', '太郎  
000003', '田中', 'タロウ000003', 'タナカ', '111-0003',  
'043-000-0003', '090-0000-0003',  
'tst3@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
postal_code, phone, cell_phone, email, created, updated,  
update_counter) VALUES (4, 'TST000004', '太郎  
000004', '田中', 'タロウ000004', 'タナカ', '111-0004',  
'043-000-0004', '090-0000-0004',  
'tst4@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
postal_code, phone, cell_phone, email, created, updated,  
update_counter) VALUES (5, 'TST000005', '太郎  
000005', '田中', 'タロウ000005', 'タナカ', '111-0005',  
'043-000-0005', '090-0000-0005',  
'tst5@example.com', now(), now(), 0);  
INSERT INTO MEMBER (pk, unique_id, given_name,  
family_name, given_name_kana, family_name_kana,  
postal_code, phone, cell_phone, email, created, updated,  
update_counter) VALUES (6, 'TST000006', '太郎  
000006', '田中', 'タロウ000006', 'タナカ', '111-0006',  
'043-000-0006', '090-0000-0006',
```

マクロお題④

Webブラウザの表示

| PK | ユニークID | 氏名 | カナ | 電話番号 | Eメールアドレス |
|----|-----------|-------------|---------------|--------------|------------------|
| 1 | TST000001 | 田中 太郎000001 | タナカ タロウ000001 | 043-000-0001 | tst1@example.com |

HTMLソース

```
<html>
<body>

  <h1>ユーザー一覧</h1>
  <table border=1>
    <tr>
      <th>PK</th>
      <th>ユニークID</th>
      <th>氏名</th>
      <th>カナ</th>
      <th>電話番号</th>
      <th>Eメールアドレス</th>
    </tr>
    <tr>
      <td>1</td>
      <td>TST000001</td>
      <td>田中 太郎000001</td>
      <td>タナカ タロウ000001</td>
      <td>043-000-0001</td>
      <td>tst1@example.com</td>
    </tr>
  </table>

</body>
</html>
```

データ件数を
増やしてみましょう。

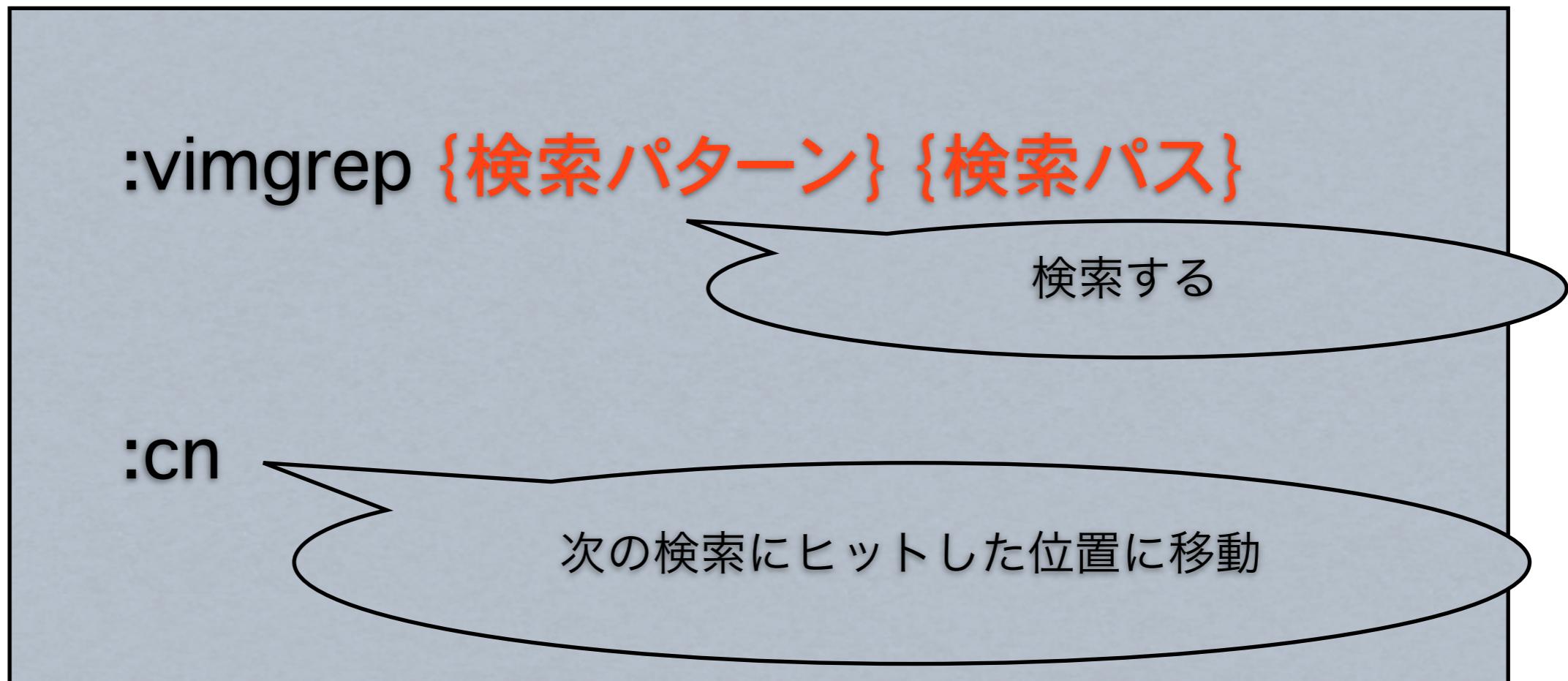
マクロ使うなら、これを
覚えておこう

マクロ使うなら、これを覚えておこう

- :vimgrep、:cn。
- インクリメント、デクリメント。
- レジスタ。
- expressionレジスタ。
- 亂数。

:vimgrep、:cn

- サーチアンドリプレースする時に使う。



インクリメント、デクリメント

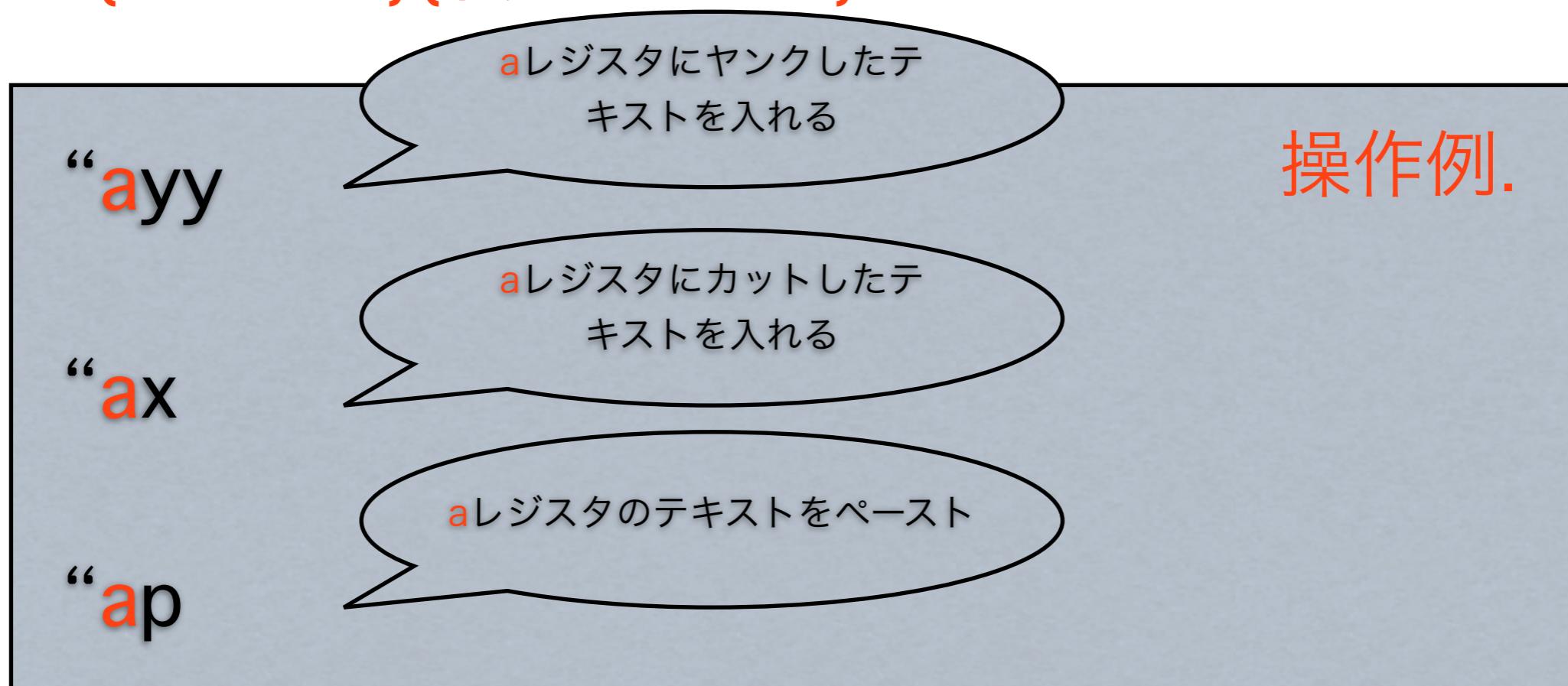
- データ生成する時に使う。
- 数字の上にカーソルを移動して、
Control-aで、数字をインクリメント。
- Control-xで、数字をデクリメント。
- 数字の桁の増減に注意。

レジスタ

- データを一時的に逃がしたい時などに使う。
- expressionレジスタにデータを渡す時に使う。

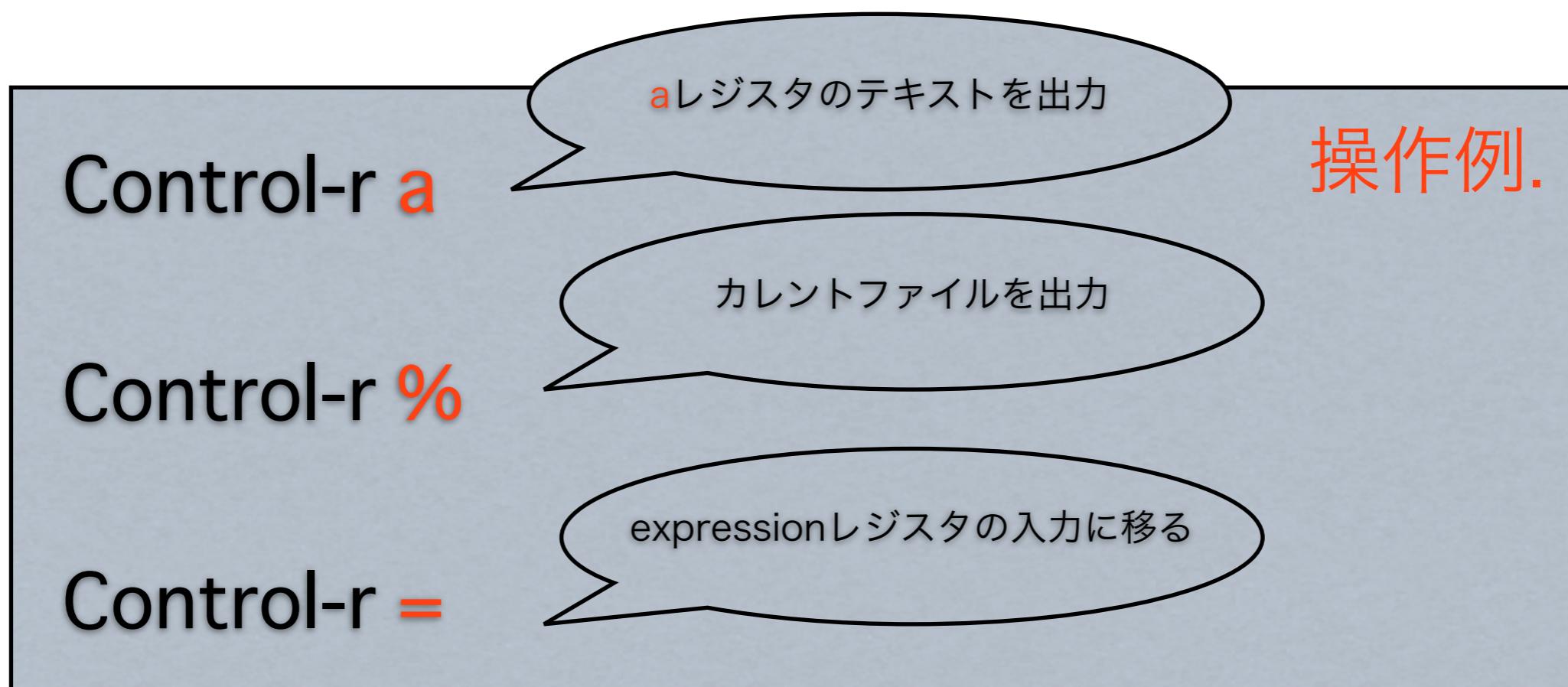
ノーマルモード、ビジュアルモードで レジスタを操作

- “{レジスタ}”でレジスタを操作できる。
- “{レジスタ}{編集コマンド}”



コマンドラインモード、インサート モードでレジスタを操作

- **Control-r {レジスタ}**で、指定したレジスタのデータを出力できる。



コマンドラインモード、インサートモードで レジスタを操作

| レジスタ | | 読込 | 書込 |
|------|---|-----------------------|-----------------------|
| “ | 無名レジスタ。編集作業やレジスタの操作を行うと、書き込まれる。 | <input type="radio"/> | <input type="radio"/> |
| 0 | 無名レジスタと似ているが、別のレジスタを指定してレジスタ操作した場合には、書き込まれない。 | <input type="radio"/> | <input type="radio"/> |
| 1~9 | 番号つきレジスタ。1行以上の編集作業を行うと、書き込まれる。 | <input type="radio"/> | <input type="radio"/> |
| a~z | 名前つきレジスタ。 | <input type="radio"/> | <input type="radio"/> |
| A~Z | 名前つきレジスタ。書き込むとa~zレジスタに追記する。 | <input type="radio"/> | <input type="radio"/> |
| / | 最後に検索されたパターン | <input type="radio"/> | <input type="radio"/> |
| : | 最後に実行されたコマンド | <input type="radio"/> | × |
| % | カレントファイル | <input type="radio"/> | × |
| # | 代替ファイル | <input type="radio"/> | × |
| = | expressionレジスタ。式の結果を取り出せる。 | <input type="radio"/> | × |

この表を覚えていませんか？

Control-rで、これらのデータを出力できます。

vimスクリプト、コマンドで レジスタを操作

- @{レジスタ}でレジスタにアクセスできる。

```
:let @a = 'registers'
```

aレジスタに'registers'という
テキストを入れる

```
:echo @a
```

aレジスタを出力する

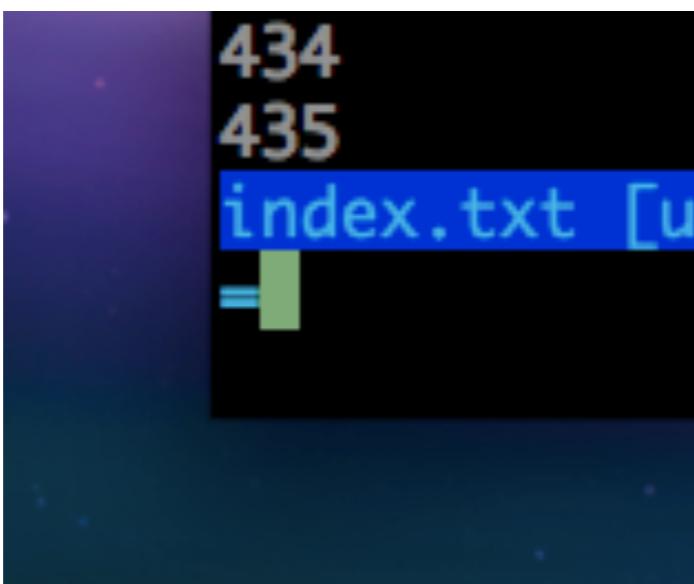
操作例.

expressionレジスタ

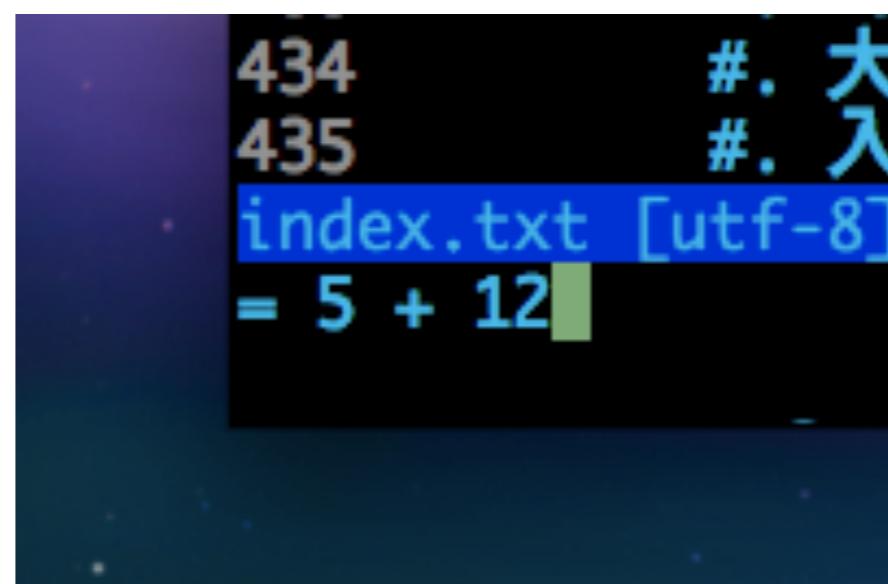
- レジスタっぽくないレジスタ。でも、レジスタのように操作できる。
- 演算の結果、ファンクションの戻り値、変数の値を取り出せる。
- 式には、変数が使える。

expressionレジスタの使い方①

- インサートモードにする。コマンドラインモードでも可。
- 「Control-r =」と入力する。
- 式を入力する。



```
434
435
index.txt [u
=
```



```
434          #. 大
435          #. 入
index.txt [utf-8]
= 5 + 12
```

expressionレジスタの使い方②

= 5 + 12

計算

式の例.

= \$HOME

変数の値を読み込む

= &encoding

= test_var

ファンクションを呼び出す

= Test()

= strftime('%Y/%m/%d %H:%M:%S')

時間

= printf('%08d', 3)

書式文字列

expressionレジスタの使い方③

- マクロの中でexpressionレジスタを使うのなら、次の方法を覚えておきましょう。
- データをレジスタのどれかに書き込み、そのレジスタの値を式の中で使う。

```
= printf('%08d', @x)
```

✗レジスタの値を整形出力

乱数

- 特に良い方法なし。知恵と努力で回避。

```
function! Rand(max)
    perl << EOF
        my $max = VIM::Eval('a:max');
        my $value = int(rand($max)), '\n';
        VIM::DoCommand("let l:r = $value")
    EOF
    return l:r
endfunction
```

知恵と努力で回避した例。

まとめ

- Vimエディタのマクロは何でもできる。
- しかも、簡単、即使える。覚える事もなし。
- 使い捨て基本だから、エレガントである必要もなし。多少強引でも許される。
- 超便利だから、みんなもマクロ厨になろう。
- でも、別の解法も考慮しつつ作業しないと、駄目。

ご静聴ありがとうございました。