

# Neovim プラグイン dial.nvim の紹介

monaqa

2023年6月30日

# 自己紹介

---

**名前** monaqa

**GitHub** @monaqa

**Bluesky** monaqa.bsky.social

**テキストエディタ** Neovim (4年半)

**よく書く言語** Python, Rust, Lua, TypeScript

**最近の Vim 活** Vim 駅伝で記事を書くなど

**好きな Vim キーバインド** `f`, `t`, `<C-a>`, `<C-x>`

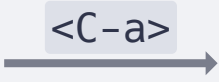
今日のお話



# Vim では数値の増減が簡単にできる

- ◆ NORMAL モードで `<C-a>` や `<C-x>` を押すだけ

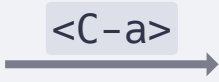
- ▶ `<C-a>`: カーソル上の（またはカーソル後の）数字を 1 増やす

ゴリラ.vim #27  ゴリラ.vim #28

- ▶ `<C-x>`: カーソル上の（またはカーソル後の）数字を 1 減らす

font-size: 10px;  font-size: 9px;

- ◆ 同じ行にあれば、カーソルより右の数字を見つけて自動で移動してくれる

ブレーキランプ 5回点滅  ブレーキランプ 6回点滅

## <C-a> / <C-x> の便利なところ

- ◆ **カウント**：数字を前置して増減値を変更できる

123  $\xrightarrow{20<C-a>}$  143      0x32  $\xrightarrow{10<C-x>}$  0x28

- ◆ **ドットリピート**：. で直前の増減操作を繰り返せる

4px 12px red;  $\xrightarrow{5<C-a>}$  9px 12px red;  $\xrightarrow{.}$  14px 12px red;  
 $\xrightarrow{w}$  14px 12px red;  $\xrightarrow{.}$  14px 17px red;

- ◆ **連番作成**: VISUAL モードで g<C-a> と押すと連番が作成できる

0. aaa  
0. bbb  
0. ccc  $\xrightarrow{g<C-a>}$  1. aaa  
2. bbb  
3. ccc

# 増減したいものはこの世にたくさんある

## 整数

`0` ↔ `1` ↔ `2` ↔ ... ↔ `10` ↔ ...

`0x00` ↔ `0x01` ↔ ... ↔ `0x0f` ↔ ...

## 小数

... ↔ `0.99` ↔ `1.00` ↔ `1.01` ↔ ...

... ↔ `9.9` ↔ `10.0` ↔ `10.1` ↔ ...

## 日付・時刻

`2022-06-30`, `06/30/2022`,

`2022年06月30日`, `6/30`, ...

## 固定文字列

`true` ↔ `false`

`月` ↔ `火` ↔ `水` ↔ ... ↔ `土` ↔ `日`

## Semantic Version

`3.8.12` → `3.8.13` ⇒ `3.9.0`

## Markdown ヘッダのレベル

`#` ↔ `##` ↔ `###` ↔ ... ↔ `#####`

# 増減したいものはこの世にたくさんある

## 整数

0 ↔ 1 ↔ 2 ↔ … ↔ 10 ↔ …

0x00 ↔ 0x01 ↔ … ↔ 0x0f ↔ …

## 小数

… ↔ 0.99 ↔ 1.00 ↔ 1.01 ↔ …

… ↔ 9.9 ↔ 10.0 ↔ 10.1 ↔ …

## 日付・時刻

2022-06-30, 06/30/2022,

2022年06月30日, 6/30, …

## 固定文字列

true ↔ false

月 ↔ 火 ↔ 水 ↔ … ↔ 土 ↔ 日

## Semantic Version

3.8.12 → 3.8.13 ⇒ 3.9.0

## Markdown ヘッダのレベル

# ↔ ## ↔ ### ↔ … ↔ #####

標準でのサポートは基本的に整数のみ

# そこで dial.nvim

---

 [monaqa/dial.nvim](https://github.com/monaqa/dial.nvim)

enhanced increment/decrement plugin for Neovim.

 Lua     568     8

<https://github.com/monaqa/dial.nvim>

(必要な環境: Neovim 0.6.1 以上)

# dial.nvim が提供するもの

## ◆ 様々なテキストの増減

- ▶ 整数
- ▶ 小数
- ▶ 日付・時刻
- ▶ 固定文字列
- ▶ Semantic Version
- ▶ Markdown ヘッダのレベル
- ▶ RGB color
- ▶ カッコの種類

etc.

## ◆ 個別に有効化・無効化可能

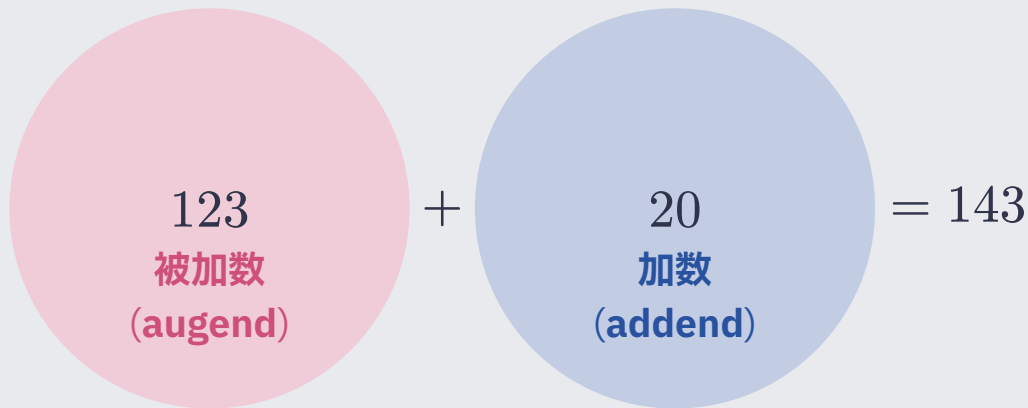
## 設定例

```
local augend = require("dial.augend")
require("dial.config").augends:register_group{
  default = {
    -- 10進整数
    augend.integer.alias.decimal,
    -- "2023/06/30" のような日付
    augend.date.new {pattern = "%Y/%m/%d"},
    -- "2023年6月30日 (金)" のような日付
    augend.date.new {
      pattern = "%Y年%-m月%-d日 (%J)",
    },
    -- true/false
    augend.constant.new {
      elements = { "true", "false" },
    },
  }
}
```



# dial.nvim における増減ルール

- ◆ dial.nvim では増減ルールのことを **被加数 (augend)** と呼んでいる



- ◆ 「日付」や「固定文字列」などは被加数の一種

# 被加数の例 (1)：日付・時刻

- ◆ バッファ上の日付や時刻を好きな単位で増減する

date: 2023/06/30  $\xrightarrow{\text{<C-a>}}$  date: 2023/07/01

date: 2023/06/30  $\xrightarrow{\text{<C-a>}}$  date: 2023/07/31

date: 2023/06/30  $\xrightarrow{\text{<C-a>}}$  date: 2024/06/30

- ◆ 様々な形式の日付や時刻に対応（設定でパターンを指定可能）

2023/06/30	2023-06-30	06/30	6月30日
06/30/2023	23/06/30	6/30	2023年6月30日(金)
30/06/2023	23/6/30	20:30	Fri 30 Jun 2023

# 被加数の例 (1) : 日付・時刻

- ◆ バッファ上の日付や時刻を好きな単位で増減する

date: 2023/06/30  $\xrightarrow{\text{<C-a>}}$  date: 2023/07/01

date: 2023/06/30  $\xrightarrow{\text{<C-a>}}$  date: 2023/07/31

date: 2023/06/30  $\xrightarrow{\text{<C-a>}}$  date: 2024/06/30

- ◆ 様々な形式の日付や時刻に対応 (設定でパターンを指定可能)

%Y/%m/%d

%Y-%m-%d

%m/%d

%-m月%-d日

%m/%d/%Y

%y/%m/%d

%-m/%-d

%Y年%-m月%-d日(%J)

%d/%m/%Y

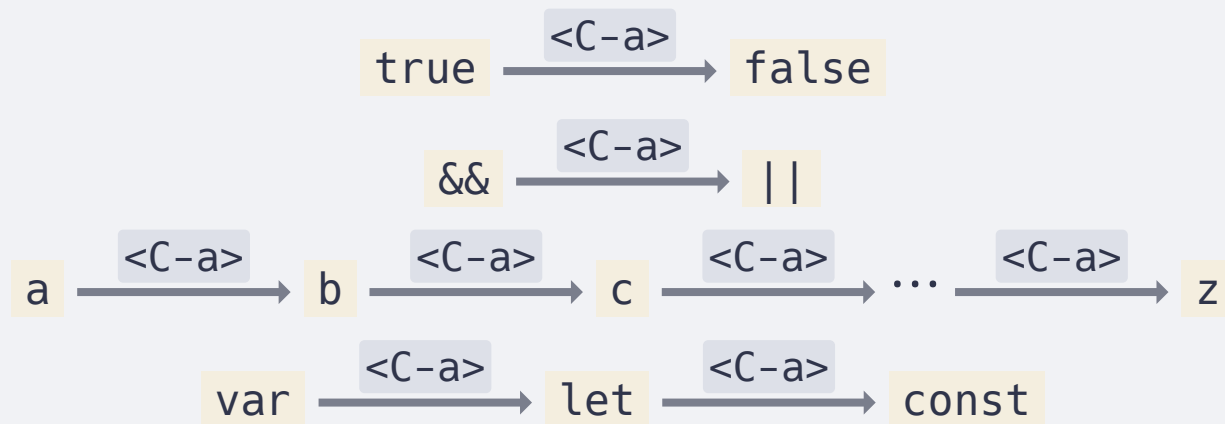
%y/%-m/%-d

%M:%S

%a %d %b %Y

## 被加数の例 (2)：固定文字列のトグル

- ◆ 複数の文字列を切り替える

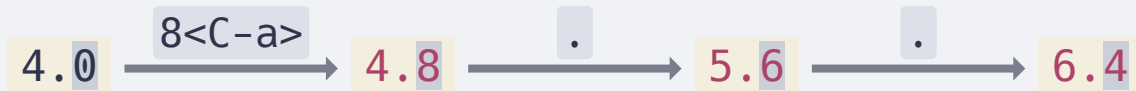


- ◆ 文字列パターンは自由に設定可能
- ◆ デフォルトでは単語として独立しているもののみ増減対象となる
  - ▶ “letter” が誤って “constter” となるのを防ぐ

# dial.nvim の特徴 (1)：標準にある機能のサポート

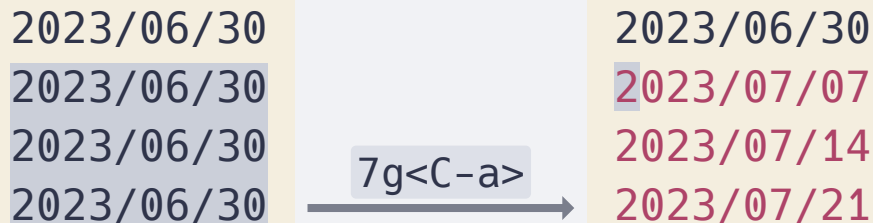
## ◆ カウント・ドットリピート をサポート

- ▶ 例: 小数の増減ルールを有効にしている場合



## ◆ ビジュアルモードでの増減、 `g<C-a>` / `g<C-x>` などをサポート

- ▶ 例: 日付の増減ルールを用いて1週間単位での連番を作る



## dial.nvim の特徴 (2)：賢いドットリピート

- ◆ 増減対象の種類を記憶して再現する

(true, 1, false)  $\xrightarrow{\text{<C-a>}}$  (false, 1, false)  $\xrightarrow{\text{W}}$  (false, 1, false)  
 $\xrightarrow{\cdot}$  (false, 1, true)

- ◆ 日付の増減では、年月日のどれを増減したか記憶して再現する

2023/06/29, 2023/07/12  $\xrightarrow{\text{<C-a>}}$  2023/07/29, 2023/07/12  
 $\xrightarrow{\text{f,}}$  2023/07/29, 2023/07/12  
 $\xrightarrow{\cdot}$  2023/07/31, 2023/08/14

# まとめ（宣伝）

---

- ◆ 標準の `<C-a>` / `<C-x>` だけでも今日は覚えて帰ってください
  - ▶ 標準の `<C-a>` / `<C-x>` だけでもぶっちゃけ便利
- ◆ `<C-a>` / `<C-x>` が好きになったら `dial.nvim` も検討してみてください
- ◆ Issue/PR 歓迎
  - ▶ Issue から生まれた機能もいくつかあります



[monaqa/dial.nvim](#)

enhanced increment/decrement plugin for Neovim.



Lua



568



8

**:qa!**

## おまけ: dial.nvim の仕組み



# そもそも <C-a> / <C-x> の役割とはなにか

## ◆ <C-a> / <C-x> の3つの仕事

1. カーソル行から、増減対象の文字列を見つける
2. 増減対象の文字列にカーソルを移動する
3. 増減後の文字列へと増減対象を書き換える

## ◆ 具体例

ある日の暮方の事である。 1 人の下人が、羅生門の下で雨やみを待っていた。

# そもそも <C-a> / <C-x> の役割とはなにか

## ◆ <C-a> / <C-x> の3つの仕事

1. カーソル行から、増減対象の文字列を見つける
2. 増減対象の文字列にカーソルを移動する
3. 増減後の文字列へと増減対象を書き換える

## ◆ 具体例

ある日の暮方の事である。 1 人の下人が、羅生門の下で雨やみを待っていた。

カーソル行から、増減対象の文字列を見つける

# そもそも <C-a> / <C-x> の役割とはなにか

## ◆ <C-a> / <C-x> の3つの仕事

1. カーソル行から、増減対象の文字列を見つける
2. 増減対象の文字列にカーソルを移動する
3. 増減後の文字列へと増減対象を書き換える

## ◆ 具体例

ある日の暮方の事である。 1 人の下人が、羅生門の下で雨やみを待っていた。

増減対象の文字列にカーソルを移動する

# そもそも <C-a> / <C-x> の役割とはなにか

## ◆ <C-a> / <C-x> の3つの仕事

1. カーソル行から、増減対象の文字列を見つける
2. 増減対象の文字列にカーソルを移動する
3. 増減後の文字列へと増減対象を書き換える

## ◆ 具体例

ある日の暮方の事である。 2 人の下人が、羅生門の下で雨やみを待っていた。

増減後の文字列へと増減対象を書き換える

# そもそも <C-a> / <C-x> の役割とはなにか

## ◆ <C-a> / <C-x> の3つの仕事

1. カーソル行から、増減対象の文字列を見つける
2. 増減対象の文字列にカーソルを移動する
3. 増減後の文字列へと増減対象を書き換える

## ◆ 具体例

ある日の暮方の事である。2 人の下人が、羅生門の下で雨やみを待っていた。

増減後の文字列へと増減対象を書き換える

これらの操作をうまく抽象化すると実装しやすそう

# 増減ルールの定式化

- ◆ dial.nvim における被加数とは、以下の2つのメソッドを持つテーブル
  - find** カーソル行から増減対象を見つけるメソッド
  - add** 増減対象の文字列を実際に増減するメソッド
- ◆ 例: 整数の増減ルールを表す **augend**

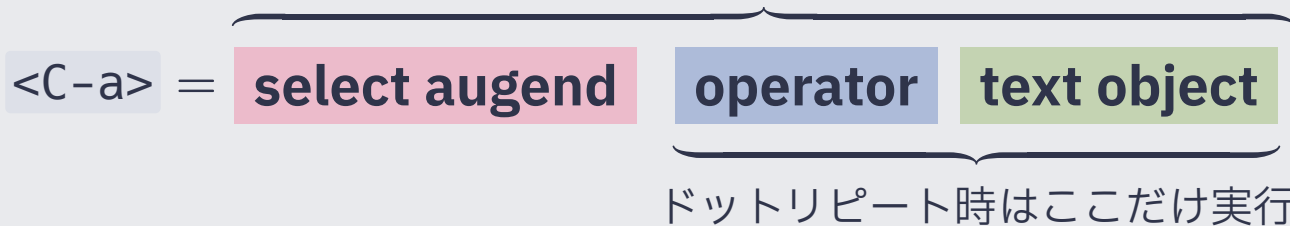
```
augend:find("foo 123", 1)  -- "foo 123" という行の頭から整数を探す
--> {from = 5, to = 7}
augend:find("123 bar", 5)  -- "123 bar" という行の5文字目から整数を探す
--> nil
```

```
augend:add("123", 1)      -- "123" を整数として見て1を足す
--> {text = "124", cursor = 3}
augend:add("123", -30)    -- "123" を整数として見て30を引く
--> {text = "93", cursor = 2}
```

# dial.nvim における増減操作

0. 予め候補となる被加数（増減ルール）のリストを設定しておく
1. **select augend**: 設定で与えた全ての被加数の `find` メソッドを呼び出し、候補が見つかった被加数のうち最適な被加数 A を選ぶ
2. **text object**: A の `find` メソッドが返した範囲を選択する
3. **operator**: A の `add` メソッドを呼び出し、戻り値でテキストを置換する

`<C-a>` を押したときは全体を実行



operator + text object と見なせば、自然にドットリピート対応できる

# 最適な被加数をどのように選ぶか？

---

## 2023年6月30日 20:00 @ 千駄ヶ谷



# 最適な被加数をどのように選ぶか？

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

短い1行の中にも様々な被加数が存在しうる。どれを選ぶべき？

# 被加数の優先順位

カーソルに重なる被加数が最優先

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

カーソル後ろの被加数は次点

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

カーソル手前の被加数は優先度低

## 2023年6月30日 20:00 @ 千駄ヶ谷

# 被加数の優先順位

## 2023年6月30日 20:00 @ 千駄ヶ谷

さらに以下の規則で優先

- ◆ 開始位置が最も左のものの優先
- ◆ 開始位置が同じ場合、終了位置が最も右のものの優先

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

## 2023年6月30日 20:00 @ 千駄ヶ谷

カーソル位置と増減範囲の相対的な位置関係のみで優先度を決めるため  
どの被加数が選ばれるか直感的に判断しやすい。

原則、**カーソルに近く、範囲の広い**被加数が優先的に選ばれる。

# まとめ

---

- ◆ dial.nvim における被加数とは「`find`, `add` メソッドを持つもの」
- ◆ dial.nvim が実際に行う操作は3種類
  - ▶ **select augend**: 最適な増減ルールを選択
  - ▶ **text object**: 増減対象の選択
  - ▶ **operator**: 増減の実行
- ◆ ドットリピートが使えるのは operator + text object による実装のおかげ
- ◆ 被加数の優先度はカーソルと増減範囲の相対位置関係のみから判断
  - ▶ 様々な種類のルールがあっても、それなりに高い精度で直感的な増減ルールを選択できる