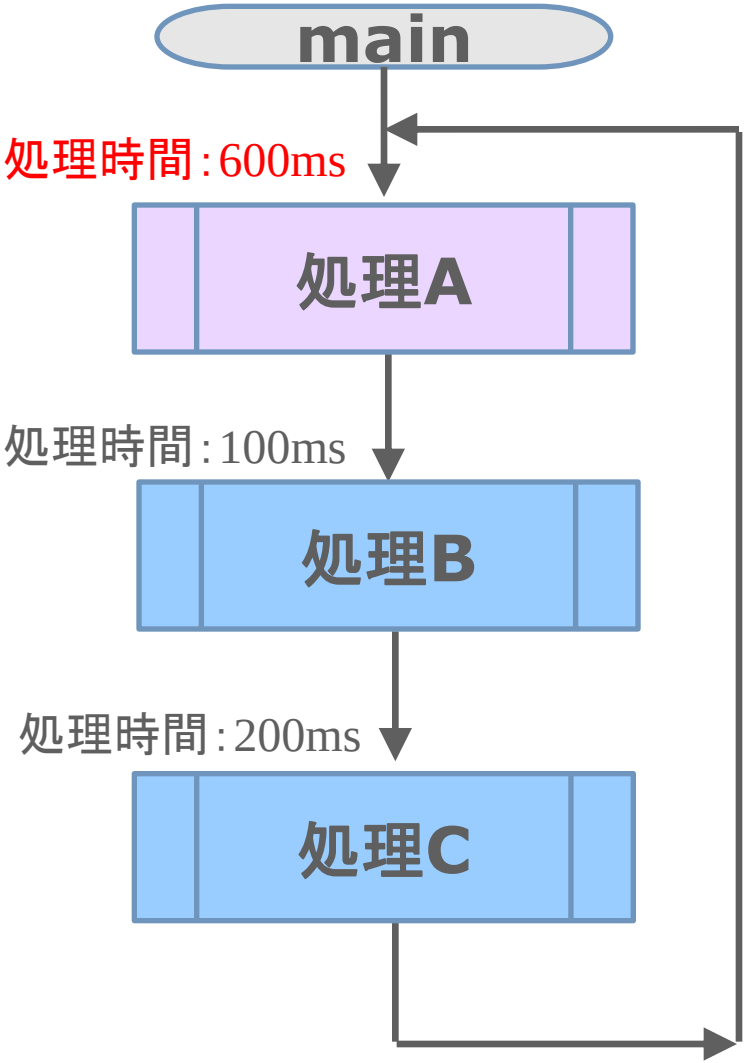
ステートマシンによる
並列処理プログラミング入門

2018/05/20

Ashita倶楽部

main関数は無限ループ



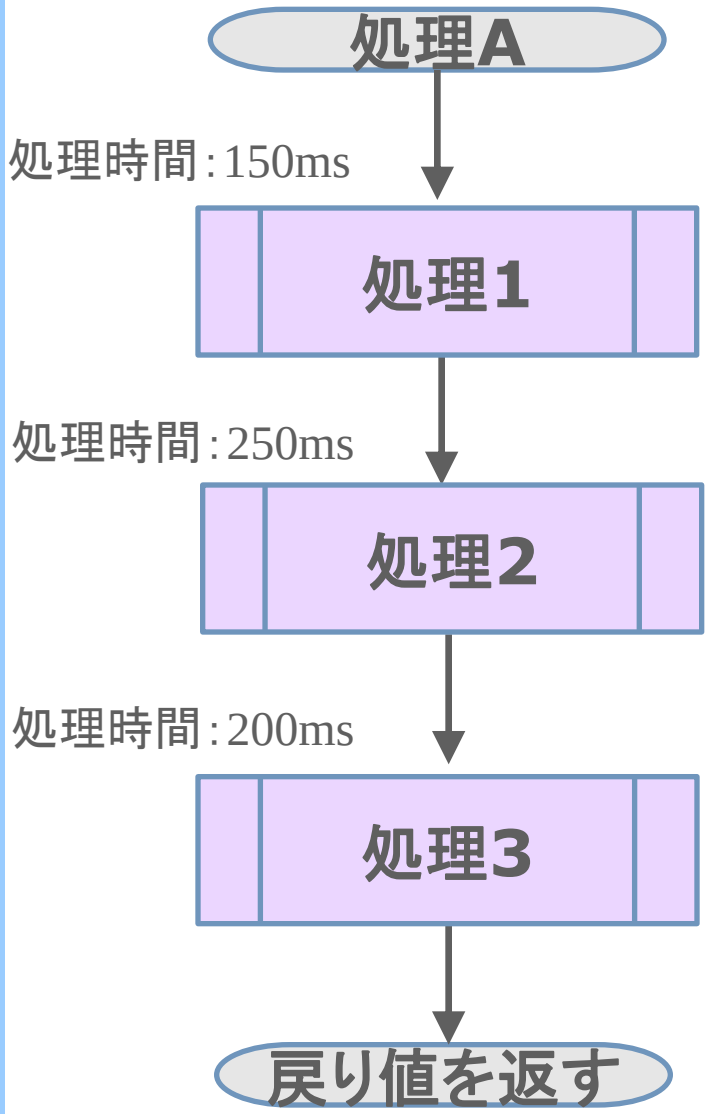
[解説]

- システム起動時に
  自動的に呼び出される

- 処理A~Cを上から順に
  繰り返し実行

※途中に別の繰り返しが
あっても良い。

その他の関数は呼び出し元に戻る

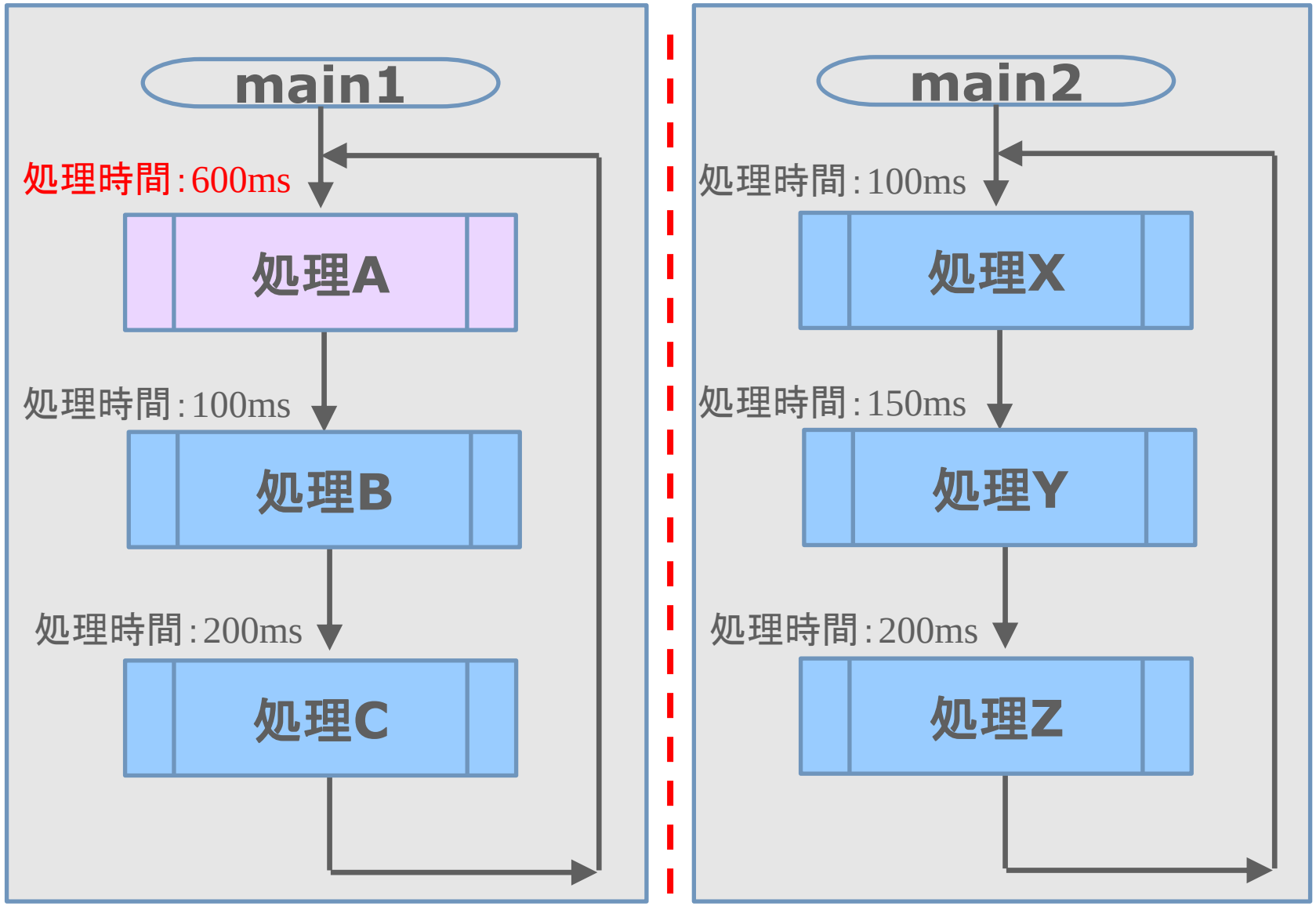


[解説]

- 別の関数から呼ばれる。
- 上から順に実行される。
  ※途中に繰り返しが
  あっても良い。
- 最後は呼び出し元に戻る。

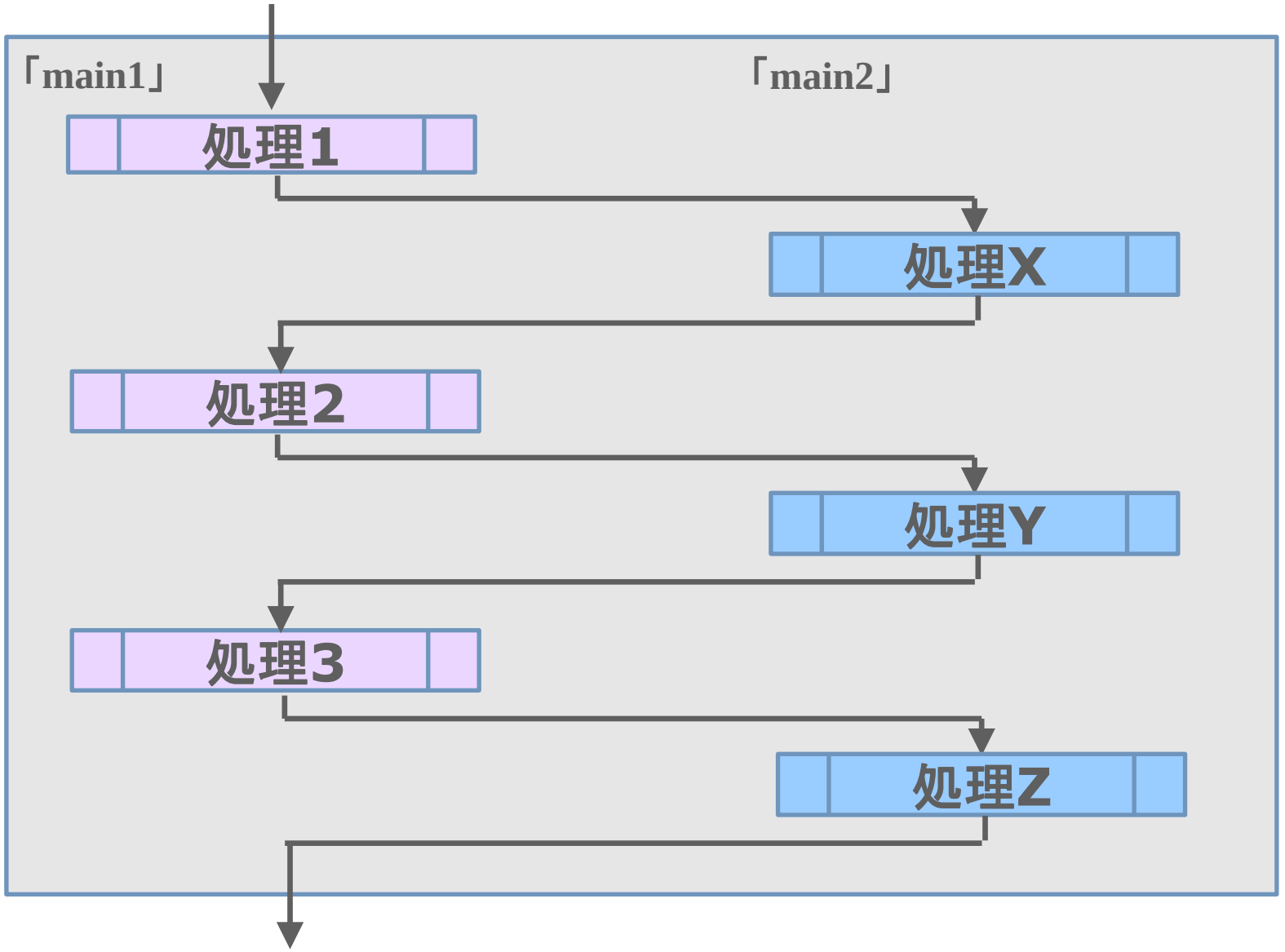
並列処理の例

2つのmain関数を「同時」に「並列」に実行させたい！

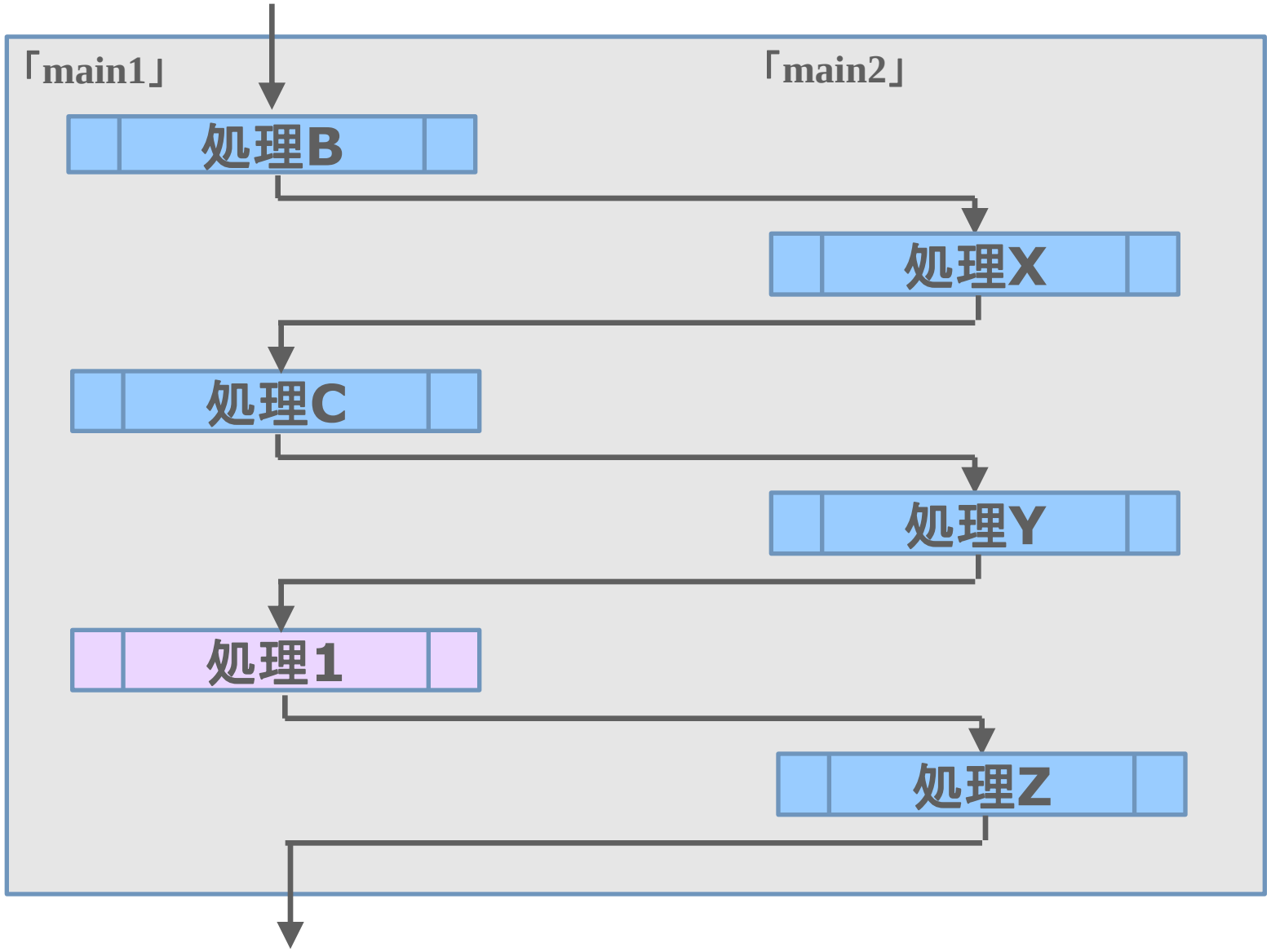


実現方法

少しずつ交互に実行する



つづき



問題点

独立していた2つの処理(main1とmain2)が混じり合ってしまった

- 2つのソースファイルを1つにマージ
  (もしくは局所関数をグローバル関数化)

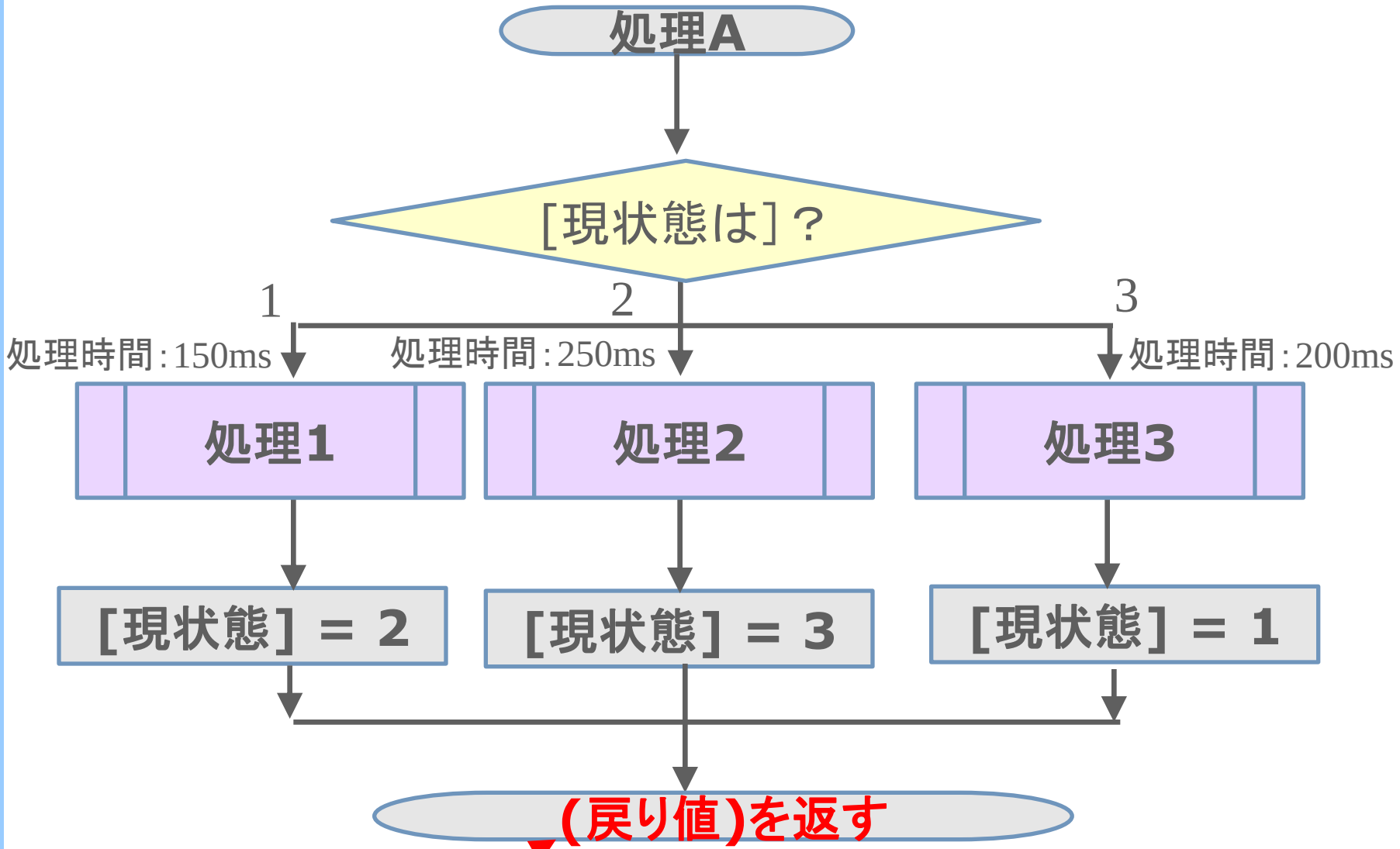
  ⇒ モジュール独立性(モジュラリティ)が崩壊

対策

関数のステートマシン化

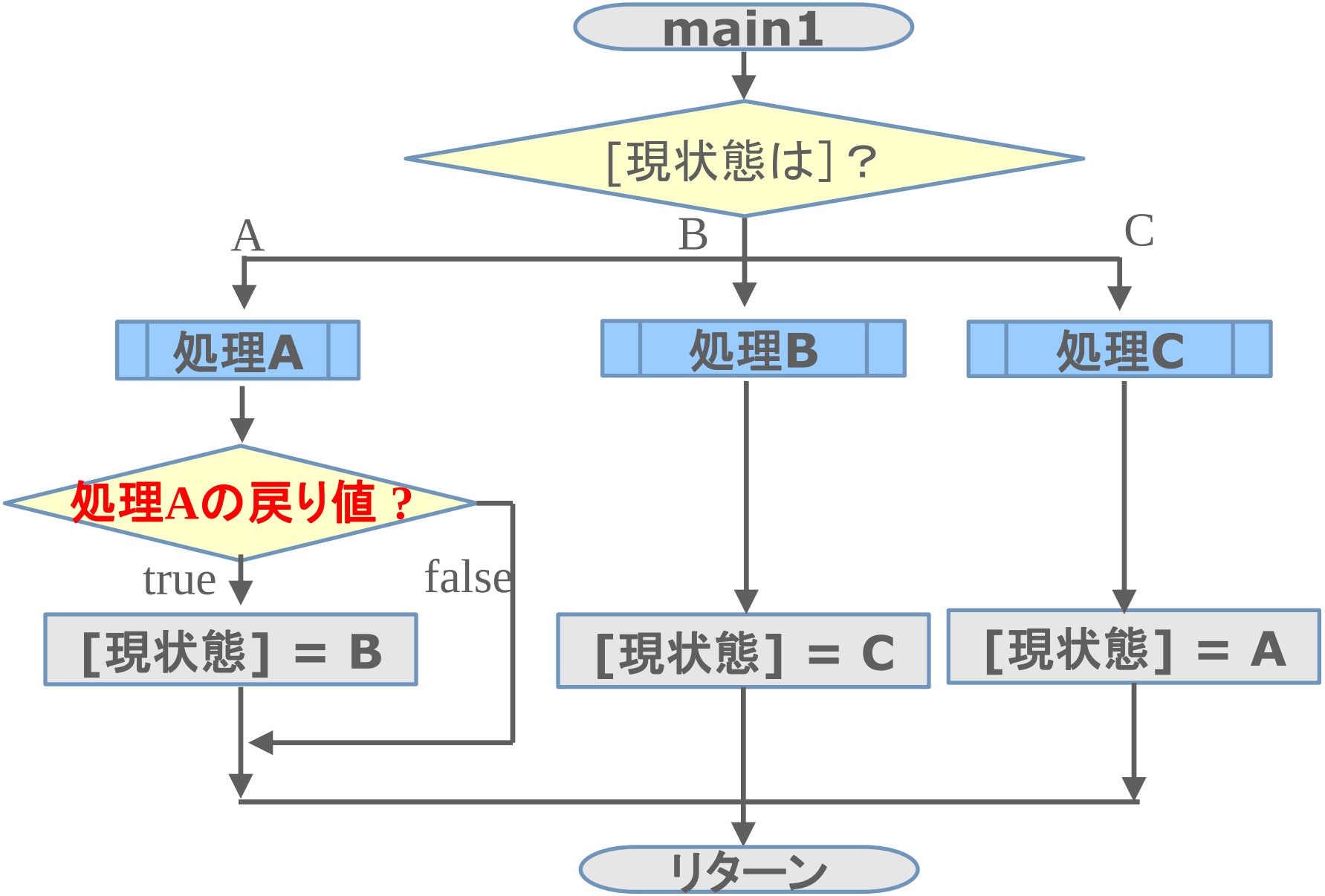
- 1つの処理を終えた後に一旦、戻る。
- 現状態をstatic変数で記憶し、次回は続きから実行する。

「処理A」のステートマシン化

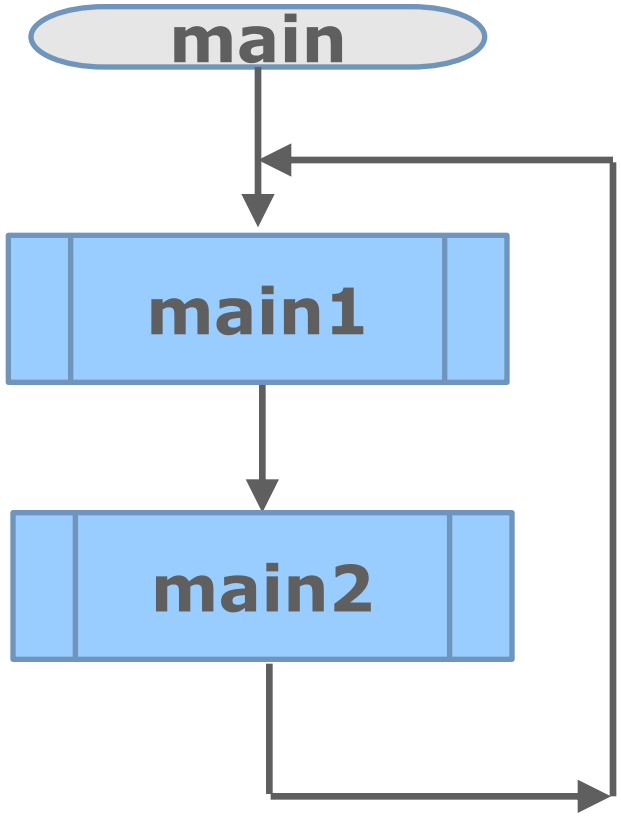


「[現状態]が1に戻ったならtrue、途中ならfalse

「main1」のステートマシン化



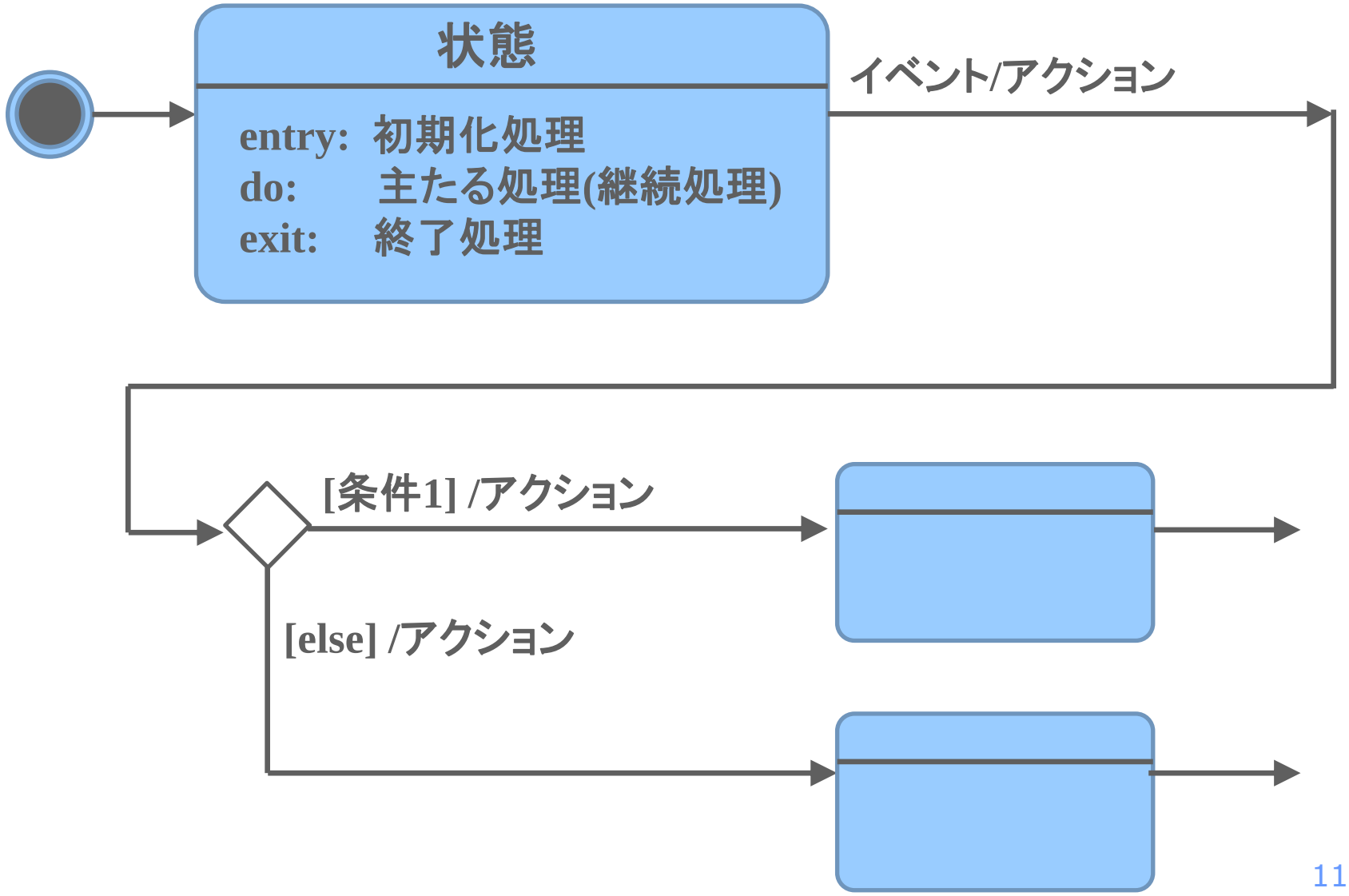
ステートマシン関数の呼び出し例



[解説]

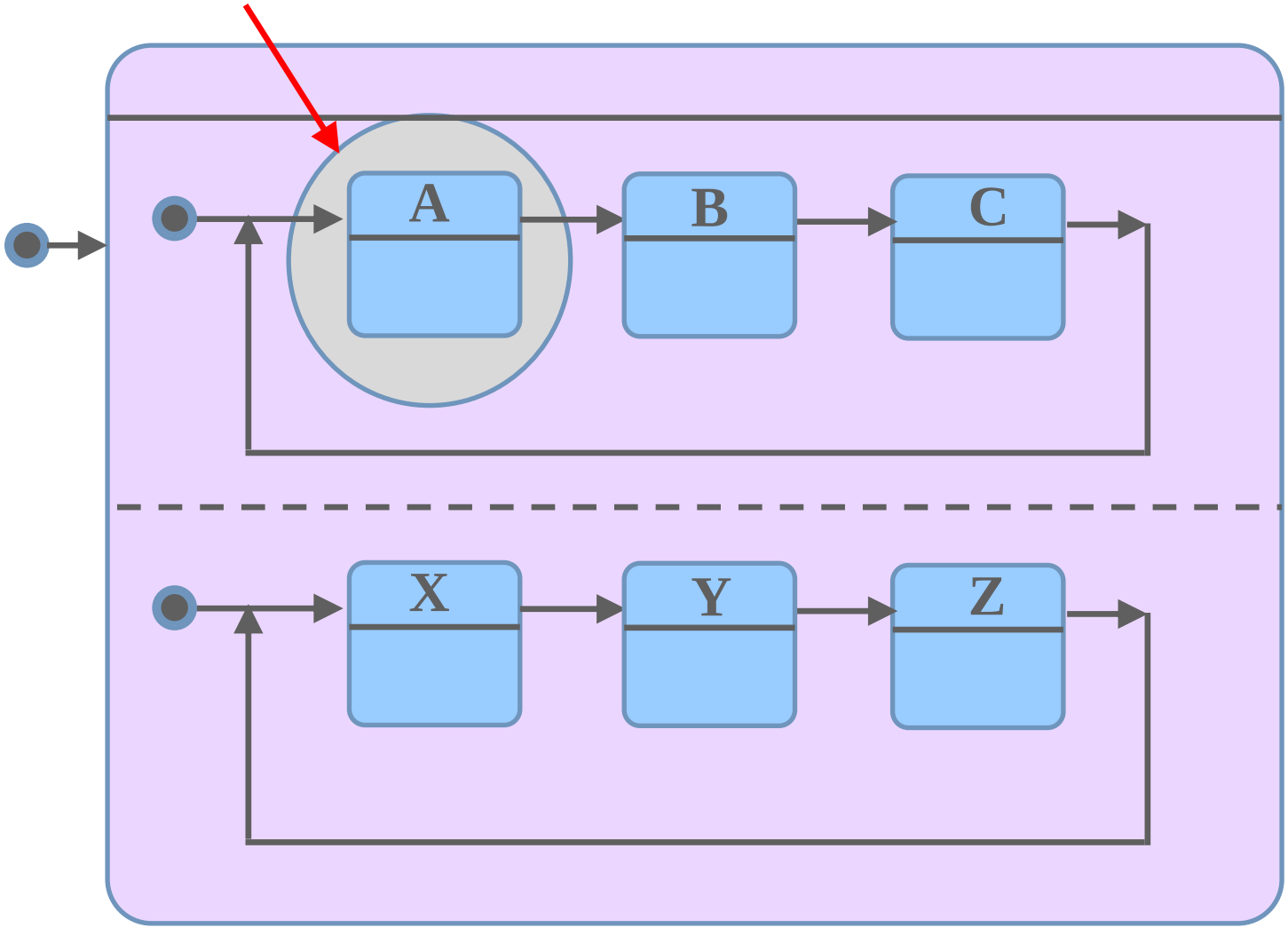
- 繰り返し呼び出すだけ。

状態遷移仕様の記述方法 「ステートチャート」



状態A~B、状態X～Z、状態1～3の記述例

処理1~3は、この中、もしくは別のステートチャートに書く



条件分岐ありのステートチャート記述例