アルゴリズムの表現

アルゴリズムとは

- 問題を解く手順
- 手順の例：カレーのレシピ
    i. 次のものを準備します
        - じゃがいも（XX切り，100g），牛肉，たまねぎ，にんじん，etc.
    ii. 油を引いて熱したフライパンで牛肉を炒めます
    iii. 牛肉がXXとなってきたら，じゃがいも，にんじん・・・

- 「手順」に登場するもの**操作**と**状態**
  - 操作：対象物に対する操作．～を準備する，～を炒める
  - 状態：対象の状態（操作によって変化する）．じゃがいもXX切り100g，牛肉がXXとなる
- 「手順」の書き方
  - 順次：**操作**の順番（～を準備する．次に～を炒める）
  - 選択：**状態**によって**操作**を選択する
  - 反復：**状態**になるまで**操作**を繰り返す

コンピュータ上での手順

- 対象物：データ（変数）
- 状態：データそのもの
- 操作：データを更新

アルゴリスムの表現方法

- 途中のデータ（状態）を明示しない表現
  - プログラム（記述的な表現）
  - フローチャート（視覚的な表現）
- 途中のデータ（状態）を明示する表現
  - 変数トレース（プログラム）
  - モノによる表現

「鶴と亀が合わせて6匹います．足の合計が20本でした．亀は何匹いますか？」（鳥は1羽2羽と数えますがここでは匹としておきます）

➡️ 答え: 亀4匹

```
a = [2, 2, 2, 2, 2, 2]
while True:
        s = 0
        for i in range(len(a)):
                s += a[i]
        if s == 20:
                break
        for i in range(len(a)):
                if a[i] == 2:
                        a[i] += 2
                        break
b = 0
for i in range(len(a)):
        if a[i] == 4:
                b += 1

print(b)
```

わかりにくさの原因

- **while-for-if-break の入れ子**
- **各処理が何を意図しているかわからない**
- 変数・数字が何を表しているかわからない
- データ変化がわからない

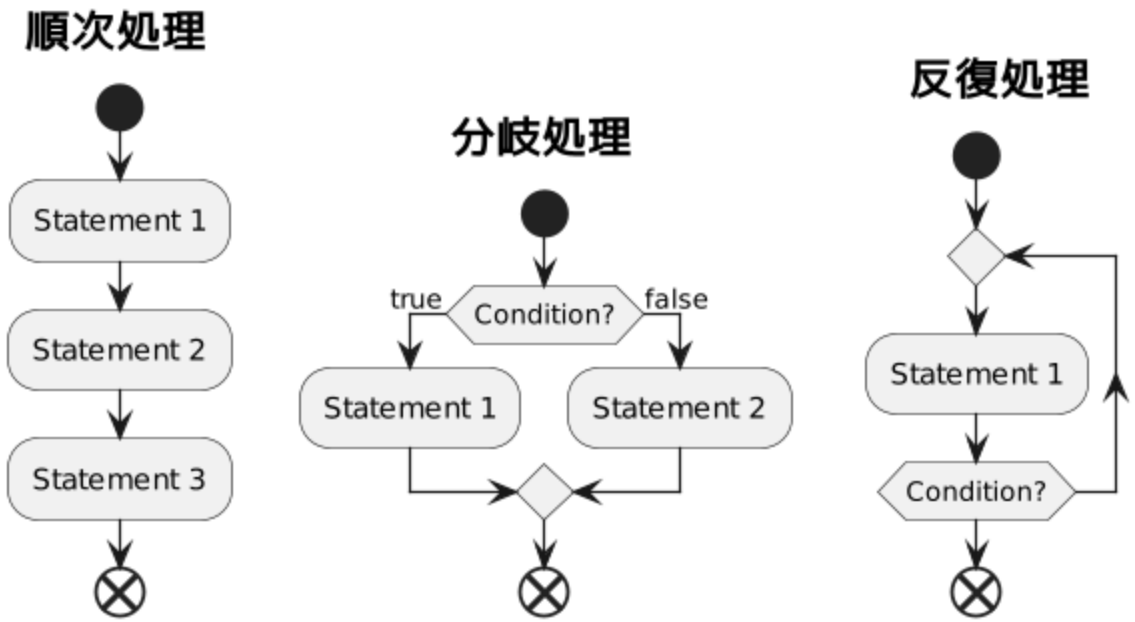
構造化

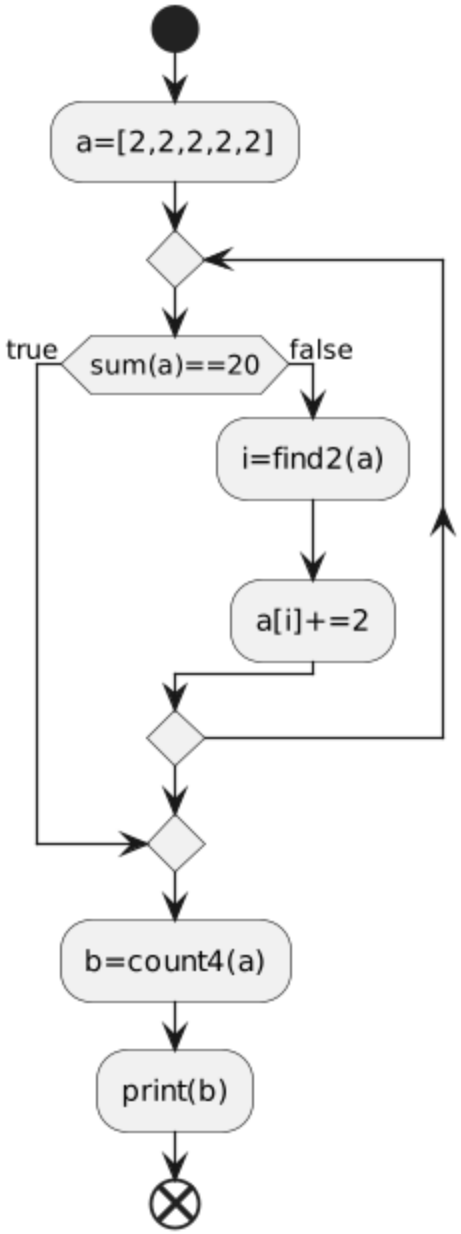
- 「手順」の書き方（再掲）
  - 順次：**操作**の順番
  - 選択：**状態**によって**操作**を選択する
  - 反復：**状態**になるまで**操作**を繰り返す
- 構造化
  - ひとかたまりの「順次・選択・反復」と「処理内容」を対応させる
  - モジュール化
  - 関数化

```
def sum(a):
        s = 0
        for i in range(len(a)):
                s += a[i]
        return s

def find2(a):
        for i in range(len(a)):
                if a[i] == 2:
                        return i

def count4(a):
        b = 0
        for i in range(len(a)):
                if a[i] == 4:
                        b += 1
        return b

## main

a = [2, 2, 2, 2, 2, 2]
while True:
        if sum(a) == 20:
                break
        i = find2(a)
        a[i] += 2
b = count4(a)
print(b)
```

フローチャート

「手順」を図にして読みやすくする ➡️ フローチャート




a=[2,2,2,2,2]
true
sum(a)==20
false
i=find2(a)
a[i]+=2
b=count4(a)
print(b)

わかりにくさの原因

- while-for-if-break の入れ子
- **各処理が何を意図しているかわからない**
- **変数・数字が何を表しているかわからない**
- データ変化がわからない

```
def total_legs(legs):
        total = 0
        for i in range(len(legs)):
                total += legs[i]
        return total

def find_index_2legs(legs):
        for i in range(len(legs)):
                if legs[i] == 2:
                        return i

def count_4legs(legs):
        tortoise = 0
        for i in range(len(legs)):
                if legs[i] == 4:
                        tortoise += 1
        return tortoise

## main

legs = [2, 2, 2, 2, 2, 2]
while True:
        if total_legs(legs) == 20:
                break
        i = find_index_2legs(legs)
        legs[i] += 2
tortoise = count_4legs(legs)
print(tortoise)
```

わかりにくさの原因

- while-for-if-break の入れ子
- **各処理が何を意図しているかわからない**
- **変数・数字が何を表しているかわからない**
- データ変化がわからない

対象の変更

- 対象物：データ（変数） ➡️ モノ
- 状態：データそのもの ➡️ モノの状態
- 操作：データを更新 ➡️ モノの状態変化

プログラムから離れる

1. 丸を6個を書く
2. それぞれの丸に2本ずつ線をつける
3. 線が合計20本になるまで次の手続きをする
   3-1. 2本しか線がない丸を見つけて2本付け加える
4. 4本線がついている丸を数える

わかりにくさの原因

- while-for-if-break の入れ子
- 各処理が何を意図しているかわからない
- 変数・数字が何を表しているかわからない
- **データ変化がわからない**