第 1 回 擬似コードゼミ

ゼミ担当者　：　中村彰之, 芝野功一郎
開催日　：　2009 年 6 月 25 日

ゼミ内容: 本ゼミでは，アルゴリズムの構造を記述するために用いられる擬似コードについて述べる．
また，擬似コードを記述する際に必要となる主な文法を紹介する．

1　はじめに

擬似コードとは，アルゴリズムの構造を記述する際に用いられる擬似的な言語のことである．擬似コードは自然言語に近い形式で記述され，特定のプログラミング言語について知識のない人にも理解できるものでなければならない．特定のプログラミング言語の文法にとらわれない形式で記述されることにより，プログラミングにおけるアルゴリズムの設計と実際のコーディングを分離することが可能になる．開発者は，プログラミング言語の文法にとらわれることなく，アルゴリズムに集中することができ，擬似コードが完成した後にはそれをプログラミング言語へと書き直すことによってコーディングを行う．

2　擬似コードの用途

擬似コードの主な用途は，コーディング前にアルゴリズムの設計を行うことである．これにより，開発の効率化をはかることができる．擬似コードの利用例の一つとして，下記に示した SA における初期点の生成処理のように，プログラムの上部に処理内容を示すことが考えられる．

例

```
#FOR   設計変数の次元数
#     乱数を制約条件内で発生させる
#     設計変数に生成された乱数を追加
#ENDFOR

for item in range(dimension):
    random = random.uniform(Min, Max)
    designValue.append(random)
```

例に示したように，擬似コードによってプログラムがどのような処理を行うかを記述することにより，どのような処理が行われるのかがわかりやすくなる．また，擬似コードで書かれた 1 行 1 行の下に，実際のコードを記述することによって，擬似コードをコメントのように扱うこともできる．擬似コードをコメントとして利用した例を下記に示す．

例

```
#FOR   設計変数の次元数
for item in range(dimension):
    #乱数を制約条件内で発生させる
    random = random.uniform(Min, Max)
    #設計変数に生成された乱数を追加
    designValue.append(random)
#ENDFOR
```

このように，プログラミングを行う際にまず擬似コードを書くことによって，第三者にもわかりやすいコードを書くことが可能となる．

3　擬似コードの書き方

擬似コードの書き方には，標準化された形式は存在しない．そのため，本章で説明する書き方は本研究室内で定めたものである，以降，擬似コードの書き方について，条件分岐，繰り返し，演算子，サブルーチンなどに分けて説明する．

3.1　演算子

算術演算子，比較演算子，論理演算子に関しては，以下のものが使用できるため，論理関係なども文章で記述しなくてもよい．

- 算術演算子： `+`, `-`, `*`, `/`, `%`, `()`, `=`
- 比較演算子： `>`, `<`, `<=`, `>=`,`=`
- 論理演算子： `AND`, `OR`, `NOT`

3.2　コマンドライン入力・出力

コマンドラインからの変数への値の代入，また値の出力は次のように記載する．

- コマンドライン入力

書き方

```
Input 変数 1, 変数 2, ...
```

例

```
Input 名前，年齢，...
---
実際の入力：西岡 23
```

- コマンドライン出力

書き方

```
Display "テキスト 1"，変数 1，"テキスト 2"，変数 2，...
```

例

```
Display 名前，"さんは現在"，年齢，"歳です"
---
実際の出力：西岡さんは現在 23 歳です
```

3.3 条件分岐

擬似コードで用いられる条件分岐には IF-THEN-ELSE 文と CASE 文がある．下記にそれぞれの記述方法を述べる．

- IF-THEN-ELSE 文
  IF-THEN-ELSE 文では，条件を指定することによってその真偽を判定し，処理を分岐させる．IF-THEN-ELSE 文で用いられるキーワードは IF，THEN，ELSE，ENDIF の 4 つである．記述方法と IF-THEN-ELSE 文を用いた擬似コードの例を下記に示す．なお，ELSE 文は省略しても良い．

書き方

```
IF　条件　THEN
    処理 1
ELSE IF　条件　THEN
    処理 2
ELSE
    処理 3
ENDIF
```

例

```
BMI 指数を体重（kg）/ 身長の 2 乗（m）として求める
IF　BMI 指数が 18.5 未満　THEN
    ”痩せ型 ”と表示
ELSE IF　BMI 指数が 18.5〜25 未満　THEN
    ”標準 ”と表示
ELSE IF　BMI 指数が 25〜30 未満　THEN
    ”肥満 ”と表示
ELSE
    ”高度肥満 ”と表示
ENDIF
```

- CASE 文
  CASE 文では与えられた値に対してそれぞれ分岐した処理を行う．CASE 文で用いられるキーワードは CASE，OF，OTHERS，ENDCASE の 4 つである．記述方法と CASE 文を用いた擬似コードの例を下記に示す．デフォルトの処理は OTHERS に記述されるが，これは省略しても良い．

書き方

```
CASE　式　OF
    値 1：処理 1
    値 2：処理 2
    値 3：処理 3
    値 4：処理 4
    OTHERS：
    デフォルトの処理
ENDCASE
```

例

```
CASE　役職　OF
    社長：Display "年収 1 億円"
    専務：Display "年収 5 千万円"
    部長：Display "年収 2 千万円"
    課長：Display "年収 1 千万円"
    OTHERS：
    Display "年収 500 万円"
ENDCASE
```

3.4 繰り返し

繰り返しは FOR 文，WHILE 文，および REPEAT 文の 3 種類である．

- FOR 文
  FOR 文では，決まった回数の繰り返しを 2 通りの方法で記述できる．1 つはある整数 1 からある整数 2 までの範囲を繰り返しの範囲として指定する方

法，もう一方は配列などのリストの値を順に取り出して，リストに格納している値の数だけ繰り返す方法である．

書き方

```
FOR カウンタ = 整数 1 to 整数 2 DO
   処理
ENDFOR

FOR 値 in リスト DO
   処理
ENDFOR
```

例

```
FOR カウンタ = 1 to 10  DO
   合計 = 合計 + カウンタ
ENDFOR

FOR 従業員 in 従業員リスト DO
   Display 従業員の名前，"さん，お疲れ様です"
ENDFOR
```

- WHILE 文
  WHILE 文では，条件式が真である間，処理が繰り返される．最初に実行した際に条件式が真でない場合，一度も処理が実行されない．

書き方

```
WHILE 条件式 DO
   処理
ENDWHILE
```

例

```
WHILE カウンタ <= 10 DO
  Display カウンタ
  カウンタ = カウンタ + 1
ENDWHILE
```

- REPEAT 文
  REPEAT 文では，条件式が偽である間，処理が繰り返される．REPEAT 文は WHILE 文とは異なり，必ず一度は実行される．

書き方

```
REPEAT
   処理
UNTIL 条件式
```

例

```
REPEAT
   ビラの印刷
   印刷枚数 = 印刷枚数 +1
UNTIL 印刷枚数 > 1000
```

3.5 サブルーチン

サブルーチンは返り値がない場合 (手続き)，ある場合 (関数) の 2 つに大別できる．また，引数は必要なければ省略してよいが，() は省略できない．

書き方

```
返り値がない場合（手続き）:
サブルーチン名 (引数 1，引数 2，...)
   処理
EXIT

返り値がある場合（関数）:
サブルーチン名 (引数 1，引数 2，...)
   処理
RETURN with 返り値
EXIT
```

例

```
返り値がない場合（手続き）:
あいさつ (名前，天気)
   Display 名前，"さん，こんにちは．"
   Display "今日は"，天気，"ですね"
EXIT

返り値がある場合（関数）:
ラジアン変換 (角度)
   変換した角度 = 角度 * π / 180
RETURN with 変換した角度
```

3.6 プログラム (実行文)

各プログラムの実行部は次のように記述する．また実行部の中でサブルーチンの呼び出しも記述可能である．

書き方

```
プログラム名 ()
    処理
    サブルーチン (引数 1，引数 2，...)
STOP
```

例

```
degreeToRadian()
    角度 = 90
    ラジアン = ラジアン変換(角度)
    Display  ラジアン
STOP
```

4 注意事項

- プログラミング言語に依存しない書き方をしなければならない.

悪い例

```
a = int(Math.random() * 6 + 1)
System.out.println ("サ イ コ ロ の
目：" + a)
```

良い例

```
a = 1 以上 7 未満の整数をランダムに発生
Display  "サイコロの目：", b
```

- 処理の記述は, 複数の処理が必要なものを一文に纏めず, 一つ一つの処理を分けて記述しなければならない.

悪い例

```
prime-judgement()
    Input   num
    IF   judgement(num) = 1   THEN
        Display "素数です"
     ELSE IF    judgement(num) = 0
THEN
        Display "素数ではありません"
    ENDIF
STOP

judgement(num)
    IF   num が素数   THEN
        RETURN with 1
    ELSE IF   num が素数でない   THEN
        RETURN with 0
    ENDIF
```

良い例

```
prime-judgement()
    Input   num
    IF   judgement(num) = 1   THEN
        Display "素数です"
     ELSE IF    judgement(num) = 0
THEN
        Display "素数ではありません"
    ENDIF
STOP

judgement(num)
    IF   num < 2   THEN
        RETURN with 0
    ELSE IF   num = 2   THEN
        RETURN with 1
    ELSE IF   num % 2 = 0   THEN
        RETURN with 0
    ENDIF
    a = 3
    WHILE   a の 2 乗 <= num   DO
        IF   num % a = 0
            RETURN with 0
        ENDIF
        a = a + 2
    ENDWHILE
RETURN with 1
```

5 練習問題

以下に示すようなピラミッド図形を作成するためのアルゴリズムを擬似コードで記述し, その擬似コードに基づいてプログラムを作成せよ.

ピラミッド図形

```
      A
     AAA
    AAAAA
   AAAAAAA
```

参考文献

1) PSEUDOCODE STANDARD, http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html

2) Pseudo Code Guide, http://ironbark.bendigo.latrobe.edu.au/subjects/PE/2005s1/other_resources/pseudocode_guide.html