計算の方法

山本昌志*

2007 年 12 月 14 日

概 要

計算の方法について説明する．

1　本日の学習内容

モデル化したデータを処理する方法—計算—を学ぶ．計算の記述方法から，計算方法，そしてコンピューターでの実行方法が概略を学習する．本日のゴールは，以下の通りである．

- 計算の代表的な処理とその記述方法が分かる．
- 代表的な計算モデルが分かる．特に，手続き型と関数型の違いが分かる．
- 様々なプログラムの実行方法が分かる．

教科書 [1] の pp.97–120 が本日の範囲である．

2　計算とその記述方法

モデル化されたデータに対する操作 (処理) のことを計算という．この計算を行う場合，必要なことを考えよう．

2.1　計算の方法

教科書では，数を数える方法について，2 通りの方法を示している．**取り出し型**と**分割型**である．これらの 2 つの方法が計算方法の全てというわけではない—ことに注意が必要である．

ここでの問題は，袋があってその中の玉の数を調べることである．さあ，諸君ならどうやって数えるか? いろいろな方法を想像せよ．

*独立行政法人　秋田工業高等専門学校　電気情報工学科

取り出し型　袋から玉を一つずつ取り出し，数える．袋に玉が無くなったら，数えるのを止める．これを，もう少し情報の講義らしく記述すると，教科書のように，次のようになる．

- <玉の数>をゼロにする．
- 袋に玉が無くなるまで，以下の処理を繰り返す．
  - 玉を取り出す
  - <玉の数>を 1 増やす

分割型　袋の中に玉が大量にある場合を考える．このようなとき，何人かで手分けして数えることがあるだろう．このように自分では手に負えない仕事を，他の人—教科書では下請け—に依頼し，各々の合計を答えとする方法を，分割型と言う．下請けは，さらに下請けに仕事を依頼することができる．究極には，末端の下請けは一つの玉を受け取りそれを数え，一つ上の元請けに「1 個です」と報告する．次のようにする．

- 袋がからならば<玉の数>をゼロ，袋のなかに玉が一つならば<玉の数>は 1
- そうでなければ
  - 袋を分割し，玉を数える仕事を下請けに
  - <玉の数>は，下請けが数えた和

2.2　計算の記述

計算を行うためには，次のようなことが必要である．ほとんどすべてのプログラミング言語では，これらの機能が用意されている．

変数 計算につかうデータを記憶するために，変数を使う．変数には変数名をつけて，それぞれの値を区別する.「代入により」と呼ばれる操作により，変数に関連づけられている値を変えることができる．

条件判断 計算の状態により，処理の順序を変更する．

繰り返し 同じような計算を，繰り返し行う．ある条件に達したら，繰り返しの処理をやめる．

添え字付き変数 プログラミング言語では，配列 (array) として実装されていることが多い．データの記憶領域は，変数名と整数でアクセスできる．

2.3　計算と意味

データ型　意味が付属していないデータに意味を与える．

- 自然科学の分野では，単位によりデータ (数値) に意味を与える．
- プログラミング言語では，コンピューター内のデータの取り扱い方法を直接指定するために，整数型や実数型，文字型などがある．

計算の意味

- あいまいに表現してはならない．

再帰と意味

- ある計算を定義するときに，それ自身を使うことがある．このような定義を再帰的 (recusive) と言う．丁度，数学の漸化式に似ている．

3　計算の表現と進行

ここでは，二つの整数の割り算を行い「商」と「あまり」を求める方法を 3 通りの方法で説明している．これはプログラムの方法に密接に関わる．

3.1　手続き型

手続き型 (procedual) の計算では，数多くの要素的計算を逐次的に処理する．たとえば，a を b で割った商とあまりは，次のように処理する．

- a から b を引けるだけ引いて，引いた回数を商，残りをあまりとする．

これを C 言語で実装すると，リスト 1 のようになる．

リスト 1: 44 を 13 で割った商とあまりを計算する手続き型のプログラム例．教科書 p.106 の手続きを C 言語で記述．

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    int a, b;
6    int r, q;
7
8    a=44;       // a <- 44
9    b=13;       // b <- 13
10
11   r=a;        // r <- a
12   q=0;        // q <- 0
13
14   while(r >= b){
15     r=r-b;
16     q=q+1;
17   }
18
19   printf("quotient=%d\tremainder=%d\n", q, r);
20
21   return 0;
22 }
```

実行結果

```
quotient=3      remainder=5
```

3.2 関数型

ある関数が別の関数を呼び出して計算の一部を実行させることにより処理を行う．その処理は，教科書の p.107 の下の方の通り．C 言語のプログラム例をリスト 2 に示す.

リスト 2: 44 を 13 で割った商とあまりを計算する手続き型のプログラム例．教科書 p.107 の手続きを C 言語で記述.

```
1  #include <stdio.h>
2
3  int q(int a, int b);          // プロトタイプ宣言
4  int r(int a, int b);
5
6  //=========================================================================
7  // メイン関数
8  //=========================================================================
9  int main(void)
10 {
11   int a, b;
12   int quot, remi;
13
14   a=44;       // a <- 44
15   b=13;       // b <- 13
16
17   quot=q(a,b);
18   remi=r(a,b);
19
20   printf("quotient=%d\tremainder=%d\n", quot, remi);
21
22   return 0;
23 }
24
25
26 //=========================================================================
27 // 商を計算する関数
28 //=========================================================================
29 int q(int a, int b)
30 {
31
32   if(a<b){
33     return 0;
34   }else{
35     return q(a-b, b)+1;
36   }
37
38 }
39
40 //=========================================================================
41 // あまりを計算する関数
42 //=========================================================================
43 int r(int a, int b)
44 {
45
```

```
46    if(a<b){
47      return a;
48    }else{
49      return r(a-b, b);
50    }
51
52  }
```

実行結果

```
quotient=3      remainder=5
```

3.3 宣言型

宣言型では関数型と同じように多くの計算要素によって計算が行われる．計算要素は，条件，性質，事実を記述した形で示す．たとえば，a を b で割ったあまりは，教科書の例のように示すことができる．

C 言語では宣言型でプログラムすることはできない．教科書の例は，Prolog の記述と思われる．

3.4 計算モデルとその間の関係

手続き型と非手続き型

- 関数型と宣言型をまとめて非手続き型 (non-procedual) と呼ぶことがある．
- それぞれの型には計算処理の得手不得手がある．
- 通常のプログラミングでは，手続き型と非手続き型をあわせて使うことが多い．

他の計算モデル

- 書き換え型やネットワーク型，データフロー型などがある．
- 計算の性質を考える多恵の理論的なモデルがある．再帰的関数やチューリング機械，ラムダ計算などである．

計算モデルのあいだの関係　計算記述の方法は，表現できる内容という意味で等価である．こことを最大値を探索する計算でしめす．

最大値を探索する問題は，関数型では教科書 p.111 の下の方のように記述できる．それを C 言語で表現すると，リスト 3 のようになる．

リスト 3: 最大値を探索する関数型のプログラム例．

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  #define N 30
6
```

```
7  int max(int p, int a[]);
8  void set_rand(int a[]);
9  //============================================================
10 // main関数
11 //============================================================
12 int main(void)
13 {
14   int a[N+1], max_int;
15
16   set_rand(a);                        // a[]に乱数を設定
17
18   max_int = max(N,a);                 // 1～Nの最大値探索
19   printf("max=%d\n", max_int);        // 最大値を表示
20
21   return 0;
22 }
23
24 //============================================================
25 // 最大値を探す再帰関数 教科書 p.105 p.111
26 //============================================================
27 int max(int p, int a[])
28 {
29
30   if(p==1){
31     return a[1];
32   }else{
33     if(a[p]>max(p-1, a)){
34       return a[p];
35     }else{
36       return  max(p-1, a);
37     }
38   }
39
40 }
41
42 //============================================================
43 // 配列 a[i] に乱数を設定
44 //============================================================
45 void set_rand(int a[])
46 {
47   int i;
48
49   srand((unsigned int)time(NULL));       // 乱数の初期値設定
50
51   for(i=1; i<N+1; i++){
52     a[i]=rand();                        // 配列に乱数を代入
53   }
54
55 }
```

実行結果

```
max=2100562050
```

この手続きを分解すると図 1 のようになる．関数型の処理の順序通りに，手続き型で表すとリスト 4 のように記述できる．両者は全く同じ計算を行っている．

全く同じ計算を行っているが，計算速度が大きく異なる場合がある．この場合だと，手続き型のプログラ

ムの方が圧倒的に早い．このように，計算に依存して，得手不得手がある．

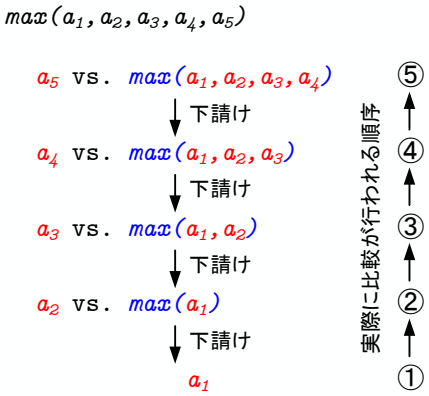

図 1: 教科書 p.111 の最大値の問題を解く関数型の実行順序.

リスト 4: 最大値を探索する手続き型のプログラム例..

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  #define N 30
6
7  void set_rand(int a[]);
8  //=============================================================
9  // main関数
10 //=============================================================
11 int main(void)
12 {
13   int i, a[N+1], max_int;
14
15   set_rand(a);                        // a[]に乱数を設定
16
17   max_int = a[1];
18   for(i=2; i<N+1; i++){
19     if(max_int < a[i]) max_int = a[i];
20   }
21
22   printf("max=%d\n", max_int);       // 最大値を表示
23
24   return 0;
25 }
26
27 //=============================================================
28 // 配列 a[i] に乱数を設定
29 //=============================================================
30 void set_rand(int a[])
31 {
32   int i;
33
34   srand((unsigned int)time(NULL));        // 乱数の初期値設定
35
```

```
36    for(i=1; i<N+1; i++){
37      a[i]=rand();                          // 配列に乱数を代入
38    }
39
40  }
```

実行結果

```
max=2100562050
```

4 プログラムとプログラム言語

プログラムは，最終的には'0' と'1' のビット列である機械語に変換されて実行される．しかし，どれも同じような方法で．機械語に変換される訳ではない．

アセンブリ言語 機械語のビットパターンを人間に分かりやすいように記述した言語である．人間が書いたプログラムは，機械語と 1 対 1 の関係があり，簡単に機械語に直せる．これは，アセンブラーというプログラムで機械語に直す．諸君が学習した CASL II はアセンブリ言語の一つである．

高級言語 人間に分かりやすいプログラミング言語である．マシン語と 1 対 1 の関係はなく，コンパイラーが記述内容に従いマシン語に翻訳する．同じ言語で書いた同一のプログラムでも，コンパイラーが異なれば，異なるマシン語になる．諸君が学習した C 言語や Fortran は高級言語である．

インタプリタ 高級言語のコンパイラーは実行に先立って，すべて機械語に変換する．変換した機械語を一気に実行するために，計算速度は速い．それに対して，インタプリタは，実行時にプログラムの各行ごと，機械語に変換して実行を行う．私は，インタプリタであるプログラミング言語 Perl や PHP をよく使う．

中間言語 高級言語とインタプリタの中間的な方法で機械語に変換する．プログラムは実行に先立って，機械語に近い中間言語に変換する．CPU が異なってもこの中間言語は，同一とする．このようにすることにより，インタプリタより高速に動作させることができる．また，高級言語のように CPU 毎に機械語を用意する必要もない．プログラミング言語 Java がこの方式で実行されることが多い．

5 課題

5.1 課題内容

以下の課題を実施し，レポートとして提出すること．

[問 1] (復予) 教科書 [1]pp.98–152 を読み，重要な部分には赤線でアンダーラインを入れよ．レポートには「ちゃんと読んで，アンダーラインを引きました.」と書け．

[問 2] (復) 教科書の章末問題の [5.3]

[問 3] (復) 教科書の章末問題の [5.4]

[問 4] (復) 教科書の章末問題の [5.5]

5.2 レポート提出要領

期限　12 月 21 日 (金) AM 8:45
用紙　A4 のレポート用紙．左上をホッチキスで綴じて，提出のこと．
提出場所　山本研究室の入口のポスト
表紙　表紙を 1 枚つけて，以下の項目を分かりやすく記述すること．
　授業科目名「情報理論」
　課題名「課題　計算の方法」
　提出日
　5E　学籍番号　氏名
内容　2 ページ以降に問いに対する答えを分かりやすく記述すること．

参考文献

[1] 河合慧（編）. 情報. 東京大学出版会, 2006.