構造体

山本昌志*

2007 年 4 月 10 日

概 要

構造体を使う理由と文法を学ぶ.

1　本日の学習内容

本日は**情報処理応用**の最初の授業であるため，はじめに授業の概要を説明する．シラバスを使って，注意点および学習内容の重要な点についての述べる．

授業の概要の説明の後，本日のテーマである**構造体**について説明する．そして練習問題を通して，構造体を理解する．本日の学習の範囲は，教科書 [1] の p.294–303 である．以下，学習のゴールを示す.

- データ構造分かり，構造体が便利なことが理解できる.
- 構造体の書き方が分かる.

2　アルゴリズムとデータ構造

現在，諸君は C 言語の文法の勉強をしている．実際プログラムの勉強は，文法の勉強だけでは全く不十分である．そのため，この講義では文法の学習が済むと,「アルゴリズムとデータ構造」の学習を進める．アルゴリズムとは問題を解く手順のことである．コンピュータープログラムでは，入力データから目的のデータを作成する手順を言う．プログラマーはアルゴリズムを考え，それをプログラミング言語で表現しなくてはならない．アルゴリズムが大事であることは，万人が認めるところである．これは，コンピュータープログラムのみならず，数学や電気の問題を解く場合も同じである．科学の問題を解く場合は手順が重要で，それが分かれば，問題が解けたのも同様である.

一方，データ構造となると，少し様子が異なってくる．数学や電気の問題のデータ構造となると，想像がつかない．科学の問題では，データ構造は重要視されないので，それも仕方ないことである．データ構造を気にするのは，なんと言ってもビジネスの分野である．たとえば，私の場合，成績処理のデータ構造を考えなくてはならない．それには，年度と学年，学籍番号，学生氏名，教科名，試験名，点数，レポート提出状況，出欠などのデータを分かり易くまとめなくてはならない．実際には図 1 に示す Excel の表を使っている．これがデータ構造である．この例のように，データのまとめ方をデータ構造という.

*独立行政法人　秋田工業高等専門学校　電気情報工学科

氏名	中間	学年末	レポート	後期成績	授業時間	公欠	欠課	出停	忌引き	備考
？？？？	86	53	18	74	30	0	0	0	0	
？？？？	70	68	15	70	30	0	0	0	0	
？？？？	69	65	20	74	30	0	6	2	0	
？？？？	68	27	13	51	30	0	2	0	0	
？？？？	25	63	20	55	30	0	0	0	0	
？？？？	90	93	11	84	30	0	0	0	0	
？？？？	80	23	18	59	30	0	0	0	0	
？？？？	75	65	17	73	30	0	0	0	0	
？？？？	78	76	13	75	30	0	2	0	0	
？？？？	63	63	15	65	30	0	0	0	0	
？？？？	78	89	18	85	30	0	0	0	0	
？？？？	69	49	19	66	30	0	2	0	0	
？？？？	78	78	20	82	30	0	0	0	0	
？？？？	78	42	14	62	30	0	2	0	0	
？？？？	42	85	8	59	30	0	0	0	0	

図 1: 成績の Excel ファイル.

プログラムを作成する場合，そのアルゴリズムとデータ構造を考えなくてはならない．プログラムの性能は，アルゴリズムの善し悪しでが決まる—ことはすぐに理解できる．しかし，データ構造については，無頓着になってしまいがちである．先の成績処理のように単純な問題であれば，誰でも同じようなデータ構造を考え，大した問題は生じない．しかし，複雑なデータの場合には，いろいろなデータ構造が考えられる．大企業の顧客データなどがそれに当たる．住所や氏名，年齢，何時，どこで，購入した商品のデータがある．これらに加えて，アンケートに答えていれば，それこそ数十の項目で，数百万人分のデータがあることも容易に想像できる．このデータをまとめ方，すなわち構造は重要で，その後の処理の方法にも大きく影響する．データ構造に従ったイメージがプログラマーの頭の中にインプットされる．そのため，アルゴリズムも大体決まってしまうのである．データ構造が処理のアルゴリズムを決めてしまうことがあるのだ!

データ構造が悪いと，効率の悪いプログラムになってしまう．たとえば，ファイルに 100 個の整数が書かれており，その合計を求めるプログラムでは配列を使うべきであろう．変数を使ったプログラムでは，効率が悪くなるのはすぐに分かるであろう．

3　これまで学習したデータ構造

データ構造には，表 1 に示すようなものがある．このうち基本データ型と配列型は既に学習したはずだ! 本日はレコード型—C 言語では構造体で表現—を学習する．構造体を学習する前に，データ構造というものを頭に入れるために，既に学習した単純型と配列型の復習をしておく．さらに，構造体の役割を簡単に述べる．

表 1: データ構造の種類

データ構造	基本データ構造	基本データ型	単純型	整数型
				実数型
				文字型
				論理型
				数え上げ型
			ポインタ型	
		構造型	配列型	
			レコード型	
		抽象データ型		
	問題向きデータ構造	線形リスト	単純リスト	
			双リスト	
			環状リスト	
		木	二分木	完全二分木
				二分探索木
				バランス木
			多分木	
			バランス木	AVL 木
				B 木
		スタック		
		キュー		

3.1 単純型変数

単純型の変数は，次のように変数に一つの数値[1]しか代入できないものを言う．

```
char c, h, moji;
int i, j, seisu;
double x, y, jisu;
```

通常，これを変数と言う．変数というと，この単純型を示す場合が多いが，配列や構造体を含める場合もあるから，文脈から適当に判断しなくてはならない．

これのイメージは，図 2 に示しているとおりで，変数とは数値を入れる箱のようなものである．整数型と倍精度実数型の変数は，数学の変数と全く同じである．

図を見て分かるように，箱の大きさが型によって異なる．これは，一つのデータを表現するために必要な情報量が異なるためである．情報量の単位は，ビット (bit) が使われる．2 進数の 1 桁を 1 ビットと言う．8 ビットで 1 バイトとなり，それがコンピューターで使われる基本単位となる．

[1] 一つの文字のみ代入可能なものは文字型の変数である．文字も整数として扱えるので，この範疇と考える．

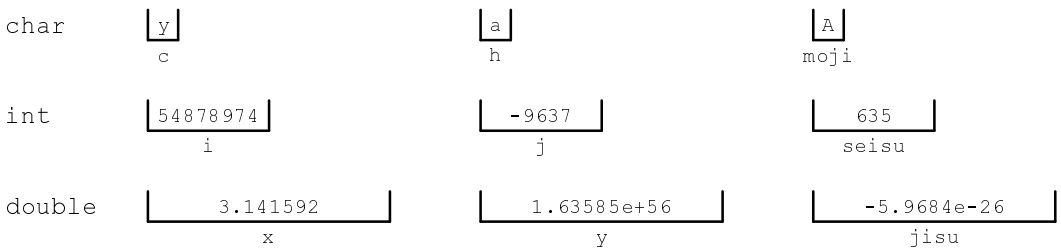

図 2: 変数のイメージ．変数とはデータを入れる箱のようなもの．

同じ int 型でもいろいろあり，表現できる範囲が異なっている．これは一つの変数の情報量の差から生まれる．C 言語で使われる型によって表現できる範囲を 2 に示す．全ての C 言語は同じとなっておらず，諸君が使っているシステムではこの表のようになっている．いろいろな型があるが，ほとんどの場合，char，int，double で十分である．諸君が作るプログラムでは，これらで十分，間に合うが，問題が生じたときのみ他の型を使えば良い．

表 2: 型によるデータの表現の違い

型	バイト長	範囲	有効精度
char	1	-128〜127	
signed char	1	-128〜127	
unsigned char	1	0〜255	
short int	2	-32768 〜 32767	
signed short int	2	-32768 〜 32767	
unsigned short int	2	0 〜 65535	
int	4	-2147483648 〜 2147483647	
signed int	4	-2147483648 〜 2147483647	
unsigned int	4	0 〜 4294967295	
long int	4	-2147483648 〜 2147483647	
signed long int	4	-2147483648 〜 2147483647	
unsigned long int	4	0 〜 4294967295	
float	4	およそ 10^{-38}〜10^{38}	およそ 6 桁
double	8	およそ 10^{-308}〜10^{308}	およそ 15 桁
long double	12	およそ 10^{-4932}〜10^{4932}	およそ 18 桁

3.2 配列

一次元の配列は数学のベクトルと，二次元の配列は行列とよく似ている．実際，C 言語でベクトルや行列の演算を行うときには，構造が同じ配列を使うことになる．順序づけられた同じ型のデータが複数ある場合，配列の出番となる．添え字 (これが順序を表す) により，それらにアクセスできるので，データの操作が簡単にできる．

配列を使う場合，

```
int i[10], j[100][100];
```

のように宣言を行う．そうすると図 3 のように，メモリー領域が確保され，配列が使えるようになる．この配列のデータにアクセスするためには，配列名と添え字を指定する．次のようにである．

```
i[3]=5;
c=i[3];
```

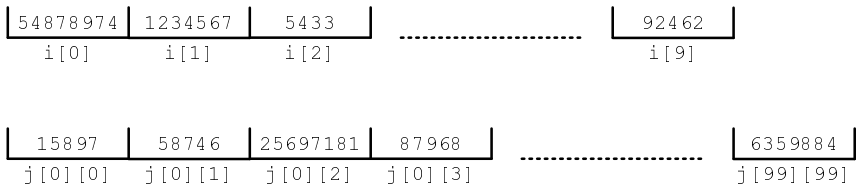

図 3: 配列のイメージ．データを入れる箱がいっぱいある．ただし箱の大きさは全て同じ．

3.3 構造体

構造体は，データの内容をわかりやすくするためにある．例えば，住所録を作るとなると一つのデータの集まりは，

- 名前
- 郵便番号
- 住所
- 電話番号

となるであろう．そうすると，これでひとつのデータの集まりとしたくなる．これを実現するために構造体がある．具体的には，つぎに述べることにする．

データの集まりということでは，配列と似ている．しかし，配列は同じ型のデータの集まりであるが，構造体は異なる型[2]のデータの集まりであることに注意が必要である．使い方も全く異なる．

[2]同じ型でも良い

4 構造体の文法

4.1 構造体型の宣言 (構造体テンプレート)

4.1.1 基礎

構造体は，メンバーと呼ばれる変数が集合した型である．プログラマーが新たに型を定義するようなものである．定義の方法は，一般的には次のようになる．

```
struct タグ名 {
  型 メンバー名;
  型 メンバー名;
  型 メンバー名;
   ・
   ・
   ・
  型 メンバー名;
} 変数リスト;
```

予約語 (キーワード)struct は，構造体を定義するとコンパイラーに伝える役目がある．タグ名と言うのは，新たに定義した構造体の名前である．型を定義するのであるから，その型の名前が必要となる．型は，メンバーの型を示す．これは，C 言語で使うことができる通常の型である．たとえば，int, double, char 等で，構造体もメンバーの型として，使える．メンバー名は，構造体を構成する変数の名前で，そのデータにアクセスするために必要である．変数リストは，ここで定義された構造体を使う場合の変数名である．

タグ名と変数リストのどちらか一方は省略可能である．タグ名を省略すると構造体の型名が無くなるので，その内容がわかりにくくなり，通常は省略しない．一方，変数リストが省略されることは，しばしば生じる．

それでは，例として学生の名前と成績，身長，体重を示した構造体を定義してみる．

```
struct gakusei{
  char name[80];
  int mathematics;
  int english;
  int japanese;
  int electrical_eng;
  int info_eng;
  double height;
  double weight;
} sato, tanaka, yamamoto;
```

これで，gakusei 型の構造体変数の sato と tanaka，yamamoto が使えるようになる．さらに，watanabe という変数を追加したい場合は，

```
struct gakusei watanabe;
```

とすればよい．

変数リストを省略すると，

```
struct gakusei{
  char name[80];
  int mathematics;
  int english;
  int japanese;
  int electrical_eng;
  int info_eng;
  double height;
  double weight;
};
```

となる．こうすると，この gakusei 型の構造体変数を使うためには，

```
struct gakusei sato, tanaka, yamamoto;
```

のようにプログラム中で書く必要がある．

一方，タグ名を省略すると，

```
struct {
  char name[80];
  int mathematics;
  int english;
  int japanese;
  int electrical_eng;
  int info_eng;
  double height;
  double weight;
} sato, tanaka, yamamoto;;
```

となる．この場合は，新たにこの型の構造体変数を追加することができないので，不便な場合がある．変数の数があらかじめ分かっている場合に使われる．

4.1.2 変数を配列に

先の例だと，学生数が多くなると不便になる．多くのデータを扱う場合は配列が便利なのは以前述べたとおりで，ここでもそれが使える．たとえば，クラス毎に

```
struct gakusei{
  char name[80];
  int mathematics;
  int english;
  int japanese;
  int electrical_eng;
```

```
  int info_eng;
  double height;
  double weight;
} M2[50], E2[50], C2[50], B2[50];
```

とする事ができる.
変数リストを省略して,

```
struct gakusei{
  char name[80];
  int mathematics;
  int english;
  int japanese;
  int electrical_eng;
  int info_eng;
  double height;
  double weight;
};
```

と構造体を定義して,プログラム中で,

```
struct gakusei M2[50], E2[50], C2[50], B2[50];
```

と書き,変数を使うこともできる.

4.1.3 メンバーを構造体型に

先のデータを見ると,成績を表すメンバー (mathematics, english, japanese, electrical_eng, info_eng) は似たようなデータで関連が強い.これは一つにまとめるとデータの内容が分かり易くなり,プログラムの見通しが良くなる.そのためには,次に示すように構造体のメンバーを使えば良い.

```
struct seiseki{
  int mathematics;
  int english;
  int japanese;
  int electrical_eng;
  int info_eng;
};

struct gakusei{
  char name[80];
  struct seiseki test;
  double height;
  double weight;
};
```

このように入れ子にすることを，構造体のネストと言う．ここでは 2 段のネストであるが，3 段でも 4 段でも可能である．

4.2 構造体型のメンバーの参照

構造体のメンバーの参照—アクセス—には，. 演算子 (ドット演算子) をつかう．次のようにするのである．

```
構造体変数 . メンバー                                    /* 通常 */
構造体変数 . メンバー . メンバー                          /* メンバーが構造体 */
構造体変数 . メンバー . メンバー . メンバー                /* メンバーのメンバーも構造体*/
構造体変数 [配列の添え字] . メンバー                      /* 構造体が配列*/
構造体変数 . メンバー [配列の添え字]                      /* メンバーが配列*/
構造体変数 [配列の添え字] . メンバー [配列の添え字]       /*構造体もメンバーも配列*/
```

ようするに，構造体といえども，ドット演算子を使う以外は通常の変数とほとんど同じである．それでは，実際のプログラム例で，それを見てみよう．

リスト 1: 構造体をつかったプログラム例.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  struct seiseki{       //構造体テンプレート(成績)
5    int mathematics;
6    int english;
7  };
8
9  struct gakusei{       //構造体テンプレート(学生)
10   char name[80];
11   struct seiseki test;
12   double height;
13 };
14
15 //=========================================================
16 //   メイン関数
17 //=========================================================
18 int main(void)
19 {
20
21   struct gakusei E2[10];       //構造体を使う
22
23   strcpy(E2[0].name,"yamamoto");
24   E2[0].test.mathematics = 95;
25   E2[0].test.english = 65;
26   E2[0].height = 174.8;
27
28   printf("%s\n", E2[0].name);
29   printf("   Mathematics : %d\n", E2[0].test.mathematics);
30   printf("   English     : %d\n", E2[0].test.english);
31   printf("   Height      : %f [cm]\n", E2[0].height);
32
33   return 0;
34 }
```

実行結果

```
yamamoto
  Mathematics : 95
  English     : 65
  Height      : 174.800000 [cm]
```

構造体は通常の変数のように取り扱うことができるので，scanf をつかって，キーボードからデータを読み込む場合は，次のようにする．先ほどのプログラムで，キーボードから入力された値をメンバーの mathematics に格納するためには次のようにする．

```
scanf("%d",&E2[0].test.mathematics);
```

通常の変数同様，格納する変数 (ここではメンバー) の頭に&をつけるだけである．

4.3 構造体型を使うときの注意

- 構造体のメンバー名と通常の変数には同じ名前が使える．すなわち，以下は問題ない．

```
struct seisu{
  int i;
  int j;
};

struct seisu k,l;
int i,j;
```

- 次のように構造体を宣言するときに，初期ががができる．

```
struct seisu{
  int i;
  int j;
};

struct seisu k={1,2};
struct seisu l[2]={{3,4},{5,6}};
```

- 同じ構造体であれば，代入演算子 (=) を用いて，そのままコピーできる．いちいちメンバー毎，コピーする必要はない．

```
struct seisu{
  int i;
  int j;
};

struct seisu k, l={1,2};
k=l;
```

- 構造体同士の比較は無意味であるし，できない．比較する場合は，メンバーを比較する．
- 関数の引数に構造体を用いることは，問題なく可能である．ふつうの変数のように考えればよい．

5　プログラム作成の練習

[**練習 1**] 次のデータを示す構造体を作成せよ．そして，データを格納して，表示するプログラムを作れ．

- 名前
- 年齢

[**練習 2**] 問 1 を拡張して，3 人分のデータを格納できるようにせよ．ヒント:構造体の配列を使う．そして，データを入力して表示せよ．

[**練習 3**] 次のようなデータ構造を表す構造体を作成せよ．構造体のネストをつかえ! そしてデータを入力して，表示せよ．

- 学年
- クラス (M, E, C, B)
- 名前
- テストの成績
  - 数学
  - 英語
  - 情報処理

6　課題

6.1　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]p.294–323 を 3 回読め．以下の問に答えよ．レポートには問いの答えとともに「3 回読んだ」と書け．

- 構造体とは何か? 3 行程度で説明せよ．
- 構造体のメンバとは何か?
- ユーザー定義型とは何か?

[**問 2**] (復) 本日配布したプリントを 2 回読め．レポートには「2 回読んだ」と書け．さらに，誤字・脱字，表現の悪いところ，間違いを指摘せよ．

[**問 3**] (復) 構造体を使ったプログラムを自分で作成せよ．

[**問 4**] ここでの学習内容でわからないところがあれば，具体的に記述せよ．

6.2 レポート提出要領

期限　　4 月 17 日 (火) 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「情報処理応用」
　　課題名「課題　構造体」
　　提出日
　　2E　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] 内田智史監修, (株) システム計画研究所編. C 言語によるプログラミング　基礎編　第 2 版. (株) オーム社, 2006.