構造体

山本昌志*

2005 年 4 月 14 日

1　本日の学習内容

本日は、配列と構造体の違い示し、後者が便利なデータ構造であることを学習する。そして、それを C 言語で実装する方法を学ぶ。これを学ぶことにより、分かりやすいプログラムが書けるようになる。

構造体は、プログラマーがデータの内容を分かりやすくするための変数を定義するようなものである。変数といっても、それはデータの集合体となっている。そのことを理解して、それが便利だと分かって欲しい。

*独立行政法人　秋田工業高等専門学校　電気情報工学科

2　構造体とは

以前示したように、プログラミングで使われるデータ構造は、表 1 のようなものがある。本日は、構造体を学習する。

構造体を学習するとほぼ基本データ構造の学習は終了する。問題向きデータ構造については、後期に学習することになる。

構造体を配列の違い、そして、構造体が便利な理由を、教科書に沿って説明する。

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　構造体の文法

教科書は分かり易く書かれているが、説明不足の所がある。そこで、退屈ではあるが細かい説明をしておく。

3.1 構造体型の定義

3.1.1 基礎

構造体は、メンバーと呼ばれる変数の集合体の型である。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１名;
  型 メンバー２名;
  型 メンバー３名;
  ・
  ・
  ・
  型 メンバー N 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なのである。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。

3.1.2 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。

3.1.3 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。

3.2 構造体型のメンバーの参照

構造体のメンバーの参照には、. 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。

```
#include <stdio.h>
#include <string.h>

struct seiseki{
  int mathematics;
  int english;
};

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

int main(){

  struct gakusei e2[10];

  strcpy(e2[0].name,"yamamoto");
  e2[0].test.mathematics = 95;
  e2[0].test.english = 65;
  e2[0].height = 174.8;

  printf("%s\n", e2[0].name);
  printf("  Mathematics : %d\n", e2[0].test.mathematics);
  printf("  English     : %d\n", e2[0].test.english);
  printf("  Height      : %f [cm]\n", e2[0].height);

  return 0;
}
```

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。

3.3 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。

4 練習問題 (プログラム作成)

教科書の練習問題 p.288 のプログラムを作成せよ。どうしてもわからない場合は、答えをみて、そのプログラムの内容を理解せよ。

4.1 アンケート集計

教科書の練習問題 Lesson 8-1 のプログラムを作成せよ。

Lesson 8-1

以下のような予想結果を 10 人分入力し、男女別の予想結果を「勝ち・引き分け・負け」に分けて集計し、表示するプログラムを構造体を利用して作成しなさい。

4.2 -応用問題-データをまとめて管理する 1

教科書の練習問題 Lesson 8-2 のプログラムを作成せよ。

Lesson 8-2

音楽 CD 10 枚に入っている曲 (各 CD には 6 曲) を入力し、テキストファイルに記録するプログラムを作成しなさい。ただし、作成するテキストファイル (CDdata.txt) は次の通りとします。

```
CD1 title:Zokkon C GENGO
Artist: Hoge Maru
song1: Ikeike hensuu
song2: Hairetu BANZAI
   ・
   ・
   ・
song6: Saraba Kozoutai
CD2 Title: Love Love C
Artist: etc etc
```

プログラム中の CD データの取り扱いには、構造体を使うこと。

4.3 -応用問題-データをまとめて管理する 2

教科書の練習問題 Lesson 8-3 のプログラムを作成せよ。

Lesson 8-3

Lesson 8-2 で作成したテキストファイルを読み込み、曲名を入力するとその曲が入っている CD の名前と何番目かを表示してくれるプログラムを作成しなさい。

5　レポート

5.1　内容

練習問題のプログラムのうち、「アンケート集計」と「-応用問題-データをまとめて管理する 1」を完成させること。実行の確認ができたならば、それをプリントアウトして、レポートとして提出する。

将来、情報処理関係の仕事に就きたいと思っている者は、頑張って、最後の「-応用問題- データをまとめて管理する 2」を完成させよ。最後の問題は、できた者のみレポートとして提出のこと。

自宅のプログラムを作っても良い。そして、実行の確認がとれたならば、それをプリントアウトし、レポートとして提出しても良い。

5.2　レポート提出要領

提出方法は、次の通りとする。

期限	後日、指定する。
用紙	A4
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「情報工学」 課題名「課題　構造体の練習」 1E　学籍番号　氏名 提出日
内容	ソースプログラム (プリントアウトのみ、手書きは不可)