構造体の使い方

山本昌志*

2007 年 4 月 17 日

概 要

ここでは，ユーザー定義型をつかい構造体変数を分かり易く記述する方法，ユーザー定義関数の引数と戻り値に構造体を使う方法，構造体のポインターの使い方の基礎的なことを学ぶ．

1 前回の復習と本日の学習内容

1.1 復習

先週の授業では，構造体の基礎的なことを説明した．構造体はデータをまとめて管理することができる—ということを理解しなくてはならない．構造体が便利な例として，2 年電気情報工学科の 45 人の英語と数学のテストの点を管理することを考える．構造体の定義と宣言は次のようにする．

```
struct student{
  char name[80]
  int math;
  int eng;
};

struct student E2[45];
```

これに対して，構造体が無いプログラミング言語の場合，次のように配列を使うことになる．

```
char name[45][80];
int math[45], eng[45];
```

配列を使った例だと，データはバラバラである．出席番号 23 番—添字 23—の名前と数学，英語の点の関係がはっきりしない．添字の番号が同じであれば，名前と数学，英語の成績が対応するという暗黙の了解があるのみである．一方，構造体だと，名前と数学，英語の成績でひとつの変数となっており，強いつながりがある．構造体を使うと，データをまとめることができ，分かり易いプログラムを書くことができる．

1.2 本日の学習内容

前回に引き続き，構造体について学習する．教科書 [1] の pp.304–323 が範囲であるが，共用体 (union) と列挙型は学習範囲外とする．共用体は組込みシステムのプログラムでは必須ではあるが，諸君が作るプ

*独立行政法人　秋田工業高等専門学校　電気情報工学科

ログラムで使うことはないからだ．列挙型は便利あるが，混乱する学生も出てくるのでここでは教えない．自分で勉強して，使ってみると良いだろう．

本日は，構造体を上手に使うことを学習する．次の通り，学習のゴールを設定している．

- ユーザー定義型を使い，構造体を定義できる．
- 関数の引数に構造体を使うことができる．
- 構造体のポインターを使うことができる．

2 ユーザー定義型

2.1 構造体型の定義と宣言

学生を表す構造体 gakusei の定義[1]は，次のように書くことができる．これで，学生の名前と数学，英語，情報処理の成績を入れる型を作ったことになる．

```
struct gakusei{
  char name[80];
  int mathematics;
  int english;
  int info_eng;
};
```

これは，gakusei という構造体のテンプレート (template:鋳型) を作成しただけで，メモリーの確保は行っていない．

実際に，メモリーを確保するためには，次のように書く．gakusei 型構造体の変数の宣言である．

```
struct gakusei shimada, yamamoto, E2[45];
```

これで，先ほど定義した構造体を使うためのメモリーを確保したことになる．その領域にには名前—変数名のこと—が付けられており，それぞれ shimada と yamamoto, そして配列型は E2[0], E2[1],$\cdots$,E2[44] となっている．

賢明な諸君は，あたかも struct gakusei が型のように振る舞っていることが分かるだろう．すなわち，int のように struct gakusei が型名で，それに続く shimada と yamamoto が変数名，E2 が配列名である．

2.2 構造体とユーザー定義型

それにしても，構造体型名 struct gakusei は長すぎる．そして分かりにくい．通常は構造体型定義—テンプレート作成—のとき，次のようにユーザー定義型を使う．

```
typedef struct{
  char name[80];
  int mathematics;
  int english;
```

[1]教科書の p.324 とこのプリントでは，定義と宣言が逆の意味で使われている．このプリントのように，テンプレートを作ることを定義，変数名を付けてメモリーを確保することを宣言と言う．それもはっきり決まっていないので注意が必要である．

```
    int info_eng;
  } student;
```

このようにすると，student 型構造体の変数宣言は，次のようにできる．

```
student shimada, yamamoto, E2[45];
```

これで，student 型の変数 shimada と yamamoto，配列 E2[45] を使うことができる．typedef を使うことで，変数宣言が分かり易くなることが理解できるであろう．構造体を使うときには，このようにユーザー定義型 typedef とあわせて使いプログラムを分かり易くするテクニックは，常套手段である．構造体を使うときには，typedef といっしょに使え!!! プログラム例をリスト 1 に示す．

リスト 1: 構造体にユーザー定義型 typedef を使った例．

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

int main(void)
{
  typedef struct{
    char name[80];
    int mathematics;
    int english;
    int info_eng;
  } student;

  student shimada, yamamoto, E2[45];
  int av;

  strcpy(shimada.name,"shimada masaharu");
  shimada.mathematics = 92;
  shimada.english = 88;
  shimada.info_eng = 45;

  av=(shimada.mathematics + shimada.english + shimada.info_eng)/3;

  printf("%s no heikin = %d\n", shimada.name, av);


  return 0;
}
```

実行結果

```
shimada masaharu no heikin = 75
```

typedef は構造体のためにあるわけではない．型に別名を付けるためにある．例えば，int 型に hogehoge と言う型名を付けることもできる．

```
typedef int seisu;
```

このようにすると，int のかわりに seisu という型名も使うことができる．変数の宣言を次のようにするのである．

```
seisu i, j;
```

上手に使えば，分かり易いプログラムを書くことができる．

3　ユーザー定義関数との構造体の受け渡し

普通の変数同様に，構造体型もユーザー定義関数との受け渡しができる．

3.1　構造体を渡す

引数に構造体を使うときは，普通の変数と同じである．リスト 2 の場合，student 型の変数 shimada の値は，関数 cal_av() を呼び出したときに，ローカル変数 gakusei にコピーされる．そして，平均値を計算する関数での処理に使われる．通常の変数と全く同じ．ただし，構造体の定義の位置は重要である．構造体を使う前に定義しなくてはならない．

リスト 2: 構造体のデータをユーザー定義関数に渡す．

```
1  #include <stdio.h>
2  #include <string.h>              // strcpy を使うため
3
4  typedef struct{                  // 構造体型 student の定義
5    char name[80];
6    int mathematics;
7    int english;
8    int info_eng;
9  } student;
10
11 int cal_av(student gakusei);     //プロトタイプ宣言
12
13 //=========================================================
14 // メイン関数
15 //=========================================================
16 int main(void)
17 {
18   student shimada;
19   int hei;
20
21   strcpy(shimada.name,"shimada masaharu");
22   shimada.mathematics = 92;
23   shimada.english = 88;
24   shimada.info_eng = 45;
25
26   hei=cal_av(shimada);
27
28   printf("%s no heikin = %d\n", shimada.name, hei);
29
30   return 0;
31 }
32
33 //=========================================================
34 // 平均点を計算するユーザー定義関数
35 //=========================================================
36 int cal_av(student gakusei)
37 {
38   int av;
39
40   av=(gakusei.mathematics + gakusei.english + gakusei.info_eng)/3;
41
42   return av;
43 }
```

実行結果

```
shimada masaharu no heikin = 75
```

3.2 構造体データを受け取る

戻り値の型を構造体型とすると，構造体を返すユーザー定義関数を作ることができる．これも通常の変数と同じ．その例をリスト 3 に示す．

リスト 3: 構造体のデータをユーザー定義関数に渡す.

```
1  #include <stdio.h>
2  #include <string.h>              // strcpy を使うため
3
4  typedef struct{                  // 構造体型 student の定義
5    int math;
6    int eng;
7  } student;
8
9  student cal_av(student a[]);       //プロトタイプ宣言
10
11 //=========================================================
12 // メイン関数
13 //=========================================================
14 int main(void)
15 {
16   student E2[3], heikin;
17
18   E2[0].math = 92;
19   E2[0].eng = 88;
20   E2[1].math = 42;
21   E2[1].eng = 38;
22   E2[2].math = 82;
23   E2[2].eng = 23;
24
25
26   heikin=cal_av(E2);
27
28   printf("mathematics no heikin = %d\n",heikin.math);
29   printf("English     no heikin = %d\n",heikin.eng);
30
31   return 0;
32 }
33
34 //=========================================================
35 // 平均点を計算するユーザー定義関数
36 //=========================================================
37 student cal_av(student a[])
38 {
39   student hei;
40
41   hei.math = (a[0].math+a[1].math+a[2].math)/3;
42   hei.eng = (a[0].eng+a[1].eng+a[2].eng)/3;
43
44   return hei;
45 }
```

実行結果

```
mathematics no heikin = 72
English     no heikin = 49
```

4 構造体のポインター

構造体のポインターも，通常の変数と同じように使うことができる．その例をリスト 4 に示す．この例では，構造体のポインターをユーザー定義関数とともに用いている．以下，その方法を示すが，アロー演算子の他は通常の変数と同じ取り扱いである．

- 構造体のポインターの宣言は，アスタリスク (*) をつける (17 行目)．通常の変数と同じ．
- 仮引数がポインターで宣言された関数 (10 行目と 36 行目) を呼び出すときに，渡す実引数はアドレスである (20 行目)．変数からアドレスを引き出すときには&演算子をつかう．
- 構造体のポインターを戻り値にすることもできる (10 行目と 36 行目)．
- 構造体のポインターをつかってメンバーにアクセスする場合，矢印 (アロー) 演算子を使う．

リスト 4: 構造体のデータをユーザー定義関数に渡す．

```
1  #include <stdio.h>
2  #include <string.h>              // strcpy を使うため
3
4  typedef struct{                  // 構造体型 student の定義
5    char name[80];
6    int math;
7    int eng;
8  } student;
9
10 student *better(student *st1, student *st2);    //プロトタイプ宣言
11
12 //=========================================================
13 // メイン関数
14 //=========================================================
15 int main(void)
16 {
17   student shimada, yamamoto, *yoi;
18
19   strcpy(shimada.name,"shimada masaharu");
20   shimada.math = 92;
21   shimada.eng = 88;
22   strcpy(yamamoto.name,"yamamoto masao");
23   yamamoto.math = 42;
24   yamamoto.eng = 61;
25
26   yoi = better(&shimada, &yamamoto);        // アドレスを送る
27
28   printf("平均点が高い方は, %sです. \n",yoi->name);
29
30   return 0;
31 }
32
33 //=========================================================
34 // 平均点を計算するユーザー定義関数
```

```
35 //=========================================================
36 student *better(student *st1, student *st2)
37 {
38   int av1, av2;
39
40   av1 = (st1->math + st1->eng)/2;          // アロー演算子を使い
41   av2 = (st2->math + st2->eng)/2;          // メンバーにアクセス
42
43   if(av1<av2){
44     return st2;
45   }else{
46     return st1;
47   }
48
49 }
```

実行結果

平均点が高い方は，shimada masaharu です．

5 プログラム作成の練習

[**練習 1**] ユーザー定義型を使い次のようなメンバーを持つ構造体を作成せよ．

- 商店名
- パソコンの価格
- ディスプレイの価格

そして，適当な値を入れて，表示させてみよ．

[**練習 2**] 問 1 を拡張して，パソコンとディスプレイの合計価格を計算する関数を作成せよ．その関数を利用して，商店名と合計価格を表示せよ．

[**練習 3**] 問 2 を拡張して，2 つの商店の合計価格の安い商店名を表示せよ．このプログラムは，ユーザー定義関数にポインターを渡し，ポインターを受け取ること．

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．

- 構造体のメンバーとはなにか?
- ユーザー定義型の使い方 (プログラム例) を示せ．
- 矢印 (アロー) 演算子とはなにか?

[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) 構造体とユーザー定義関数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 24 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　構造体の使い方」
　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.