目次

★8　構造体

★8.1　構造体を使おう

とある健康診断で得られたデータを扱うプログラムを作ることを考える．データは,「名前」「身長」「体重」の 3 つが組になっているとする．kenshin1.c は，2 人分のデータ（○○ A という変数が 1 人目の分，○○ B が 2 人目の分を表す）のキー入力を受け取り，それらを交換してから表示しようとしたものである．

このようにいくつかの変数が組になっているような場合には，それらをひとまとめにした**構造体**を定義して使うのがよい．実際，kenshin1.c には誤りが含まれているのにそれに気がつきにくいが，kenshin2.c のように構造体を使うとソースを見通しよく書けてバグも混入しにくくなる．

● 構造体の作り方・使い方

(1)

(2)

(3)

kenshin1.c

```
 1  #include <stdio.h>
 2  #include <string.h>  // for strcpy()
 3
 4  int main(void)
 5  {
 6    char nameA[100], nameB[100];
 7    double heightA, heightB;
 8    double weightA, weightB;
 9    char ntmp[100];
10    double htmp, wtmp;
11
12    scanf("%s %lf %lf", nameA, &heightA, &weightA);
13    scanf("%s %lf %lf", nameB, &heightB, &weightB);
14
15    // A <--> B
16    strcpy(ntmp, nameA);
17    htmp = heightA;  wtmp = weightA;
18    strcpy(nameA, nameB);
19    heightA = heightB;  weightA = weightB;
20    strcpy(nameB, ntmp);
21    heightB = htmp;
22
23    printf("[A] %s %3.1fcm %3.1fkg\n",
24           nameA, heightA, weightA);
25    printf("[B] %s %3.1fcm %3.1fkg\n",
26           nameB, heightB, weightB);
27    return 0;
28  }
```

実行例

```
$ ./kenshin1
Hogeo 169.8 65.7      ← キー入力
Fugayo 203.4 45       ← キー入力
[A] Fugayo 203.4cm 45.0kg
[B] Hogeo 169.8cm 45.0kg
```

kenshin2.c

```
 1  #include <stdio.h>
 2
 3
 4
 5
 6
 7
 8
 9  int main(void)
10  {
11
12
13    scanf("%s %lf %lf", a.name, &a.height, &a.weight);
14    scanf("%s %lf %lf", b.name, &b.height, &b.weight);
15
16    tmp = a; a = b; b = tmp;  // A <=> B
17
18    printf("[A] %s %3.1fcm %3.1fkg\n", a.name, a.height, a.weight);
19    printf("[B] %s %3.1fcm %3.1fkg\n", b.name, b.height, b.weight);
20    return 0;
21  }
```

★8.2 typedef による型定義

typedef を用いると，自分で新しい型 (type) を定義する (define) ことができる．例えば，右の場合，Kazu という型が新しく定義されている．このプログラム中では，整数を格納する変数の型名として int と同じ意味で Kazu を使える．また，typedef の所を

```
typedef double Kazu;
```

とすれば，今度は Kazu 型は double 型と同じになる．このように，typedef はわかりやすく保守しやすいプログラムの作成に役立つ．

typedef は，構造体の型名を短く表すために利用されることが多い．例えば，上記の struct kcard という構造体の定義の後で右の例 (2) のように typedef しておけば，struct kcard と書くかわりに KCard と書いて済ませることができるようになる（struct kcard と書いてもよい）．

また，typedef による構造体型名の定義は，右の例 (3),(4) のように構造体の定義と組み合わせてしまうこともできる．この場合，例 (4) では構造体タグ名 (例 (2),(3) の kcard) を省略しているので，struct なんたらとは書けず，KCard という形名のみが使えることになる．

typedef の使用例 (1)

```
typedef int Kazu;

Kazu Add(Kazu x, Kazu y){
   return x + y;
}
```

typedef の使用例 (2)

```
struct kcard{
   char name[100];
   double height;
   double weight;
};
typedef struct kcard KCard;

KCard a, b, tmp;
```

typedef の使用例 (3)

```
typedef struct kcard{
   char name[100];
   double height;
   double weight;
} KCard;

KCard a, b, tmp;
```

typedef の使用例 (4)

```
typedef struct {
   char name[100];
   double height;
   double weight;
} KCard;

KCard a, b, tmp;
```

★8.3 構造体と関数

普通の変数と同様に，構造体変数も関数の引数とできる (☆ 1)．以下のプログラムとその実行例 (☆ 2) からわかるように，関数 Futorou を呼び出しても main の側の b.weight の値は変化しないが，関数 Futorou2 のようにポインタを渡せば，関数内で構造体メンバの値を変化させることができる．これは，普通の変数の場合と同じである．ただし，構造体変数へのポインタを用いる場合，プログラムの書き方に注意が必要である．kcard.c の 19 行目に示すように，構造体変数へのポインタを用いてメンバを参照する際には"->"という演算子を用いる．x->weight は，(*x).weight を略記したものである．

☆ 1) 説明は省略するが，関数の戻り値に構造体を指定することもできる．

☆ 2) BMI(Body Mass Index) は，身長 (単位は [m]) と体重 ([kg]) から計算される人間の肥満度指数．22 が標準らしい．BMI = (体重)/(身長)^2

kcard.h

```
 1  #ifndef _KCARD_H
 2  #define _KCARD_H
 3
 4  // 構造体の定義
 5  typedef struct {
 6    char name[100];
 7    double height;
 8    double weight;
 9  } KCard;
10
11  // プロトタイプ宣言
12  double BMI(KCard x);
13  void Pocha(KCard x);
14  void Pocha2(KCard *x);
15
16  #endif
```

kcard.c

```
 1  #include "kcard.h"
 2
 3  // 引数で構造体変数を受け取る関数 (1)
 4  double BMI(KCard x)
 5  {
 6    double h = x.height / 100;
 7    return x.weight / (h * h);
 8  }
 9
10  // 引数で構造体変数を受け取る関数 (2)
11  void Pocha(KCard x)
12  {
13    x.weight += 30;
14  }
15
16  // 引数で構造体変数を受け取る関数 (3)
17  void Pocha2(KCard *x)
18  {
19    x->weight += 30;
20  }
```

kenshin3.c

```
 1  #include <stdio.h>
 2  #include "kcard.h"
 3
 4  int main(void)
 5  {
 6    KCard a, b, tmp;
 7
 8    scanf("%s %lf %lf", a.name, &a.height, &a.weight);
 9    scanf("%s %lf %lf", b.name, &b.height, &b.weight);
10
11    printf("%s さんの体重は %f[kg], BMI は %f です\n", a.name, a.weight, BMI(a));
12    printf("%s さんの体重は %f[kg], BMI は %f です\n", b.name, b.weight, BMI(b));
13    printf(" Pocha  => "); Pocha(b);
14    printf("%s さんの体重は %f[kg], BMI は %f です\n", b.name, b.weight, BMI(b));
15    printf(" Pocha2 => "); Pocha2(&b);
16    printf("%s さんの体重は %f[kg], BMI は %f です\n", b.name, b.weight, BMI(b));
17    return 0;
18  }
```

実行例

```
$ ./kenshin3
Hogeo 169.8 65.7
Fugayo 203.4 45
Hogeo さんの体重は 65.700000[kg], BMI は 22.787149 です
Fugayo さんの体重は 45.000000[kg], BMI は 10.877037 です
 Pocha  => Fugayo さんの体重は 45.000000[kg], BMI は 10.877037 です
 Pocha2 => Fugayo さんの体重は 75.000000[kg], BMI は 18.128395 です
```

★8.4 構造体の配列

普通の変数と同様に，構造体変数の配列を作ることもできる．C 言語では，関数に配列を渡す際にはポインタを渡すので，呼び出された関数側で配列の要素の値を変更すれば呼び出した側の値も変化する．やはり普通の変数と同じである．

kcard.h

内容は省略 (関数 Pocha3 のプロトタイプ宣言を追加)

kcard.c（前頁のものに以下を追加）

```
22  // 引数で構造体配列を受け取る関数
23  void Pocha3(KCard x[], int n)
24  {
25    int i;
26    for(i = 0; i < n; i++) x[i].weight += 30;
27  }
```

kenshin4.c

```
 1  #include <stdio.h>
 2  #include "kcard.h"
 3
 4  #define NDATA 100
 5
 6  int main(void)
 7  {
 8    KCard a[NDATA];
 9    int i, n;
10
11    for(n = 0; n < NDATA; n++){
12      if(3 != scanf("%s %lf %lf", a[n].name, &a[n].height, &a[n].weight)) break;
13    }
14    Pocha3(a, n);
15    for(i = 0; i < n; i++){
16      printf("%s さんの身長は%.1fcm，体重は%.1fkg です\n", a[i].name, a[i].height, a[i].weight);
17    }
18    return 0;
19  }
```

data.txt

```
Hogeo   169.8  62.3
Fugayo  203.4  45
Issun     3.03  0.33
```

実行例

```
$ ./kenshin4 < data.txt
Hogeo さんの身長は 169.8cm，体重は 92.3kg です
Fugayo さんの身長は 203.4cm，体重は 75.0kg です
Issun さんの身長は 3.0cm，体重は 30.3kg です
```