構造体とデータ構造

長谷川「よくわかるC言語」近代科学社
に基づく

構造体とは

配列(array)---同じ型の要素を多数記憶できる

構造体(structure)---いろいろな型の変数をひとまとめにして扱える

構造体を使うには

定義と宣言 --- 例: 名簿を作る

名簿(meibo)の項目には

名前(name 文字列)

年齢(age 整数)

性別(gender 文字(FかM))

身長(height 浮動小数点数)

を書くものとする

構造体名

```
struct meibo {
    char name[11];
    int age;
    char gender;
    float height;
}
```

メンバー名

x;

この構造体「型」の変数

構造体の使用例 (structure-0.c)

```
#include <stdio.h>
#include <string.h>
int main(void) {
  struct meibo {
     char name[11];
     int age;
     char gender;
     float height;
  } x;
  strcpy( x.name, "中京　太郎");
  x.age = 21;
  x.gender = 'M' ;
  x.height = 170.5;
  printf("名前: %s　年齢: %d歳  性別: %c  身長: %.1f(cm)¥n",
                          x.name, x.age, x.gender, x.height);
  return 0;
}
```

構造体のメンバーの書き方に注意
　変数名.メンバー名
例: x.age　(間はピリオド)

構造体の定義のいろいろ

構造体を定義して、その型の変数を宣言するには、いろいろな方法がある

(1) 型と変数宣言を同時に行う

```
struct meibo {
    char name[11];
    int age;
    char gender;
    float height;
} x;
```

(2) 型と変数宣言を別に行う

```
struct meibo {
    char name[11];
    int age;
    char gender;
    float height;
};
struct meibo x;
```

構造体の定義のいろいろ(2)

(3) 型を定義し、型に名前をつける

```
struct meibo {
    char name[11];
    int age;
    char gender;
    float height;
};
typedef struct meibo MEIBO;

MEIBO x;
```

(4)型を定義し、型に名前をつける
ことを一遍にする(多く用いられる)

```
typedef struct meibo {
    char name[11];
    int age;
    char gender;
    float height;
} MEIBO;

MEIBO x;
```

```
#include <stdio.h>
#include <string.h>
int main(void) {
  struct meibo {
     char name[11];
     int age;
     char gender;
     float height;
  } x;
  strcpy( x.name, "中京　太郎");
  x.age = 21;
  x.gender = 'M' ;
  x.height = 170.5;
  printf("名前: %s　年齢: %d歳　性別: %c　身
                          x.name, x.age, x.gender,
  return 0;
}
```

構造体の初期化 (structure-1.c)

structure-0.cと異なる方法で構造体変数に値を与える方法の紹介

```
typedef struct meibo {
    char name[11];
    int age;
    char gender;
    float height;
} MEIBO;
```

```
MEIBO x = {"中京　太郎", 21, 'M', 170.5};
```

構造体の配列

構造体の配列も作れる: (structure-2.c)

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

int main(void) {
  typedef struct meibo {
     char name[11];
     int age;
     char gender;
     float height;
  } MEIBO;
  int i;
```

```
MEIBO list[3] = { {"中京　太郎", 21, 'M', 170.5},
                  {"愛知 花子", 20, 'F', 155.3},
                  {"名古屋 始", 19, 'M', 165.4} };
for (i=0; i < 3; i++)
  printf("名前: %s　年齢: %d歳  性別: %c  身長: %.1f(cm)¥n",
          list[i].name, list[i].age, list[i].gender, list[i].height);
  return 0;
}
```

構造体の配列

構造体のポインタ: (structure-3.c)

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

int main(void) {
  typedef struct meibo {
     char name[11];
     int age;
     char gender;
     float height;
  } MEIBO;
```

```
MEIBO list[3] = { {"中京　太郎", 21, 'M', 170.5},
                  {"愛知 花子", 20, 'F', 155.3},
                  {"",, 0, 'M', 0.0} };
MEIBO *ptr;
for (ptr = list; ptr -> age > 0; ptr++)
  printf("名前: %s　年齢: %d歳  性別: %c  身長: %.1f(cm)¥n",
          ptr->name, ptr->age, ptr->gender, ptr-> height);
  return 0;
}
```

構造体を指すポインタの場合
メンバは　ポインタ->メンバ名

問題: 複素数の構造体(complexTemplate.c)

1) 複素数 を構造体として定義せよ
   構造体の型名は Comp
   メンバー名は re と im (ともに型はdoule) とする
2) 以下のそれぞれの関数を作れ
   複素共役を求めるc_conj (戻り値の型は Comp）, 絶対値を求める c_abs (戻り値の型はdouble)
3) 以下のそれぞれの関数を作れ(いずれも戻り値の型は Comp）
   2つの複素数の和を計算する c_add
   差を求める c_sub, 積を求める c_mult,
   商を求める c_div