データ構造とアルゴリズム (第11回)

静岡大学システム工学科

安藤 和敏

2008.01.28

データ構造(先週の話の繰返し)

Kruskal のアルゴリズムを実装するためには，集合を表現する変数やそのような変数の合併 $(\cup)$ を計算する操作を，プログラミング言語で実現しなければならない． それを実現するのがデータ構造 **(data structure)** である．

一般に，データ構造とは，構造化，組織化されたデータの集まりのことであり，プログラミング言語においては，構造を持った変数の事である．

データ構造には，

- リスト
- スタック，待ち行列 (キュー)
- 2分木，平衡2分木

などがある．

構造体とは何か

様々なデータ構造をC言語で実装するためには，構造体を用いる．

構造体とは，

操作しやすいように一つの名前でまとめされた，一つまたはそれ以上のおそらくは異なった型の変数の集まり

である（カーニハン& リッチー「プログラミング言語C」より）．

C＋＋やJavaなどのオブジェクト指向言語においては，クラスと呼ばれる概念が基本的であるが，構造体はクラスの最も原始的な形態.

構造体の例

- 複素数は，一つの構造体であり，実部を表す実数と虚部を表す実数によって構成される．
- 平面上の点も，一つの構造体であり，x座標を表す実数とy座標を表す実数から成る．
- アドレス帳の中のアドレスも，一つの構造体であり，氏名，住所，電話番号，... などの文字列変数から成る．

構造体タグと構造体メンバ

構造体に付けられる名前のことを構造体タグと呼ぶ．(例：複素数，点，アドレス)

一つの構造体中に含まれている変数のことを，その構造体のメンバと呼ぶ．

例:

- 構造体複素数のメンバは，実部と虚部
- 点のメンバは，x座標とy座標
- アドレスのメンバは，氏名，住所，電話番号,....

構造体の宣言

ある名前(構造体タグ)をもった構造体の宣言． これはデータの型の宣言である．

構造体の宣言のしかた I

`struct` ＜構造体タグ＞ {＜メンバ宣言の並び＞};

例 (複素数を表すデータ型の宣言):

```
struct COMPLEX {    /* 構造体タグ COMPLEX */
    double real;     /*  double 型のメンバ変数 real */
    double img;      /*  double 型のメンバ変数 img */
};
```

前ページの宣言は，ある構造体の型を宣言しただけである．型の宣言と同時に変数も宣言できる．

構造体の宣言のしかた I(a)

struct ＜構造体タグ＞ {＜メンバ宣言の並び＞} 構造体識別子;

例:

```
struct COMPLEX {      /* COMPLEX型の宣言*/
    double real;
    double img;
} x, y;               /* COMPLEX型の変数 x,y の宣言*/
```

普通の変数のときと同様に，初期化もできる．

構造体の宣言のしかた I(b)

struct ＜構造体タグ＞ {＜メンバ変数宣言の並び＞} 構造識別子＝{ メンバの並び};

例:

```
struct COMPLEX {
    double real;
    double img;
} x = { 15,5};  /* COMPLEX型の変数 x を宣言して
                   x = 15 + 5i で初期化*/
```

普通の変数のときと同様に，構造体を指すポインタ変数も宣言できる．

構造体の宣言のしかた I'(a)

struct ＜構造体タグ＞ {＜メンバ宣言の並び＞} 構造ポインタ体識別子;

例:

```
struct COMPLEX {
    double real;
    double img;
} *p;  /* COMPLEX型の変数を指すポインタ p を宣言*/
```

構造体を指すポインタ変数も宣言と同時に初期化できる.

構造体の宣言のしかた I'(b)

struct ＜構造体タグ＞ {＜メンバ宣言の並び＞} 構造ポインタ体識別子＝ ポインタ;

形式I によって，構造体の型名が宣言されていれば，後でその型を持った(ポインタ) 変数を宣言できる．

構造体の変数宣言しかたII

```
struct ＜構造体タグ＞ 構造体識別子＝{メンバの並び};
```

又は，

構造体の変数宣言しかたII'

```
struct ＜構造体タグ＞ *構造体ポインタ識別子＝ポインタ;
```

例:

```
struct COMPLEX a, *p = &a;
```

構造体のメンバの参照のしかた

実体から参照する場合

構造体変数名.メンバ変数

例:

```
struct COMPLEX a = {1, 4};

printf("%d\n", a.real);

printf("%d\n", a.img);
```

ポインタから参照する場合

構造体ポインタ変数名->メンバ変数

例:

```
struct COMPLEX a = {1, 4};

struct COMPLEX *p = &a;

printf("%d\n", p->real);

printf("%d\n", p->img);
```

ついでに,

```
struct COMPLEX a = {1, 4};
struct COMPLEX *p = &a;

printf("%d\n", (*p).real);
printf("%d\n", (*p).img);

printf("%d\n", (&a)->real);
printf("%d\n", (&a)->img);
```

というのもできる.

名前の付けかたに関する注意

構造体タグ，メンバ変数名，構造体変数名は，異なる名前空間において定義されるので，同じ名前を用いることができる．例えば，

```
struct something {
    int something;
    double other;
} something;
```

というのもありだ．しかし，こういうことは混乱するからやめたほうがいい．

プログラム例 Prog10.1 (complex1.c)

```
#include <stdio.h>
#define PrintComplex(x) \
  printf("(%f%s%fi)", (x).real, ((x).img>=0)?"+":"",(x).img)
struct COMPLEX {double real; double img;};
main(){
  struct COMPLEX a={1,2}, b={1,-2}, c;
  c.real = a.real+b.real;
  c.img = a.img+b.img;
  PrintComplex(c);putchar('\n');
}
```

プログラム例 Prog10.2 (complex2.c)

```
#include <stdio.h>
#define PrintComplex(x) \
  printf("(%f%s%fi)", (x).real, ((x).img>=0)?"+":"",(x).img)
struct COMPLEX {double real; double img;};
struct COMPLEX AddComplex(struct COMPLEX a,
                          struct COMPLEX b);
main(){
  struct COMPLEX a={1,3}, b={5,3}, c;
  c = AddComplex(a,b);
```

```
  PrintComplex(a);putchar('+');
  PrintComplex(b);putchar('=');
  PrintComplex(c);putchar('\n');
}

struct COMPLEX AddComplex(struct COMPLEX a,
                          struct COMPLEX b) {
  struct COMPLEX ans;
  ans.real = a.real + b.real;
  ans.img = a.img + b.img;
  return(ans);
}
```

プログラム例 Prog10.3 (complex3.c)

```
#include <stdio.h>
#define PrintComplex(x) \
  printf("(%f%s%fi)", (x).real, ((x).img>=0)?"+":"",(x).img)

struct COMPLEX {double real; double img;};
void AddComplex(struct COMPLEX *c, struct COMPLEX *a,
                           struct COMPLEX *b);
```

```
main(){
  struct COMPLEX a={1,3}, b={5,3}, c;
  AddComplex(&c, &a, &b);    /* c = a + b */
  PrintComplex(a); putchar('+');
  PrintComplex(b); putchar('=');
  PrintComplex(c); putchar('\n');
}
void AddComplex(struct COMPLEX *c, struct COMPLEX *a,
                          struct COMPLEX *b) {
  c->real = a->real + b->real;
  c->img  = a->img + b->img;
}
```

課題 (課題番号 1)

- 問題: テキストの演習問題 10.1
- 締切: 2008年02月01日(金)17:00
- 提出先: lecsys ホームページ