11. 構造体と共用体

(1)構造体・・・事始め

たとえば，住所録を作るプログラムを作ろうとする場合，データは，氏名や住所です。

プログラミング言語を離れて考えると，次のように整理できます。

```
Address_data
        Name  :諸星忠文
         Age  :20
     Mail_no  :189-0002
     Address  :東村山市久米川町 3-59-27
      Tel_no  : 042-394-2798
```

これらのデータを配列で表現すると

```
int Age[100];
char Name [100][19], Mail_no[100][9],
Address[100][31],     Tel_no[100][15];
```

のように表現して，もちろんプログラムも書けますが，せっかく概念的な意味で整理したのに，これらの間の意味的なつながりが無視されてしまっています。

また，最初の添え字で個人を識別しているということは，プログラマだけが分かっており，プログラム上は明確ではありません。

Cでは，上記で整理した概念と同一の形でデータ構造を表現できます。これを，「**構造体**」といいます。

構造体は，配列と対比すると「異なるデータ型の集まり」といえます。

(2)構造体の宣言と参照方法

構造体を宣言するには，たとえば，次のように宣言します。

[宣言 1]

```
typedef struct
{   char Name [19];
    int  Age;
    char Mail_no[9] ;
    char Address[31];
    char Tel_no[15];
} Address_data;
Address_data data[100], A, B;
```

[宣言 2]

```
struct Address_data
{   char Name [19];
    Int  Age;
    char Mail_no[9] ;
    char Address[31];
    char Tel_no[15];
};
struct Address_data data[100], A, B;
```

データが構造体であることを常に意識したい時は宣言 2，意識せずデータの型のひとつとしてみなしたいときは宣言 1 を使います。これらのデータを参照するには，以下のように記述します。

[参照方法]

```
    A.Age = 20;
    data[15].Age=22;
```

typedef は型を識別する名前を宣言する方法です。たとえば次のような使われ方もします。

```
typedef long secNo, *secPtr;
secNo  deta[100]; /* long の配列 */
secPtr sp;   /* long へのポインタ */
```