データ構造 (配列)

山本昌志*

2004 年 10 月 21 日

1　これまでの復習と今週の内容

1.1　これまでの復習

今まで、学習してきたことをまとめると、次のようになる。

- 入出力

`scanf`	キーボードからの入力
`printf`	ディスプレイ出力

- 変数宣言

`int`	変数名を書くことにより、整数を入れる入れ物を用意する。
`double`	変数名を書くことにより、倍精度実数を入れる入れ物を用意する。

- 演算子

`=`	代入を行う
`+,-,*,-`	四則演算を行う
`<, <=, >, >=`	大小を判定する関係演算子
`!=, ==`	等しいか、否かを判定する演算子
`&&, ||, !`	論理演算子

- 制御文

`if`	`else if` や `else` と組み合わせて、制御構造を作る
`switch`	`case` と組み合わせて、多分岐構造を作る。

- 繰り返し文

*国立秋田工業高等専門学校　電気情報工学科

`for`	前判定繰り返し。あらかじめ繰り返し回数が分かっているときに、使われることが多い。
`while`	前判定繰り返し。繰り返し回数が分からないときに、使われることが多い。
`do while`	後判定繰り返し。繰り返し回数が分からないときに、使われることが多い。

1.2 今週の学習内容

本日は、配列というデータ構造について、学習する。いままで、諸君が知っているデータ構造は、単純型と呼ばれるもので、

```
int a;
double x;
```

のように宣言される。これは、

- a という名前が付いた整数を格納する入れ物を用意する。
- x という名前が付いた倍精度実数を格納する入れ物を用意する。

というように解釈する。このデータ構造では、変数名、たとえば a や w を指定することで、データにアクセスする。

ここでは、もっと進んだデータ構造を学習する。それは、配列と呼ばれるもので、

```
int b[10000];
double y[10000];
```

のように宣言され、名前と自然数によりデータにアクセスできる。この配列の使い方を理解することで、とんでもなく大量のデータを取り扱うことができるようになる。

2 配列の必要性と宣言

2.1 たくさんの値を記憶する必要性

コンピュータでは、データを記憶する場所には、プログラマーが名前を付ける必要がある。そうしないと、プログラマーは記憶したデータを取り扱うことができない。これまでに、諸君は単純型と言われるデータ構造を学習した。それを使うためには、記憶するデータの型と変数名をプログラムの最初に記述する。次のようにである。

```
int a;
double x;
```

このようにいちいち名前を指定する方法だと、大量のデータを処理することは不可能である。たとえば、100万個の名前を付けるだけで、大変である。

2.2 配列の宣言

配列宣言の書式は、次の通りである。

書式 (1 次元配列)

データ型名 配列名 [サイズ];

たとえば、

```
int hairetsu[100];
```

のように宣言をすれば、整数を格納する hairetsu[0]〜hairetsu[99] までの 100 個のデータ領域が使える。

多次元配列の場合の書式は

書式 (多次元配列)

データ型名 配列名 [サイズ 1][サイズ 2][サイズ 3]...[サイズ n];

とする。

たとえば、

```
int tajigen[10][20][30];
```

のように宣言をすれば、整数を格納する tajigen[0][0][0]〜tajigen[9][19][29] までの 6000 個のデータ領域が使える。

3 いろいろな配列

3.1 1 次元配列

多くのデータを取り扱うときには、配列と言うデータ構造を用いる。その宣言は、データの型名と配列名、そしてその大きさを記述する。たとえば、次のようにである。

```
int ab[1000000], cd[1000];
double xy[1000000], z[60];
```

これで、

- 整数が格納できる ab[0]〜ab[999999] の 100 万個の記憶場所
- 整数が格納できる cd[0]〜cd[999] の 1000 個の記憶場所

- 倍精度実数が格納できる xy[0]〜xy[999999] の 100 万個の記憶場所
- 倍精度実数が格納できる z[0]〜z[59] の 60 個の記憶場所

が確保できた。

このデータ構造で、個々の内容にアクセスするためには、配列名と自然数である添え字 (インデックス) を指定する。次のようにする。

```
a = ab[12345];
ab[23456] = 654321;
scanf("%d",&ab[34567]);
printf("%d\n",ab[45678]);
```

今まで、使ってきた単純型と同じである。

大量のデータを取り扱うとき、単純型のように、その都度、アクセスする配列名と添え字を指定するプログラムを書くのでは、配列を使う意味はほとんどない。繰り返し (ループ) 文とともに使って、配列の有用性が発揮できる。たとえば、つぎのようにすると、

```
for(i=0; i<=360; i++){
  s[i] = sin(i*3.1415/180);
  c[i] = cos(i*3.1415*180);
}
```

の三角関数の値が 0〜360 度まで計算できる。

一次元配列のイメージを図 2 に示す。

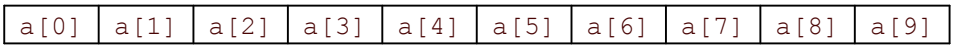

図 1: 一次元配列のイメージ

3.2 2 次元配列

1 次元配列は、配列名とひとつの添字 (自然数) でデータにアクセスする。それに対して、配列名と 2 つの添字を用いるものを 2 次元配列と言う。たとえば、1 日の最高気温を年間に渡って、データとして格納したい場合、

```
double max_temp[13][32];
```

と言うように配列を宣言して用いる。これで、max_temp[0][0]〜max_temp[12][31] まで、使える。

これに、最高気温をキーボードから入力する場合、

```
int max_temp[13][32];
int dates[13] = {0,31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
int i,j;

for(i=1; i<=12; i++){
  for(j=1; j<=dates[i]; j++){
   printf(%d 月%d 日の最高気温を入力して下さい\n", i, j);
   scanf("%lf", &max_temp[i][j]);
  }
}
```

のように書く。配列 dates には、月の日数を入れておく。閏年は考えていない。2 次元配列にアクセス屡ときには、二重ループが使われることが多い。

二次元配列のイメージを図 2 に示す。

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]	a[2][5]
a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]	a[3][5]
a[4][0]	a[4][1]	a[4][2]	a[4][3]	a[4][4]	a[4][5]
a[5][0]	a[5][1]	a[5][2]	a[5][3]	a[5][4]	a[5][5]
a[6][0]	a[6][1]	a[6][2]	a[6][3]	a[6][4]	a[6][5]

図 2: 二次元配列のイメージ

3.3 多次元配列

配列の添字の数を次元と言う。それが 1 個だと 1 次元配列、2 個だと 2 次元配列、それ以上のものも考えられる。先程の最高気温を年月日で処理する場合、3 次元配列が適当であろう。

4 データ構造

コンピューターで処理するデータは、全て、数字[1]に置き換えられる。絵や音、あるいは文字なども全て、数字に直して取り扱っているのである。そして、その取り扱う数字の個数は莫大である。たとえば、画素数が 600×800 のカラー画像を考える。

- 一つの画素は、赤 (R) と緑 (G)、青 (B) の強弱を変えることで表現されている。それぞれの色 (RGB) は、0〜255 までの値を取る。
- カラーがを構成するために必要な数の総数は、$3 \times 800 \times 600 = 1440000$ である。

[1]正確に言うと、0 と 1 のビット列

表 1: データ構造の種類

データ構造	基本データ構造	基本データ型	単純型	整数型
				実数型
				文字型
				論理型
				数え上げ型
			ポインタ型	
		構造型	配列型	
			レコード型	
		抽象データ型		
	問題向きデータ構造	線形リスト	単純リスト	
			双リスト	
			環状リスト	
		木	二分木	完全二分木
				二分探索木
				バランス木
			多分木	
			バランス木	AVL 木
				B 木
		スタック		
		キュー		

たった、一枚のカラー画に 144 万の数字が必要である。動画になると、これを 1 秒間に 30 回も計算している。驚くべきことである。

画像以外にも、コンピューターが処理するデータは沢山ある。それらは、全て数字となっており、それをどのように表現するかがプログラムのポイントなる。データの表現の方法をデータ構造と言う。このデータ構造をまとめたもの表 1 に示す。これまでは、基本データ型の単純型のみを使ってプログラムを組んできたが、今回で配列を学習したわけである。

諸君は、配列をつかうことで、大量のデータを効率よく扱えるようになったのである。ただ、表を見てわかるようにこれまで学習したものは、まだデータ構造の一部であることが分かる。2 年生以降、これらのデータ構造の詳細を学習することになるであろう。プログラミングの修得の道のりは、長いのである。

5　課題

5.1　プログラム作成

以下のプログラムを作成せよ。

- 11 月の毎日の 1 時間毎の気温のデータがある。図 2 のようになっている。
- 日毎の最高気温と最低気温、平均気温をディスプレイに書き出す。
- 11 月の最高気温と最低気温、平均気温をディスプレイに書き出す。

表 2: 11 月の気温

	0 時	1 時	2 時	3 時	···	23 時
1 日	8.3	7.9	7.5	7.2	···	9.8
2 日	9.3	9.2	9.1	9.0	···	6.3
3 日	6.2	5.8	5.3	4.9	···	12.0
⋮	⋮	⋮	⋮	⋮	⋱	⋮
30 日	4.3	3.9	3.3	2.8	···	3.8

5.2　レポート 提出要領

提出方法は、次の通りとする。

期限　　11 月 08 日 (月)PM5:00 まで
用紙　　A4
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 I」
　　課題名「課題　配列 (その 1)」
　　1E　学籍番号　氏名
　　提出日
内容　　ソースプログラム (手書き、プリントアウトどちらでも可)