5. 配列

今までの学習で、変数は

```
int     i;
```

の指定で 1 つの整数型データをしまう箱を用意した。
では 100 個の整数型のデータを使う場合はどうすればよいだろう。
例えば

```
int     a0,a1,a2,a3,a4,a5,a6,a7,a8,a9;
int     a10,a11,a12,a13,a14,a15,a16,a17,a18,a19;
int     a20,a21,a22,a23,a24,a25,a26,a27,a28,a29;
int     a30,a31,a32,a33,a34,a35,a36,a37,a38,a39;
  .....
  .....
int     a80,a81,a82,a83,a84,a85,a86,a87,a88,a89;
int     a90,a91,a92,a93,a94,a95,a96,a97,a98,a99;
```

と書かなければならない (紙の都合上省略しているが..... ではなく実際に書く)。これでは取り扱いが面倒である。これを楽に扱うことができる配列についてここでは学習する。

配列の宣言

a0,a1,..... ,a99 の変数を宣言するのではなく、

```
int   a[100];
```

で、a[0] から a[99] までの 100 個の箱を用意できる。

配列の各要素の操作

配列の各要素を扱う場合には、

```
a0 ではなく a[0]
a5 ではなく a[5]
a99 ではなく、 a[99]
```

とする。ここで、a[0],a[1],...,a[99] はそれぞれ今まで学習した普通の変数と同じように使える。例えば、

```
result = n + a[10] * 23;
scanf("%d", &a[3]);       /* a[3] にキーボードから整数を代入する */
```

といった使い方ができる。
配列で便利なことは、

```
a[i*2+j+5]       (i と j は、int i,j; で宣言されているものとする)
```

のように [ ] の中に具体的な数字そのものだけでなく、式や変数を書くことができることである。ただし、[ ] の中は (計算して) 整数となる値だけで、 a[1.1] や、a[4.0/3] などとすることはできない。[ ] の中の数値、変数、式などを配列 a の**添字 (インデックス)** と呼ぶ。

例

```
int   a[100];
```

と宣言しただけでは、a[0],a[1],....., a[99] の各変数の初期値は不定なので、各配列要素に 0 を代入する時は、宣言後

```
for(i=0; i<100; i++) {
       a[i]=0;
}
```

のようにする。

[練習 2] 10 個の整数値を順番に $a[0]$,$a[1]$,...,$a[9]$ に読み込み、その順序で出力するプログラムを作りなさい。

[練習 3] 上の問題で、入力したのとは逆の順番で出力するようにしなさい。

[練習 4] [練習 2] で、奇数番目に入力した数だけを順に出力するようにしなさい。

[練習 5] $0 \leq i \leq 30$ (i は整数) において

$$a[i] = \begin{cases} 0, & \text{i が偶数の時} \\ 1, & \text{i が奇数の時} \end{cases}$$

となるようなプログラム (の断片) を作りなさい。

[練習 6] 9999 が入力されるまで数を読み込み、入力したのとは逆の順番で出力するプログラムを作りなさい。