1.6　配列

1.6.1　1 次元配列

配列(Array) とは変数をまとめてリストにしたものです．関連性のある複数のデータをまとめて扱いたい場合などに使用します．宣言のしかたは変数の場合とほぼ同じですが，配列のサイズ (配列要素の数) を指定する必要があります．配列の各要素が 1 つの変数と同じようなものになります．

配列の宣言

```
型　配列名 [サイズ];
```

要素数 100 の int 型の配列 array を用意する場合は次のように宣言します．

```
int array[100];
```

同じことを変数でやろうとすると大変です．

```
double array001;
double array002;
...
double array100;
```

配列の要素にアクセスする場合には配列に**添え字 (インデックス)** を付け配列要素を指定します．**配列の要素は 0 からサイズ-1** となります．上の例では要素を 100 個作りましたが，要素の番号は 0 から 99 となります．array の 3 番目の要素に 100 を代入する場合は次のようにします．

```
array[2] = 100;
```

配列の宣言時に初期値を与える場合は次のようにします．

```
int array[5] = {134, 12, 34, 4332, 243};
```

例題 14　1 次元配列

example-14.c

```
#include <stdio.h>

int main(void)
{
        int i;
        int array[5] = {134, 12, 34, 4332, 243};
        for (i = 0; i < 5; i++) printf("array[%d] = %dY=n", i, array[i]);
                array[0] = 10000;
                array[1] = array[2] + array[3] + array[4];
                printf("Y=narray[0] = %dY=narray[1] = %dY=n", array[0], array[1]);

        return 0;
}
```

実行結果

```
array[0] = 134
array[1] = 12
array[2] = 34
array[3] = 4332
array[4] = 243

array[0] = 10000
array[1] = 4609
```

配列は変数とほとんど同じ使い方ができますが，配列を別の配列にまるごと代入することはできません．for 文を使うなどして要素をひとつずつ代入する必要があります．

また，コンパイル時に配列の添え字の範囲はチェックされません．存在しない要素にアクセスした場合，プログラムや OS がクラッシュしたり不安定な状態になったりする可能性があります．

1.6.2 多次元配列

次のように宣言することで 2 次元の配列を作ることができます．

2 次元配列の宣言

型　配列名 [サイズ][サイズ];

`int` 型の 10×5 の配列を宣言する場合は次のようにします．

```
int array[10][5];
```

3 次元の配列なら

```
int array[10][5][17];
```

のように宣言します．このように各次元ごとに要素のサイズを追加すればいくらでも次元を増やすことができます．使い勝手やメモリーサイズの都合上から 4 次元配列以上はあまり使われないようです．

2 次元配列 array の要素番号 [2][3] に 100 を代入する場合は次のようにします．

```
array[2][3] = 100;
```

2 次元配列の宣言時に初期値を与える場合は次のようにします．

```
int array[3][2] = {{1, 2}, {3, 4}, {5, 6}};
```

2 次元配列は行列のイメージそのままに使用することができます．次の例題では以下の計算を 2 次元配列を使って行います．

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} + \begin{pmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{pmatrix} = \begin{pmatrix} 8 & 10 \\ 12 & 14 \\ 16 & 18 \end{pmatrix}$$

例題 15 2 次元配列で 3 行 2 列の行列の和を計算

example-15.c

```
#include <stdio.h>

int main(void)
{
        int i, j;
        int matrixA[3][2] = {           /* 3 行 2 列の行列 A を宣言 */
                {1, 2},
                {3, 4},
                {5, 6}
        };

        int matrixB[3][2] = {            /* 3 行 2 列の行列 B を宣言 */
                {7,  8},
                {9,  10},
                {11, 12}
        };

        int matrixC[3][2];              /* 3 行 2 列の行列 C を宣言 */

        for (i = 0; i < 3; i++) {       /* 行列 A + 行列 B を 行列 C に代入 */
                for (j = 0; j < 2; j++) {
                        matrixC[i][j] = matrixA[i][j] + matrixB[i][j];
                }
        }

        for (i = 0; i < 3; i++) {       /* 行列 C の内容を表示 */
                for (j = 0; j < 2; j++) {
                        printf("%d¥t", matrixC[i][j]);
                }
                printf("¥n");
        }

        return 0;
}
```

実行結果

```
8       10
12      14
16      18
```

1.6.3 課題

課題 12 3 次元ベクトルの内積を求めるプログラムを作ってみよう.

課題 13 3 行 3 列の行列 A と B の積 AB を求めるプログラムを作ってみよう.

ヒント: 行列 C を A と B の積とすると C の要素 C_{ij} は次式で与えられる.

$$C_{ij} = \sum_{n=1}^{k} A_{in}B_{nj} = A_{i1}B_{1j} + \cdots + A_{ik}B_{kj}$$

for ループを 3 重にネストして C_{ij} を求める.