システム科学科 情報処理演習テキスト 6
– 配列その 1 –

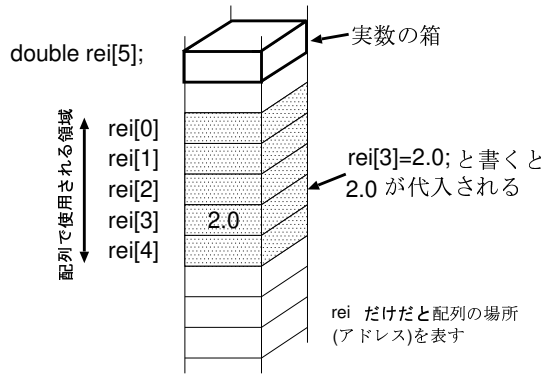
原案:佐藤宏介 修正: 升谷 保博, 庄野 逸, 鳩野逸生, 橋本守, 才脇直樹，田中宏喜，日浦慎作

ポイント 1: 同じ型のデータを一挙に多数取り扱いたい場合、 “配列” を用いる。配列を用いるには、普通の変数と同様に以下のような “宣言” をする必要がある。ただし、要素数は “定数” である必要がある。

配列の宣言: 配列の要素の型 配列名 [必要な要素数];

例:
```
int tensu[100];                    /* 100 名分の試験の点数 */
double heikin[5];                  /* 5 教科分の平均 */
```

ポイント 2: 配列は、同じ型の変数の箱が積み重なったものとイメージすれば良い (下図参照)



ポイント 3: 配列の添字番号は **0** から始まり、必要要素数**-1** 番目までしか使えないことに注意

ポイント 4: 配列 a の i 番目の要素に値を代入したり、その値を参照したりするには a[i] と書けば良い

例
```
double rei[5];      /* 配列の宣言 */
rei[0] = 1.0;       /* 0 番目の要素に 1.0 を代入 */
n=3;
rei[n] = 3.1415;    /* 添字には整数変数も使える */
```

ポイント 5: 配列の各要素は、普通の変数と同様に取り扱うことができる

example6_1.c

```
/*
 * 入力された 2 つの 3 次元ベクトルの内積を求めるプログラム
 */

#include<stdio.h>
#include<math.h>

int main (void)
{

    int i;
    double vector1[3];  /* 3 次元ベクトル 1 のための配列 */
    double vector2[3];  /* 3 次元ベクトル 2 のための配列 */
    double naiseki = 0;

    printf("1 つめのベクトル成分\n");
    for (i = 0; i<3; i++){
        scanf("%lf", &vector1[i]);
    }

    printf(" 2 つめのベクトル成分\n");
    for (i = 0; i<3; i++){
        scanf("%lf", &vector2[i]);
    }

    for ( i = 0; i<3; i++){
        naiseki += vector1[i] * vector2[i];  /*内積の計算*/
    }
    printf("%lf\n",naiseki);

    return 0;
}
```

ポイント 6: 配列および変数は宣言をしたときに初期値を代入できる。

ポイント 7: 初期値の要素数は、宣言する要素数より少なくてもかまわない。また要素数を省略して初期値を設定した場合の配列の要素数は、その初期値を代入するのに 必要な要素数となる。

変数の場合 変数の型 変数の名前 = 変数の初期値 (定数)

例:
```
int number = 10;
double value = -1.0;
```

配列の場合 変数の型 配列の名前 [要素数]= { 要素 0 の値, 要素 1 の値, ··· }

例:
```
int average[5] = {3.3, 5.1, 0.3, 3.2, 1.1 };
int tensu[3] = { 30, 50, 20 };
```

example6_2.c

```
/*
 * あるクラスの数学の成績を 1 次元配列を使って表し，その平均値と標準偏差を求めるプログラム
 */
#include<stdio.h>
#include<math.h>

int main (void)
{

    double heikin = 0;
    double sd = 0;
    int i;

    int math[15] = {58, 72, 83, 36, 94, 67, 85, 78, 70, 81, 77, 69, 90, 83, 75};

    for (i = 0;i<15;i++){
        heikin += math[i];
    }
    heikin /= 15;
    printf("平均値は%lf です\n", heikin);

    for (i = 0;i<15;i++){
        sd += (math[i] - heikin) * (math[i] - heikin);
    }
    sd = sqrt(sd) / 15;

    printf("標準偏差は%lf です\n", sd);
    return 0;
}
```

example6_3.c

```
/* 乱数を発生する関数 rand() を用いて、0 から 99 までの乱数を
 *  100 個発生させ、平均値と、最大値を計算するプログラム
 *
 */
#include<stdio.h>
#include<stdlib.h> /* rand, srand のプロトタイプ宣言が書かれている*/
#include <time.h>  /*time のために必要*/

int main (void)
{

    int ransuu[100];
    int i;
    double heikin;
    int max;

    srand(time(NULL));
                               /* srand は，疑似乱数列の初期値を与えるライブラリ関数
                                * time は現在の暦時間を返すライブラリ関数
                                * NULL はおまじない
                                */

    for (i = 0; i<100; i++) {
       ransuu[i] = rand() % 100;
                               /* 0 から 99 までの乱数を 100 個発生
                                * rand は，0 から RAND_MAX までの疑似乱数列を発生する標準ライブラリ関数
                                * RAND_MAX の値が大きいので，100 で割った余りも乱数となる
                                */
    }

    heikin = 0;

    for (i = 0; i<100; i++){
        heikin += ransuu[i];
    }

    printf("平均値は%lf です。\n", heikin / 100);

    max = ransuu[0];
    for (i = 1; i<100; i++){    /* max の初期値が ransuu[0] なので，1 からループを回す*/
        if (ransuu[i] > max){
            max = ransuu[i];
        }
    }

    printf("最大値は%d です。\n", max);

    return 0;
}
```

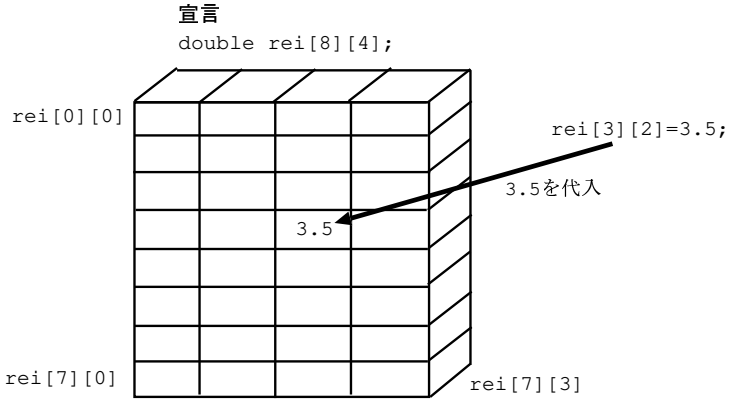
ポイント 8: 2 次元の配列は以下のように宣言すれば良い

配列の宣言: 要素の型 配列名 [要素数][要素数];

例:
```
int tensu2[5][100];          /* 5 クラス × 100 名分の試験の点数 */
double matrix[5][5];         /* 5 × 5 の行列 */
```

ポイント 9: 2 次元の配列は下図のように “行列” をイメージすれば良い

ポイント 10: 2 次元の配列要素 (i,j) に対して代入や参照を行いたいときには、“配列名 [i][j]” として用いる。



example6_4.c

```
/*
 * 2 つの行列の和と差を表示するプログラム
 */
#include<stdio.h>

#define ROW 2
#define COLUMN 2

int main (void)
{
    int a[ROW][COLUMN];
    int b[ROW][COLUMN];
    int c[ROW][COLUMN];
    int d[ROW][COLUMN];
    int i,j;

    printf("1 番目の 2 x 2 行列の値を整数でいれてください\n");
    for (i = 0;i < ROW;i++){
        printf("%d 行目\n",i+1);
        for (j = 0;j < COLUMN;j++){
            printf("    %d 列目",j+1);
            scanf("%d",&a[i][j]);
        }
    }
    printf("2 番目の 2 x 2 行列の値を整数でいれてください\n");
    for ( i = 0;i < ROW;i++){
        printf("%d 行目\n",i+1);
        for (j = 0;j < COLUMN;j++){
            printf("    %d 列目",j+1);
            scanf("%d", &b[i][j]);
        }
    }
    for (i = 0;i < ROW;i++){
        for (j = 0;j < COLUMN;j++){
            c[i][j] = a[i][j] + b[i][j];
            d[i][j] = a[i][j] - b[i][j];
        }
    }
    printf("\n2 つのベクトルの和を表示します。\n");
    for ( i = 0;i < ROW;i++){
        for (j = 0;j < COLUMN;j++){
            printf("%d ",c[i][j]);
        }
        printf("\n");
    }
    printf("\n2 つのベクトルの差を表示します。\n");
    for ( i = 0;i < ROW;i++){
        for (j = 0;j < COLUMN;j++){
            printf("%d ",d[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

ポイント 11: 2 次元の配列も宣言時に値の初期化ができる。

2 次元配列の場合:

```
変数型 配列名 [n 行][m 列]  = {  { 要素 (0,0) の値, 要素 (0,1) の値, ···, 要素 (0,m-1) の値 }
                                 { 要素 (1,0) の値, 要素 (1,1) の値, ···, 要素 (1,m-1) の値 }
                                 ···
                                 { 要素 (n-1,0) の値, 要素 (n-1,1) の値, ···, 要素 (n-1,m-1) の値 } };
```

example6_5.c

```
/*
 * 行列とベクトルの掛け算
 */
#include <stdio.h>

#define ROWMAX 100
#define COLMAX 100

int
main(void)
{
    double m[ROWMAX][COLMAX] = {{0.0, 0.1, 0.2, 0.3},
                                {1.0, 1.1, 1.2, 1.3},
                                {2.0, 2.1, 2.2, 2.3}},
           v1[COLMAX], v2[ROWMAX];
    int    i, j;

    /* v1 への入力 */
    for ( i = 0; i < 4; i++ ) {
        printf("v1[%d] → ", i);
        scanf("%lf", &v1[i]);
    }

    /* v2 = m * v1 の計算 */
    for ( j = 0; j < 3; j++ ) {
        v2[j] = 0.0;
        for ( i = 0; i < 4; i++ ) {
            v2[j] += m[j][i]*v1[i];
        }
    }

    /* v2 の出力 */
    for ( j = 0; j < 3; j++ ) {
        printf("v2[%d] = %8.3f\n", j, v2[j]);
    }
    return 0;
}
```