システム科学科 情報処理演習テキスト 7
– 関数その 2 –

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: 配列内のデータ全体 (配列全体) を関数の引数として、関数に渡すには、呼び出し側の引数に配列のみを書けば良い

ポイント 2: 配列を受け取る関数側では、対応する引数の “配列の型” と “要素数” とを呼び出し側の宣言と同じにする。ただし 1 次元配列の場合、関数側で要素数を指定する必要は無い。

example7_1.c

```
/*
 * 要素数が N である int 型の配列の要素の最小値を返す関数を用いて,
 * 10 人の英語と数学の成績を入力して, それぞれの最低点を求めるプログラム.(配列の受け渡しについて)
 */
# include <stdio.h>
# define  NUM  10

int minimum( int x[], int n);

int main(void)
{
    int  i, engmin, mathmin;
    int eng [NUM] = { 80, 78, 93, 62, 55, 93, 72, 68, 74, 82};
    int math[NUM] = { 88, 92, 48, 62, 88, 93, 95, 68, 83, 75};

    engmin = minimum(eng, NUM);
    mathmin = minimum(math, NUM);
    printf("英語の最低点は%d です. \n", engmin);
    printf("数学の最低点は%d です. \n", mathmin);

    return 0;
}
int minimum(int x[], int n)        /* 要素数を引数とすることで, プログラムを一般化 */
{
    int i;
    int min;

    min = x[0];                    /* min を x[0] で初期化 しているため */
    for (i = 1; i < n; i++){       /* ループは, 1 からでよい */
        if (x[i] < min)
            min = x[i];
    }
    return min;
}
```

ポイント 3: 引数で渡された配列の内容を関数内で書き換えると、その変更はたとえ呼び出し側の関数に制御が戻っても保持される (引数で渡された変数を書き換えても、この値は保持されない)

example7_2.c

```
/*
 * 配列の全要素を 0 にする関数.
 * (呼び出された関数側で受け取った配列を変更すると, 元の配列にも影響が及ぶことを学ぶ.
 * 普通の引数の値渡しとの 違いを理解する. これを防止するためには,
 * void zeroset(int x[], int n) を void zeroset(const int x[], int n) とすれば良い.
 * こうすると代入する式 x[i] = 0  が翻訳時にエラーとなる.)
 */
# include <stdio.h>

void zeroset(int x[], int n);

int main(void)
{
    int  i;
    int ary1[] = {1, 2, 3, 4, 5};
    int ary2[] = {4, 3, 2, 1};

    for (i = 0; i < 5; i++)
        printf("ary1[%d] = %d \n", i, ary1[i]);
    for (i = 0; i < 4; i++)
        printf("ary2[%d] = %d \n", i, ary2[i]);

    zeroset(ary1, 5);
    zeroset(ary2, 4);

    for (i = 0; i < 5; i++)
        printf("ary1[%d] = %d \n", i, ary1[i]);
    for (i = 0; i < 4; i++)
        printf("ary2[%d] = %d \n", i, ary2[i]);

    return 0;
}
void zeroset(int x[], int n)
{
    int i;
    for(i = 0; i < n; i++)
        x[i] = 0;
}
```

ポイント 4: const 修飾子を用いると、代入によるデータの破壊を制限できる

example7_3.c

```
/*
 * 配列に入力した要素から，目的の値をもつ要素を探索するプログラム.
 * （逐次探索の実験.）
 */
# include <stdio.h>
#define NUM 5
#define FAIL -1

int search(const int x[], int ky,  int n);

int main(void)
{
    int  i, key, index;
    int  data[NUM];

    for(i = 0; i < NUM; i++){
        printf("data[%d] : ", i);
        scanf("%d", &data[i]);
    }
    printf("探索したい値 : ");
    scanf("%d", &key);

    index = search( data, key, NUM);

    if( index == FAIL)
         printf("探索失敗！");
    else
         printf("%d は%d 番目のデータです. \n", key,  index+1);

    return 0;
}
int search(const int x[], int ky,  int n)
/* 配列 x を const 宣言しているために，この関数内で配列 x[] に値を代入することはできない*/
{
    int i =0;
    while(1) {
        if (i == n)
            return FAIL;
        if (x[i] == ky)
            return i;
        i++;
    }
}
```

ポイント 5: ２次元配列を関数の引数として関数に渡す場合は、その要素数を省略できない。(正確には、1 番目の要素数は省略できるが、2 番目の要素数は省略できない。多次元配列の場合、先頭の要素数のみ省略できる。)

example7_4.c

```
/*
 * ２行３列の行列 L と３行２列の行列 M の積を２行２列の行列 N に格納する関数を用いたプログラム
 * なお行列の値は任意の整数をあらかじめ与えられている.（多次元配列の受け渡し）
 */
# include <stdio.h>

void mul( int x[2][3], int y[3][2], int z[2][2]);

int main(void)
{
    int  i, j;
    int  l[2][3] = { {1, 2, 3}, {4, 5, 6} };
    int  m[3][2] = { {7, 8}, {9, 1}, {2, 3} };
    int  n[2][2];

    mul(l, m, n);

    for (i = 0; i < 2; i++){
        for (j = 0; j < 2; j++)
            printf("%3d", n[i][j]);
        printf("\n");
    }
    return 0;
}

void mul( int x[2][3], int y[3][2], int z[2][2])
{
    int  i, j, k;
    for (i = 0; i < 2; i++){
        for (j = 0; j < 2; j++){
            z[i][j] = 0;
            for (k = 0; k < 3; k++)
               z[i][j] += x[i][k] * y[k][j];
        }
    }
}
```