

プログラミング演習I

配列

```
/*
 * 5人の学生の点数を読み込んで合計点と平均点を表示
 */

#include <stdio.h>

int main(void)
{
    int uchida;          /* 内田君の点数 */
    int satoh;           /* 佐藤君の点数 */
    int sanaka;          /* 佐中君の点数 */
    int hiraki;          /* 平木君の点数 */
    int masaki;          /* 真崎君の点数 */
    int sum = 0;         /* 合計点 */

    printf("5人の点数を入力してください。 \n");
    printf(" 1番 : ");   scanf("%d", &uchida);   sum += uchida;
    printf(" 2番 : ");   scanf("%d", &satoh);    sum += satoh;
    printf(" 3番 : ");   scanf("%d", &sanaka);    sum += sanaka;
    printf(" 4番 : ");   scanf("%d", &hiraki);    sum += hiraki;
    printf(" 5番 : ");   scanf("%d", &masaki);    sum += masaki;

    printf("合計点 : %5d\n", sum);
    printf("平均点 : %5.1f\n", (double)sum / 5);

    return 0;
}
```

実行例

5人の点数を入力してください。

1番 : 83 2番 : 95 3番 : 85 4番 : 63 5番 : 89

合計点 : 415

平均点 : 83.0

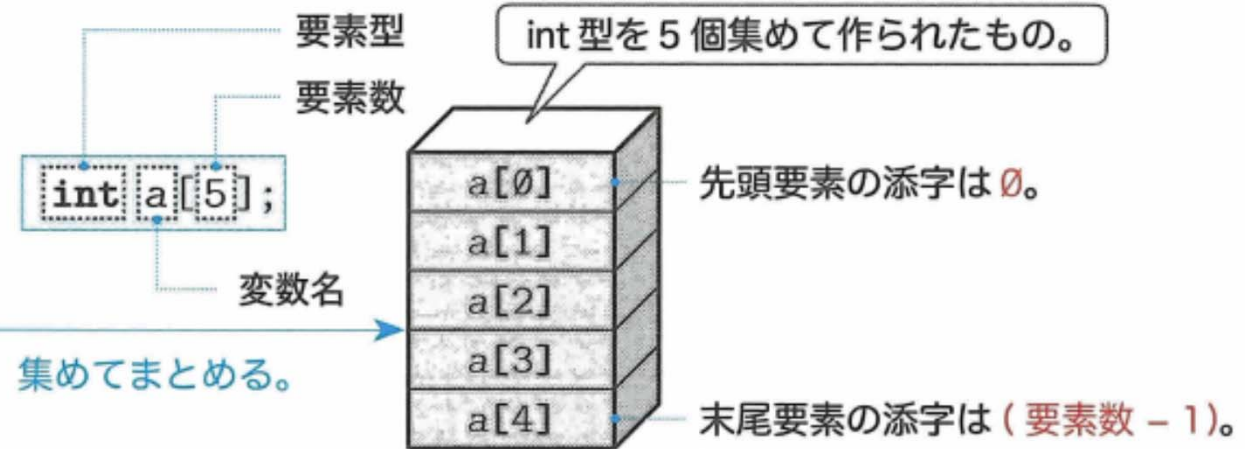
+= は左辺に右辺を加える
複合代入演算子。

配列(array)とは

a 変数のよせ集め



b 配列



● Fig.5-1 配列

注意！！ C言語では配列の添え字は0から始まる。
(覚えていますか？ FOR文の開始は1からではなく、0からのほうが相性が良いといいましたね。実は配列が0から始まるからです。なぜ0から始まるかは、プログラミング演習IIで触れられると思います。)

配列とFOR文(1)

List 5-2

chap05/list0502.c

```
/*
  配列の各要素に先頭から順に1,2,3,4,5を代入して表示
*/
#include <stdio.h>

int main(void)
{
    int v[5];      /* 要素型がint型で要素数が5の配列。 */

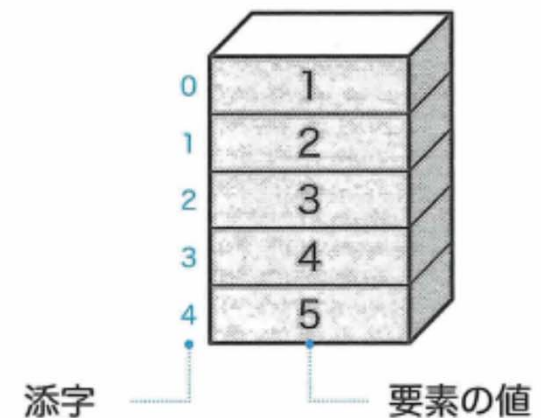
    v[0] = 1;
    v[1] = 2;
    v[2] = 3;
    v[3] = 4;
    v[4] = 5;

    printf("v[0] = %d\n", v[0]);
    printf("v[1] = %d\n", v[1]);
    printf("v[2] = %d\n", v[2]);
    printf("v[3] = %d\n", v[3]);
    printf("v[4] = %d\n", v[4]);

    return 0;
}
```

実行結果

```
v[0] = 1
v[1] = 2
v[2] = 3
v[3] = 4
v[4] = 5
```



● Fig.5-2 添字と要素の値

配列とFOR文(2)

List 5-3

chap05/list0503.c

```
/*
  配列の各要素に先頭から順に1,2,3,4,5を代入して表示 (for文)
*/


#include <stdio.h>

int main(void)
{
    int i;
    int v[5];    /* int[5]型の配列 */

    for (i = 0; i < 5; i++)    /* 要素に値を代入 */
        v[i] = i + 1;

    for (i = 0; i < 5; i++)    /* 要素の値を表示 */
        printf("v[%d] = %d\n", i, v[i]);

    return 0;
}
```



実行結果

v[0]	=	1
v[1]	=	2
v[2]	=	3
v[3]	=	4
v[4]	=	5

なお、配列の要素を一つずつ順番になぞっていくことを^{そうさ}**走査** (traverse) といいます。

一般に、要素型がTypeである配列のことを“**Typeの配列**”と呼びます。これまでのプログラムの配列は、すべて“**intの配列**”です。

練習

List 5-3

chap05/list0503.c

```
/*
 配列の各要素に先頭から順に1,2,3,4,5を代入して表示 (for文)
*/

#include <stdio.h>

int main(void)
{
    int i;
    int v[5]; /* int[5]型の配列 */

    for (i = 0; i < 5; i++)
        v[i] = i + 1;

    for (i = 0; i < 5; i++) /* 要素の値を表示 */
        printf("v[%d] = %d\n", i, v[i]);

    return 0;
}
```

解答1: $v[i]=i$;
解答2: $v[i]=5-i$;

要素の値
添字

実行結果

```
v[0] = 1
v[1] = 2
v[2] = 3
v[3] = 4
v[4] = 5
```

演習 5-1

List 5-3 を書きかえて、先頭から順に 0, 1, 2, 3, 4 を代入するプログラムを作成せよ。

演習 5-2

List 5-3 を書きかえて、先頭から順に 5, 4, 3, 2, 1 を代入するプログラムを作成せよ。

配列の初期化

- 配列の宣言時に、初期化が行われるかどうかは、いろいろややこしいところがある(変数と同様)。従って通常は、配列(変数)は初期化されておらず、なにが入っているかわからないと考えること。
- 初期化は明示的に行うことができる。

List 5-5

chap05/list0505.c

```
/*
 * 配列の各要素を先頭から順に1,2,3,4,5で初期化して表示
 */
#include <stdio.h>

int main(void)
{
    int i;
    int v[5] = {1, 2, 3, 4, 5};    /* 初期化 */

    for (i = 0; i < 5; i++)        /* 要素の値を表示 */
        printf("v[%d] = %d\n", i, v[i]);

    return 0;
}
```

実行結果

```
v[0] = 1
v[1] = 2
v[2] = 3
v[3] = 4
v[4] = 5
```

配列操作の練習(並べ替え)

List 5-8

chap05/list0508.c

```
/*  
  配列の全要素の並びを反転する  
*/
```

値の交換の仕方

(1) aの値をtempに代入

(2) aにbの値を代入

(3) 保存していたtempの値をbに代入

```
    printf("x[%d] : ", i);  
    scanf("%d", &x[i]);  
}
```

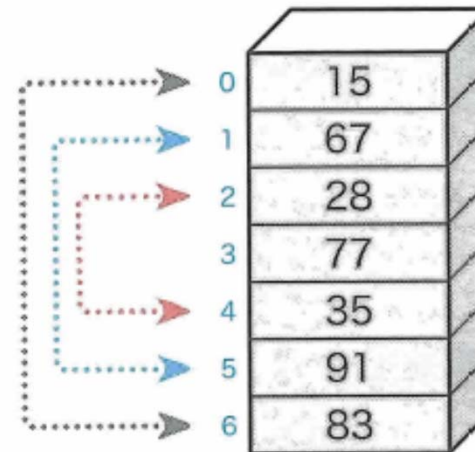
```
for (i = 0; i < 3; i++) { /* 要  
    int temp = x[i];  
    x[i]      = x[6 - i];  
    x[6 - i] = temp;  
}
```

```
puts("反転しました。");  
for (i = 0; i < 7; i++) /* 要素の値を  
    printf("x[%d] = %d\n", i, x[i]);
```

```
return 0;
```

```
}
```

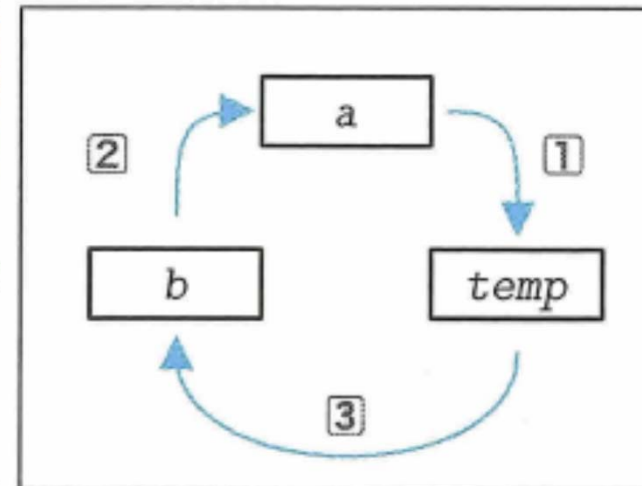
交換



実行例

```
x[0] : 15  
x[1] : 67  
x[2] : 28  
x[3] : 77  
x[4] : 35  
x[5] : 91  
x[6] : 83  
反転しました。  
x[0] = 83  
x[1] = 91  
x[2] = 35  
x[3] = 77  
x[4] = 28
```

● Fig.



配列による成績処理

List 5-9

chap05/list0509.c

```
/*
 * 5人の学生の点数を読み込んで合計点と平均点を表示
 */
#include <stdio.h>

int main(void)
{
    int i;
    int tensu[5];
    int sum = 0;

    printf("5人の点数を入力してください。 \n");
    for (i = 0; i < 5; i++) {
        printf("%2d番: ", i + 1);
        scanf("%d", &tensu[i]);
        sum += tensu[i];
    }

    printf("合計点: %5d \n", sum);
    printf("平均点: %5.1f \n", (double)sum / 5);

    return 0;
}
```

tensu[8];

8

8

5

5

8

実行例

5人の点数を入力してください。

1番: 83

2番: 95

3番: 85

4番: 63

5番: 89

合計点: 415

平均点: 83.0

5 ... 学生の人数

5 ... 表示の桁数

たった、要素が5から、8に変わっただけで、なんでこれだけ書き換えなければならないのか？ バグの温床になる。

読みやすく、保守性の高いプログラムへ

List 5-9

chap05/list0509.c

```
/*
 * 5人の学生の点数を読み込んで合計点と平均点を表示
 */
#include <stdio.h>

int main(void)
{
    int i;
    int tensu[5];          /* 5人の学生の点数 */
    int sum = 0;           /* 合計点 */

    printf("5人の点数を入力してください。 \n");
    for (i = 0; i < 5; i++) {
        printf("%2d番: ", i + 1);
        scanf("%d", &tensu[i]);
        sum += tensu[i];
    }

    printf("合計点: %5d\n", sum);
    printf("平均点: %5.1f\n", (double)sum / 5);

    return 0;
}
```

実行例

5人の点数を入力してください。

1番: 83

2番: 95

3番: 85

4番: 63

5番: 89

合計点: 415

平均点: 83.0

5 ... 学生の人数

5 ... 表示の桁数

#define 文

List 5-10

chap05/list0510.c

```
/*
 * 学生の点数を読み込んで合計点と平均点を表示（人数をマクロで定義）
 */
#include <stdio.h>

#define NUMBER 5 /* 学生の人数 */

int main(void)
{
    int i;
    int tensu[NUMBER]; /* NUMBER人の学生の点数 */
    int sum = 0; /* 合計点 */

    printf("%d人の点数を入力してください。 \n", NUMBER);
    for (i = 0; i < NUMBER; i++) {
        printf("%2d番：", i + 1);
        scanf("%d", &tensu[i]);
        sum += tensu[i];
    }

    printf("合計点： %5d\n", sum);
    printf("平均点： %5.1f\n", (double)sum / NUMBER);

    return 0;
}
```

コメントも重要！！！！

5人の点数を入力してください。
1番： 83
2番： 95
3番： 85
4番： 63
5番： 89
合計点： 415
平均点： 83.0

NUMBER ... 翻訳時に5に置換される。

理想的には、プログラム本文中には数字は書かないこと。ただし、初期化の意味での0と、最小単位増加するという意味で、1は使うことはOK

プログラムを読む

- LIST5－12を読んでみてください。
- プログラムの読みかた
 - － 上から下に読む（当たり前）
 - プログラムの構造を考える。最小構造毎、読む。
 - － 多少わからない変数などがあっても流し読みする。
 - 流し読みできるプログラムは綺麗
 - － プログラムにはパターンがある。
 - できる限りたくさんのプログラムを読むこと
 - 今は、OpenSourceの時代。綺麗で品質の良いプログラムのソースが無料で読める良い時代となった。

LIST5-12

```
#define NUMBER 80      /* 人数の上限 */
```

```
int main(void)
{
```

```
    int i, j;
    int num;                /* 実際の人数 */
    int tensu[NUMBER];      /* 学生の点数 */
    int bunpu[11] = {0};    /* 点数の分布 */
```

```
    printf("人数を入力してください：");
```

```
    do {
        scanf("%d", &num);
        if (num < 1 || num > NUMBER)
            printf("\a1～%dで入力してください：", NUMBER);
    } while (num < 1 || num > NUMBER);
```

```
    printf("%d人の点数を入力してください。 \n", num);
```

```
    for (i = 0; i < num; i++) {
        printf("%2d番：", i + 1);
```

```
        do {
            scanf("%d", &tensu[i]);
            if (tensu[i] < 0 || tensu[i] > 100)
                printf("\a1～100で入力してください：");
        } while (tensu[i] < 0 || tensu[i] > 100);
        bunpu[tensu[i] / 10]++;
    }
```

読み込む値を1～NUMBERに
制限するためのdo文。

読み込む値を1～100に
制限するためのdo文。

1番：17
2番：38
3番：100
4番：95
5番：23
6番：62
7番：77
8番：45
9番：69
10番：81
11番：83
12番：51
13番：42
14番：36
15番：60

---分布グラフ---

100：*
90 ～ 99：*
80 ～ 89：**
70 ～ 79：*
60 ～ 69：***
50 ～ 59：*
40 ～ 49：**
30 ～ 39：**
20 ～ 29：*
10 ～ 19：*
0 ～ 9：


```

} while (num < 1 || num > NUMBER);

printf("%d人の点数を入力してください。\\n", num);

for (i = 0; i < num; i++) {
    printf("%2d番 :", i + 1);
    do {
        scanf("%d", &tensu[i]);
        if (tensu[i] < 0 || tensu[i] > 100)
            printf("\\a1~100で入力してください : ");
    } while (tensu[i] < 0 || tensu[i] > 100);
    bunpu[tensu[i] / 10]++;
}

```

読み込む値を1~100に
制限するためのdo文。

---分布グラフ---

```

100 : *
90 ~ 99 : *
80 ~ 89 : **
70 ~ 79 : *
60 ~ 69 : ***
50 ~ 59 : *
40 ~ 49 : **
30 ~ 39 : **
20 ~ 29 : *
10 ~ 19 : *
0 ~ 9 :

```

```

puts("\\n---分布グラフ---");
printf("      100 : ");

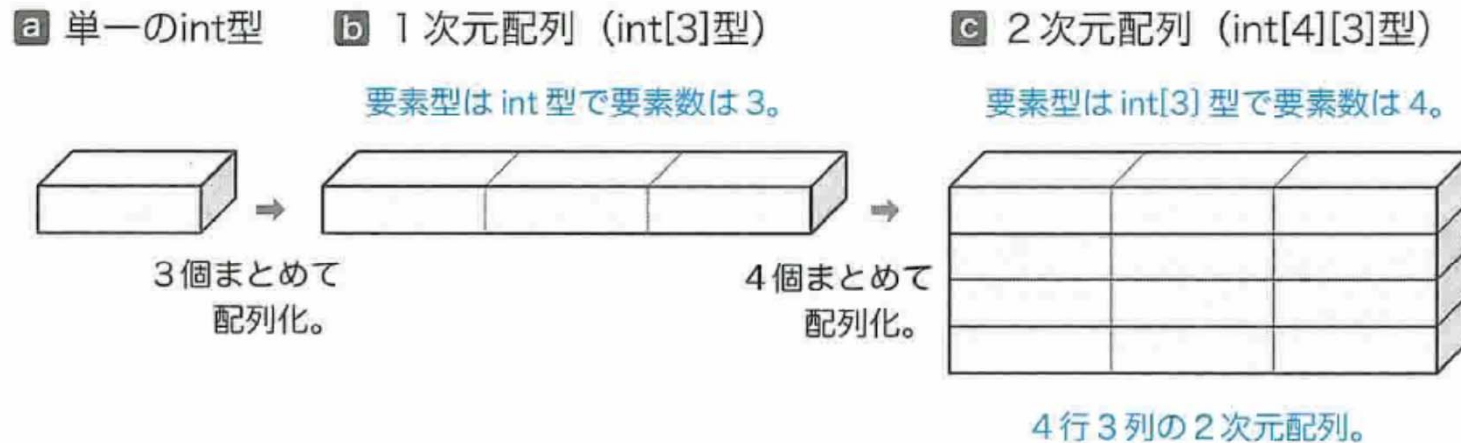
for (j = 0; j < bunpu[10]; j++) /* 100点 */
    putchar('*');
putchar('\\n');

for (i = 9; i >= 0; i--) { /* 100点未満 */
    printf("%3d ~%3d :", i * 10, i * 10 + 9);
    for (j = 0; j < bunpu[i]; j++)
        putchar('*');
    putchar('\\n');
}

return 0;

```

2次元配列

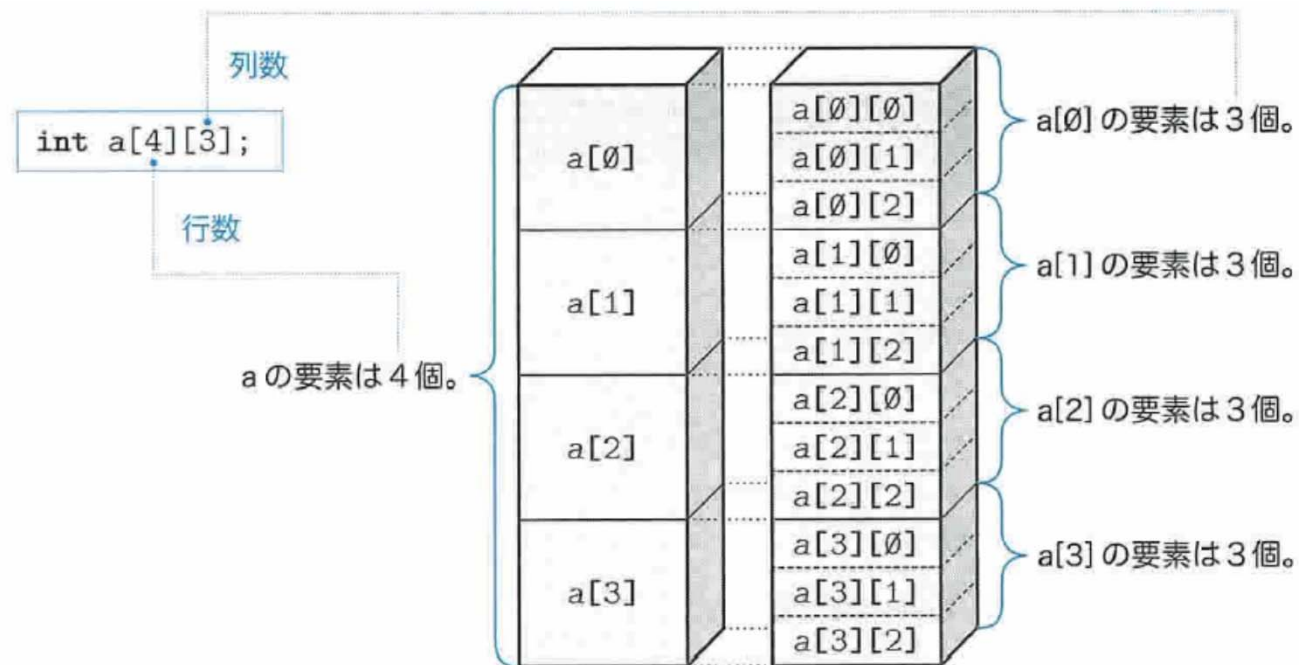


● Fig.5-8 1次元配列と2次元配列の派生

- **a** ⇨ **b** : int 型をまとめて1次元配列を派生 (ここでは3個集めています)。
- **b** ⇨ **c** : 1次元配列をまとめて2次元配列を派生 (ここでは4個集めています)。

それぞれの型は、以下ようになります。

- **a** : int 型
- **b** : int[3] 型 “int”を要素型とする要素数3の配列
- **c** : int[4][3] 型 《“int”を要素型とする要素数3の配列》を要素型とする要素数4の配列



● Fig.5-9 4行3列の2次元配列

1次元配列と同様、多次元配列の全要素／全構成要素は記憶域上に一直線に並びます。構成要素の並びでは、まず末尾側の添字が順に0, 1, ... と増えていき、それから先頭側の添字が0, 1, ... と増えていきます。すなわち、以下の順番です。

`a[0][0] a[0][1] a[0][2] a[1][0] a[1][1] a[1][2] ... a[3][1] a[3][2]`

そのため、たとえばa[0][2]の直後にa[1][0]が位置する、あるいは、a[2][2]の直後にa[3][0]が位置する、といったことが保証されます。


```

/*
 4人の学生の3科目のテスト2回分の合計を求めて表示
*/

#include <stdio.h>

int main(void)
{
    int i, j;
    int tensu1[4][3] = { {91, 63, 78}, {67, 72, 46}, {89, 34, 53}, {32, 54, 34} };
    int tensu2[4][3] = { {97, 67, 82}, {73, 43, 46}, {97, 56, 21}, {85, 46, 35} };
    int sum[4][3];          /* 合計 */

    /* 2回分の点数の合計を求める */
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 3; j++)
            sum[i][j] = tensu1[i][j] + tensu2[i][j];
    }
}

```

int[3] 型の要素に対する初期化子が4個あるため4は省略可能。
※省略した場合は自動的に4とみなされる。

int[3] 型の要素 tensu1[0] に対する初期化子。

/* 4人分の */
/* 3科目の */
/* 2回分を加算 */

		tensu1... 1回目の点数				tensu2... 2回目の点数				sum ... 合計		
1番の学生	0	91	63	78	0	97	67	82	0	188	130	160
2番の学生	1	67	72	46	1	73	43	46	1	140	115	92
3番の学生	2	89	34	53	2	97	56	21	2	186	90	74
4番の学生	3	32	54	34	3	85	46	35	3	117	100	69
		0	1	2		0	1	2		0	1	2
		国語 英語 数学				国語 英語 数学				国語 英語 数学		

● Fig.5-10 4行3列の2次元配列に格納されたテストの点数