

cp-7. 配列

(C プログラミング入門)

URL: <https://www.kkaneko.jp/cc/adp/index.html>

金子邦彦



内容

例題 1. 月の日数

配列とは. 配列の宣言. 配列の添え字.

例題 2. ベクトルの内積

例題 3. 合計点と平均点

例題 4. 棒グラフを描く

配列と繰り返し計算の関係

例題 5. 行列の和

2次元配列

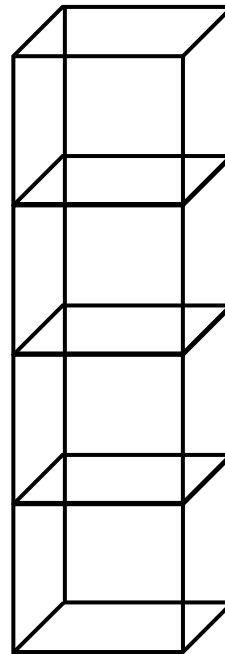
目標



- 配列とは何かを理解し, `int`, `double` を使った配列の宣言と使用ができるようになる
- 配列と繰り返し文を組み合わせて, 多量のデータを扱えるようになる

配列

添字



0

1

2

3

- データの並びで, 0 から始まる番号 (添字) が付いている

例題 1. 月の日数

- 年と月を読み込んで、日数を求めるプログラムを作る
 - うるう年の2月ならば29
 - 日数を求めるために、サイズ12の配列を使う

例) 2001年11月 → 30

月の日数

```
#include <stdio.h>
```

```
#pragma warning(disable:4996)
```

```
int main()
```

```
{
```

```
int num_days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

```
int y;
```

```
int m;
```

```
printf( "y=" );
```

```
scanf( "%d", &y );
```

```
printf( "m=" );
```

```
scanf( "%d", &m );
```

```
if ( (m == 2) && (((y % 400) == 0) || (((y % 100) != 0) && ((y % 4) == 0)))){
```

```
    printf( "number of days(%d) = 29¥n", m );
```

```
}
```

```
else {
```

```
    printf( "number of days(%d) = %d¥n", m, num_days[m-1] );
```

```
}
```

```
return 0;
```

```
}
```

} 配列の宣言

} 配列からの
読み出し

「m - 1」に意味がある

月の日数

実行結果の例

y=2001

m=11

number of days(11) = 30

メモリ

```
num_days[m-1];
```

配列からの値の
読み出し

num_days[0]	31
num_days[1]	28
num_days[2]	31
num_days[3]	30
num_days[4]	31
num_days[5]	30
num_days[6]	31
num_days[7]	31
num_days[8]	30
num_days[9]	31
num_days[10]	30
num_days[11]	31

配列の宣言

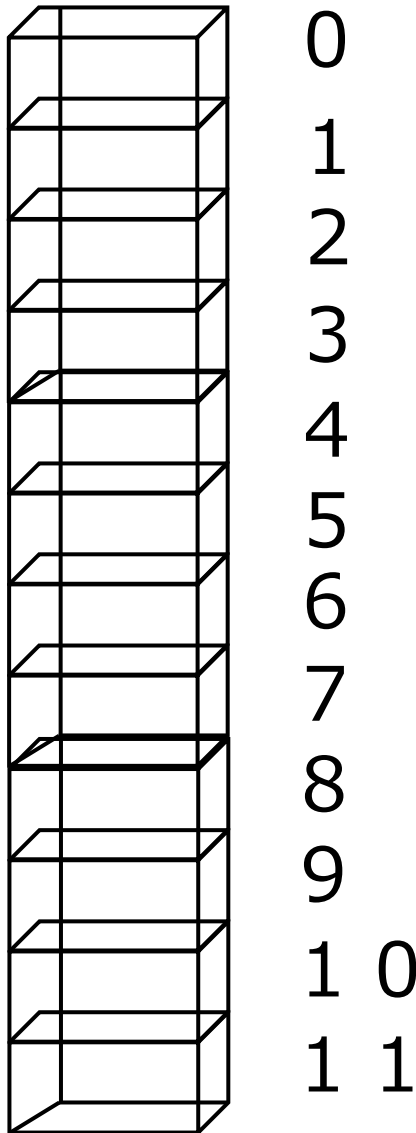
- 配列には、名前と型（データの種類のこと）とサイズがある
 - 整数データ int
 - 浮動小数データ double
- 配列を使うには、配列の使用をコンピュータに伝えること（宣言）が必要

```
int num_days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

整数 名前は
データ num_days

配列のサイズ
は 12

配列の添字



- 配列の中身を読み書きするときには、配列の名前と添字を書く

例) num_days[m-1]

これが添字

- 添字は0から(サイズ-1)まで

例) サイズ12の配列の
添字は、0から11まで

- 添字を付けて、普通の変数と同じように使う

例)

```
v[0]=1.3;
```

```
a=v[1];
```

```
printf("%f", v[2]);
```

```
scanf("%lf", &v[3]);
```

例題 2. ベクトルの内積

- ベクトル (1.9, 2.8, 3.7) と, ベクトル (4.6, 5.5, 6.4) の内積を表示するプログラムを作る
 - 2つのベクトルの内積の計算のために, サイズ3の配列を2つ使う

例題 2 . ベクトルの内積

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    double u[]={1.9, 2.8, 3.7};
```

```
    double v[]={4.6, 5.5, 6.4};
```

```
    int i;
```

```
    double ip;
```

```
    ip = 0;
```

```
    for (i=0; i<3; i++) {
```

```
        ip = ip + u[i]*v[i];
```

```
    }
```

```
    printf("内積=%f¥n", ip);
```

```
    return 0;
```

```
}
```

} 配列の宣言

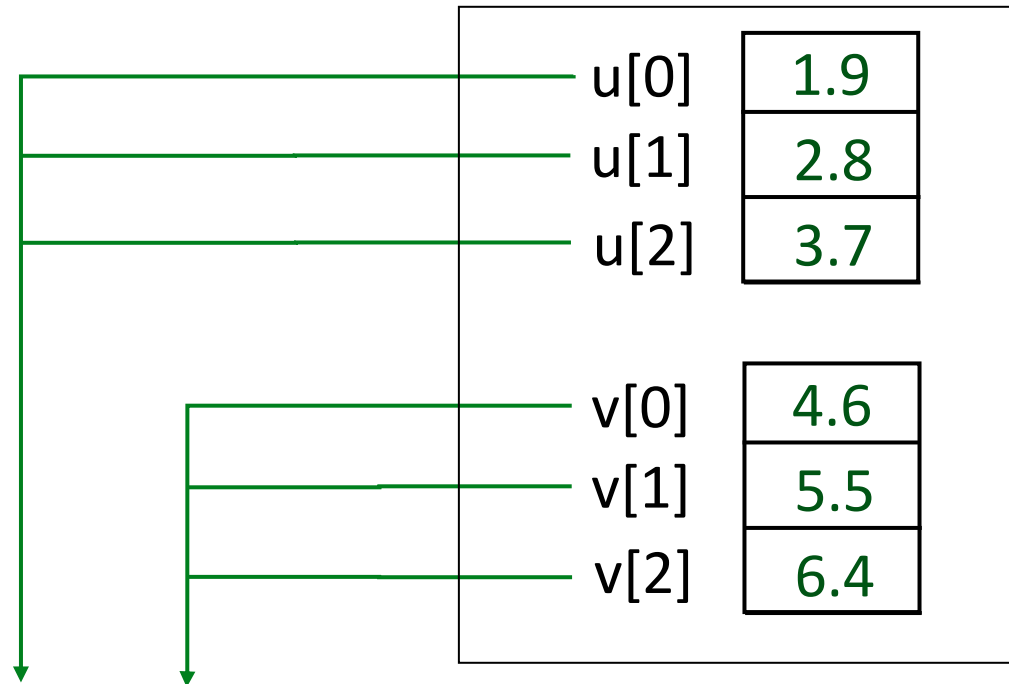
} 配列からの
読み出し

ベクトルの内積

実行結果の例

内積=47.820000

メモリ



```
ip = ip + u[i]*v[i];
```

配列からの値の
読み出し

配列の宣言

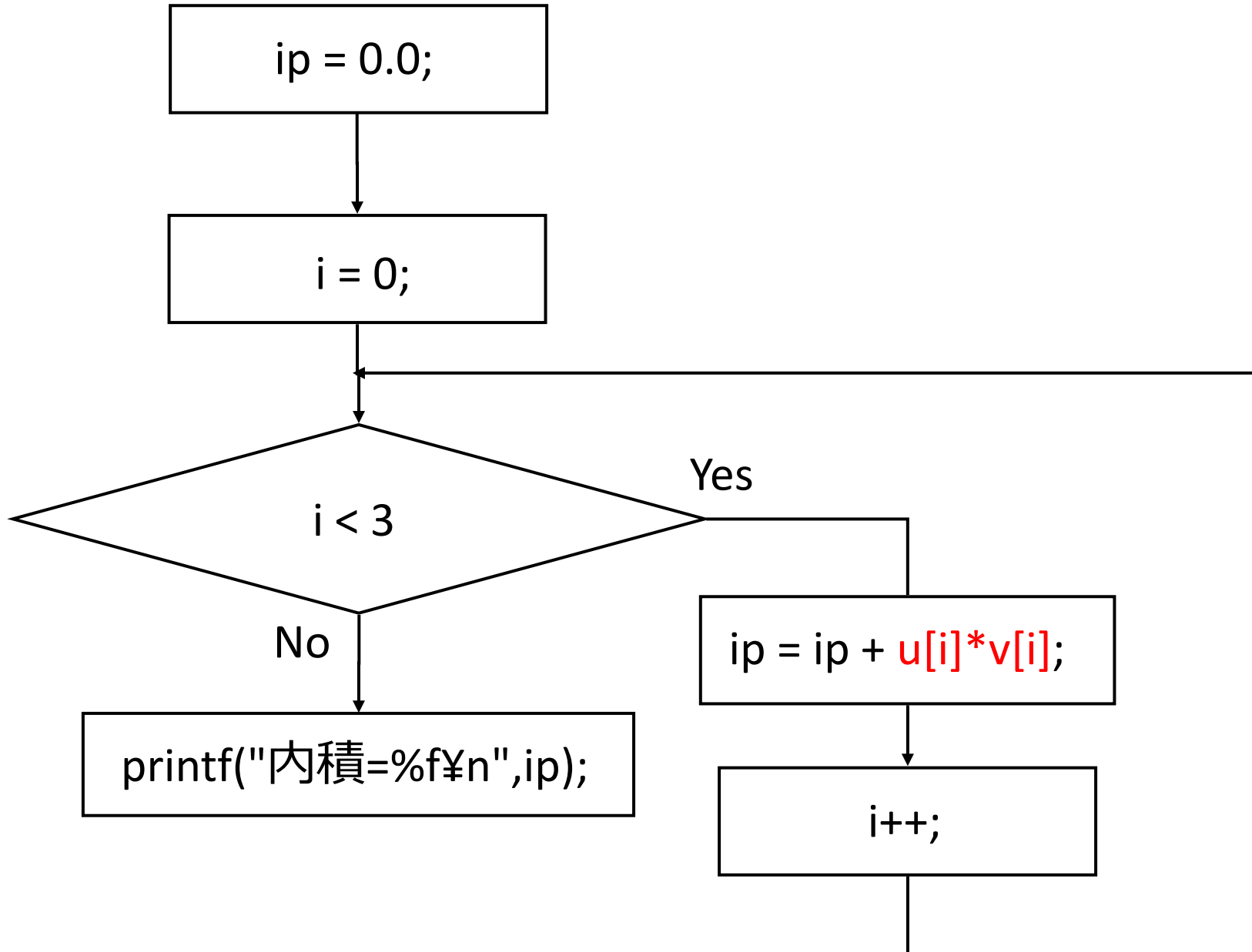
```
double u[]={1.9, 2.8, 3.7};  
double v[]={4.6, 5.5, 6.4};
```

浮動小数
データ

名前
uとv

配列のサイズ
は3

プログラム実行順



ベクトルの内積

	i の値	繰り返し条件式 が成り立つか	ip の値
繰り返し 1 回目	$i = 0$	$i < 3$ が成り立つ	$ip = ip + u[0] * v[0];$ つまり ip の値は $u[0]*v[0]$
繰り返し 2 回目	$i = 1$	$i < 3$ が成り立つ	$ip = ip + u[1] * v[1];$ つまり ip の値は $u[0]*v[0] + u[1]*v[1]$
繰り返し 3 回目	$i = 2$	$i < 3$ が成り立つ	$ip = ip + u[2] * v[2];$ つまり ip の値は $u[0]*v[0] + u[1]*v[1]$ $+u[2]*v[2]$
繰り返し 4 回目	$i = 3$	$i < 3$ が成り立たない	

例題 3 . 合計点と平均点

- 点数を読み込んで、合計点と平均点を表示するプログラムを作る。
 - 平均点の計算では、小数点以下切捨て
 - プログラムは、点数を読み込み続けるが、読み込んだ点数が「-1」のときには、合計点と平均点を表示して終了する
 - 読み込んだ点数は、いったんサイズ100の配列に格納する（点数の数は100を超えないものと仮定する）

例) 50, 85, 30, 20

合計点 185

平均点 46

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int x[100];
    int sum;
    int i;
    int n;
    n = 0;
    do {
        printf( "x[%d]=", n );
        scanf( "%d", &x[n] );
        n++;
    } while ( ( x[n-1] >= 0 ) && ( n < 100 ) );
    sum = 0;
    for (i=0; i<(n-1); i++) {
        sum = sum + x[i];
    }
    printf("合計=%d, 平均=%d¥n", sum, sum/(n-1) );
    return 0;
}
```

} 配列の宣言

合計点と平均点

実行結果の例

x[0]=50

x[1]=85

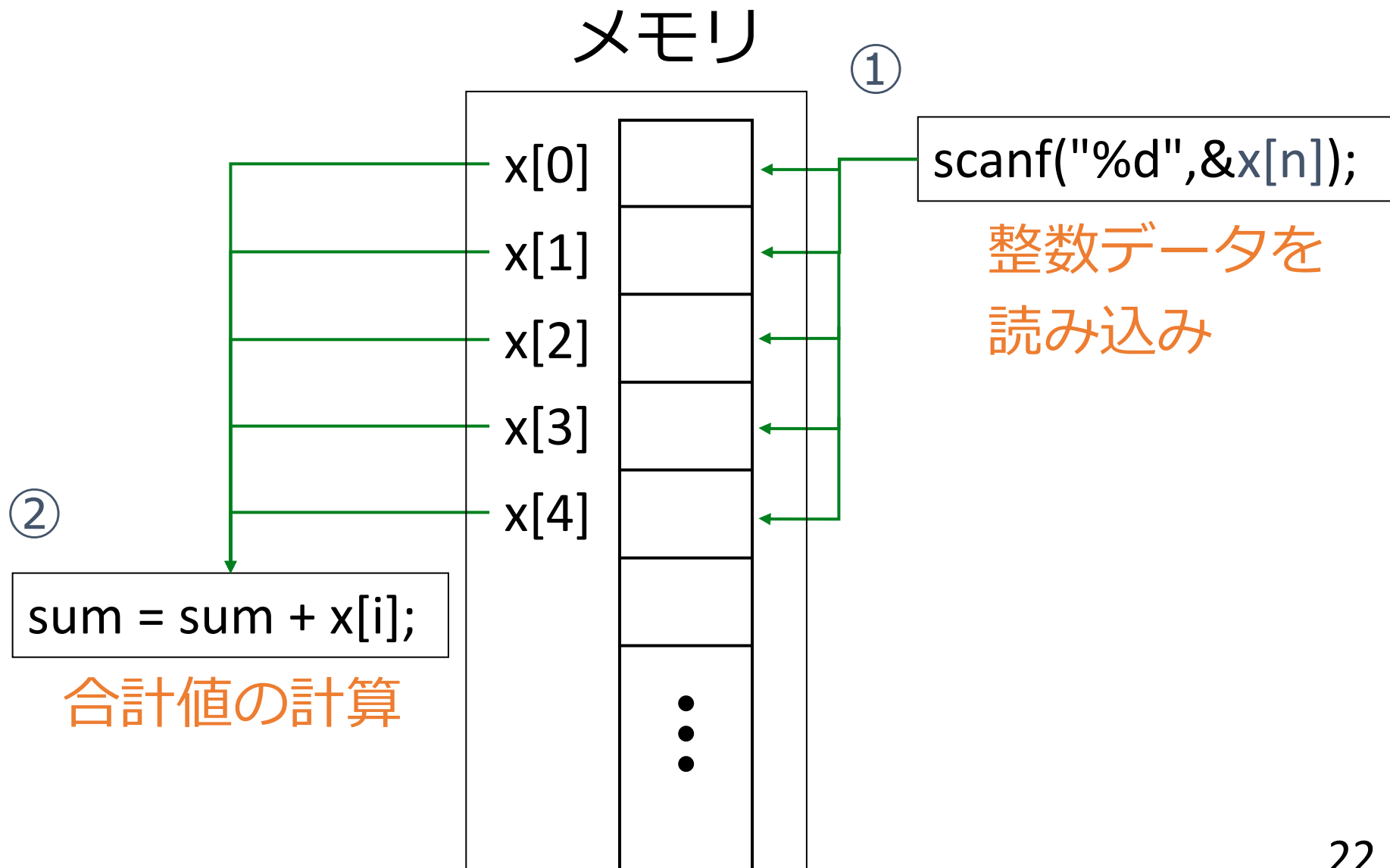
x[2]=30

x[3]=20

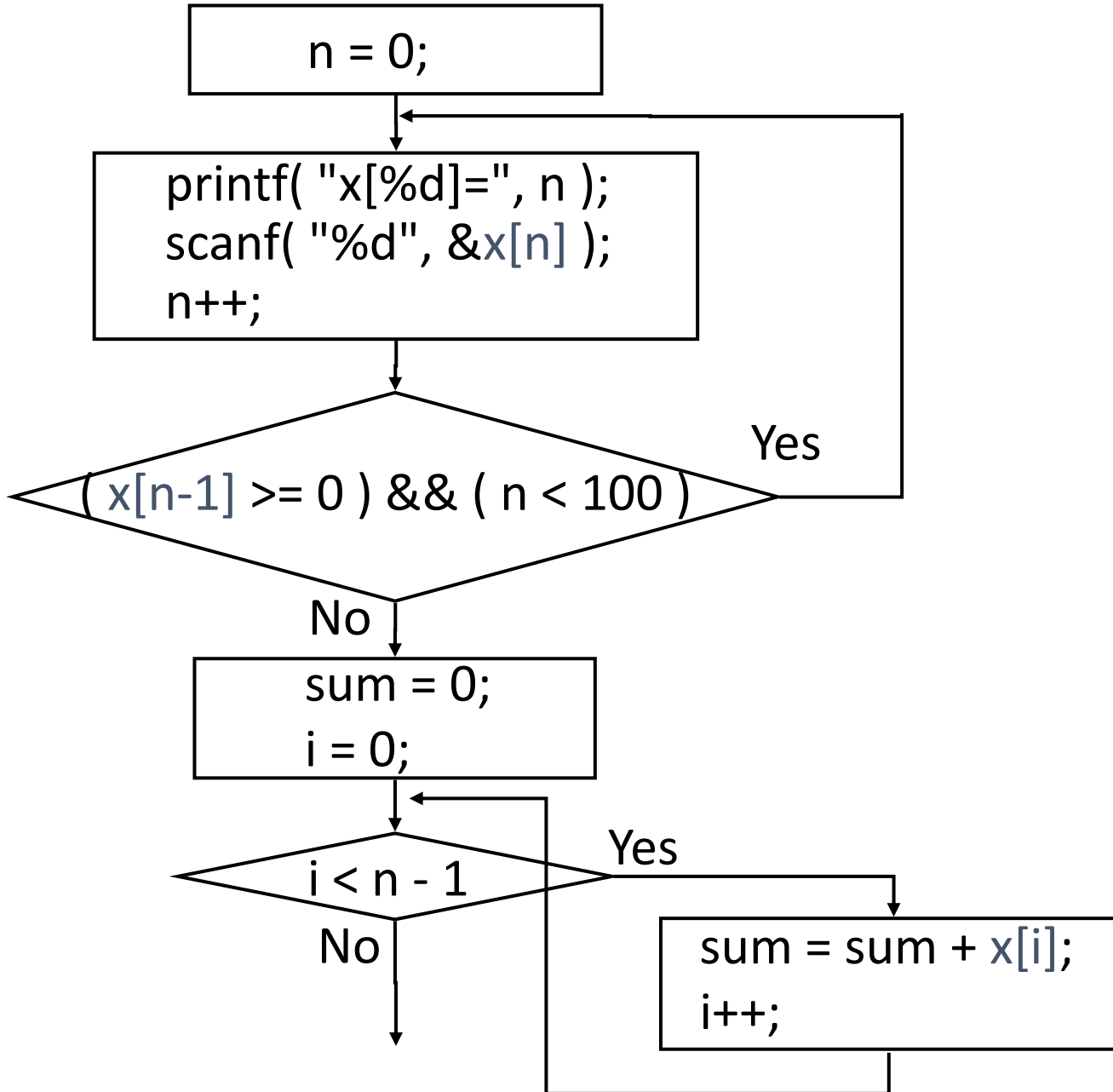
x[4]=-1

合計=185, 平均=46

プログラムとデータ



プログラム実行順



合計点と平均点 (5回目で「-1」を読み込んだとき)



n の値	繰り返し条件式 が成り立つか	読み込まれる 配列の要素
繰り返し 1回目	$n = 1$ $(x[0] > 0 \ \&\& \ n < 100)$ が成り立つ	$x[0]$
繰り返し 2回目	$n = 2$ $(x[1] > 0 \ \&\& \ n < 100)$ が成り立つ	$x[1]$
繰り返し 3回目	$n = 3$ $(x[2] > 0 \ \&\& \ n < 100)$ が成り立つ	$x[2]$
繰り返し 4回目	$n = 4$ $(x[3] > 0 \ \&\& \ n < 100)$ が成り立つ	$x[3]$
繰り返し 5回目	$n = 5$ $(x[4] > 0 \ \&\& \ n < 100)$ が成り立たない	$x[4]$

「-1」を読み込んだら
この部分が成り立たない

配列の宣言

1. 配列の宣言時にサイズを指定

例)

```
int x[100];
```

← サイズとして「100」
を書いている

2. 配列の宣言時に初期値を設定

例)

```
double u[]={1.9, 2.8, 3.7};
```

```
double v[]={4.6, 5.5, 6.4};
```

```
int a[]={6,4,7,1,5,3,2};
```

初期値を並べて
書いている

例題 4 . 棒グラフを描く

- 整数の配列から, その棒グラフを表示するプログラムを作る.
 - ループの入れ子で, 棒グラフの表示を行う

棒グラフを描く

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a[]={6,4,7,1,5,3,2};
```

```
    int i;
```

```
    int j;
```

```
    for (i=0; i<7; i++) {
```

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

```
            printf("*");
```

```
        }
```

```
        printf("¥n");
```

```
    }
```

```
    return 0;
```

```
}
```

} 配列の宣言

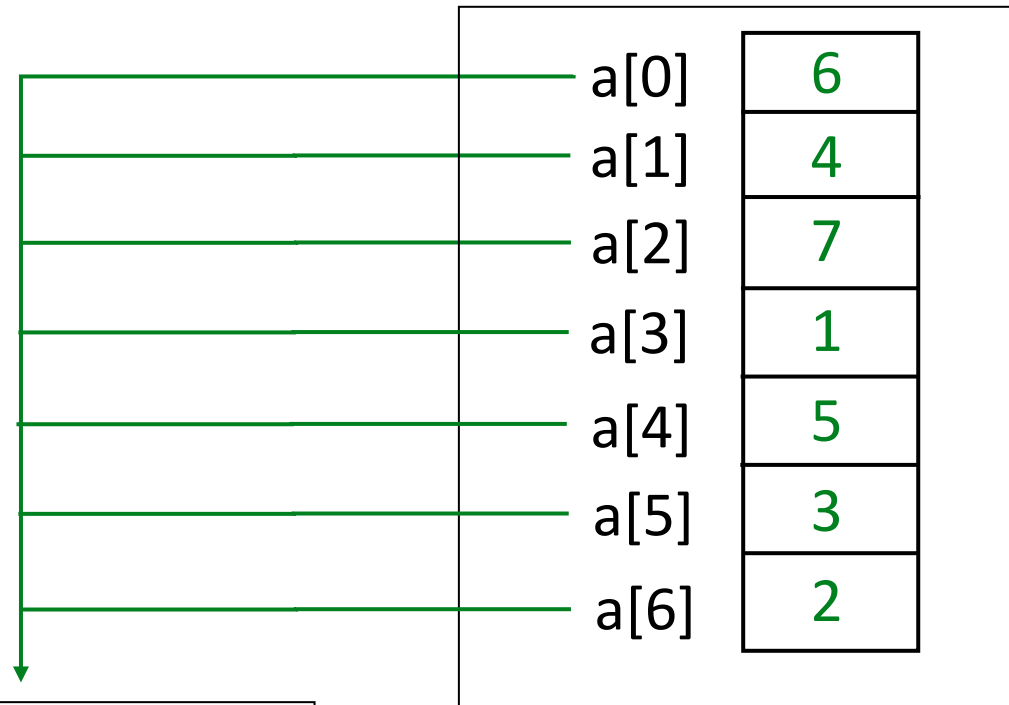
} 配列からの
読み出し

棒グラフを書く

実行結果の例

```
*****  
  
****  
  
*****  
  
*  
  
*****  
  
***  
  
**
```

メモリ



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

配列からの値の
読み出し

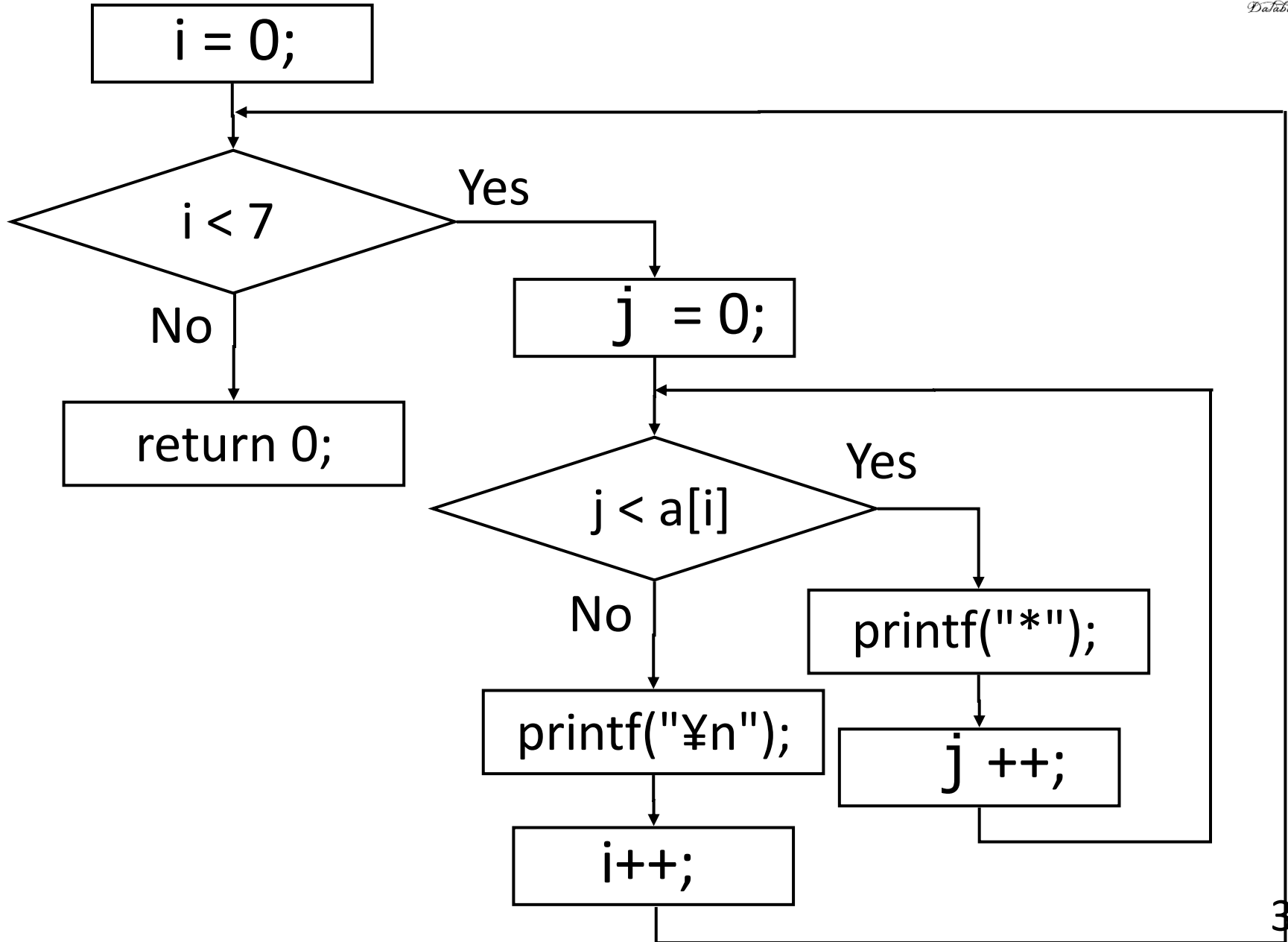
配列の宣言

```
int a[]={6,4,7,1,5,3,2};
```

整数 データ
名前が
a

配列のサイズ
は7

プログラム実行順



ここまでのまとめ

[] の意味

- 配列の宣言
 - [] の中に, 配列のサイズを書く. 但し, 配列の初期値を設定するときには「空」にする.
- 配列の使用
 - [] の中に, 使用したい配列の添字を書く

課題 1

- n次の多項式

$$f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_n \cdot x^n$$

について，次数 n と，係数 a_0 から a_n を読み込んで， $f(x)$ を計算するプログラムを作りなさい

- n は高々 20 までとする。（ユーザが 20 以上の数を n として与えたら，メッセージを表示して止まること）。
- 次ページで説明する Horner 法を使うこと
- 読み込んだ点数は，いったんサイズ 21 の配列に格納する
- x を繰り返し入力できるようなプログラムであること（つまり $f(x)$ を何度も計算する）

Horner法

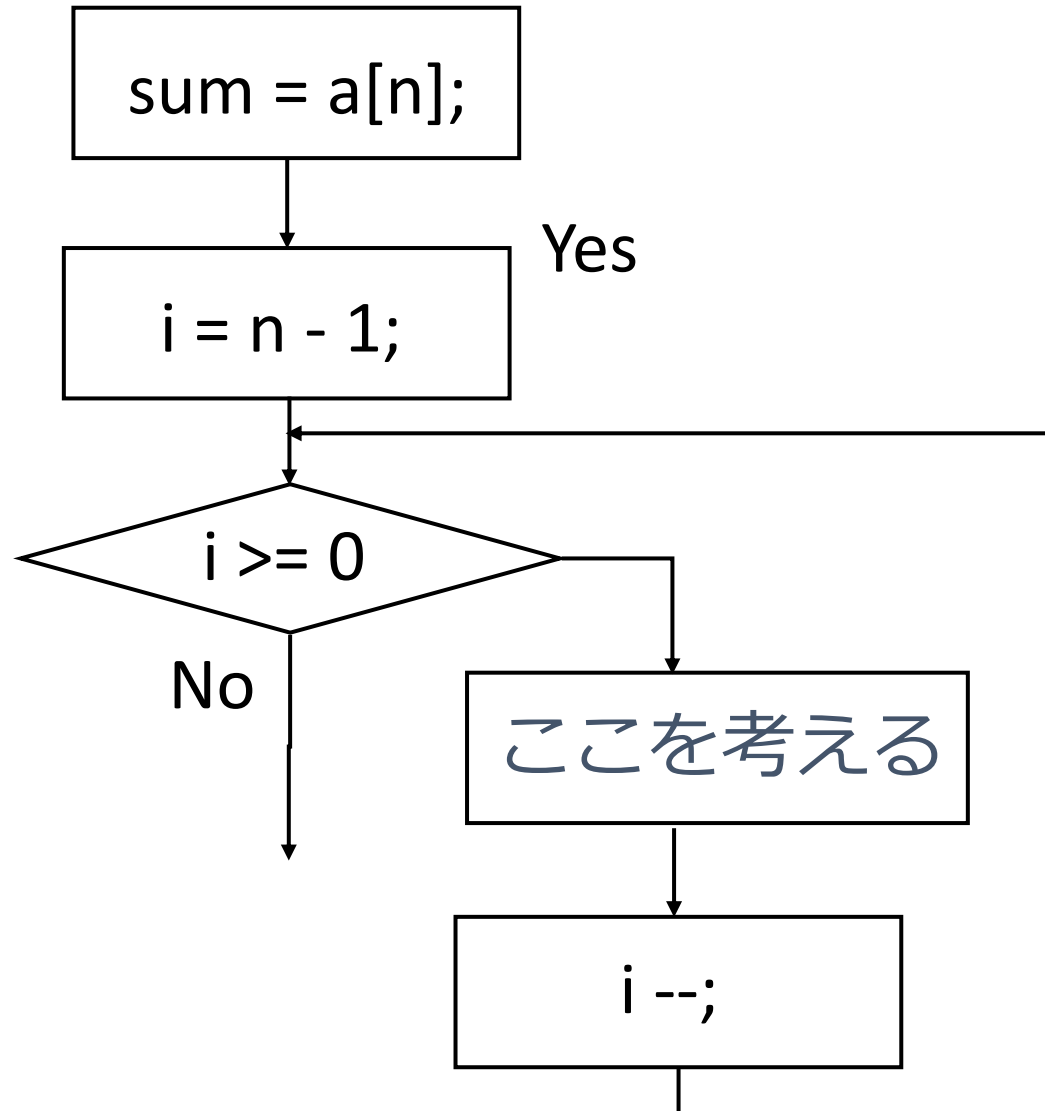
$$\begin{aligned} f(x) &= a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_n \cdot x^n \\ &= a_0 + (a_1 + (a_2 + \dots + (a_{n-1} + a_n \cdot x) x \dots) x) x \end{aligned}$$

例えば, $5 + 6x + 3x^2$
 $= 5 + (6 + 3x) x^2$

計算手順

- ① a_n
- ② $a_{n-1} + a_n \cdot x$
- ③ $a_{n-2} + (a_{n-1} + a_n \cdot x) x$
- • • (a_n まで続ける)

課題 1 のヒント



課題 2. エラトステネスのふるい

- 「エラトステネスのふるい」の原理に基づいて100以下の素数を求め、結果を表示するプログラムを作成せよ。
 - 100以下の素数を求めるために、サイズ100の配列を使う
 - 添字が素数の要素に1, そうでない要素に0を代入する

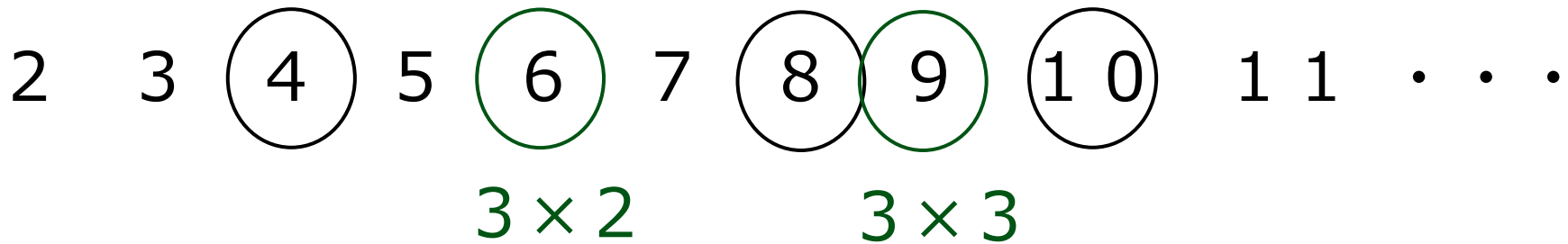
エラトステネスのふるい (1/4)

2 3 4 5 6 7 8 9 10 11 . . .

2×2 2×3 2×4 2×5

まず、2の倍数を消す

エラトステネスのふるい (2/4)



次に, 3の倍数を消す

エラトステネスのふるい (3/4)

2 3 4 5 6 7 8 9 10 11 ...

5 × 2

次に, 5の倍数を消す

(「4の倍数」は考えない.

それは, 「4」がすでに消えているから)

エラトステネスのふるい (4/4)

2 3 (4) 5 (6) 7 (8) (9) (10) 11 ...

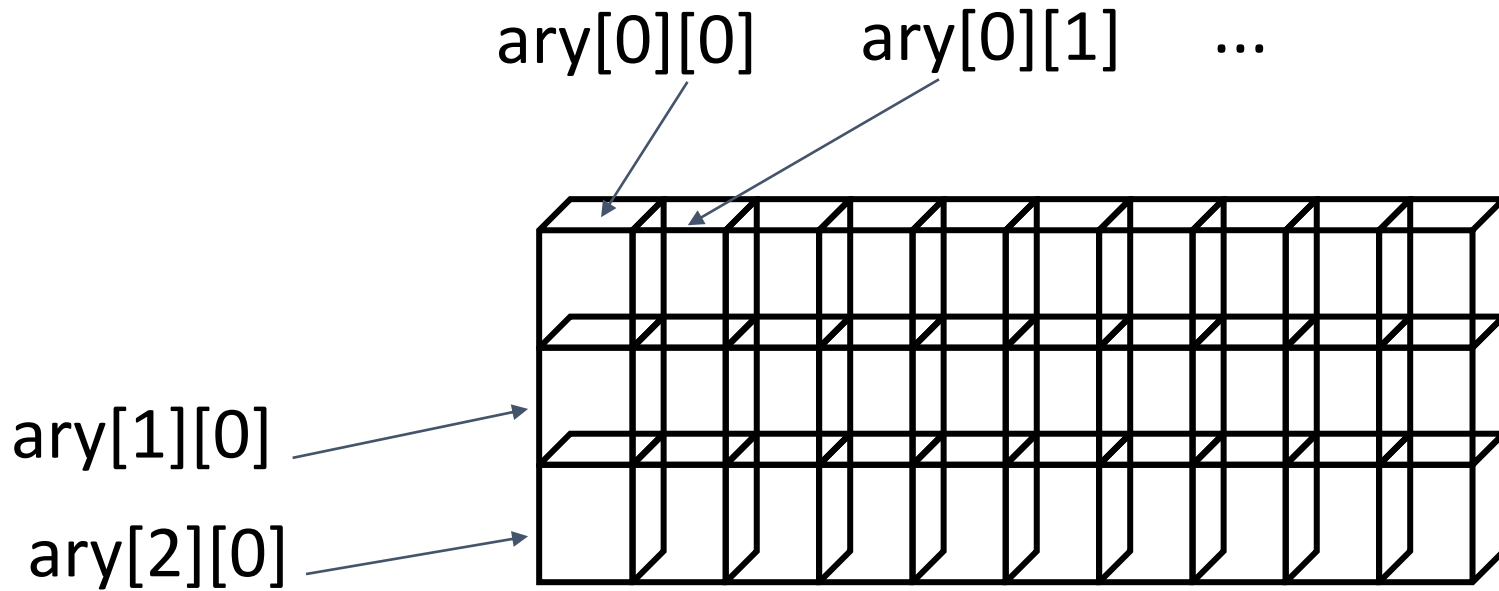
以上のように, 2, 3, 5, 7の倍数を消す.

10 (これは100の平方根) を超えたら, この操作を止める
(100以下で, 11, 13...の倍数はすでに消えている)

例題 5 . 行列の和

- 2行3列の行列の和を計算して表示するプログラムを作る.
 - 2つの行列を扱うために, 2行3列の2次元配列を2つ使う
 - a 2行3列の2次元配列 浮動小数
 - b 2行3列の2次元配列 浮動小数

2次元配列とは



行列の和

```
#include <stdio.h>
int main()
{
    int a[2][3]={{1,2,3},{4,5,6}};
    int b[2][3]={{9,8,7},{6,5,4}};
    int i;
    int j;
    for (i=0; i<2; i++) {
        for (j=0; j<3; j++) {
            printf("%d, ", a[i][j]+b[i][j]);
        }
        printf("¥n");
    }
    return 0;
}
```

} 配列の宣言

} 配列からの
読み出し

行列の和

実行結果の例

```
10, 10, 10,  
10, 10, 10,
```

2次元配列の宣言

- 2次元配列の宣言では、名前、型、サイズを指定することが必要

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

↑
整数
データ

↑
名前は
a

↑
配列のサイズ
は2×3

2次元以上の配列の宣言では、サイズを書く
(省略できない) ことになっている

課題 3



- 3×3行列の積を計算して表示するプログラムを作成しなさい
- 2つの行列を扱うために, 3行3列の2次元配列を2つ使う

多次元配列についての補足

- 多次元配列は、普通、最大7次元まで（処理系により異なる）。

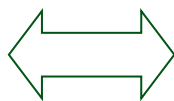
例) 整数型の3次元配列

```
int a[2][3][4];
```

変数の初期化

- 変数を宣言すると同時に、値の設定もできます。

```
int i;  
i = 0;
```



```
int i = 0;
```

同じ意味