プログラミング入門
第8回講義

- 配列（その2）
  - 2次元配列
  - 配列宣言と構造
  - 配列の操作

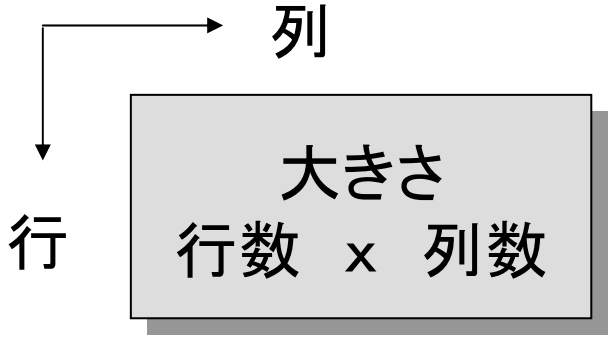
マークのあるサンプルプログラムは
`/home/course/prog0/public_html/2006/lec/source/`
下に置いてありますから、各自自分のディレクトリに
コピーして、コンパイル・実行してみてください

２次元配列の宣言と構造(p.228)

どこで宣言　→　プログラムの先頭部分

どのように　→　まず　配列名　を決める

次に　サイズを指定



配列名[行数][列数]

行数、列数
どちらも
整数型定数

2次元配列の宣言と構造(2)

```
int data[3][5];
```

3行5列の
int型
2次元配列

全体は配列
data

列添字は0から
列数－1まで

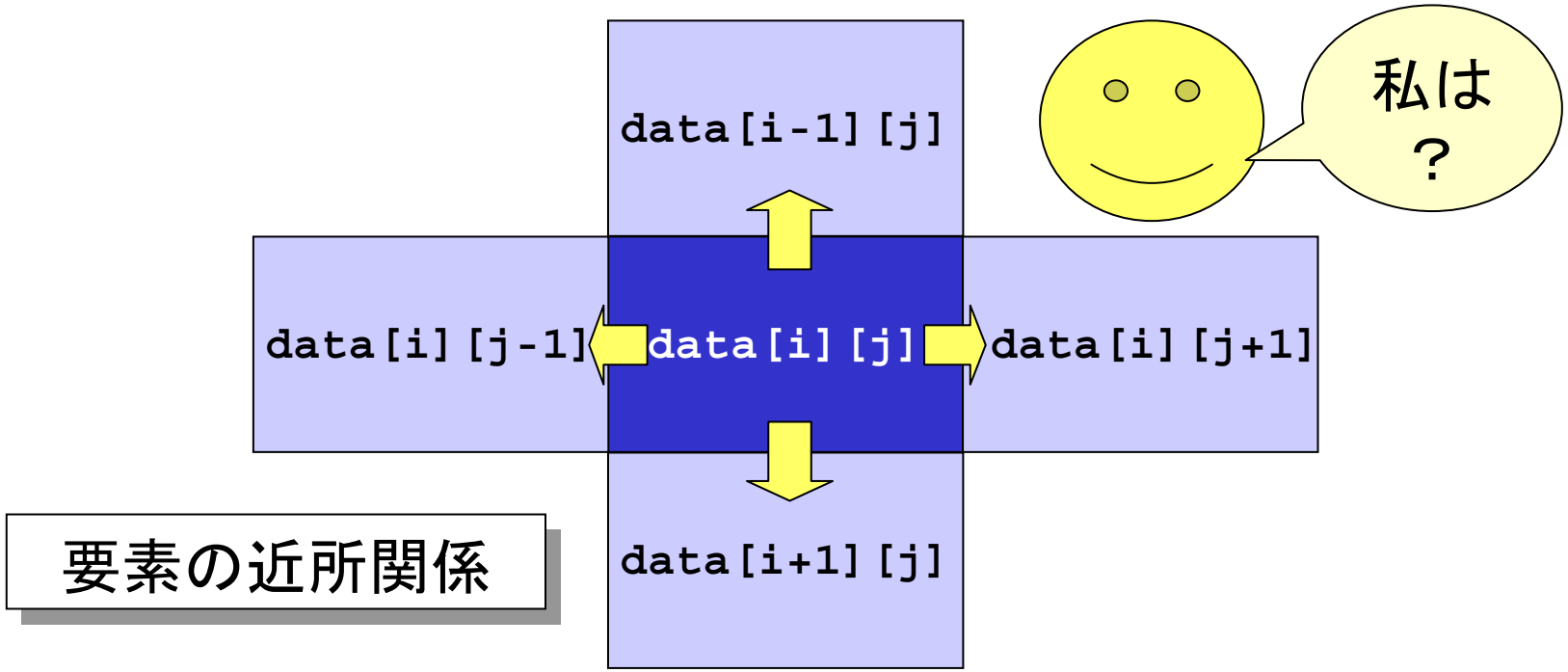
[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]

行添字は0から
行数－1まで

一つずつを「要素」と言う
data[行添字][列添字]

２次元配列の配列要素

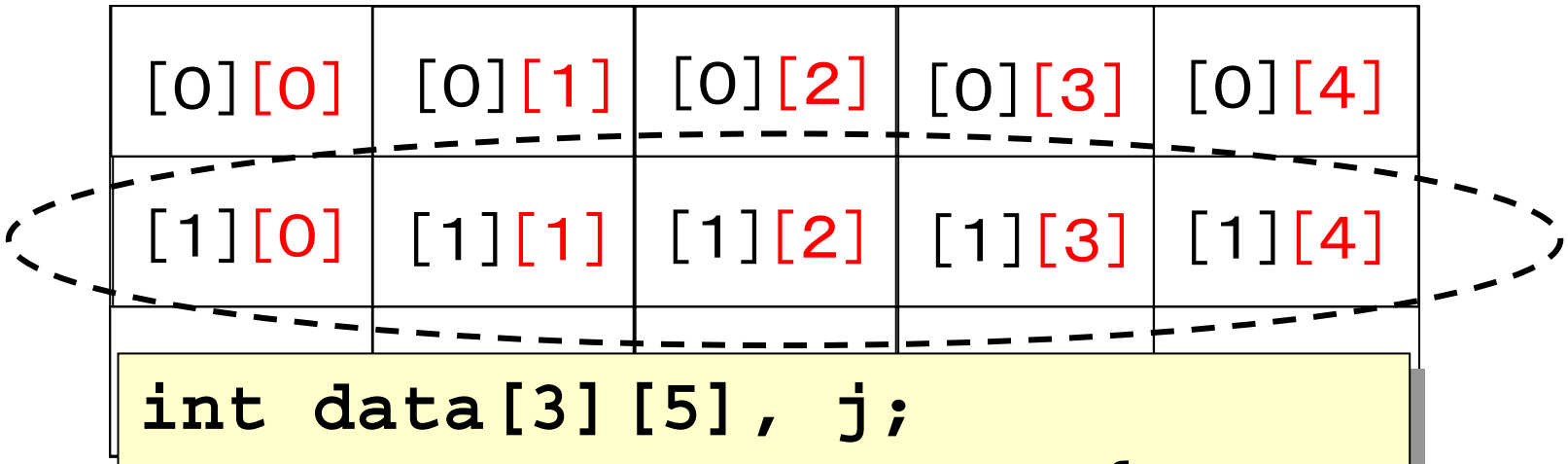
配列名[行添字][列添字]



要素の近所関係

2次元配列要素の操作(1)

配列 int data[3][5]; の行1を表示



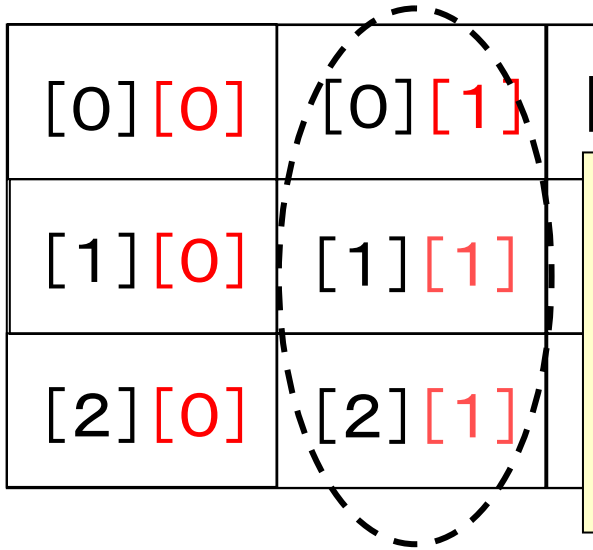

```
int data[3][5], j;
for(j = 0; j < 5; j++){
  printf("%d ",data[1][j]);
}
```

行番号固定

前頁の答え `data[i-1][j+1]`

2次元配列要素の操作(2)

配列 int data[3][5]; の列1の和

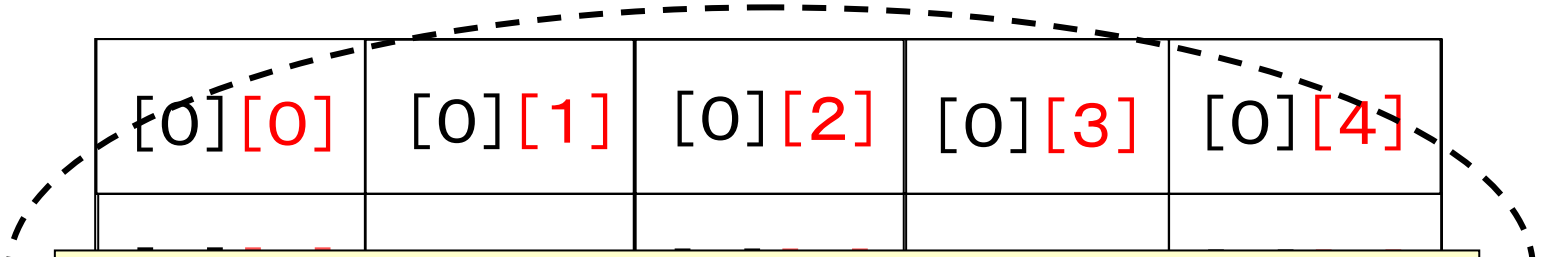



```
int data[3][5], i, sum = 0;
for(i = 0; i < 3; i++){
  sum += data[i][1];
}
```

列番号固定

2次元配列要素の操作(3)

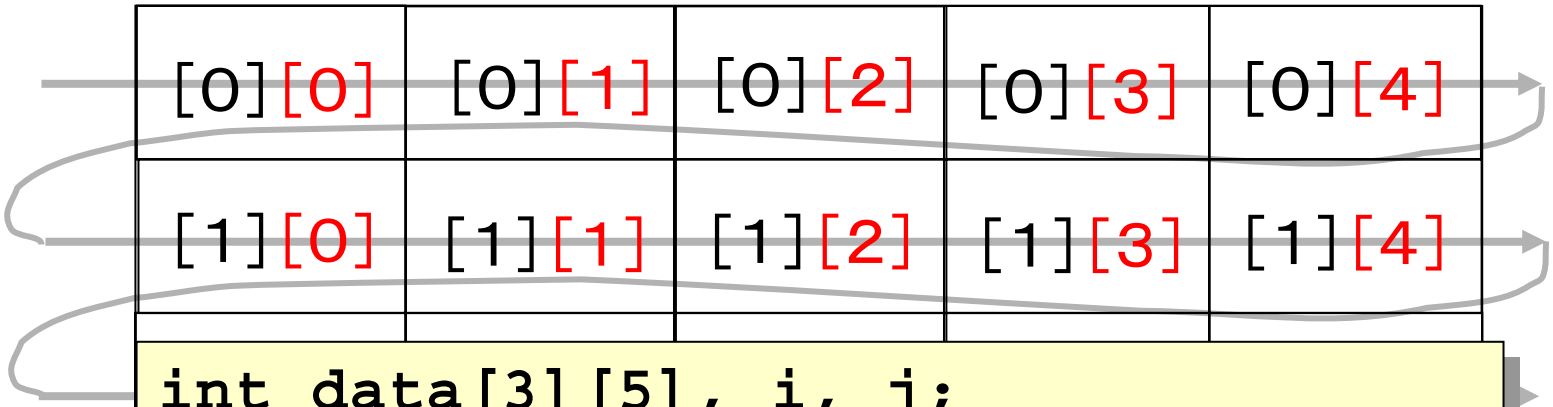
配列 `int data[3][5];` の全ての要素の和




```
int data[3][5], i, j, sum = 0;
for(i = 0; i < 3; i++){
  for(j = 0; j < 5; j++){
    sum += data[i][j];
  }
}
```

二重ループ

◆2次元配列要素の入力




```
int data[3][5], i, j;
for(i = 0; i < 3; i++){
  for(j = 0; j < 5; j++){
    scanf("%d",&data[i][j]);
  }
}
```

/home/course/prog0/public_html/2006/lec/source/lec08-1.c

◆2次元配列要素の出力(1)

```
#include <stdio.h>
int data[3][5], i, j;
for(i = 0; i < 3; i++){
  for(j = 0; j < 5; j++){
    data[i][j] = i * 10 + j;
    printf("%2d ",data[i][j]);
  }
  printf("\n");
}
```

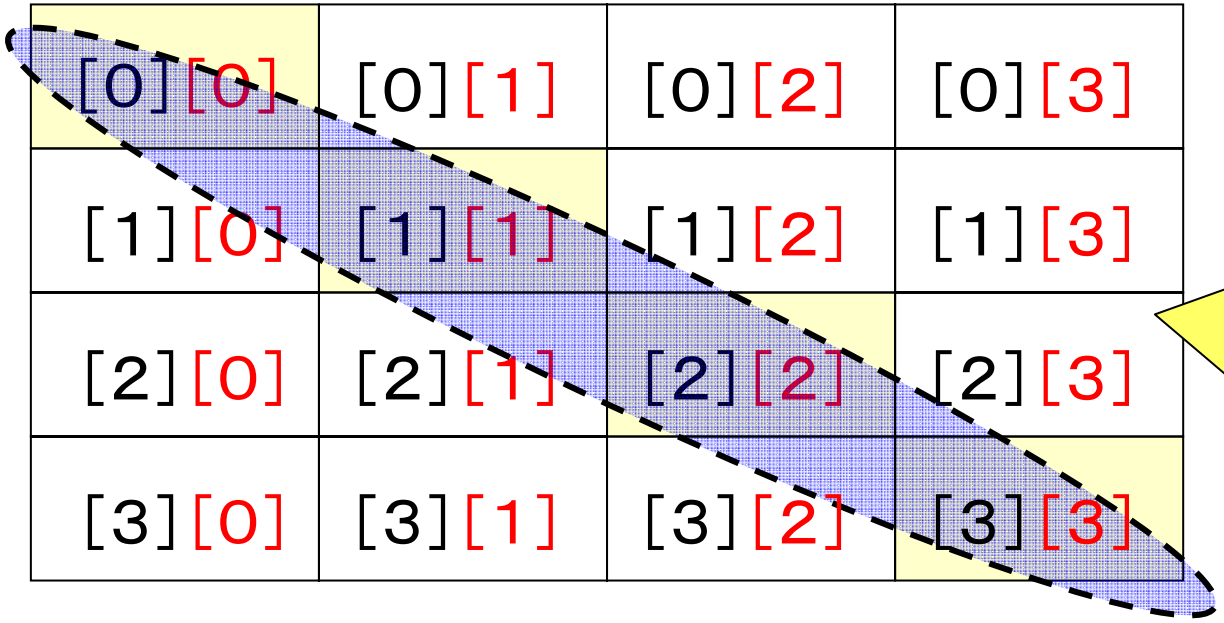
```
std1dc1{s1000000}1: ./a.out
 0  1  2  3  4
10 11 12 13 14
20 21 22 23 24
std1dc1{s1000000}2:
```

内側のforループが終了する度に
(つまり1行分の表示が終わる度に)
改行すると、表が表示できる

/home/course/prog0/public_html/2006/lec/source/lec08-2.c

2次元配列要素の出力(2)

配列 int data[4][4]; の対角要素の表示



/home/course/prog0/public_html/2006/lec/source/lec08-3.c

マクロの利用(p.224)

- プログラムの最初にマクロを定義することが出来る
- マクロとは定数や簡単な計算式に名前を付け、プログラム中で使用すること
- プログラミング入門では定数マクロのみを紹介する。定数マクロの定義書式は以下の通り:

  `#define マクロ名　値`

- コンパイル時（正確にはコンパイルの前処理時）にプログラム内のマクロ名は全て値に置き換わる（右例ならNは全て10に置き換わる）
- マクロの利点は値の変更を一括して行えることである（一括書き換えが容易）

名前がNで値が10のマクロを定義

マクロの使用

```
#include <stdi...n>
#define   N   10

main(){
int data[N],i;
for(i = 0; i < N; i++){
  scanf("%d",&data[i]);
}
....
```

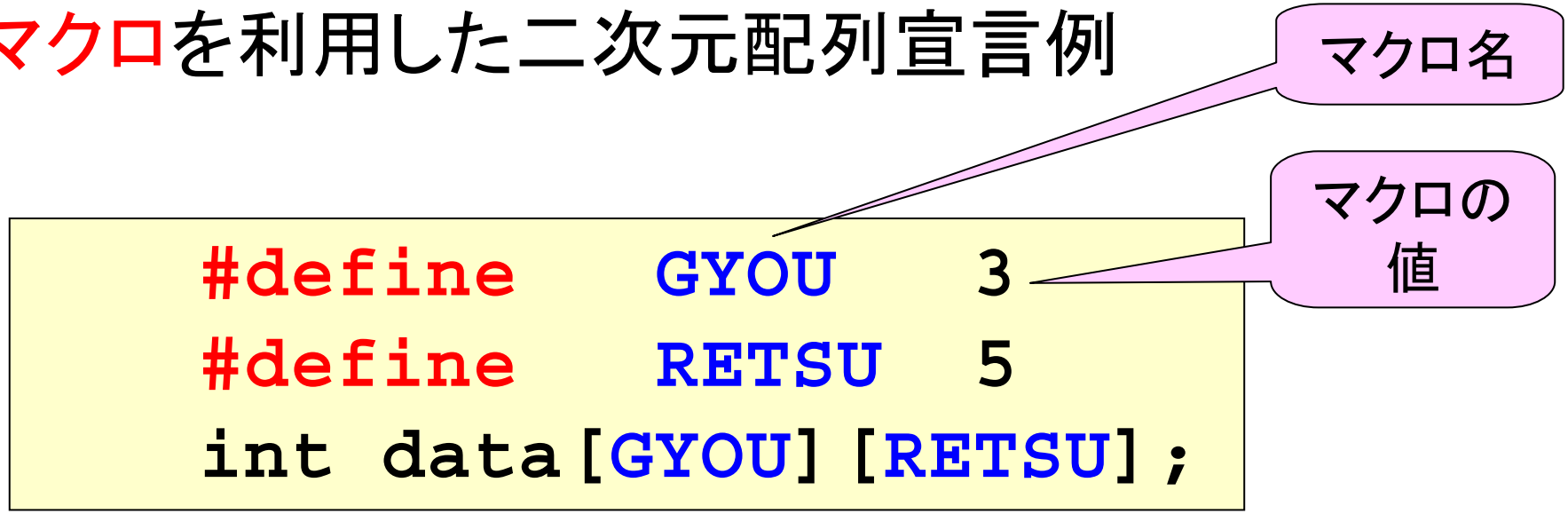
このプログラムをマクロを使わずに書いて、その後値を20に変更することを考えると理解出来る

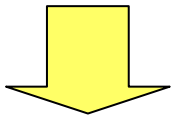
前頁の答え 1重

```
for(i=0;i<4;i++)
  printf("%d¥n",data[i][i]);
```

マクロを利用した 2次元配列の宣言(p.229)

マクロを利用した二次元配列宣言例





int data[3][5]; と同じ

配列の初期化(1)(p.238)

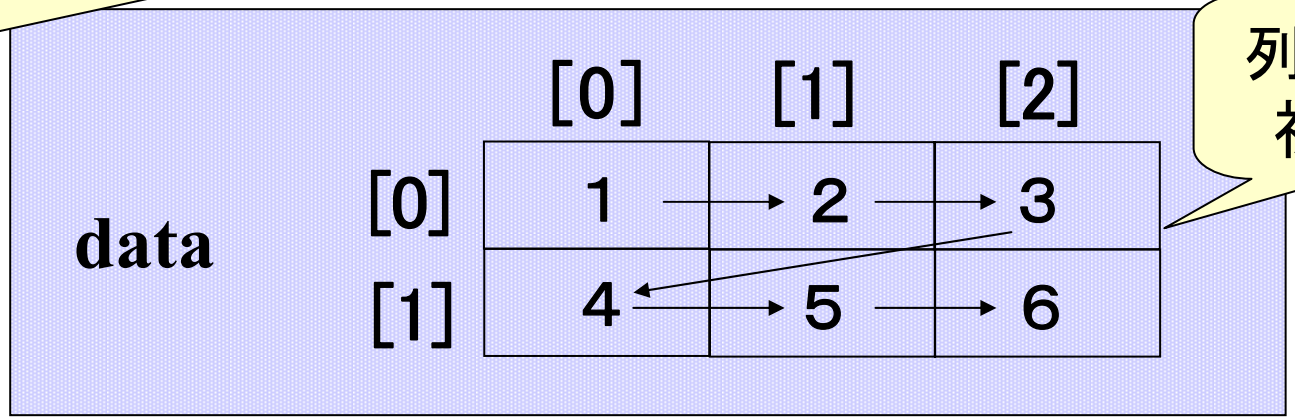
２次元配列の場合、列優先で若番から初期化

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

又は

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

行毎に中括弧にまとめるのが一般的



列優先で初期化

配列の初期化(2)

初期化のバリエーション(3つとも同じ初期化を行う)

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

行サイズを省略可能
(初期化データ
の個数で判断)

data

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6

/home/course/prog0/public_html/2006/lec/source/lec08-4.c

2次元配列の応用(1)
行列の加算(p.230)

```
double a[ROW][COLUMN];
double b[ROW][COLUMN];
double c[ROW][COLUMN];
int i,j;
for(i = 0; i < ROW; i++) {
  for(j = 0; j < COLUMN; j++) {
    c[i][j] = a[i][j] + b[i][j];
  }
}
```

$$c_{ij} = a_{ij} + b_{ij}$$

2次元配列の応用(2)
行列の積(p.232)

```
int a[P][Q],b[Q][R],c[P][R],i,j,k;
for(i = 0; i < P; i++) {
  for(j = 0; j < R; j++) {
    c[i][j] = 0;
    for(k = 0; k < Q; k++){
      c[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

P行Q列× Q行R列の行列の積はP行R列になり、

その第i行第j列の要素は $c_{ij} = \sum_{k=0}^{q-1} a_{ik} \cdot b_{kj}$ となる