第5回　配列

1

今回の目標

- マクロ定義の効果を理解する。
- １次元配列を理解する。
- ２次元配列を理解する。

☆２×２の行列の行列式を求めるプログラムを作成する

2

行列式

$$\begin{vmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{vmatrix} = x_{00}x_{11} - x_{01}x_{10}$$

3

マクロ定義

`#define　文字列1 文字列2`

ソースのなかの文字列1を文字列2に書き換える（置換する。）

この定義をマクロ定義と呼び、
文字列1をマクロ名、
文字列2をマクロ展開という。

通常の変数と区別するため、本演習ではマクロ名に英小文字を用いないこと。（スタイル規則参照）

例　`#define DATANUM 1000`

重要な定数に名前をつける効果がある。

4

#difine 例

```
#define     MEMBER 50

int main(){
        double kokugo[MEMBER];
        double suugaku[MEMBER];
        double eigo[MEMBER];
        double rika[MEMBER];
```

プログラムにおける利用データ数の変更が容易になる。

同じ効果

```
int main(){
        double kokugo[50];
        double suugaku[50];
        double eigo[50];
        double rika[50];
```

配列の最大要素数は、重要なのでマクロ定義で”名前（マクロ名）”を付けること。（スタイル規則参照）

5

配列

同じ型の変数を複数集めたもの。
個々の要素（配列要素）は、普通の変数として扱える。

宣言：

```
要素のデータ型  配列名[要素数];
```

配列を宣言する際は、マクロを用いること。

例

```
#define     SIZE   4

double x[SIZE];
```

```
double x[100];
```

注意：
1. 変数は要素数分用意される。
2. 宣言で用いる要素数は定数でなくてはならない（変数で指定することは、不可）

6

使い方:

配列名[添字]

で普通の変数のようにつかえる。配列要素は、整数型の値（定数、変数、式）で指定される。要素を指定する整数値を添字（インデックス）と呼ぶこともある。
ただし、添字の値は、「0から（要素数－1）」でなければならない。

例

```
max=x[0];
```

添字の値が不正になるようなプログラムでも、コンパイルはできてしまう。十分注意する事。

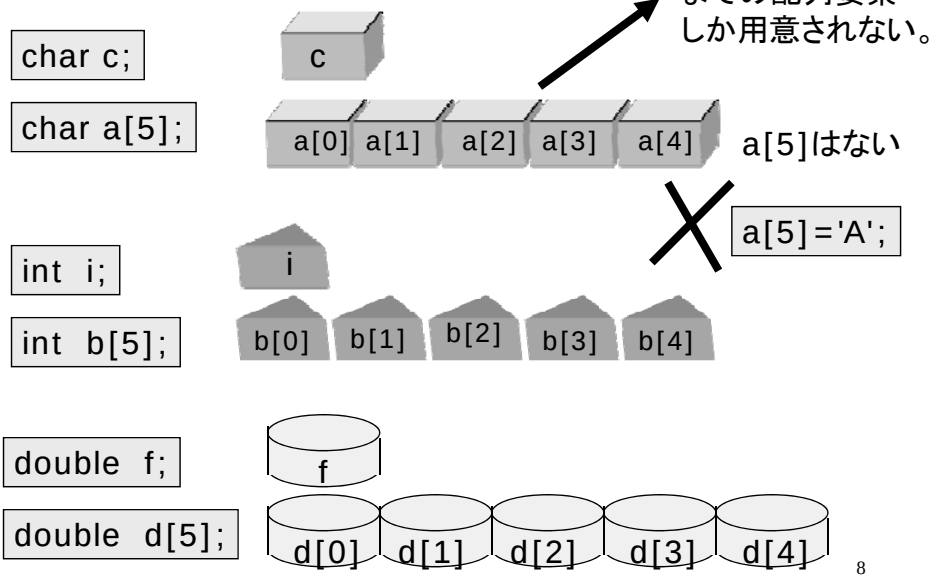
```
int i;
i=3;
min=x[i];
```

添え字にはint型の変数も利用可能である。この場合、変数に蓄えられている値に注意する事。

注意:添字にはint型の式を使い、
添字の値が取りえる範囲に気を付ける事。

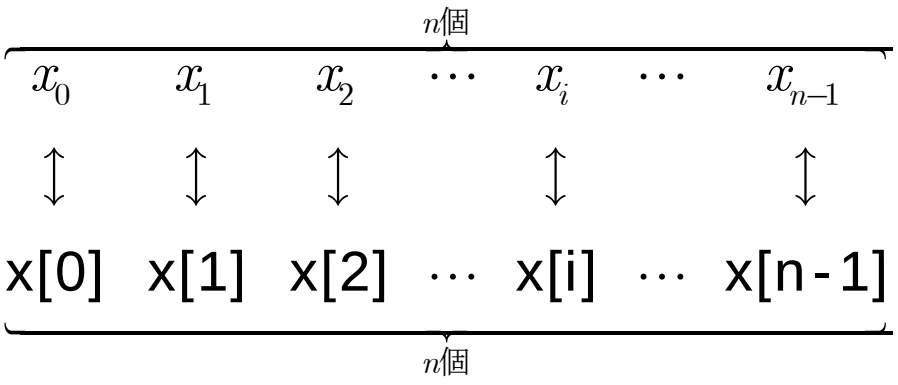
7

イメージ



8

数学の変数とC言語の配列



9

練習1

```
/*  test_array.c    配列実験  コメント省略 */
#include <stdio.h>
#include <math.h>
#define  SIZE  3
int main()
{
      int    i;
      double    a[SIZE];
      double    b[SIZE];
      double    c[SIZE];

      /*     次に続く     */
```

```
/* 配列要素への代入*/
a[0]=0.0;
a[1]=0.0;
a[2]=0.0;
b[0]=M_PI;
b[1]=0.0;
b[2]=0.0;
c[0]=0.0;
c[1]=0.0;
c[2]=0.0;

/*間違った代入*/
b[SIZE]=M_E; /*間違い*/
/*    次に続く    */
```

```
/*配列内容表示*/
printf("a[0] =%f ¥n",a[0]);
printf("a[1] =%f ¥n",a[1]);
printf("a[2] =%f ¥n",a[2]);

printf("b[0] = %f ¥n",b[0]);
printf("b[1] = %f ¥n",b[1]);
printf("b[2] = %f ¥n",b[2]);

printf("c[0] = %f ¥n",c[0]);
printf("c[1] = %f ¥n",c[1]);
printf("c[2] = %f ¥n",c[2]);

/*          続く              */
```

```
    /*配列要素の間違った利用*/
    printf("c[%d] =  %f ¥n",SIZE,c[SIZE]);

    return  0;
}
```

13

2次元配列

宣言:

```
要素のデータ型  配列名[行の要素数][列の要素数];
```

例

```
#define     TATE   5
#define     YOKO   3

double  m[TATE][YOKO];
```

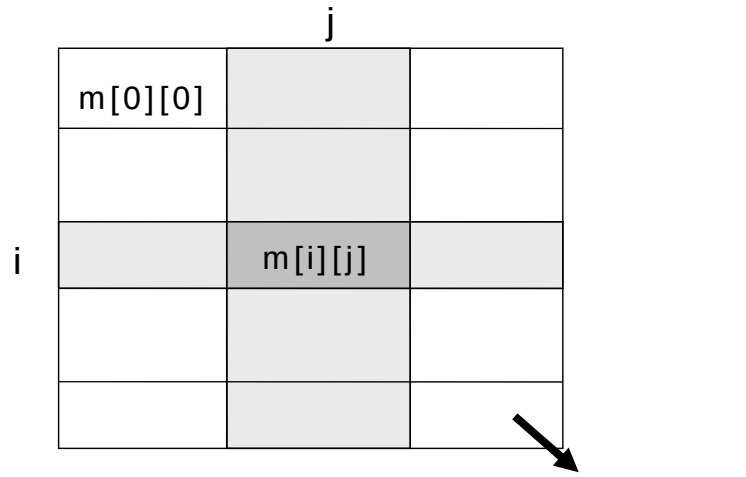
使いかた:

```
配列名[添字1][添字2]
```

で普通の変数のように使える。
また、添字には(整数型の)
変数や式も使える。

14

2次元配列のイメージ



15

2次元配列でよくある間違い

✗ 間違い1

配列名[添字1,添字2]

m[i, j]

数学の座標のように、カンマで区切るのは間違い。

✗ 間違い2

配列の添字を入れ替えてしまう。

m[i][j] のつもりで、 m[j][i] とする。

16

練習2

```
/*  tenchi.c    2次元配列実験(転置行列）
      コメント省略
*/
#include <stdio.h>
#define  GYO  2
#define  RETU 2
int main()
{
      doulbe x[GYO][RETU];     /*行列*/
      double tx[RETU][GYO];    /*転置行列*/

      /*次に続く*/
```

```
      /*入力処理*/
      printf("x[0][0]?");
      scanf("%lf",&x[0][0]);
      printf("x[0][1]?");
      scanf("%lf",&x[0][1]);
      printf("x[1][0]?");
      scanf("%lf",&x[1][0]);
      printf("x[1][1]?");
      scanf("%lf",&x[1][1]);
      /*代入処理*/
      tx[0][0]=x[0][0];
      tx[0][1]=x[1][0];
      tx[1][0]=x[0][1]
      tx[1][1]=x[1][1];
      /*次に続く*/
```

```
    /*出力処理*/
    printf("x¥n");
    printf("%6.2f %6.2f ¥n",x[0][0],x[0][1]);
    printf("%6.2f %6.2f ¥n",x[1][0],x[1][1]);

    printf("xの転置行列¥n");
    printf("%6.2f %6.2f ¥n",tx[0][0],tx[0][1]);
    printf("%6.2f %6.2f ¥n",tx[1][0],tx[1][1]);

    return 0;
}
```

多次元配列

宣言:

データ型 配列名[次元1の要素数][次元2の要素数][次元3の要素数]･･･;

例

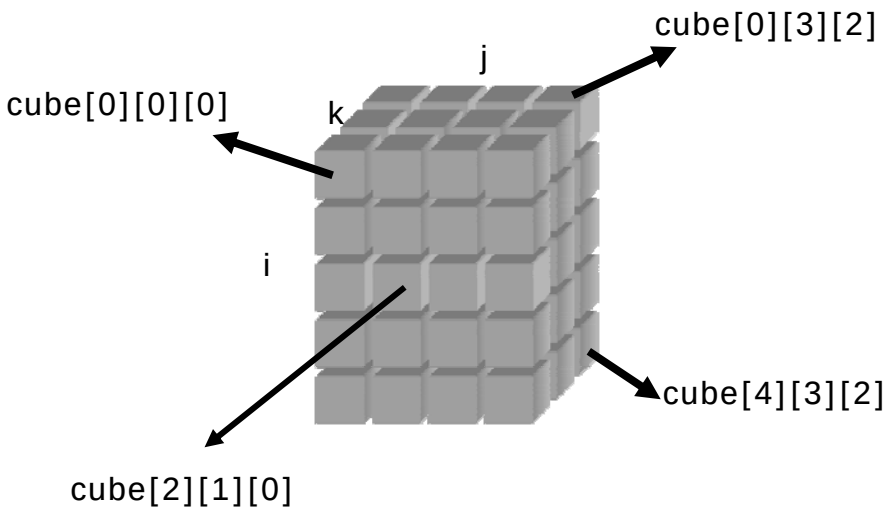
```
#define     TATE   5
#define     YOKO    4
#define     OKU     3

double  cube[TATE][YOKO][OKU];
```

使いかた:

配列名[添字1][添字2][添字3]･･･

で普通の変数のようにつかえる。
また、添え字には(整数型の)
変数や式も使える。

3次元配列のイメージ



21

行列式を求めるプログラム

```
/*
        作成日: yyyy/mm/dd
        作成者:本荘太郎
        学籍番号:B0zB0xx
        ソースファイル:determinant.c
        実行ファイル:determinant
        説明:2×2行列の行列式を求めるプログラム。
        入力:標準入力から行列の4つの成分を入力する。
                同じ行の要素が連続して入力されるとする。
        出力:標準出力に行列式を出力する。
*/
/*      次のページに続く */
```

22

```
/*    続き  */
#include <stdio.h>
/*マクロ定義*/
#define  SIZE  2
int  main()
{
     /*    変数宣言          */
     double  det;    /*行列式*/
     double  matrix[SIZE][SIZE];   /*行列*/

/*    次のページに続く */
```

23

```
     /* 入力処理*/
     printf("matrix[0][0]?");
     scanf("%lf",&matrix[0][0]);
     printf("matrix[0][1]?");
     scanf("%lf",&matrix[0][1]);
     printf("matrix[1][0]?");
     scanf("%lf",&matrix[1][0]);
     printf("matrix[1][1]?");
     scanf("%lf",&matrix[1][1]);

     /*行列式の計算*/
     det＝matrix[0][0]*matrix[1][1]
          -matrix[0][1]*matrix[1][0];
```

```
    /* 出力処理*/
    printf("行列¥n");
    printf("%6.2f %6.2f ¥n",matrix[0][0],
                            matrix[0][1]);
    printf("%6.2f %6.2f ¥n",matrix[1][0],
                            matrix[1][1]);
    printf("の行列式は、¥n");
    printf("%6.2f です。¥n",det);

    return 0;
}
```

実行例

```
$./determinant
matrix[0][0]?1.0
matrix[0][1]?2.0
matrix[1][0]?3.0
matrix[1][1]?4.0
行列
   1.00  2.00
   3.00  4.00
の行列式は
 -2.00です。
$
```