プログラミング発展

2023年度2Q　火曜日5~7時限(13:45~16:30)
金曜日5~7時限(13:45~16:30)

工学院　情報通信系

尾形わかは，松本隆太郎，
Chu Van Thiem, Saetia Supat
TA:東海林郷志，千脇彰悟

第5回　「ポインタ」

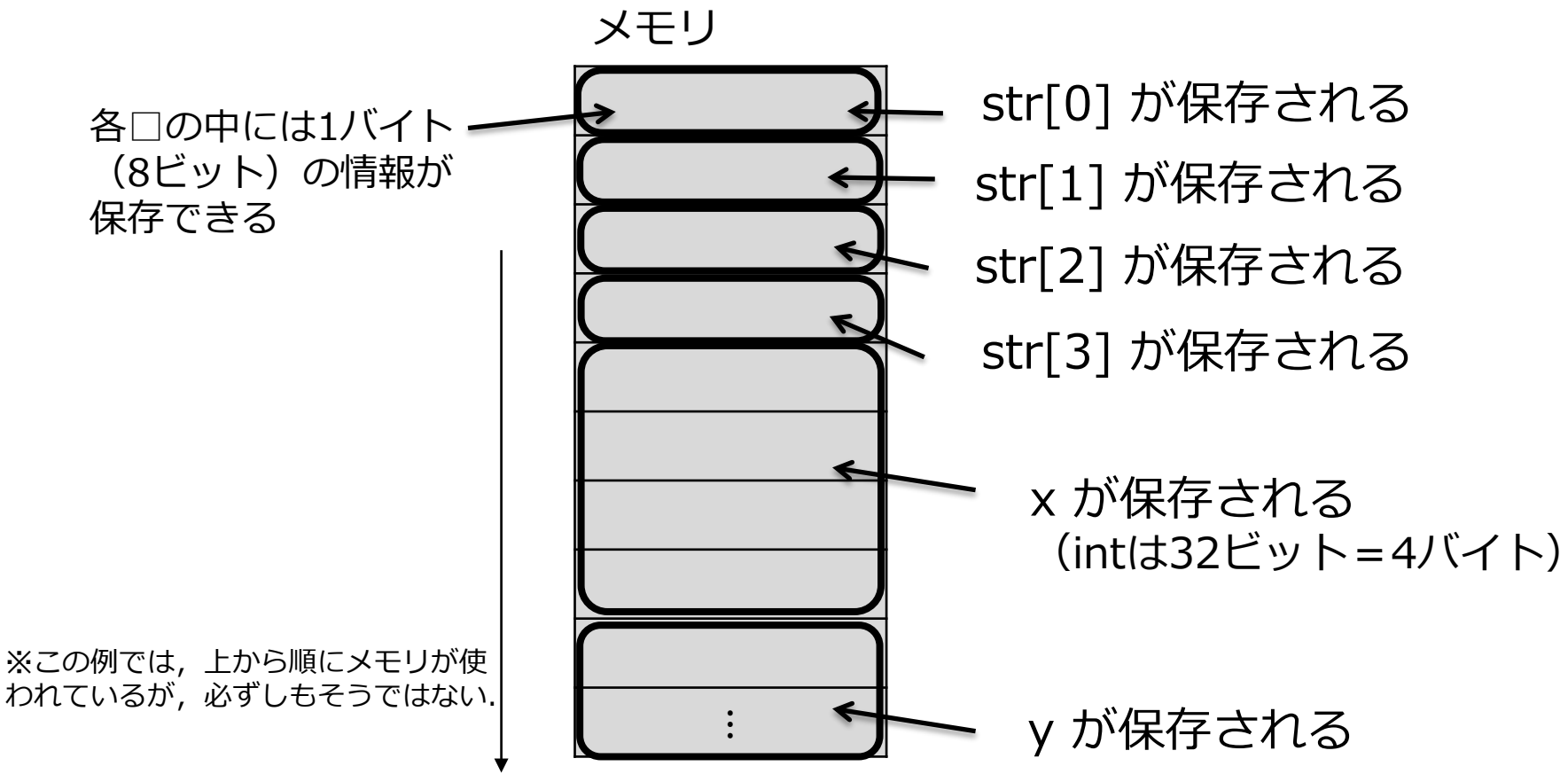
1. 前回の復習（課題の解説）
2. ポインタ

おさらい

変数とメモリ２

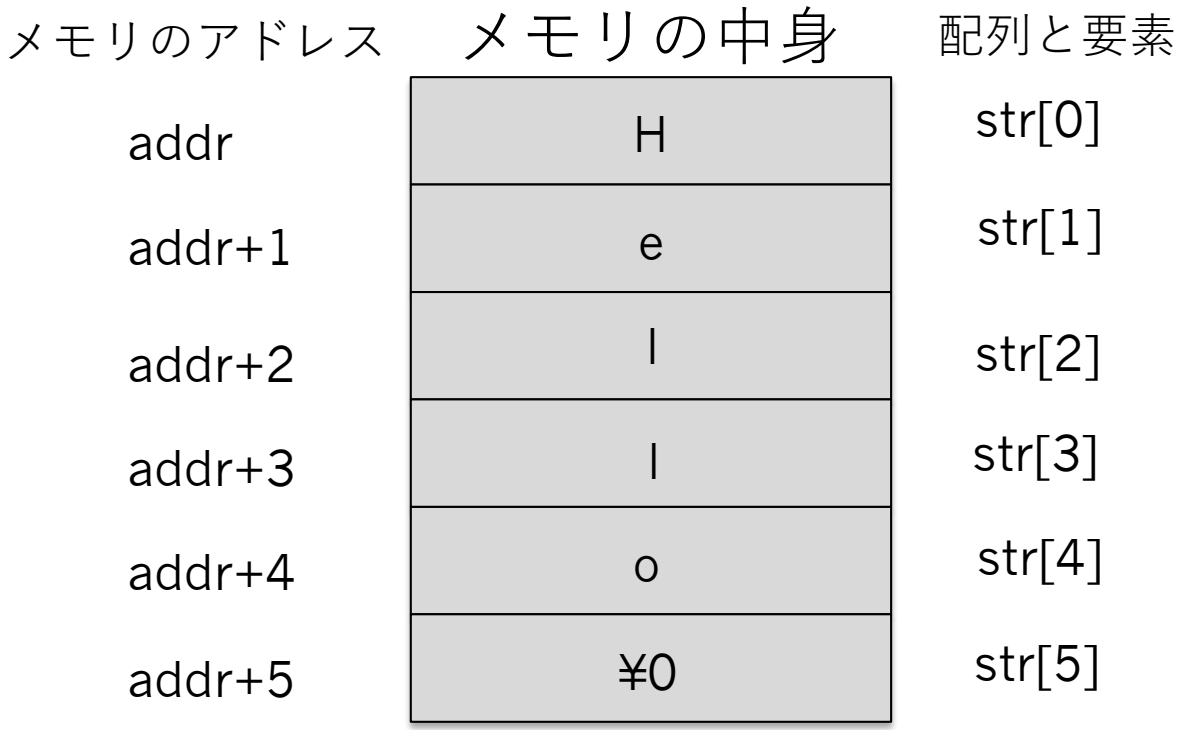
例

```
char str[4].
int x;
short y;
```



文字型変数の表現

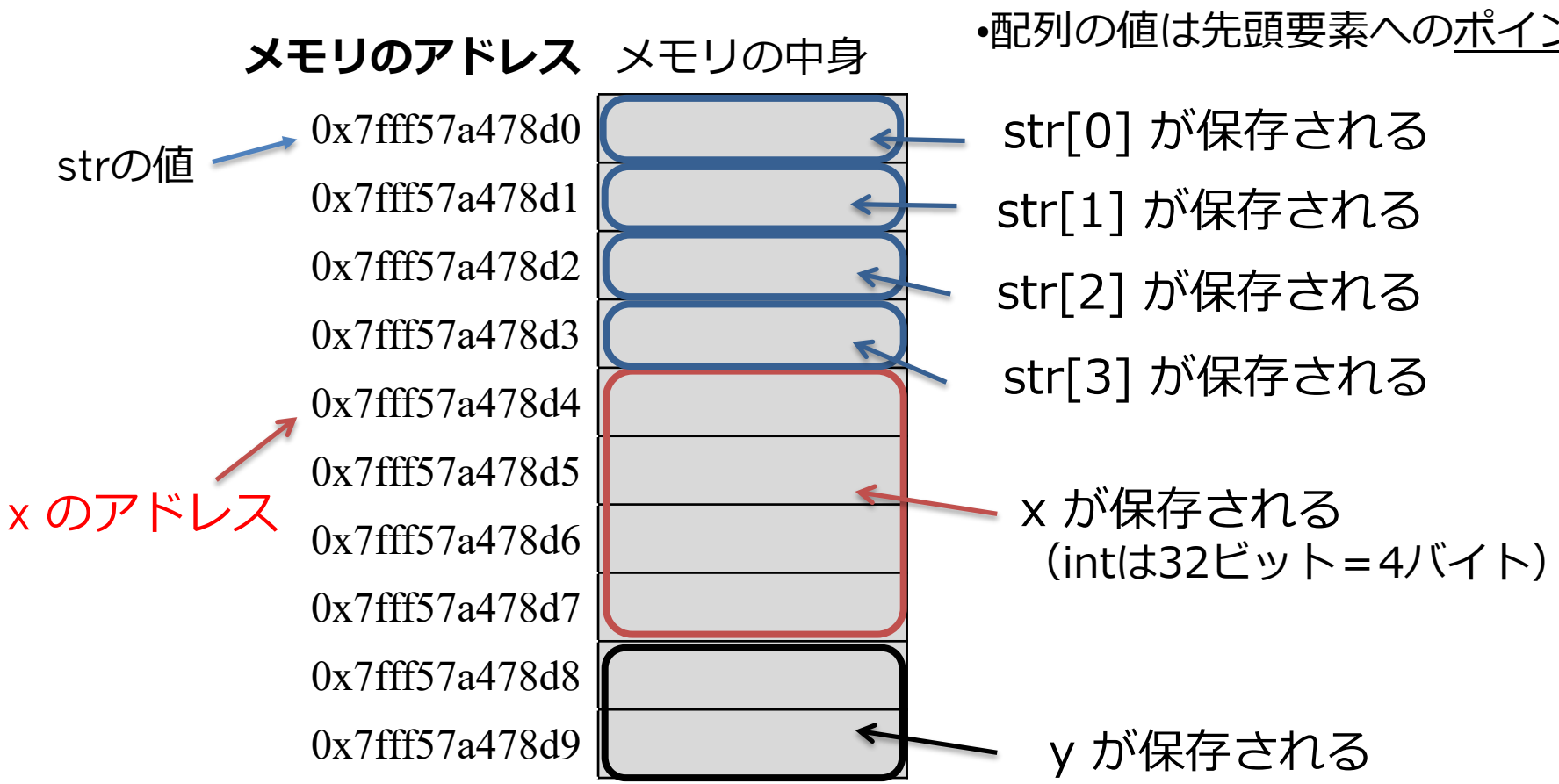
char str[6]={'H', 'e', 'l', 'l', 'o', '¥0'};



ポインタ：メモリの「アドレス」を保持する**変数**

変数の「アドレス」

- メモリには，「アドレス」がある．



- 配列の値は先頭要素へのポインタ

ポインタ

- 「ポインタ」：計算機メモリの「アドレス」を保持する変数。
  他の変数の値、配列の要素、等を参照するのに用いる

- ポインタ型：ポインタのための型。
  - ポインタ型の値 ： 64bitのデータ(この講義で使っている環境の場合)
- 参照するデータ型に応じて、異なる型になる。
  - int型の変数のアドレスを保持するポインタ形は「intポインタ型」
    int *ptr1;
  - char型の変数のアドレスを保持するポインタ型は「charポインタ型」
    char *ptr2;
  - 「intポインタ型の変数」のアドレスを保持するポインタ型　（ポインタのポインタ）
    int **ptr3;

ポインタに関する演算子 1

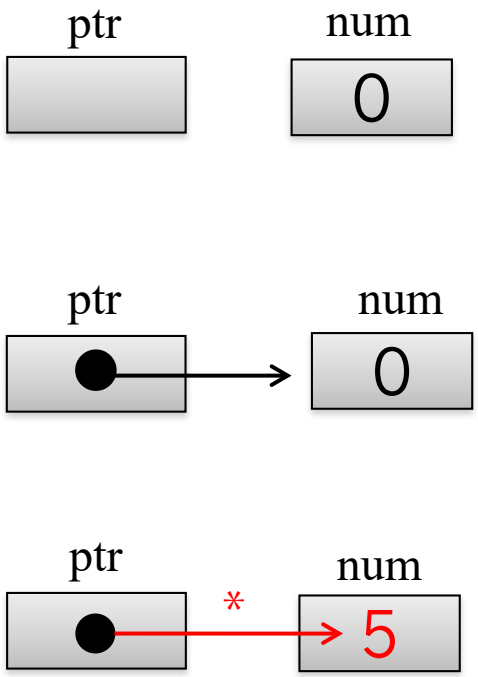
- アドレス演算子　'&'（アンパサンド，アンド）
  - 変数等のアドレス（ポインタ型の値）を表す
  - 例：
```
int i, *ptr;
ptr = &i;
```

- 間接演算子　'* '　（アスタリスク）
  - ポインタ型の変数で指定したアドレスに格納された対象 (変数等) を表す
  - 例：
```
if (*ptr > 5) *ptr = 5;
```

`ptr = &i`　であるとき，
`if (i > 5) i = 5;`
と同じことを表す．

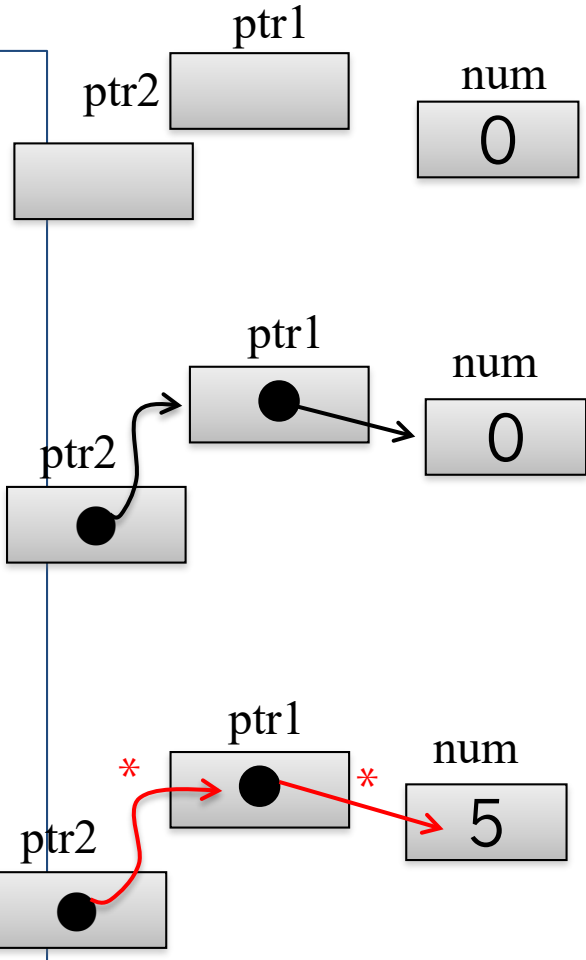
ポインタによる変数の値の操作 1

```
#include <stdio.h>

int main(void) {
    int num = 0;
    int *ptr;

   // num のアドレスをポインタ変数ptrに代入
   ptr = &num;

   // ptrで指定したアドレスで示されるメモリ領域に 5 を代入
   *ptr = 5;

   // numの値を二通りで出力
   printf("(1) num = %d¥n",num);
   printf("(2) num = %d¥n",*ptr);

   return 0;
}
```



ポインタによる変数の値の操作２

```
#include <stdio.h>
int main(void) {
    int num = 0;
    int *ptr1, **ptr2;


    // num のアドレスをポインタ変数ptr1に代入
    ptr1 = &num;
    // ptr1のアドレスをポインタのポインタptr2に代入
    ptr2 = &ptr1;


    // ptr2で指定したアドレスで示されるメモリ領域（つまりptr1）
    //  で指定したアドレスで示されるメモリ領域に５を代入
    **ptr2 = 5;

    // numの値を三通りで出力
    printf("(1) num = %d¥n",num);
    printf("(2) num = %d¥n",*prt1);
    printf("(3) num = %d¥n",**prt2);
    return 0;
}
```



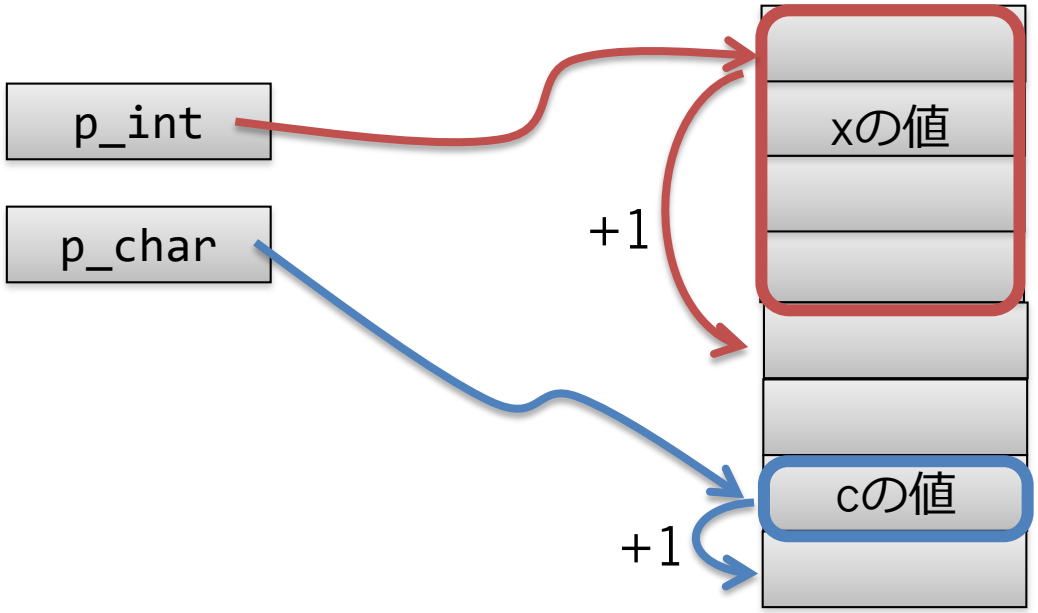
ポインタに関する演算子２

- ポインタの加算（＋），減算（－）
  - ポインタが参照する型のサイズを単位として，値（アドレス）をずらす
  - 例: `ptr++;`

例えば

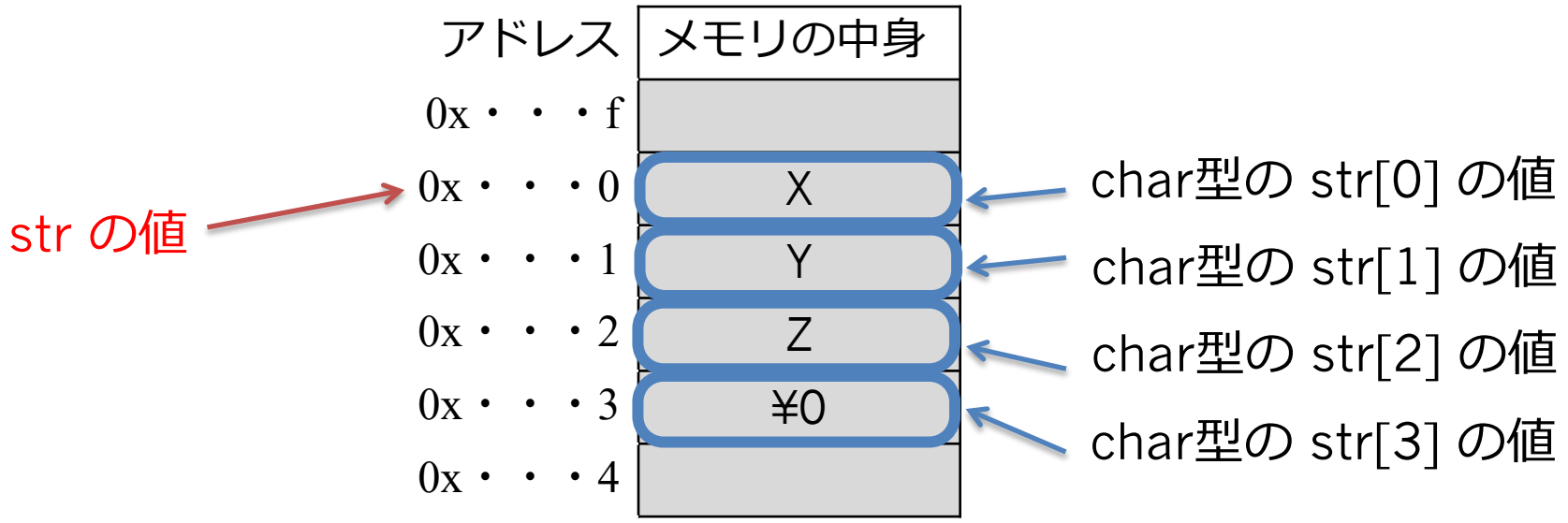
```
int x, *p_int;
char c, *p_char;

p_int = &x;
p_char = &c;

p_int++;
p_char++;
```

のとき，右図になる．



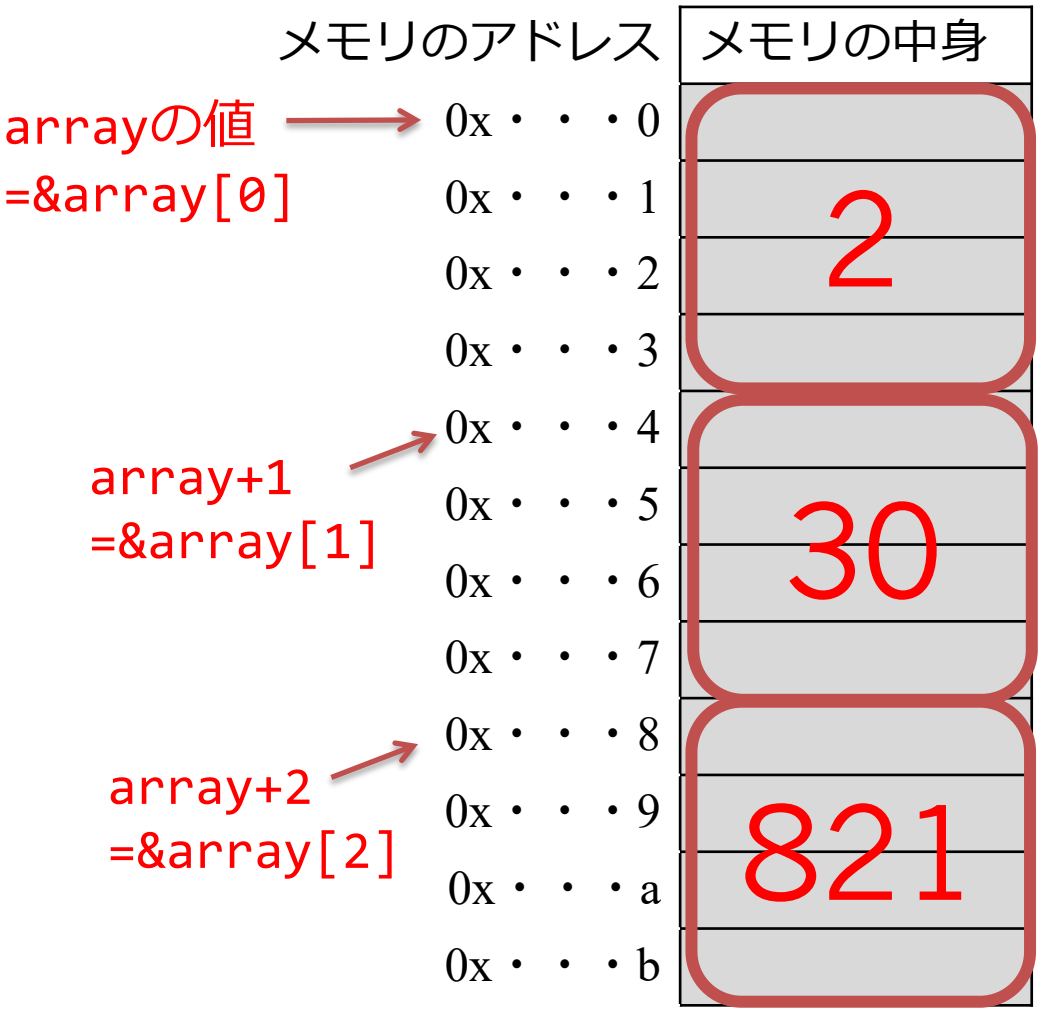
ポインタと配列

- 例：　`char str[4]="XYZ";`



- `str[i]` は、`*(str+i)` のこと.

ポインタと配列


メモリのアドレス
メモリの中身
arrayの値
=&array[0]
array+1
=&array[1]
array+2
=&array[2]
0x・・・0
0x・・・1
0x・・・2
0x・・・3
0x・・・4
0x・・・5
0x・・・6
0x・・・7
0x・・・8
0x・・・9
0x・・・a
0x・・・b
2
30
821

ポインタと関数

プログラミング基礎で出てきた set_dataという関数を考える.

```
void set_data(float a[], int n, int seed){
    int i;
    srand(seed);
    for(i=0; i<n; i++)
        a[i]=my_random(0.0, 1000.0);
}
```

0.0 ～ 1000.0 の間の乱数を選んでくる関数

set_data(a, 3, seed) を実行すると,
配列 a の3個の要素 a[0], a[1], a[2] に乱数がセットされる.

配列ではなく, 3 つの変数 a1, a2, a3 に乱数をセットしたい場合は？

```
void set_3data(float a1, float a2, float a3,  int seed){
    srand(seed);
    a1=my_random(0.0, 1000.0);
    a2=my_random(0.0, 1000.0);
    a3=my_random(0.0, 1000.0);
}
```

これではダメ！

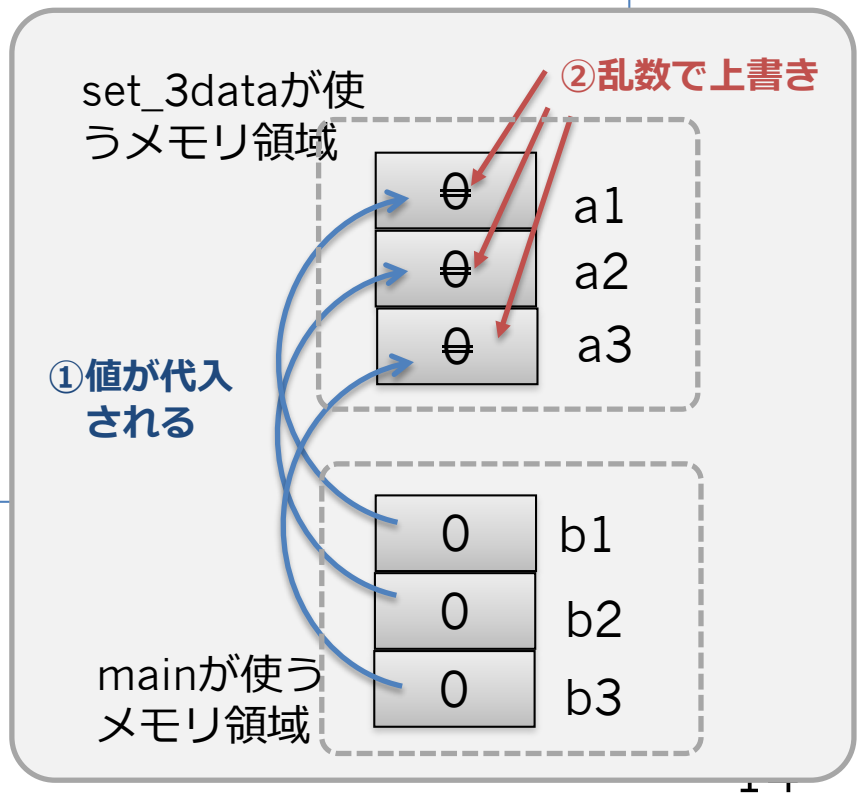
ポインタと関数

なぜうまくいかないのか？

```
void set_3data(float a1, float a2, float a3, int seed){
    srand(seed);
    a1=my_random(0.0, 1000.0);
    a2=my_random(0.0, 1000.0);
    a3=my_random(0.0, 1000.0);
}

void main()
{
    float b1=0, b2=0, b3=0;
    ...
    set_3data(b1,b2,b3,time(NULL));
    ...
}
```

②

①



set_3dataで設定した乱数値がmain の a1, a2, a3に反映されない.

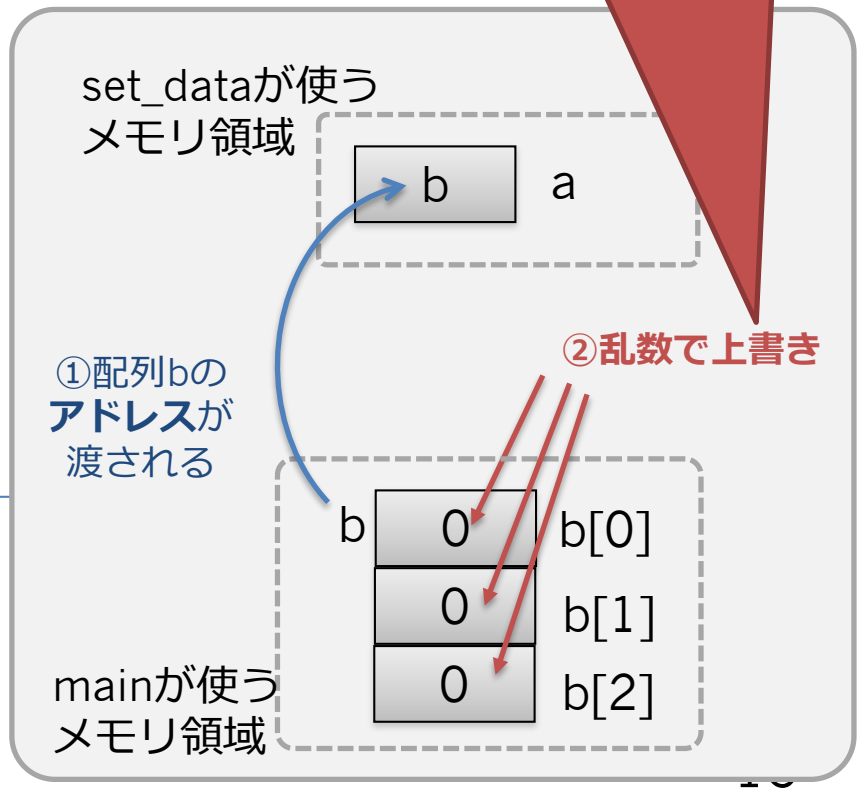
ポインタと関数

配列の場合は，なぜ上手くいくのか？

```
void set_data(float a[], int n, int seed){
    int i;
    srand(seed);
    for(i=0; i<n; i++){
        a[i]=my_random(0.0, 1000.0);}
}                                  ②

int main()
{
    float b[3]={0,0,0};
    ...
    set_data(b,3,time());
    ...                            ①
}
```

b は，配列の先頭のアドレスである.

set_data内の a[i] のアドレスは，main内の b[i] と同一なので，直に乱数を代入



ポインタと関数

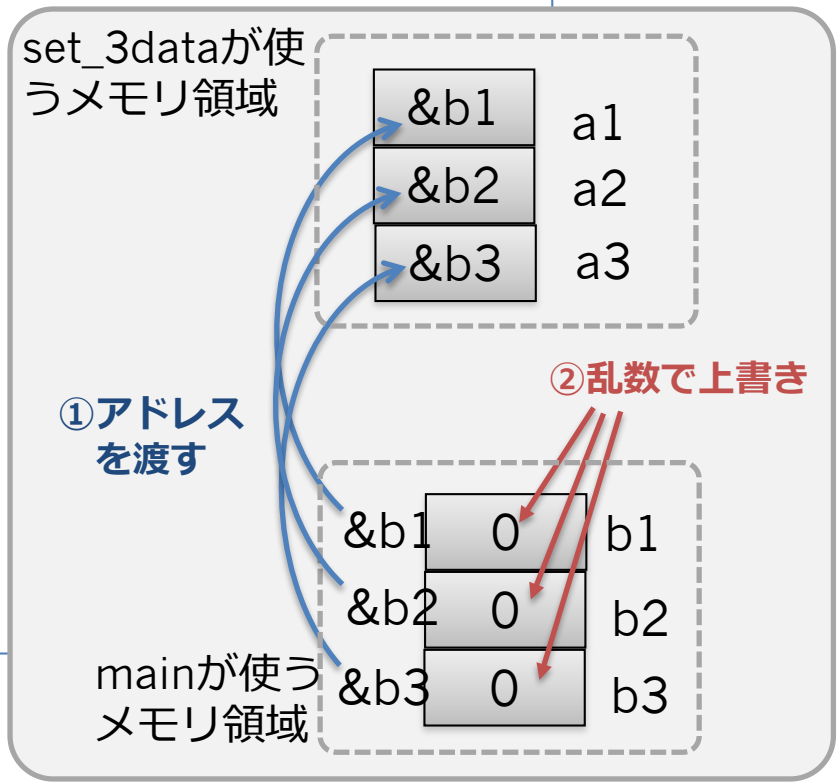
引数の値を変えたい場合は，「ポインタ渡し」とする．

アドレスを受け取る

```
void set_3data(float *a1, float *a2, float *a3, int seed){
    srand(seed);
    *a1=my_random(0.0, 1000.0);
    *a2=my_random(0.0,
    *a3=my_random(0.0,
}

void main()
{
    float b1=0, b2=0, b3=0;
    ...
    set_3data(&b1,&b2,&b3,time());
    ...
}
```

② アドレスが指すメモリの内容を書き換える

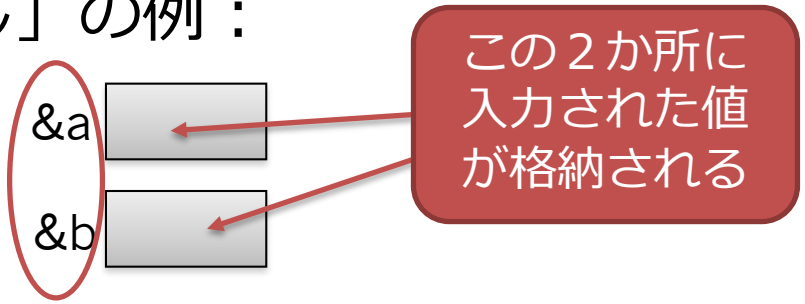
① アドレスを渡す



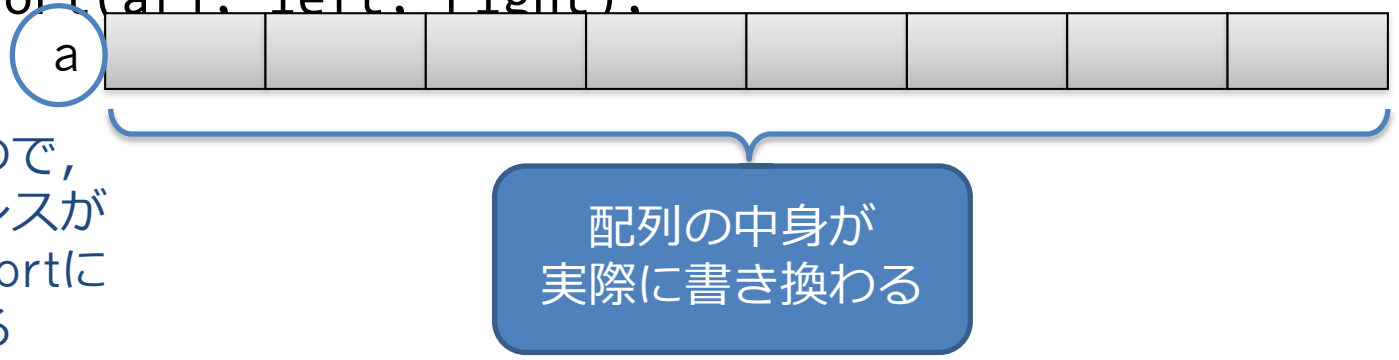
ポインタと関数

これまで出てきた「ポインタ渡し」の例：

- `scanf("%d %d", &a, &b);`



- `quick_sort(a[], left, right);`



Ex5-1

まずは，ポインタの中身を見てみよう.

アドレスを出力するときには，%p を使うこと.

例：　`printf("address of x is %p¥n", &x);`

① int型の大きさ３の配列 cを宣言し，
配列要素のアドレス (&c[0],&c[1],&c[2]）と，
c，c+1，c+2 の値（ポインタの値）を
出力するプログラムを作成し，これらが等しいことを確認せよ.

② 次ページで示す構造体 test_t について，配列 test_t a[3]を宣言し
各配列要素のアドレスと
a[0]の各メンバのアドレス
を出力するプログラムを作成せよ.

③ ②の結果をもとに，変数aの各メンバの値がどのようにメモリに格納されているか，メモリイメージの図を描きなさい.

Ex5-1 （つづき）

```
typedef struct {
    char str1[3];
    unsigned char num1;
    char str2[7];
    unsigned int num2;
    char str3[10];
} test_t;
```

★提出物：

- ②のプログラム：ex5_1_2.c
- ③のメモリイメージを描いた図： ex5_1_3.pdf

→次ページ参照

メモリイメージ図の例

- 例えば

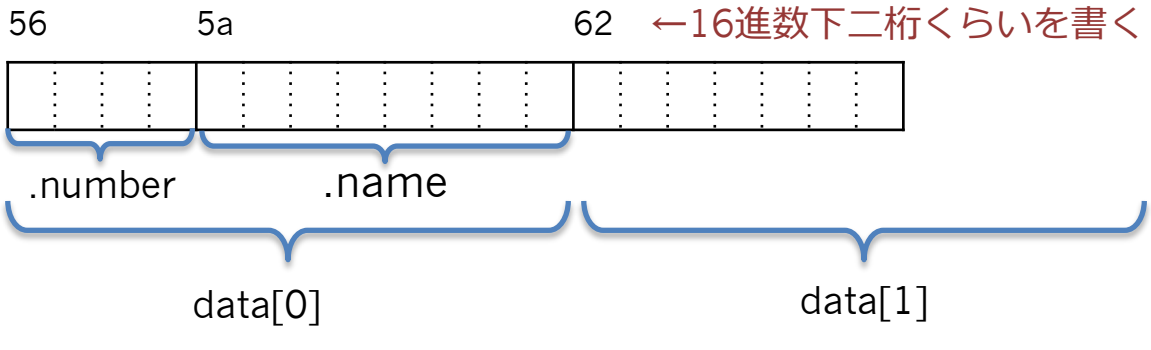
```
typedef struct {
        int number;
        char name[8];
    } sample;
sample data[N];
```

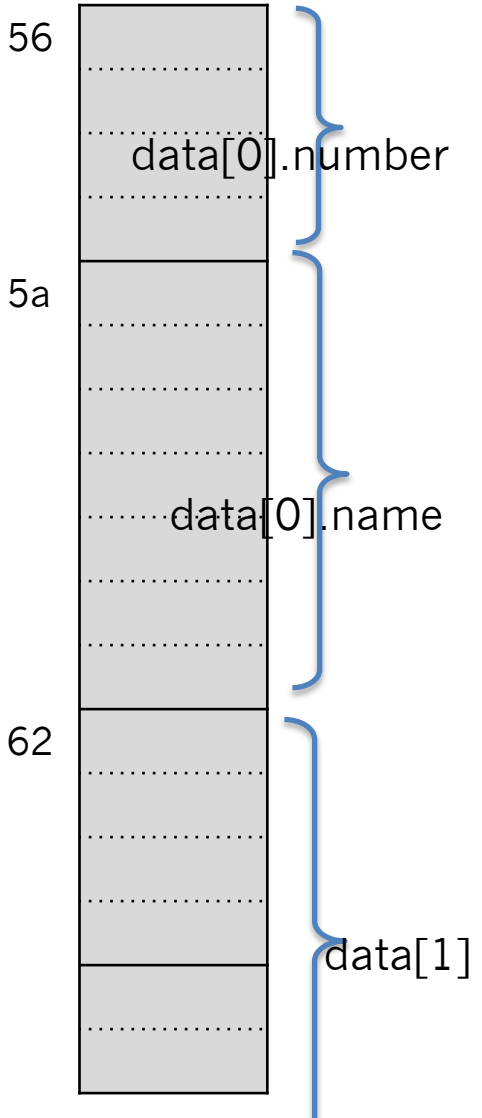
に対して

```
&data[0].number == 0x123456
&data[0].name == 0x12345a
&data[1] = 0x123462
```

だとしたら

16進数で表示される.
（もっと桁は多い）





Ex5-2

① 二つの文字型変数の値を交換する関数として，下の swap1 を考える．

```
void swap1(char a, char b){   // 誤り
    char temp;
    temp = a;
    a = b;
    b = temp;
}
```

文字型変数aとbにそれぞれ'A'と'B'を代入したのち，swap1 を用いて変数a,bの値を交換し，aとbの値を出力するプログラムを作成せよ．（値が交換されないことを確認せよ．）
さらに，関数 swap1 およびmain関数に，
変数 a および b のアドレスを出力する機能を加えよ．

実行例）

```
In main: addresses of a : xxxxxxx , address of b : xxxxxxx
In swap1: addresses of a : xxxxxxx , address of b : xxxxxxx
a = A, b = B
```

Ex5-2　（続き）

② 二つの文字型変数の値を（ちゃんと）交換する関数 swap2 を実装せよ.
①のプログラムを修正し，swap1の代わりにswap2を使って文字型変数aとbの値が交換できるようにせよ.
（値が交換されることを確認せよ．）

実行例）

```
In main: addresses of a : xxxxxxx , address of b : xxxxxxx
In swap2: addresses of a : xxxxxxx , address of b : xxxxxxx
a = B, b = A
```

★提出物：

②のプログラム　：　ex5_2_2.c

Ex5-2　（続き）

③　（発展課題）任意の型の二つの変数の値を交換する関数 swap_univ を実装せよ．下に示すプログラムを完成させ，正しく交換が行わることを確認せよ．

```
typedef struct{
     char name[30];
     int age;
     int gender;
} person_t;

void swap_univ( ... ){ ... }

void main(){
     person_t p1 = {“Alice”, 20, 1};
     person_t p2 = {“Bob”, 30, 2};
     int one=1;
     int zero=0;

     printf( ... );  // p1, p2, zero, one を表示
     swap_univ( ... ); // p1とp2のデータを入れ替える
     swap_univ( ... ); // oneとzeroの値を入れ替える
     printf( ... );  // p1, p2, zero, one を表示
}
```

swap_univの引数は３つとする．２つは交換する変数で，同じ型であると仮定してよい．３つ目は，その型のサイズ．ワーニングが出ないように！

全く同じにする

注意！

- ポインタ型の宣言：

```
char *p;
```

と宣言すると，pは「何らかのchar型の変数のアドレス」を格納するメモリ（8バイト？）が確保される．

```
char *p;
*p = 'a' ;
```

これは実行時エラーを起こします．
pが示すメモリは確保していないのに，
書き込もうとしたので．

```
char *p = "Hello";
printf("%s¥n",p);
*p='A';
```

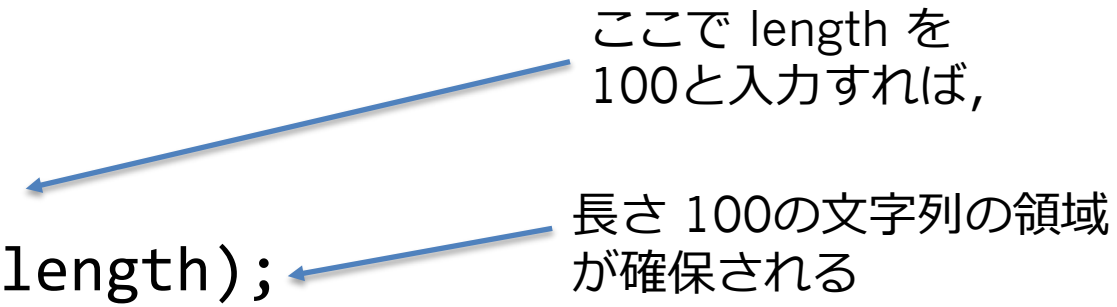
pには Helloが入ったメモリ領域のアドレスが格納され，Helloが印字されるが，
代入はできる場合とエラーになる場合がある．
（できないと思っておこう）

```
char p[] = "Hello";
printf("%s¥n",p);
*p='A';
```

この場合は，5文字分のメモリ領域（6バイト）が確保されるので，正常に代入できる．

可変長配列

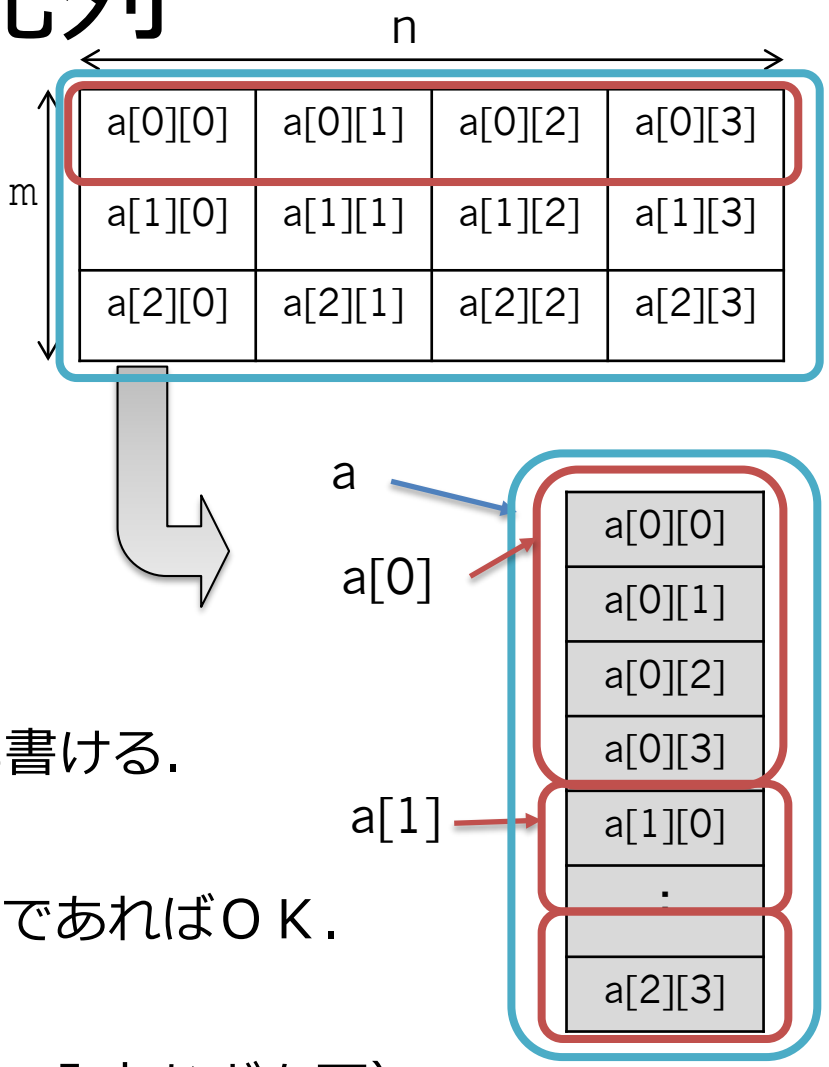
- 配列の長さの指定：
  配列の宣言時に決まっていればOK（※）




```
int length;
scanf("%d",&length);

char string[length];
int sq_matrix[length][length];
```

※静的配列(static)の場合は、コンパイル時に決まっている必要がある。

2次元配列



`int a[m][n];`
と宣言すると，
`m×n` の連続したメモリが確保される．

縦m×横nの行列として使える．

`a[i][j]` は　`*(*a+i*n+j)` のこと．
`*(*(a+i)+j)` とも書ける．

mとnは，それまでに決まっている自然数であればＯＫ．
可変長でもＯＫ
（それまでの計算結果とか，`scanf`からの入力なども可）

配列を関数に渡す

- `int a[10]` を関数 `f1` の引数にする場合,
  - ○ `void f1(int a[10]);`
  - ○ `void f1(int a[]);`
  - ○ `void f1(int *a);`

配列サイズは省略できる

- 可変長配列 `int b[m]` を関数 `f2` の引数にする場合,
  - × `void f2(int b[m]);`
  - ○ `void f2(int length, int b[length]);`
  - × `void f2(int b[length], int length);`
  - ○ `void f2(int b[]);`
  - ○ `void f2(int *b);`

配列サイズは省略できる

二次元配列を関数に渡す

- 二次元配列 `int mat[10][20]` を関数 `f3` の引数にする場合，
  サイズを省略できるのは，1つ目だけ．
  ○ : `void f3(int mat[10][20]);`
  ○ : `void f3(int mat[][20]);`
  × : `void f3(int mat[][]);`
  × : `void f3(int **mat);`

1つ目の配列サイズは省略できる

- 可変長配列 `int mat[m][n]` を関数 `f4` の引数にする場合，
  × `void f4(int mat[m][n]);`
  ○ `void f4(int m, int n, int mat[m][n]);`
  × `void f4(int mat[m][n], int m, int n);`

  ○ `void f4(int n, int mat[][n])`
  × `void f4(int **mat);`

1つ目の配列サイズは省略できる

Ex5-3

可変長二次元配列を行列と考え，行列を扱う関数を作ってみる.

① 整数nと　n×nの2次元配列matを引数とし，n×nの行列の形で表示する関数 `void print_matrix` を作る.

② 整数と，n×nの2次元配列matを引数とし、matをn×nの行列として考え，行列の乗算で2乗した結果をmat に上書きする関数 `void square_matrix` を作る.

③ 上記の関数を用いて，次の動作をするプログラムを作成する。

1. 正の整数nを scanf で受け取る。
2. ｎ×ｎの`int`型2次元配列 m を可変長二次元配列として宣言し，各要素の値を
   `m[i][j] = i*n + j;`
   で設定する。
3. `print_matrix` と `square_matrix`とを呼び出すことで，`m`が表現する行列Mと，行列Mの二乗M^2 と，四乗M^4 を表示する。

Ex5-3（続き）

★提出物：①②③を含むプログラム（ex5_3.c）

- 余裕のある人は，行列を「見栄え良く」表示してみよう.

int len が，行列要素の最大値の桁数であるとき，

```
printf("%*d ", len, data[i]);
```

とすれば，data[i]をlen桁で出力できる.

桁数の計算は，自力で行うか，log10関数を利用しても良い.

```
double log10(double x);
```

log10関数を使うには，

```
#include<math.h>
```

が必要.

この課題の締め切りは
6回目講義の課題と同一
とします.

Ex5-4

- 引き続き，二次元配列を用いた行列操作：逆行列を求める.

① 準備：サイズnのfloat配列を，長さnのベクトルと考え，次の３つの関数を実装しなさい.

- `void vec_mult_s(int n, float *in, float a, float *out);`
  長さnのベクトルinの各要素をa倍し，その結果をベクトルoutに代入する.
- `int vec_div(int n, float *in, float a, float *out);`
  長さnのベクトルinの各要素をaで割り，その結果をベクトルoutに代入する．ただし a=0　のとき，0以外を戻り値として返す.
- `void vec_sub(int n, float *in1, float *in2, float *out);`
  長さnのベクトルin1からベクトルin2を減算し，その結果をベクトルoutに代入する.

② 上記の関数を利用して，行列の逆行列を求める関数

```
int mat_inv(int n, float in[n][n], float out[n][n]);
```

を実装しなさい．inとoutはn×nの行列を表し，inに逆行列が存在するときは，その逆行列をoutに入れて ０を戻り値として返し，逆行列が存在しないときは，0以外（例えば -1）を戻り値として返すこと.

(つづく)

Ex5-4 (つづき)

③ ファイル”matrix.txt”から行列を読み込み，
mat_invを用いて逆行列を計算し，
元の行列と逆行列を表示するプログラムを作りなさい.
逆行列が存在しない場合は，Not invertible　と表示すること.
matrix.txt ファイルは以下の形式とする（数値は一例）.

”matrix.txt”

```
4
1  2  3  4
0  0  0  1
10  32  3  7
5  5  2  0
```

行列のサイズ（int）.
この例では，行列は4×4

各行に，サイズで指定される個数の値が，
スペースで区切られている.
サイズで指定される行数だけ並ぶ.
この例では全てが整数だが，
floatとして扱うこと.

★提出物：①～③を含んだプログラムのソースファイル (ex5_4.c)

※逆行列が存在しない行列も含め，いくつかの行列で試してみること！

Ex5-4 (つづき)

~matsumoto.r.aa/bin/random-matrix.sh
を用いると右のように
matrix.txt をランダムに
生成する.

何度か試してみることで,
正しく動作することを
確認してみよう.

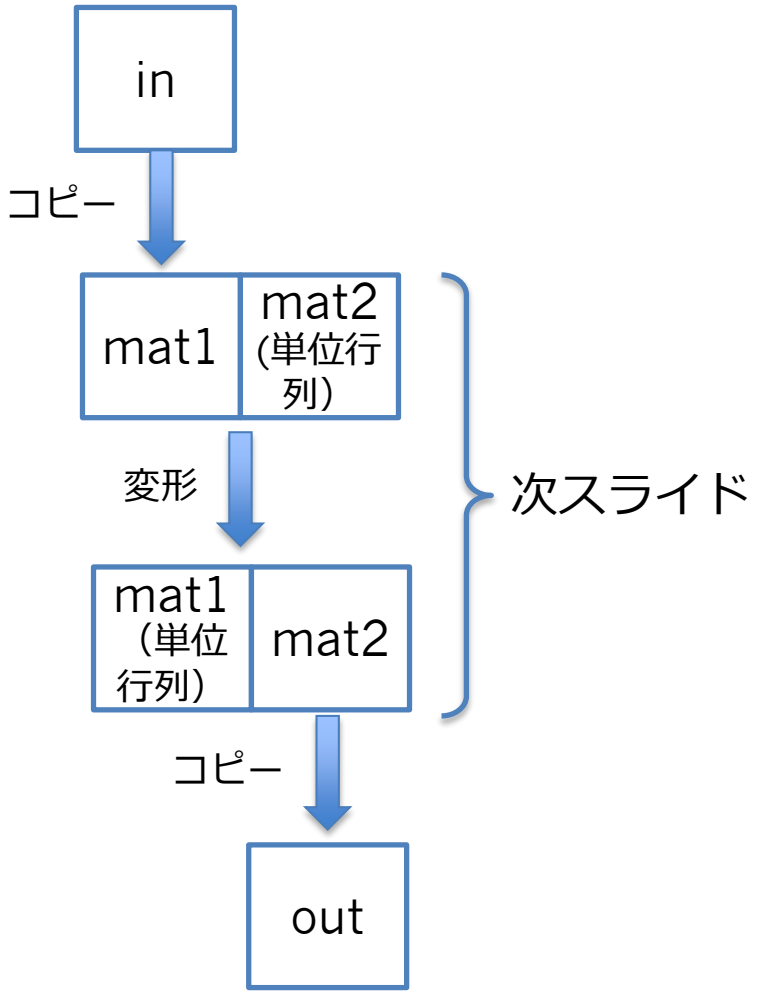
★時間のある人は，以下も見てみよう

- 逆行列に元の行列を掛け算し，単位行列となるかを確認．丸めの誤差により，戻らない可能性あり.
- floatの代わりにdoubleを使うことにより，上記の誤差が小さくなるか？

補足

- 逆行列の求め方：知ってますか？

掃き出し法



補足（つづき）

mat1

$$\begin{pmatrix} 2 & 2 & 5 \\ 3 & 3 & 6 \\ 3 & 2 & 1 \end{pmatrix}$$

mat2

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

1. mat1の1行1列成分(例では 2)で，mat1とmat2の第1行を割る(`vec_div`)

$$\begin{pmatrix} 1 & 1 & 5/2 \\ 3 & 3 & 6 \\ 3 & 2 & 1 \end{pmatrix} \quad \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. matu1の残りの行（例では第２行と第3行）の1列成分(例では３と３)をmat1，mat2の第1行の各成分にかけたもの(`vec_mult_s`)を，残りの行の各成分からそれぞれ引く(`vec_sub`)

$$\begin{pmatrix} 1 & 1 & 5/2 \\ 0 & 0 & -3/2 \\ 0 & -1 & -13/2 \end{pmatrix} \quad \begin{pmatrix} 1/2 & 0 & 0 \\ -3/2 & 1 & 0 \\ -3/2 & 0 & 1 \end{pmatrix}$$

ここまでの手順でmat1の第1列が単位行列の第1列と同じになった.

(つづく)

補足（つづき）

$$\begin{pmatrix} 1 & 1 & 5/2 \\ 0 & 0 & -3/2 \\ 0 & -1 & -13/2 \end{pmatrix} \quad \begin{pmatrix} 1/2 & 0 & 0 \\ -3/2 & 1 & 0 \\ -3/2 & 0 & 1 \end{pmatrix}$$

同様に，１と２の手順を，mat1の２行２列成分～n行n列成分に対して繰り返す（手順１と２の行や列を指していた１を，２，３，…，ｋ，…，ｎと読み替え）．
ただし，もしk行k列成分が0のときは，２の手順の割り算ができないので，
別途，下記の手順を行う

３．（k行k列成分が0のとき；上の例では２行２列成分が0）
第(k+1)行以降で第k列成分が0でない行（m行）を探す（例では第３行）．
mat1, mat2それぞれのk行とm行を入れ換える．

$$\begin{pmatrix} 1 & 1 & 5/2 \\ 0 & -1 & -13/2 \\ 0 & 0 & -3/2 \end{pmatrix} \quad \begin{pmatrix} 1/2 & 0 & 0 \\ -3/2 & 0 & 1 \\ -3/2 & 1 & 0 \end{pmatrix}$$

この状態から，１と２の手順を続ける．
左のmat1が単位行列になったとき、右のmat2が求める逆行列になる．

＊手順３で入れ換える行が見つからない場合には，**逆行列が存在しない．**

補足（つづき）

$$\begin{pmatrix} 1 & 1 & 5/2 \\ 0 & -1 & -13/2 \\ 0 & 0 & -3/2 \end{pmatrix} \quad \begin{pmatrix} 1/2 & 0 & 0 \\ -3/2 & 0 & 1 \\ -3/2 & 1 & 0 \end{pmatrix}$$

実際にこの例題について続けて　k = 2 として手順 1 を実施すると

$$\begin{pmatrix} 1 & 1 & 5/2 \\ 0 & 1 & 13/2 \\ 0 & 0 & -3/2 \end{pmatrix} \quad \begin{pmatrix} 1/2 & 0 & 0 \\ 3/2 & 0 & -1 \\ -3/2 & 1 & 0 \end{pmatrix}$$

次に手順 2 を実施すると（残りの行（第 1 行）の成分も対象となるので注意！）

$$\begin{pmatrix} 1 & 0 & -4 \\ 0 & 1 & 13/2 \\ 0 & 0 & -3/2 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 & 1 \\ 3/2 & 0 & -1 \\ -3/2 & 1 & 0 \end{pmatrix}$$

さらに　k = 3 として手順 1 と 2 を実施すると

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 3 & -8/3 & 1 \\ -5 & 13/3 & -1 \\ 1 & -2/3 & 0 \end{pmatrix}$$

で逆行列が求められた.

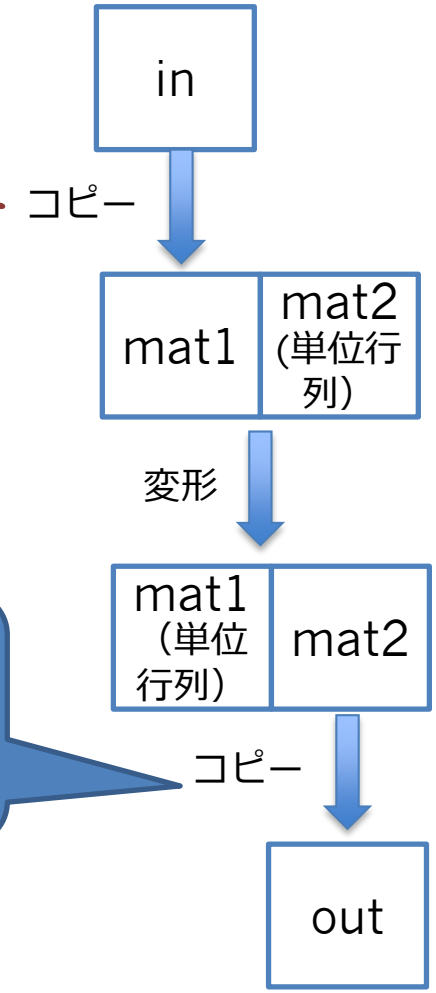
注意!

`mat_inv(n, in, out);`

を実行した時に，in の内容が変わらないようにすること！

（in = out のとき以外）

ここで，一時的に用意した変数にコピーしないと，in の値が変わってしまいます.



mat2を用意せずにもできるが，mat2を用意して最後にoutにコピーすれば，outとinが同じでもＯＫ.（逆行列で上書きされる）

今日の提出物 まとめ

- Ex5-1:
  - ②のプログラム：ex5_1_2.c
  - ③のメモリイメージを描いた図： ex5_1_3.pdf（pdfでなくても良い）
- Ex5-2：
  - ②のプログラム：ex5_2_2.c
- Ex5-3：プログラム　ex5_3.c

③のプログラム　ex5_2_3.c
も提出すれば加点します

- （Ex5-4：プログラム　ex5_4.c ）

この課題は，次回の課題として提出してください.

- ファイル名は指定のものを使うこと.

コピペレポートについて

プログラムや考察などが他の提出者と重複している場合、不正とみなして減点および問い合わせをすることがあります