

第12回ポインタの応用 (ポインタと関数、配列)



1

今回の目標

- 関数呼び出しとポインタの関係を理解する。
- 配列とポインタの関係を理解する。
- 関数間での配列を引き渡し方を理解する。

☆他の関数内の2つの変数の値を交換する関数を作成し、その関数を用いて関数内の2次元配列の2つの行を交換する関数を作成する。

2

関数とポインタ

2つの関数間の引数の受け渡しに、ポインタは重要な役割を果たす。
関数の仮引数や戻り値として、アドレスを用いることができる。

書式

```
戻り値の型 関数名(仮引数の型 * 仮引数名)
{
    return 戻り値;
}
```

例

```
void func_ref(int *p)
{
    return;
}
```

```
int main()
{
    func_ref(&i);
    a=i;
}
```

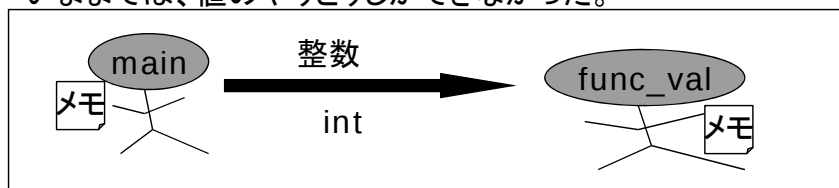
アドレスを渡して関数を呼び出す方法。この場合の仮引数はpで、intへのポインタ型。(int *)型であって、int型ではないことに注意。

仮引数がポインタ型の場合、呼び出す際にはアドレスを指定しなければならない。

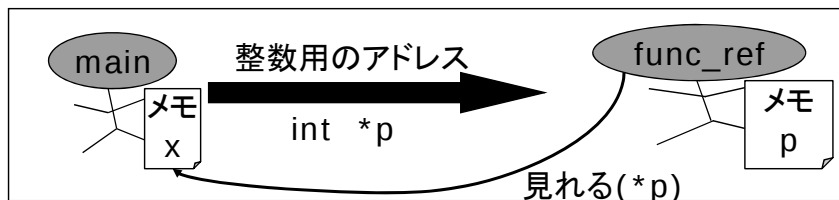
3

イメージ

いままでは、値のやりとりしかできなかった。



ポインタの仮引数を用いると、アドレスのやりとりができる。



アドレスのやりとりをすると、
他の関数内のローカル変数の内容を変更できる。

4

値による呼び出し(call by value)

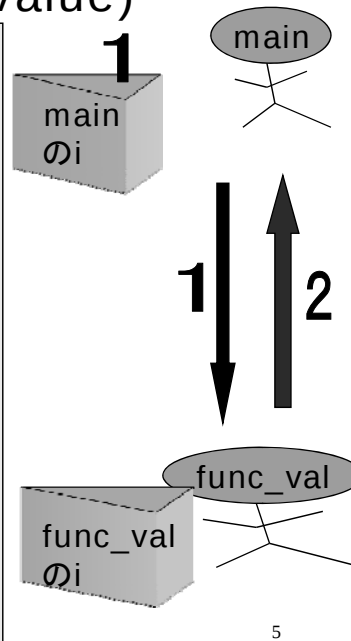
```

/*test_cbv.c 値による呼び出し実験*/
#include <stdio.h>
int func_val(int);
int main()
{
    int i=1;
    int j=0;
    printf("i= %d j= %d\n",i,j);

    j=func_val(i);

    printf("i= %d j= %d\n",i,j);
    return 0;
}
int func_val(int i)
{
    i++;
    return i;
}

```



参照による呼び出し(call by reference)

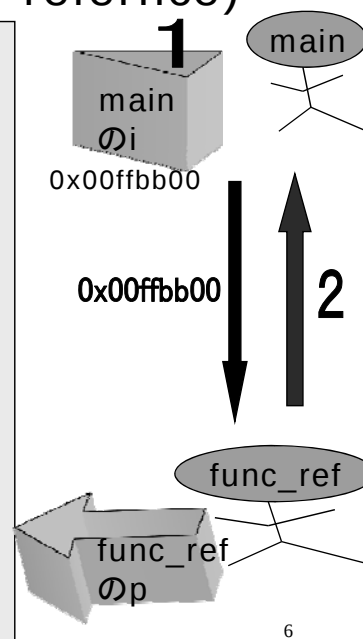
```

/*test_cbr.c 参照による呼び出し実験*/
#include <stdio.h>
int func_ref(int *p);
int main()
{
    int i=1;
    int j=0;
    printf("i= %d j= %d\n",i,j);

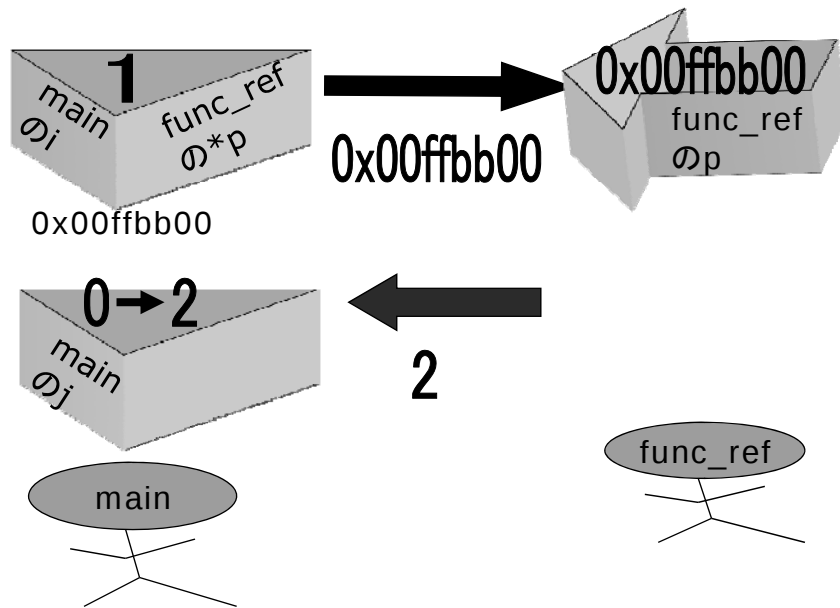
    j=func_ref(&i);

    printf("i= %d j= %d\n",i,j);
    return 0;
}
int func_ref(int *p)
{
    (*p)++;
    return (*p);
}

```



イメージ



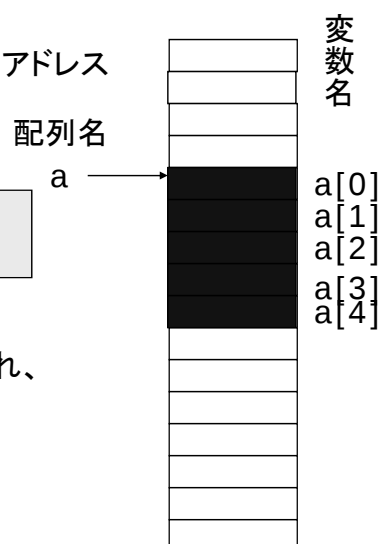
配列とポインタ

C言語では、配列名は先頭の要素のアドレスを指す。

例えば、

```
#define MAX 5
char a[MAX];
```

と宣言するとアドレスの連続した5個のchar変数がメモリ上に確保され、その先頭のアドレスがaにはいる。つまり、aには「&a[0]の値(アドレス)」が保持されている。



char a[MAX];
char *p;
p=a;

配列名

a

p

ポインタ
(配列の先頭要素の
アドレスを持つ)

とすると、
a[i]とp[i]は同じ変数(配列要素)
を表す。

a[0] p[0]
a[1] p[1]
a[2] p[2]
a[3] p[3]
a[4] p[4]

9

イメージ

char a[3];
a[0]='A';
a[1]='B';
a[2]='C';

A B C

a[0] a[1] a[2]

0x00ffaa00 0x00ffaa01 0x00ffaa02

0x00ffaa00

a

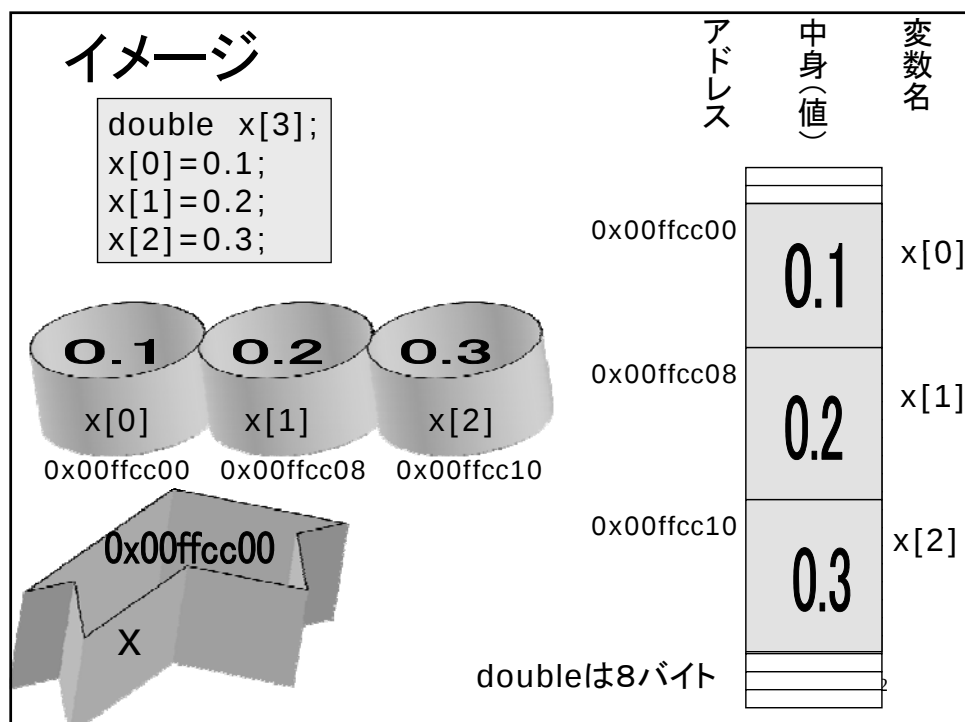
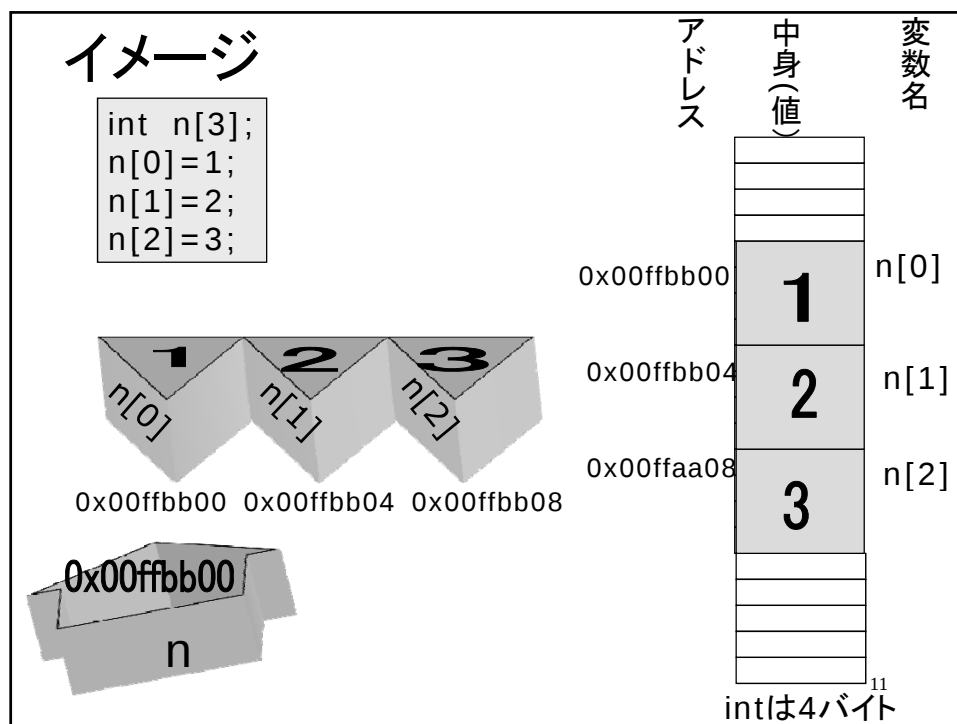
アドレス

中身(値)

変数名

0x00ffaa00 A a[0]
0x00ffaa01 B a[1]
0x00ffaa02 C a[2]

10
charは1バイト



練習2

```
/* pointer_array.c ポインタと配列実験 コメント省略 */
#include <stdio.h>
#define MAX 5
int main()
{
    int i;
    int n[MAX]; /*データを入れる配列*/
    int *p; /*上の配列の先頭を指すポインタ*/

    for(i=0;i< MAX;i++)
    {
        n[i]=i+1;
    }

    /* 次に行く */
}
```

13

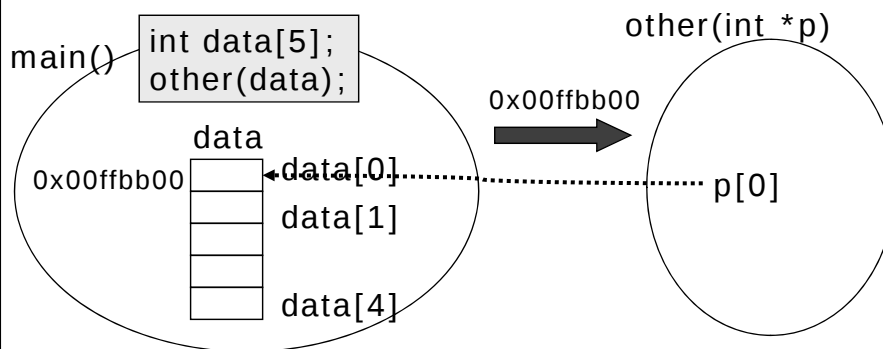
```
/* 続き */
p=n;
printf("nの値は %p ¥n",n);
printf("n[0]のアドレスは%p ¥n",&n[0]);
printf("pの値は%p ¥n",p);
printf("¥n¥n");

printf("&n[i] n[i] p[i]¥n");
for(i=0;i<MAX;i++)
{
    printf("%p %d %d ¥n",
        &n[i],n[i],p[i]);
}
return 0;
}
```

14

関数間での配列の引き渡し1

他の関数に配列(`data[**]`)の先頭要素のアドレス(`data`,すなわち、`&data[0]`)を渡すことができる。配列名が配列の先頭アドレスを保持していることに注意する。

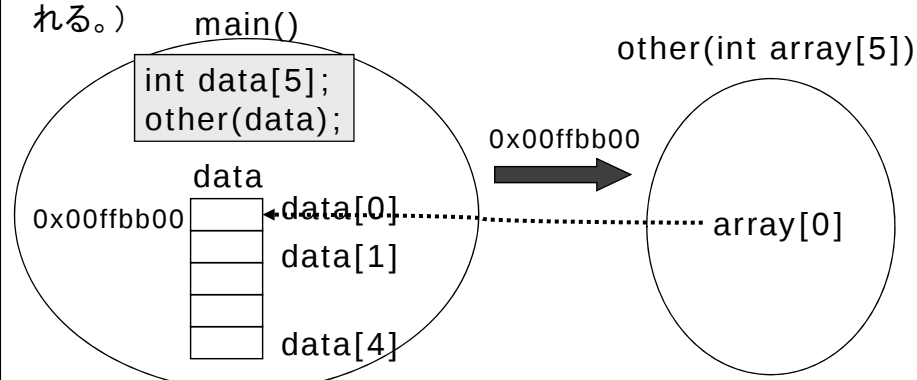


受け取った他の関数の中では、配列の先頭要素のアドレスの
入ったポインタを配列名のように使うことができる。

15

関数間での配列の引き渡し2

受け取る側では、仮引数に配列を記述しても良い。
この場合、引き渡されたアドレスが、引数の配列要素の先頭
アドレスになる。(すなわち、「`array = data`」の代入が行なわ
れる。)



注意:

呼び出し側の配列の内容が書き換わるかもしれない。
十分に注意して関数を設計すること。

16

練習3

```
/*test_sendarray.c*/
#include <stdio.h>
#define MAX 10
void print_array(int n,int p[MAX]);
void write_array(int n,int p[MAX]);
int main()
{
    int i;
    int n;
    int data[MAX];

    for(i=0;i<MAX;i++)
    {
        data[i]=i;
    }

    printf("n=?");
    scanf("%d",&n);
    /* 次が続く */
```

17

```
/*続き*/
print_array(n,data);

/*内容変更*/
write_array(n,data);

print_array(n,data);
return 0;
}
/*main関数終了*/

/*次が続く*/
```

18

```
/*    続き    */
void print_array(int n,int p[MAX])
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("data[%d] = %d ¥n",i,p[i]);
    }
    printf("¥n");
    return;
}

/*    次に行く    */
```

注意:
呼ばれる方では、配列の最後に気を付ける事。

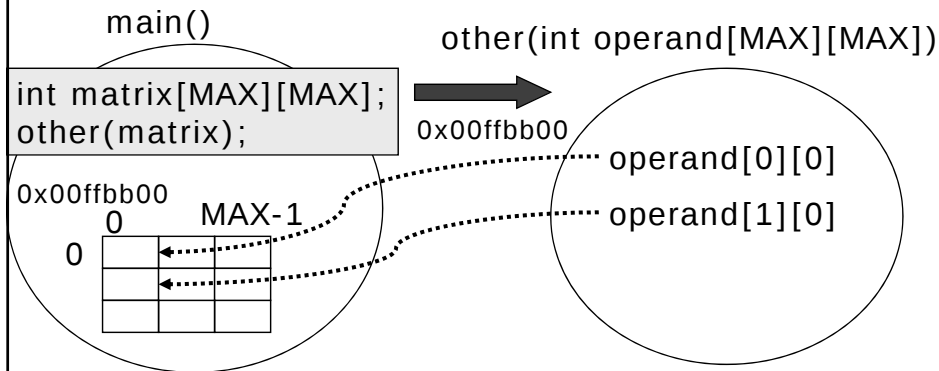
19

```
/*    続き    */
void write_array(int n,int p[MAX])
{
    int i;
    for(i=0;i<n;i++)
    {
        p[i]=(p[i])*2;
    }
    return;
}
/*全てのプログラム終了*/
```

20

関数間での2次元配列の引き渡し

2次元配列では、先頭アドレスの他に大きさも指定する必要がある。



注意:

呼び出し側の配列の内容が書き換わるかもしれない。
十分に注意して関数を設計すること。

21

他の関数内の2次元配列の2つの行を交換する

```
/*
    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: arg_matrix.c
    実行ファイル: arg_matrix
    説明: 関数間での2次元配列の受け渡しの効果を調べるための
          プログラム。他の関数内の行列(2次元配列)の2行を交換
          する関数を作成する。

    入力: 標準入力からGYO×RETU個のint値を行列の成分として
          受け取る。ただし、同じ行の値が連続しているとする。
          さらに、交換したい行番号を2つ標準入力から受け取る。
    出力: 標準出力に、行交換前の行列および行交換後の行列を
          出力する。
*/
    次のページに続く    */
```

22

```
/*      続き      */
/* ヘッダファイルの読み込み */
#include <stdio.h>
/* マクロの定義 */
#define GYO 3
#define RETU 4

/* プロトタイプ宣言 */
void print_matrix(int matrix[GYO][RETU]);
/* 配列matrixを表示する関数 */
void swap(int *p,int *q);
/* 参照呼び出しを用いて、
   2つの変数の値を入れ替える関数 */
void swap_rows(int matrix[GYO][RETU],int row1,int row2);
/* 配列matrixのrow1行とrow2行を入れ替える関数 */

/*      次のページに続く      */
```

```
/* 続き */

/* main関数開始 */
int main()
{
    /* ローカル変数宣言 */
    int matrix[GYO][RETU]; /* データ入力用配列 */
    int i; /* 行番号を制御、ループカウンタ */
    int j; /* 列番号を制御、ループカウンタ */
    int row1; /* 交換したい行番号1 */
    int row2; /* 交換したい行番号2 */

    /* 続く main関数 */
```

```
/*続き main関数内*/
/*入力処理*/
printf("行ごとに入力して下さい。¥n");
for(i=0;i<GYO;i++){
    for(j=0;j<RETU;j++){
        scanf("%d",&matrix[i][j]);
    }
}

printf("交換したい行番号を2つ入力して下さい。¥n");
printf("(0以上 %d未満)",GYO);

scanf("%d",&row1);
scanf("%d",&row2);
if(row1<0||GYO<row1||row2<0||GYO<row2){
    printf("不正な入力です。");
    return -1;
}

/*続く main関数*/
```

```
/*続き main関数内*/
/*交換前の行列の表示*/
printf("交換前¥n");
print_matrix(matrix);

/*行列の行交換*/
printf("¥n%3d 行と%3d行を交換中¥n¥n",row1,row2);
swap_rows(matrix,row1,row2);

/*交換後の行列の表示*/
printf("交換後¥n");
print_matrix(matrix);

/*正常終了*/
return 0;
}
/*main関数終了*/

/*続く*/
```

```
/*続き print_matrixの定義開始*/
/*行列を表示する関数
仮引数matrix:表示したい行列
戻り値:なし(void)
*/
void print_matrix(matrix[GYO][RETU]){
    /*ローカル変数宣言*/
    int    i;        /*行番号を制御、ループカウンタ*/
    int    j;        /*列番号を制御、ループカウンタ*/

    /*表示処理*/
    for(i=0;i<GYO;i++){
        for(j=0;j<RETU;j++){
            printf("%3d",matrix[i][j]);
        }
        printf("¥n");
    }

    return;
}
/*print_matrix関数定義終了 続く*/
```

```
/*続き 関数swapの定義開始*/
/*他の関数の2変数の値を入れ替える関数
仮引数 p,q:交換すべき整数型の変数のアドレス
(参照呼出なので、呼び出し側ではアドレスを指定する)
戻り値:なし(void)
*/
void swap(int *p,int *q){
    /*ローカル変数宣言*/
    int    temp; /*交換のために値を一時的の保存する*/

    /*交換処理*/
    temp=*p;
    *p=*q;
    *q=temp;

    return;
}
/*関数swapの定義終了 続く*/
```

```
/*続き 関数swap_rowsの定義開始*/
/*行列の2行を入れ替える関数
仮引数matrix:GYO×RETUの行列
仮引数row1,row2:交換すべき行番号
(呼び出された時点で(0<=row1<GYO) && (0<=row2<GYO)
を満たす。)
戻り値:なし(void)*/
void swap_rows(matrix[GYO][RETU],int row1,int row2){
    /*ローカル変数宣言*/
    int    j;        /*列番号を制御、ループカウンタ*/

    /*交換処理*/
    for(j=0;j<RETU;j++){
        swap(&matrix[row1][j],&matrix[row2][j]);
    }

    return;
}
/*関数swap_rowsの定義終了*/
/*プログラム(arg_matrix.c)の終了*/
```

実行例

```
$/arg_matrix<arg_matrix.in
行ごとに入力して下さい。
交換したい行番号を2つ入力して下さい。
(0以上%d未満)
交換前
 1  2  3  4
 5  6  7  8
 9 10 11 12

 0行と 2行を交換中

交換後
 9 10 11 12
 5  6  7  8
 1  2  3  4
$
```