［14］　ポインタ

1．ポインタとは？

☆データ（数字，文字）が保存されているメモリの場所（番地，アドレス）.

2．ポインタを扱うための演算子と変数

・アドレス演算子　＆　：変数のメモリ上のポインタを取り出す.

例）　&a;　　変数aのポインタ

・ポインタ演算子　＊　：ポインタにより示されるメモリの内容（変数の値）を取り出す.

例）　*pa　　ポインタ変数paが示す変数の値.

・ポインタ変数　　：ポインタを保存するための変数.

参照する変数型　＊変数名

例）
```
int *pa;
```
整数型変数を参照するためのポインタ変数pa
```
double *pb;
```
実数型変数を参照するためのポインタ変数pb
```
char *pc;
```
文字型変数を参照するためのポインタ変数pc

3．ポインタによるデータ参照

例）
```
#include <stdio.h>
int main(void){
int a;
```
整数変数aを宣言.
```
int *pa;
```
整数型ポインタ変数paを宣言.
```
a=10;
pa=&a;
```
変数aのアドレス（ポインタ）をpaに代入する.
```
printf("%d \n",&a);
```
変数aのポインタを表示.
```
printf("%d \n",pa);
```
ポインタ変数paの値（アドレス）を表示.
```
printf("%d \n",*pa);
```
paが指し示す示すメモリの内容を表示.
```
return 0;
}
```

［15］　関数（2）

1．値による呼び出し　Call by Value　　**変数の値を渡す**

☆戻り値が１つで良い場合に使用する.

2．参照による呼び出し　Call by Reference　　**ポインタを渡す**

☆戻り値が２つ以上必要な場合や，配列変数全体を引き渡す場合に使用する.

```
型　関数名（型　ポインタ変数,...）{
    文
  *ポインタ変数 = 値；          ポインタが示す内容を変更することで値を返す.
}
```

☆関数の仮引数にはポインタ変数を宣言する.

☆関数を呼び出す場合には，引数に変数のポインタを引き渡す.

例)

```
int main(void){
  int m=3, j=1, k=2;
  func(m, &j, &k);                  変数mの値と変数j，kのポインタをわたす.
  printf("%d %d %d\n",m,j,k);
  return 0;
}
void func(int c,int *a,int *b){     main関数中の変数j，kのポインタが,
                                    ポインタ変数a，bにコピーされる.
    *a = c + *a;                    ポインタa，bが示すメモリの内容が変更される.
    *b = c * *b;                    すなわちj，kの値が変わる.
    }
```

※値による呼び出しと参照による呼び出しを組み合わせた例.

```
void main(void){
   int m=3, j=1, k=2;
   x=func(m, &j,&k);                 変数mの値と変数j，kのポインタをわたす.
   printf("%d %d %d %d\n",m,j,k);
   }
int func(int c, int *a, int *b){
    *a = c + *a;
    *b = c * *b;
    return *a * *b; }                return文で戻り値を返すこともできる.
```

［16］　関数（3）

1．配列変数を渡す

・配列変数全体を渡す場合には，参照による引渡しを用いる．

○1次元配列の場合

int a[100];

と定義されていると，aはポインタ変数と見ることができるので，aの内容（ポインタ）を参照による引渡しを用いることができる．

```
void main(void){
    int x, a[100];
    x=func(a);                    1 次元配列変数aのポインタを渡す.
    ....
int func(int b[ ]){               配列変数のポインタを受け取る仮引数.
              ↓
int func(int *b){                 また，このように書いてもよい.
```

※配列の要素数は，関数のほうではわからないので，引数で引き渡す必要がある．

```
void main(void){
    static int a[3]={10,17,35};
    x=add(a,3);                   1 次元配列変数aのポインタを渡す.
    ....
int add(int b[ ], int n){         配列要素のn個の合計を求める関数
    int i, sum=0;
    for(i=0;i<n;i++)              配列要素0からn-1までを合計する.
    sum+=b[i];
    return sum;                   結果は戻り値で戻している.
}
```

○2次元配列の場合

```
void  main(void){
    int x, a[5][10];
     x=func(a);                    ←2 次元配列変数aのポインタを渡す.

void func(int b[ ][10]){          ←2 次元配列では，列要素サイズを宣言する.
              ↓
void func(int *b, int m, int n){
              またこのように書けるが，行，列の数（ここではm，n）を渡す必要がある.
```