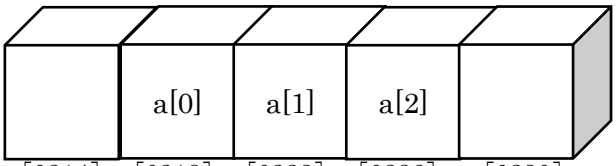
ソフトゼミ∀　第7回

# ポインタ

ポインタと配列には実は密接な関係があります。それについてちょっと解説します。

## ✤配列とアドレスと関数と

配列とは実はメモリ的に連続した位置に保存されています。お隣さんです。
つまり int a[3];と宣言すると、イメージとして次のような感じでメモリに変数が割り当てられます。




【メモリのイメージ】

さて、ここでこの配列を関数に渡してみます。void func(int a[]);って関数があったとしたらfunc(a);のように配列を渡します。
ここで、注意しておきたいのは「配列の要素の値を渡しているわけではない」点です。
配列ごと渡している、もっと言うと配列のアドレスを渡しているので、渡した先で配列の要素の値が変わると、main文や他の場所でも値が変わります。
何を言ってるかわからないと思いますが、論より証拠です。次のサンプルコードを打ち込んでみてください。

余白です。
サンプルコードは次のページへ

```
#include<stdio.h>
#define N 10
void nijou(int a[],int n){
  int i;
  for(i = 0;i < n;i++)
    a[i] *= i;              // a[i] = a[i] * i;と同じ意味です
}
int main(void){
  int a[N],i;
  for(i = 0;i < N;i++){
    a[i] = i;
    printf("a[%d]:%d¥n",i,a[i]);
  }
  printf("¥n");
  nijou(a,N);
  for(i = 0;i < N;i++)
    printf("a[%d]:%d¥n",i,a[i]);

  return 0;
}
```

どうでしょうか。関数にアドレスを渡したときと同じようにmain文の中でも配列の中身が変わったと思います。実は、配列を渡すときもアドレスを渡しているのです。
渡しているのは先頭のアドレス、つまり&a[0]で、メモリ上で連続した位置に存在しているので、一つ隣のメモリを見てあげると次のa[1],a[2]を参照することができるのです。
これをポインタを使って同じことをすると次のようになります。

```
void nijou(int *a,int n){
  int i;
  for(i = 0;i < n;i++)
    *a++ *= i;
}
```

さっきの nijou 関数をこれに差し替えてみてください。これでもさっきと同じ動作をするはずです。a++によってポインタの示す場所を動かしながら中身の値を変えています。

これらを踏まえて次のポインタと関数の問題を解いてみてください。

## ✤問題

1. 誰か問題考えてください…
2.