C言語講座
第5回

ポインタ

今回の内容である、ポインタはC言語習得の最難関の一つである。

が、同時にC言語の特徴をフルに活かすことができるのもポインタだ。

今日だけで理解しようとしなくても良い。

だが、改めて書いておく。

分からないことがあったら質問するように。

アドレス(address)

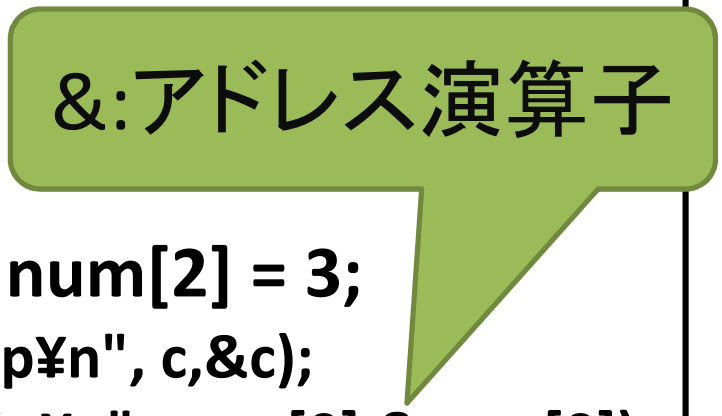
今まで扱ってきたintやcharなどの変数は、全て一時記憶としてメモリ上に記憶される。

その、保存された“番地”のことをアドレスと呼ぶ。

○プログラミング実践

アドレスを表示する

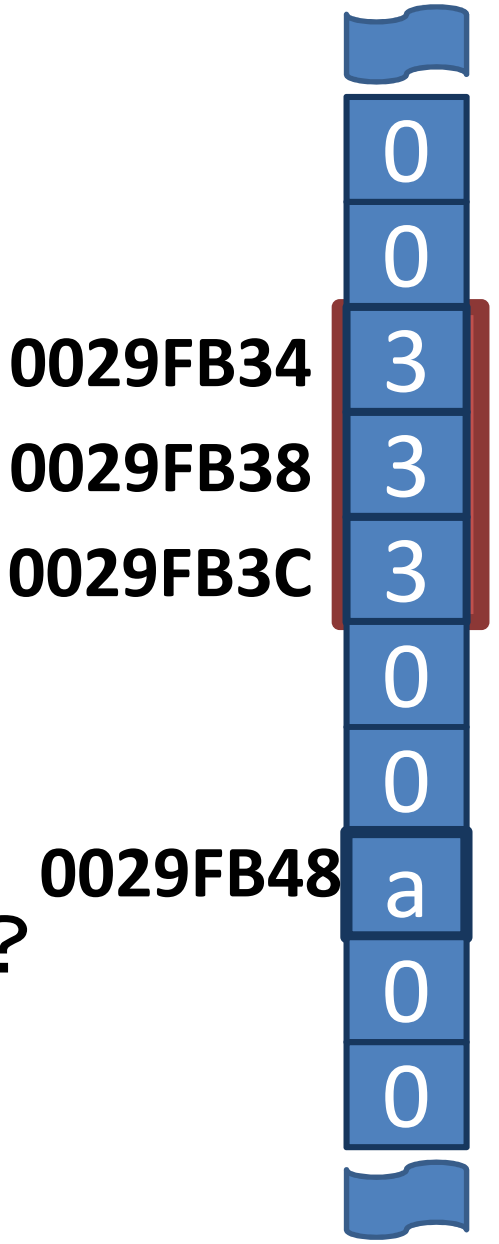
```
#include <stdio.h>
void main(void)
{
    char c = 'a';
    int num[3];
    num[0] = 3; num[1] = 3; num[2] = 3;
   printf("文字:%c 、アドレス:%p¥n", c,&c);
   printf("数値:%d 、アドレス:%p¥n", num[0],&num[0]);
   printf("数値:%d 、アドレス:%p¥n", num[1],&num[1]);
   printf("数値:%d 、アドレス:%p¥n", num[2],&num[2]);
}
```



イメージとしては、こんな感じ(例)

配列で宣言した変数の、
それぞれの要素のアドレスを
見比べてみよう。

第一回の資料にも書いたけど、
int型の大きさは4byteだったよね？



ポインタ(pointer)

ポインタとは、“変数の、メモリ上のアドレス情報を格納したもの”だ。

内容的には、前ページの&(アドレス演算子)が示すものと同じものであり、&が演算子であるのに対し、ポインタは変数の一種のようなものである。

〇構文

```
データ型 *変数名;
```

○プログラミング実践

アドレスを表示する

```
#include <stdio.h>
void main(void)
{
    int  x = 5;
    int  *p = &x;
    printf("数値:%d 、アドレス:%p¥n", x,p);
    printf("アドレス:%p¥n", &p);
}
```

アドレスを表示する

ポインタを利用した関数

アドレスを表示したところで、実際何の役にも立たない。

・・・という訳で、ポインタの有用な点の一つとして、ポインタを利用した関数について説明していこうと思う。

が、その前に、実際の例を次ページに乗せてみた。

何が違うのか、考えてみよう。

○プログラミング実践

```
#include "stdio.h"
void swap(int* x, int* y)
{
    int  tmp = *x;
    *x = *y;
    *y = tmp;
}
void main(void)
{
    int  a = 3, b = 9;
    printf("a=%d, b=%d¥n", a,b);
    swap(&a, &b);
    printf("a=%d, b=%d¥n", a,b);
}
```

x : アドレス(例 : 0029FB38)
*x : アドレスが指す値(例 : 3)

2変数の同時宣言

さて、分かっただろうか。

二つの変数の値を交換するプログラムなのだが、swap関数の引数部分がポインタで宣言されていた。

そのアドレスが示すオブジェクトを交換することで、値を交換しているのだ。

値を交換するだけなら、ポインタ要らなくね？
とか思ってる人は、↓を書いてみて。

```
#include "stdio.h"
void swap(int x, int y) {
   int  tmp = x;
   x = y;
   y = tmp;
}
void main(void) {
   int  a = 3, b = 9;
   printf("a=%d, b=%d¥n", a,b);
   swap(a, b);
   printf("a=%d, b=%d¥n", a,b);
}
```

関数中の引数は、代入ではなくてコピーしているにすぎないので、他の関数からは操作出来ない。

これは“オブジェクトの独立性”うんぬんの話になり、ややこしいから、

そういうものなんだ、とだけ理解していてほしい。

○演習

３つの要素を持つint型配列を、昇順に並び代える関数sort3 を完成させ、プログラムが正しく動作するようにしてください。

```
#include <stdio.h>
void swap(int* x, int* y) {
   int tmp = *x;
   *x = *y;
   *y = tmp;
}
void sort3(int* n0, int* n1, int* n2) {

}
void show_3_nums(char *s, int n0, int n1, int n2) {
                printf("%s:{%d, %d, %d}¥n", s, n0, n1, n2);
}
void main(void) {
                int n0 = 10, n1 = 5, n2 = 7;
                show_3_nums("sort前", n0, n1, n2);
                sort3(&n0, &n1, &n2);
                show_3_nums("sort後", n0, n1, n2);
}
```

○解答例

```
void sort3(int* n0, int* n1, int* n2) {
        if (*n1 < *n0) { swap(n0, n1); }
        if (*n2 < *n1) { swap(n2, n1); }
        if (*n1 < *n0) { swap(n1, n0); }
}
```

swap関数に渡す引数はintのポインタ型であり、n0~n2自身がポインタであるため、&はいらない。

○演習２（やる気があるなら…）

以下は現在日付から、指定された日数後の日付を取得するプログラムです。

set_date関数を完成させ、プログラムが正しく動作するようにしてください。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

void get_now(int* year, int* month, int* day) {
        time_t now = time(NULL);
        struct tm tm_now = (*localtime(&now));

        *year = tm_now.tm_year + 1900;
        *month = tm_now.tm_mon + 1;
        *day = tm_now.tm_mday;
}
```

○演習２ (続き...)

```
void show_date(char* s, int year, int month, int day)
{
        printf("%s:%d年%d月%d日¥n", s, year,month,day);
}

void set_date(int shift, int* year, int* month, int* day)
{




}
void main(void)
{
        int year, month, day;
        int shift = 31;
        char s[128], szshift[16];
        get_now(&year, &month, &day);
        show_date("今日", year, month, day);
        set_date(shift, &year, &month, &day);
        itoa(shift, szshift, 10); strcpy(s, szshift); strcat(s, "日後");
        show_date(s, year, month, day);
}
```

○解答例2

```
// うるう年判定
int is_Leap(int year)
{
	if (year % 4 == 0) {
		if (year % 100 == 0 && year % 400 != 0) {
			return 0;
		}
		return 1;
	}
	return 0;
}
```

○解答例2 (続き...)

```
// その月の最終日を返す
int  end_of_dayOfMonth(int year, int month) {
        switch (month) {
            case 1: case 3: case 5: case 7:
            case 8: case 10: case 12:
                return 31;
            case 4: case 6: case 9: case 11:
                return 30;
            case 2:
                if (is_Leap(year) == 0) { return 29; }
                else { return 28; }
        }
        return -1;
}
```

○解答例2 (続き...)

```
// shift日後の日付に書きかえる関数
void set_date(int shift, int* year, int* month, int* day) {
        int end_day;        // end_of_dayOfMonthで返す値の保持用
        *day += shift;      // shift日ずらす
        while (1 == 1) {
                end_day = end_of_dayOfMonth(year, *month);
                // 日にちが不自然(1月63日など)でなければ抜ける
                if (*day <= end_day) { break; }
                // 不自然な場合、調整する
                *day -= end_day;  (*month)++;
        }

        // 計算しやすいように、monthの値を0~11にする
        (*month)--;
        // 月が不自然場合(27月など)、調整する
        *year += (*month / 12);  *month %= 12; (*month)++;
}
```