第３章
ポインタについて

ポインタに関する質問をたくさんいただきました。ポインタの習得というハードルは、Ｃ言語の初学者にとって永遠のテーマともいうべきもののようです。

本章では、基礎的かつ重要であるにもかかわらず、あまり正しく理解されていないと思われる事項を中心に学習します。

3-1　ポインタとアドレス

まずは、ポインタの基本を学習します。

アドレスとアドレス演算子

オブジェクトは、記憶域の一部として存在します。オブジェクトが格納されている**番地**が**アドレス**（*address*）です。

Fig.3-1 は、*x*，*y*，*z* が 100 番地、102 番地、106 番地を先頭に格納されている様子です。

アドレス演算子（*address operator*）と呼ばれる & 演算子で、アドレスを調べられます。

オブジェクトのアドレスを取得して表示するプログラムを **List 3-1** に示します。

```
int   x;
int   y;
long  z;
```

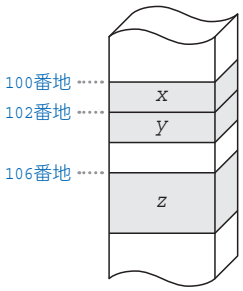

Fig.3-1　記憶域とオブジェクト

▶ アドレス演算子は、**単項 & 演算子**（*unary & operator*）とも呼ばれます。

List 3-1

```
/*
    オブジェクトのアドレスを表示
*/

#include  <stdio.h>

int main(void)
{
    int   x;
    int   y;
    long  z;

    printf("xのアドレス＝%p\n", &x);
    printf("yのアドレス＝%p\n", &y);
    printf("zのアドレス＝%p\n", &z);

    return (0);
}
```

実行結果一例

```
xのアドレス＝100
yのアドレス＝102
zのアドレス＝106
```

▶ 実行結果として表示される値は、処理系、環境、実行するタイミングなどで異なりますし、実際の物理アドレスであるとは限りません（これ以降のプログラムでも同様です）。

書式指定 "%p" による出力の形式は、処理系によって異なります。4 桁から 16 桁程度の 16 進数で出力が行われるのが一般的です。

*

もちろん、配列内の各要素のアドレスも、アドレス演算子によって取得できます。プログラム例を **List 3-2** に示します。

List 3-2

```
/*
    配列内の要素のアドレスを表示
*/

#include  <stdio.h>

int main(void)
{
    int  x[5];

    printf("x[0]のアドレス＝%p\n", &x[0]);
    printf("x[1]のアドレス＝%p\n", &x[1]);
    printf("x[2]のアドレス＝%p\n", &x[2]);
    printf("x[3]のアドレス＝%p\n", &x[3]);
    printf("x[4]のアドレス＝%p\n", &x[4]);

    return (0);
}
```

実行結果一例

```
x[0]のアドレス＝100
x[1]のアドレス＝102
x[2]のアドレス＝104
x[3]のアドレス＝106
x[4]のアドレス＝108
```


アドレス演算子は、**register** 記憶域クラス指定子（p.16）を伴って宣言されたオブジェクトには適用できません。

通常の記憶域ではなく、レジスタに格納される可能性があるからです。

重要　**register** 記憶域クラス指定子を伴って宣言されたオブジェクトのアドレスは取得できない。

したがって、**List 3-3** に示すプログラムは、コンパイル時にエラーとなって、実行できません。

List 3-3

```
/*
    register宣言されたオブジェクトのアドレスは取得できないことを確認
*/

#include  <stdio.h>

int main(void)
{
    register int  x;

    printf("xのアドレス = %p\n", &x);   /* エラー */

    return (0);
}
```

実行結果

コンパイル時にエラーとなり実行できません。

■ Column 3-1　C++ における register 記憶域クラス指定子とアドレス演算子

C++ では、**register** 記憶域クラス指定子を伴って宣言されたオブジェクトのアドレスも取得できます。したがって、**List 3-3** に示すプログラムは正しく動作します。