解答編　第 11 章

ポインタの取り扱い

演習問題 11.1　文法事項

1）アドレス演算子とポインタ演算子について説明しなさい。

解答例　アドレス演算子は、変数などの演算項に作用してそのアドレスを返します。アドレス演算子は、「**&**(アンパサンド)」で表されます。

ポインタ演算子は、ポインタ変数などの演算項に作用して、演算項が示すアドレスに格納されている値を返します。ポインタ演算子は、「*****(アスタリスク)」で表されます。

2）プログラム中で、配列名は何を表していますか。簡潔に答えなさい。

解答例　プログラム中で，配列名は配列の先頭のアドレスを表します。このため、配列名とポインタ演算子を組み合わせて、ポインタの操作によって配列を取り扱うこともできます。

3）ポインタ変数に対してインクリメント演算子を適用するとどのようなことが起こりますか。配列と関連づけて説明しなさい。

解答例　C 言語では、ポインタは単純なアドレスではなく、それが指し示すデータのデータ型の大きさを知っています。このため、ポインタ変数をインクリメントすると、アドレスの値が単純に 1 増加するのではなく、ポインタ変数が指すデータ型の大きさに合わせてアドレスの値が増加します。

従って、配列 **ary[]** とポインタ変数 **ptr** が同じデータ型を対象としており、**ptr** のアドレスが **ary** と同じ場合、どんなデータ型に対しても **ary[2]** と ***(ptr + 2)** は、同じデータを表すことになります。

4）ポインタ変数も配列名も、プログラム中ではポインタの値 (アドレス) を表します。両者の違いについて簡潔に述べなさい。

解答例　ポインタ変数は、初期化付き宣言でなければ、宣言された時点では値が定まっていません。プログラム中では、アドレスを代入したり、演算を行って値を変更することができます。

一方、配列名は、配列が宣言された時点で、配列の先頭アドレスを値として持っています。プログラム中では、配列名にアドレスを代入したり、演算を行って値を変更することができません。これは、「配列名は配列の先頭アドレスを示す」と約束されているだけで、アドレスを格納する領域が確保されているのではないからです。「配列名は、ポインタ定数のようなもので、変更できない」と考えておくとよいでしょう。

演習問題 11.2

プログラム例10.4とプログラム例11.3を参考にして、文字列を接続するプログラムを、ポインタ変数を用いて作りなさい。

解答例　解答例を解答プログラム 11.1 に示します。

文字列を接続する処理の考え方は、本文のプログラム例 10.4 の説明を参照してください。解答プログラム 11.1 は、プログラム例 10.4 の配列操作を全てポインタ変数による操作に書き換えたものとなっています。

プログラム例 10.4 と解答プログラム 11.1 を比較すると、後者の方がシンプルに見えます。この例のように、配列要素を順番にアクセスするような問題では、同じ操作をするにしても、ポインタを用いた方がシンプルになります。

解答プログラム 11.1　文字列の接続プログラム

```
/************************************************
    p11_1.c
    文字列の接続
    Tadaaki Shimizu    2006.10.22
************************************************/
#include <stdio.h>

main()
{
    char aStr[100];      // 入力される文字列
    char bStr[100];      // 複写先の文字列
    char *aPtr, *bPtr;   // ポインタ変数

    printf(" Input string 1 >> ");
    scanf("%s", aStr);
    printf(" Input string 2 >> ");
    scanf("%s", bStr);

    // bStr[] 文字列の終端を探す
    for(bPtr = bStr; *bPtr != '\0'; bPtr++)
        ;     // 空文

    // 文字列のコピー
    for(aPtr = aStr; *aPtr != '\0'; aPtr++, bPtr++)
        *bPtr = *aPtr;
    *bPtr = '\0';

    printf("String :%s\n", bStr);
    exit(0);
}
```

演習問題 11.3

プログラム例10.5とプログラム例11.3を参考にして、文字列を比較するプログラムを、ポインタ変数を用いて作りなさい。

解答例 解答例を解答プログラム 11.2 に示します。

このプログラムは、本文中のプログラム例 11.5 をポインタを使って書き換えただけです。ただし、**for** 文を **while** 文に変えてあります。その理由は、**for** 文のフィールドに書く式が増えすぎて、**for** の行が長くなりすぎたためです。ついでに、**if** 文と **break** 文の組み合わせで行っていた条件判断を、**while** の条件部に組み込んでみました。

このように、プログラムを書き換えたために、プログラム例 10.5 と解答プログラム 11.2 はかなり違って見えるかもしれません。しかし、その動作は全く同じです。同じ処理を記述するにも、その記述方法が無数にあり、そこに工夫の余地があることに注意して下さい。

解答プログラム 11.2　文字列の比較プログラム

```
/************************************************
    p11_2.c
    文字列の比較
    Tadaaki Shimizu    2006.10.22
************************************************/
#include <stdio.h>

main()
{
    char aStr[100];      // 入力される文字列1
    char bStr[100];      // 入力される文字列2
    char *aPtr, *bPtr;   // ポインタ変数

    printf(" Input string 1 >> ");
    scanf("%s", aStr);
    printf(" Input string 2 >> ");
    scanf("%s", bStr);

    aPtr = aStr;
    bPtr = bStr;
    while(*aPtr != '\0' && *bStr != '\0' && *aPtr == *bPtr) {
        aPtr++;
        bPtr++;
    }

    if(*aPtr == *bPtr)
        printf("yes\n");
    else
        printf("no\n");
    exit(0);
}
```

演習問題 11.4

例題11.2の解説とプログラム例11.4を参考にして、「名前 /名字」の形式で入力された氏名を、名前と名字に分割し、両者を入れ替えるプログラムを作りなさい。

プログラム例11.5で示した「見かけ上の文字列のコピー」と同様に、ポインタの操作による、「見かけ上の名前と名字の入れ替え」を行うことができますか。できればプログラムを作ってみなさい。

解答例　解答例を解答プログラム 11.3 に示します。氏名の分割までは、プログラム例 11.4 と全く同じです。氏名の入れ替えは、ポインタ変数 **aPtr** と **bPtr** の値 (アドレス) を入れ替えることで行っています。実際の文字データは、配列 **name[]** から少しも移動していないことに注意して下さい。

つまり、実際にデータが格納されている場所は一切変更せずに、データの格納場所

解答プログラム 11.3　氏名の分割と入れ替え

```
/************************************************
    p11_3.c
    氏名の分割／入れ替えプログラム
    Tadaaki Shimizu    2007.01.10
************************************************/
#include <stdio.h>

main()
{
    char name[40];       // 氏名の配列
    char *aPtr, *bPtr;   // ポインタ変数
    char *tmp;           // 入れ替え用テンポラリ変数

    printf(" Input string >> ");
    scanf("%s", name);

    aPtr = name;         // 名前文字列のアドレス
    bPtr = name;         // 分割文字の探索
    while(*bPtr != '/' && *bPtr != '\0') bPtr++;
    if(*bPtr == '/') {
        *bPtr = '\0';    // 文字列の分割
        bPtr++;          // 名字文字列のアドレス
    }

    printf("名前 : %s\n", aPtr);
    printf("名字 : %s\n", bPtr);

    // 氏名の入れ替え
    tmp  = aPtr;
    aPtr = bPtr;
    bPtr = tmp;
    printf("名前 : %s\n", aPtr);
    printf("名字 : %s\n", bPtr);

    exit(0);
}
```

を表す情報であるアドレスを入れ替えることにより、見かけ上のデータの入れ替えを行っているのです。このような方法は、インチキに思えるかも知れませんが、次のような素晴らしい利点があります。

データを実際に (物理的に) 入れ替えるよりも

1. 手続き (プログラム) が簡単になる
2. プログラムの実行速度が早くなる

電子機器を制御するようなプログラムを除いて、多くのプログラムでは、データが物理的にどのようなアドレスに在るかは、ほとんど問題になりません。論理上で正しくデータの操作が行えさえすれば良いのです。このような場合、ポインタを上手く利用して、実際にデータを動かすことなく、見かけ上の操作をする手法は、非常に重要なテクニックとなります。