第 6 章　便利な配列とポインタを理解しよう

概要

顧客名簿や売上データなどの膨大な数の数値や文字をプログラムで取り扱うには、一つ一つ変数を宣言するのではなく、機械的にたくさんの変数を宣言できる仕組みが必要です。ここではこのための配列について学びます。

この章の目標

配列の宣言と使用方法を覚える。
ポインタの宣言と使用方法を覚える。
変数の値と値の記憶場所の違いを理解する。

6.1　便利な入れ物、配列を覚えよう

●配列の宣言

（例）

```
int x[3];
```

これは、整数型の配列を宣言した例で、x が配列名で、括弧[]の中の 3 が、変数の個数を表します。この例の場合、三つの整数型の変数、x[0],x[1],x[2]を宣言したという意味です。
ここで注意しなければならないのは、配列の添え字は、0 から始まるということです。従って、添え字の最大は、この例の場合、2（3-1）となります。

●プログラム例

（P 例：6.1-1）

```
#include<stdio.h>

main()
{
        int i,x[3],y;

        for(i=0;i<3;i++){
                scanf("%d",&x[i]);
        }
        y = x[0] + x[1] + x[2];
        printf("y = %d¥n",y);
}
```

(P 例 : 6.1-2)
配列の宣言時に、値を代入しておく方法

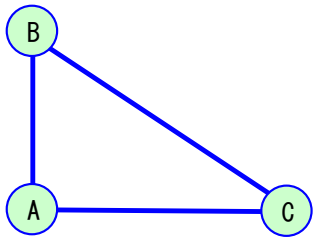
```
#include<stdio.h>

main()
{
        int i,y;
        int x[3] = {5,2,10};

        y = 0;
        for(i=0;i<3;i++){
                y += x[i];
        }
        printf("y = %d¥n",y);
}
```

6.2　配列に表中の値を保存しよう

●配列の宣言
地域間の距離を表す次のような表（距離表）があったとします。

表 6.2-1　距離表

	A	B	C
A	0	3	4
B	3	0	5
C	4	5	0



これを各地域を 0 からはじまる配列の添え字として表すと上の距離表は次のような距離データレコードとして表現することができます。

表 6.2-2　距離データ

発地	着地	距離
0	0	0
0	1	3
0	2	4
1	0	3
1	1	0
1	2	5
2	0	4
2	1	5
2	2	0

地域コード
A:0,B:1,C:2

このデータを表す変数は、二つの添え字を持つ配列として次のように宣言できます。

```
int d[3][3];
```

この配列の最初の添え字を発地とし、後の添え字を着地とすると、B 地域から C 地域への距離を表す変数は、d[1][2]と表現できます。

●プログラム例
表 6.2-1 の距離表を基に、配列「d」に距離を代入し、各地域を A-B-C の順に回ったときの総移動距離を求めるプログラムを次に示します。
(P 例 : 6.2-1)

```
#include<stdio.h>

main()
{
        int d[3][3] = {{0,3,4},
                       {3,0,5},
                       {4,5,0}};
        int y;

        y = d[0][1] + d[1][2];
        printf("y = %d¥n",y);
}
```

6.3　頻繁に使用されるポインタて何だ

●変数の値とアドレス
変数の宣言を行った際、プログラムはメモリ上に変数の値を保存しておく場所（アドレス）を確保します。従って、変数に値を代入するということは、この変数のアドレスに値を記憶させるということです。
下図にこのイメージを示します。この変数の値とアドレスの意味の相違をよく理解しておいて下さい。

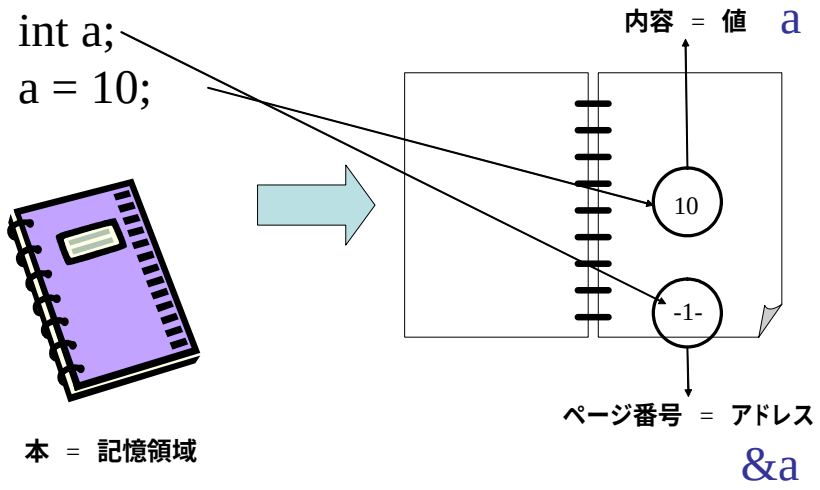


図 6.3-1　値とアドレス

●通常の変数とポインタ変数
○通常の変数
宣言例：int　　a;

値　　a
アドレス　　&a ―― 変数の値が記憶される場所（自動的に決まる。）

○ポインタ変数
ポインタ変数は、変数の値として、アドレス値を扱えるように用意された変数です。

宣言例：int　　*b;

値　　b ―― 値は、アドレス値
アドレス　　&b ―― 変数の値が記憶される場所（自動的に決まる。）

従って、ポインタ変数の値（アドレス）に記憶されている値を操作するには間接演算子を用いて次のように表現します。

値　　*b ―― b が示すアドレス値にある値

使用例：
```
int a;
int *b;

b = &a;
*b = 10;
```

この例は、変数 a に値、10 を代入したことになります。図を参考にして下さい。

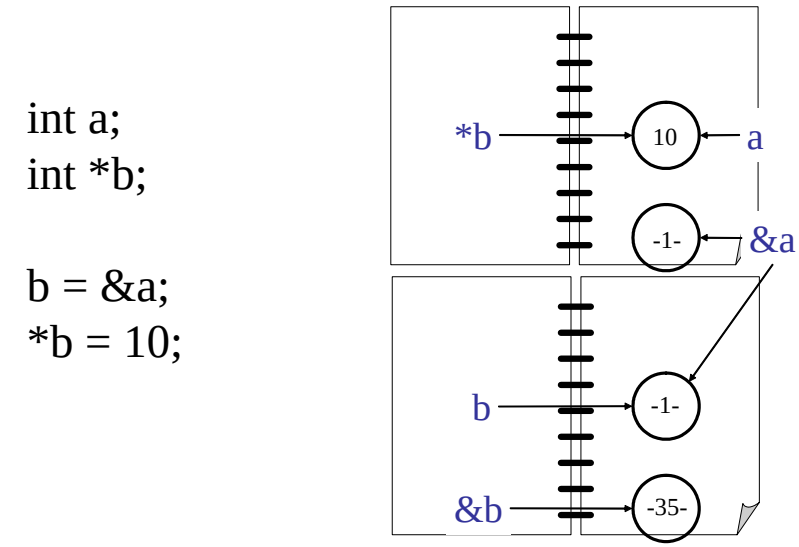


図 6.3-2　ポインタ変数の値とアドレス

○配列変数
宣言例：int　　a[3];

値　　a[1]　――― 変数 a[1]の値
アドレス　　&a[1] ――― 変数 a[1]の値が記憶される場所（自動的に決まる。）
ただし、配列の先頭 a[0] のアドレスは、&a[0] あるいは、 a と記述することができる。

int a[3];
a[0] = 2000; a[1] = 2001; a[2] = 2002;

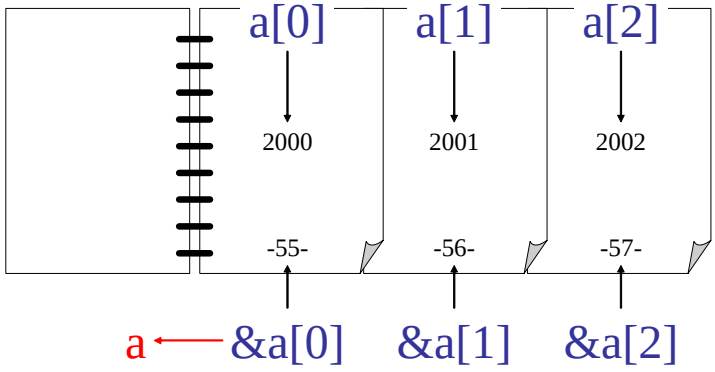


図 6.3-3　配列の値とアドレス

●プログラム例
（P 例：6.3-1）

```
#include<stdio.h>

main()
{
        int a;
        int *b;

        b = &a;
        *b = 3;
        printf("&a:%d,   a:%d¥n", &a, a);
        printf(" b:%d,  *b:%d¥n", b, *b);
        printf("&b:%d,   b:%d¥n", &b, b);
}
```

（P 例：6.3-2）

```
#include<stdio.h>
void kansu(int, int*);

main()
{
        int a,b;

        a = 2;
        b = 5;
        printf("a:%2d,   b:%2d¥n", a, b);
```

```
        kansu(a,&b);
        printf("a:%2d,   b:%2d¥n",a,b);
}

void kansu(int x,int *y)
{
        x += 10;
        *y += 10;
        printf("x:%2d,  *y:%2d¥n",x,*y);
}
```

補足:
kansu 関数が呼び出されたとき、x = a と y = &b の二つの代入が行われる。この代入の前者は、値の代入であり、後者は、アドレス値の代入である。

(P 例:6.3-3)

```
#include<stdio.h>

main()
{
        int i,a[3],*b;

        for(i=0;i<3;i++)
                a[i] = 7;
        for(i=0;i<3;i++)
                printf("%2d,",a[i]);
        printf("¥n");

        b = a;
        for(i=0;i<3;i++)
                *(b+i) += 10;
        for(i=0;i<3;i++)
                printf("%2d,",a[i]);
        printf("¥n");
}
```

●注意事項
次のプログラムは、間違ったポインタ変数の使用例です。値「8」をポインタ変数が示すアドレス値に代入していますが、メモリ上のどこのアドレス値かが確定していません。もし OS の重要な値を記憶しているアドレス値であれば、計算機が暴走し、大変なことになります。

```
#include<stdio.h>

main()   注意:このプログラムは、絶対に実行してはいけません。
{
        int *p;

        *p = 8;
        printf("%d¥n",*p);
}
```