第5章 「配列」のまとめ

5.1 「配列」のまとめ

配列 (教 p.116)

同一の型のデータを集めて、番号（___、そえじ）でアクセスできるようにしたもの。C 言語の配列の添字は___から始まる。

```
/* 初期化しないとき */
int va[5];
/* 配列の初期化は、式をコンマで区切って { } で囲む。*/
int vb[5] = { 15, 20, 30 };
/*（残りの要素は （i） で初期化される。） */

/* 初期化子を代入することはできない。— 教 p.121 */
~~vb = { 15, 20, 30, 0, 0 };~~
/* 配列同士の代入はできない。 — 教 p.130*/
~~vb = va;~~
```

(i) ___

Q 5.1.1 次のプログラムの断片の出力は?

```
int a[] = { 3, 7, 5 };
printf("%d", a[2]);
```

___

配列の走査 (教 p.118)

配列は for 文と相性が良い。5 個の要素を持つ配列の各要素に対して同じ操作を行なうときには次のような for 文を使う。

```
for (（i）; （ii）; （iii）) {
  a[i] = …;
}
```

(i) ______ (ii) ______ (iii) ____

配列の全要素の並びを反転する (教 p.123)

2 つの変数 x, y を入れ替えるのに、

```
x = y; y = x;
```

と書いてもダメで[なぜ?]、別の変数（例えば temp）を一つ用意して、

```
______________________________
```

と書く必要がある。

なお、List 5-8 は真ん中のループを、以下のようにコンマ演算子を使うかたちにすることも可能である。

```
…
for (i = 0, j = 6; i < j; i++, j--) {
  int temp = x[i];
  x[i]     = x[j];
  x[j]     = temp;
}
…
```

オブジェクト形式マクロ（定数マクロ）（教 p.124）

プログラム中で繰り返し使う定数は名前をつける。

```
#define NUMBER 5
```

この指令は NUMBER というオブジェクト形式＿＿＿＿を定義する。マクロは他のコンパイル処理に先だって、一括して置換される。マクロを定義すると、次のような利点がある。

- 値の変更が容易になる。
- 定数の意味がわかり易くなる。秘密の数値（マジックナンバー）を直接プログラムに埋め込まないこと！

マクロが使われるのは、次のような箇所である。

- 配列の要素数など、文法上定数が要求されるところ
- 円周率などの数学定数・物理定数など絶対に変らない定数

（これら以外の箇所では、通常の変数を使うのが普通である。）

C99 規格では、配列の要素数に変数を使って、次のような書き方も一応可能になった。

```
int n = 3;
int a[n];  /* C99 では許容だが */

for (i = 0; i < n; i++) {
  a[i] = 0;
}
```

しかし、すべての処理系がこれをサポートしなくても良いことになっているので非推奨とする。（演習の解答では使ってはいけない。）

マクロ名は通常すべての文字を＿＿＿＿とする慣習がある。小文字を使ってもコンパイルエラーになるわけではないが、強く非推奨とする。（逆に変数名は必ず小文字を混ぜること。）

代入式の評価（教 p.126）

代入「変数 = 式」も式であり、値（代入された値と同じ）を持つ。代入演算子は右結合である（右側から行われる）。つまり、`x = y = 0` は＿＿＿＿＿＿＿＿と解釈される。

Q 5.1.2 List 5-11 の 2 つめのループは、なぜ（`i = 0` ではなく）`i = 1` から始まるのか？

Warning 《教 p.131》

発音は /ˈwɔːnɪŋ/ で、カタカナでは＿＿＿＿＿＿が近い。警告という意味で、エラーではない（コンパイルができないということはない）が間違っている可能性が高いことを示す。

Q 5.1.3 次の break 文を使用したプログラム (breakTest.c) について考える。

```c
#include <stdio.h>

int main(void) {
    int i;
    int a[5] = {1, 2, -2, -4, 5};
    int n = 5;
    for (i = 0; i < n; i++) {
        if (a[i] < 0) {
            break; /* continue; も試せ。 */
        }
        printf("a[%d] = %2d\n", i, a[i]);
    }
    return 0;
}
```

上のプログラムの出力はどうなるか？改行は ↲ で表せ。

＿＿＿＿＿＿＿＿＿＿＿＿

また break を continue に変えたときはどうなるか？

＿＿＿＿＿＿＿＿＿＿＿＿

5.2 「多次元配列」のまとめ

多次元配列 《教 p.132》

各要素が配列であるような配列、言い替えれば 2 つ以上の添字を持つ配列のこと。ただし、物理的には一次元に配置される。（Fig.5-10参照） 《教 p.132》

```c
int x[2][3] = {{ 1, 2, 3 }, { 4, 5, 6 }};
```

Q 5.2.1 上の二次元配列 `x` のメモリ上の配置を Fig.5-10 のような図で表わせ。

Q 5.2.2 次のプログラムの断片の出力は?

```
int a[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
printf("%d", a[1][2]);
```

文法のまとめ

宣言 (declaration)

に以下を追加する。

分類	一般形	補足説明
配列宣言	*型 変数* [*定数*] = { *式* , … , *式* };	「=」以降の 灰色の部分は省略可能

式 (expression)

に以下を追加する。

分類	一般形	補足説明
配列アクセス	式 [式]	a[1],b[2][3] など