第８回

プログラミングスタイルなど

［１］ 名前のつけかた

- グローバル変数にはわかりやすい名前を，ローカル変数には短い名前を。
- 関連性のあるものには関連性のある名前をつけて，統一しよう。
- 関数には能動的な名前を。
- 名前は的確に。

《例題１》

次のコードの名前と値の選び方についてコメントせよ。

```
?    #define TRUE 0
?    #define FALSE 1
?
?    if ((ch = getchar()) == EOF)
?        not_eof = FALSE;
```

《例題２》

次の関数を改良せよ。

```
?    int smaller(char *s, char *t) {
?        if (strcmp(s, t) < 1)
?            return 1;
?        else
?            return 0;
?    }
```

［2］ 式と文

- 構造がわかるようにインデントしよう。
- 自然な形の式を使おう。
- カッコを使ってあいまいさを解消しよう。
- 複雑な式は分割しよう。
- 明解に書こう。
- 副作用（++などの演算子）に注意。

《例題３》

次の式のあいまいさをカッコを使って解消しよう。

```
?    if (x&MASK == BITS)

?    leap_year = y % 4 == 0 && y % 100 != 0 || y % 400 == 0;
```

《例題４》

次の式にはどんな副作用があるか述べなさい。

```
?    array[i++] = i;

?    scanf("%d %d", &yr, &profit[yr]);
```

《例題５》

次のそれぞれのコードを改良せよ。

```
?    if (!(c == 'y' || c == 'Y'))
?        return;

?    length = (length < BUFSIZE) ? length : BUFSIZE;

?    flag = flag ? 0 : 1;

?    quote = (*line == '"') ? 1 : 0;
```

```
?    if (val & 1)
?        bit = 1;
?    else
?        bit = 0;
```

《例題６》

さまざまな評価順によって生成される可能性のある出力を列挙してみよ。

```
?    n = 1;
?    printf("%d %d¥n", ++n, ++n);
```

［３］ 一貫性と慣用句

- インデントとブレースのスタイルを統一しよう。
- 慣用句によって一貫性を確保しよう。
- 多分岐の判定には else-if を使おう。

《例題７》

次のコードを正しく直しなさい。

```
?    if (month == FEB) {
?        if (year%4 == 0 && year%100 != 0 || year%400 == 0)
?            if (day > 29)
?                legal = FALSE;
?        else
?            if (day > 28)
?                legal = FALSE;
?    }
```

《よく使われる慣用句的な形式》

```
for (i = 0; i < n; i++)
    array[i] = 1.0;

for (p = list; p != NULL; p = p->next)
    …
```

無限ループの書き方

```
    for (;;)
        …
    while (1)
        …

int c;
while ((c = getchar()) != EOF)
    putchar(c);
```

《例題９》

次のコードをもっと明解に書き直しなさい。

```
?   if (retval != SUCCESS)
?   {
?       return retval;
?   }
?   /* All went well! */
?   return retval;

?   for (k = 0; k++ < 5; x += dx)
?       scanf("%lf", &dx);
```

［４］ マクロやリテラルな数値の使い方

- 関数マクロはなるべく使うな。
- マクロの本体と引数はかっこに入れよう。
- 数値はマクロでなく定数として定義しよう。
- 整数でなく文字定数を使おう。
- オブジェクトサイズは言語に計算させよう。

《例題１０》

次のコードの問題点は何か。

```
?    #define isupper(c) ((c) >= 'A' && (c) <= 'Z')
?        …
?    while (isupper(c = getchar()))
?        …

?    #define square(x) (x) * (x)
?        …
?    y = 1 / square(x)

?    #define ISDIGIT(c) ((c >= '0') && (c <= '9')) ? 1 : 0
```

《列挙子 enum 型の活用》

列挙子リストの中の識別子は int 型の定数として宣言される。＝で値を指定しない場合は，定数の値は 0 から始まり，順に 1 ずつ増やされる。

```
enum { NO, YES };

enum {
    MINROW = 1,       /* 上端 */
    MINCOL = 1,       /* 左端 */
    MAXROW = 120,     /* 下端 */
    MAXCOL = 80       /* 右端 */
};
```

《はっきりした定数の使い方》

例えば，0 という数値はプログラムの中で様々な文脈で登場する。コンパイラは適切な型に自動変換してくれるが，型を明記した方が読み手に 0 の役割りが分かりやすくなる。

```
?    if (c >= 65 && c <= 90)
?    if (c >= 'A' && c <= 'Z')    →    if (isupper(c))

?    str = 0;              →    str = NULL;
?    name[i] = 0;          →    name[i] = '¥0';
?    x = 0;                →    x = 0.0;
```

《sizeof 演算子の活用》

```
char buf[1024];
    …
fgets(buf, sizeof(buf), stdin);
```

注意．gets は入力行のサイズを制限する手段がないので極めて危険！

```
# define NELEMS(array) (sizeof(array) / sizeof(array[0]))
    …
double dbuf[100];
    …
for (i = 0; i < NELEMS(dbuf); i++) {
    …
```

練習問題２６

２６－１．次のフォーマットで，ある人の１日ごとの仕事時間を入力すると，日数と仕事の合計時間を出力するプログラムをつくりなさい。ただし，仕事時間の入力データが不正な場合（hh:mm の入力で，hh>23 や hh<0 や mm>59 や mm<0 の場合）は再入力をさせるようにしなさい。また，終了は 0:0 とします。

<入力例>

仕事時間 hh:mm を入力して下さい（0:0 で終了）

3:35↵
2:09↵
4:55↵
……↵
……↵
0:0↵　　（終了）

<出力例>

日　　数　17　日
合計時間　104　時間
　　23　分

２６－２．（難問）キーボードから入力された１行（空白で区切られた正整数並びの文字列）を項に分解して，i 番目の項の整数値を配列要素 d[i-1]に累算しながら格納するプログラムをつくりなさい。ただし，配列 d の要素数は 10 として宣言し，10 項目より多い入力の場合は 11 項目以降を棄却しなさい。1 行の入力には関数 fgets を用いなさい。fgets は改行文字も取り込む点に注意して，入力の終了は「改行のみの入力」で判定しなさい。

《ヒント》入力行の正整数並びの文字列を項に分解するには，空白文字は関数 isspace，数字文字は関数 isdigit で判定して行いなさい。また，一つの整数を表す文字列 s を数値化するにはライブラリ関数 int atoi(const char *)を用いて（stdlib.h をインクルードする），数字文字列 s は atoi(s)により数値化しなさい。

<入力例>
空白文字で区切って正整数の並びを入力してください！
入力終了は何も入力せずに改行です

59 37 88 32 19 44 98 78 ↵　　……8 項目の入力
21 35 57 ↵　　……3 項目の入力
↵　　……入力終了

<出力例>
d[0] = 80
d[1] = 72
d[2] = 145
d[3] = 32
d[4] = 19
d[5] = 44
d[6] = 98
d[7] = 78

２６−３．次のプログラムは，数字文字列を入力して数値に変換するものです。switch 文を用いている箇所を else-if 文を使って分かり易く書き直しなさい。

```
/* Q26-3 */
/* 数字文字列の入力 */
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>

main()
{
    char c;
    double val = 0.0;
    int sign = 1, dot = 0, n = 0;

    printf("数字文字列を入力して下さい：");

    while ((c = getchar()) != '¥n') {
        switch (c) {
            case '-':
                sign = -1;
                /* 落下 */
            case '+':
                c = getchar();
                break;
            case '.':
                break;
            default:
                if (!isdigit(c)) {
                    printf("正しい数字表記でありません！¥n");
                    exit(1);
                }
                break;
        }
        if (c == '.') {
            dot = 1;
        } else {
            if (dot == 1) n++;
            val = val * 10 + (c - '0');
        }
    }
    val /= pow(10.0, n);
    val *= sign;

    printf("入力数は%f です。¥n", val);
}
```

２６−４．（難問）以下のコードはいずれも間違っています。ループや条件文の現れるところで境界条件のテスト（条件分岐が正しく実行されるかどうか，ループの繰り返し回数が正しいかどうか）を行ってみなさい。その上で，どこに間違いがあるかを指摘して，修正を施しなさい。

(a) 階乗を計算するコード：

```
?    int factorial(int n)
?    {
?        int fac;
?        fac = 1;
?        while (n--)
?            fac *= n;
?        return fac;
?    }
```

(b) 文字列の文字を１行に１文字ずつ表示するコード：

```
?    i = 0;
?    do {
?        putchar(s[i++]);
?        putchar('¥n');
?    } while (s[i] != '¥0');
```

(c) コピー元からコピー先へ文字列を複写するコード：

```
?    void strcpy(char *dest, char *src)
?    {
?        int i;
?        for (i = 0; src[i] != '¥0'; i++)
?            dest[i] = src[i];
?    }
```

(d) これも文字列コピーであるが，s から t に n 個の文字をコピーするコード：

```
?    void strncpy(char *t, char *s, int n)
?    {
?        while (n > 0 && *s != '¥0') {
?            *t = *s;
?            t++;
?            s++;
?            n--;
?        }
?    }
```