プログラミング及び実習2

第 3 回課題（ex03）の解答例

佐野 彰

ex03 解答例: square.c, frac.c

```c
// square.c
#include <stdio.h>

int main() {
    int a;

    while (1) {
        scanf("%d", &a);
        if (a<0) break;
        printf("%d\n", a*a);
    }
    return 0;
}
```

```c
// frac.c
#include <stdio.h>

int main() {
    int a, b;
    scanf("%d%d", &a, &b);
    if (b == 0) return 0;

    printf("%f\n", (double)a/b);
    return 0;
}
```

- square.c: 負の値を読んだら、2 乗を出力する前に `break` でループを抜ける
- frac.c: `(double)a/b` は a を実数にしてから割る。`(double)(a/b)` だと整数の割り算が先になり、1 と 2 では `0.000000` になる
- `%f` は小数点以下 6 桁で出力する

ex03 解答例: twice.c, twice2.c

```c
// twice.c
#include <stdio.h>

int main() {
    int i;
    scanf("%d", &i);
    if (i <= 0) return 0;

    while(i > 0) {
        i *= 2;
        printf("%d\n", i);
    }

    return 0;
}
```

```c
// twice2.c
#include <stdio.h>

int main() {
    int i;
    scanf("%d", &i);
    if (i == 0) return 0;

    int s;
    s = i<0 ? 0 : 1;   // 入力の符号（負なら 0、正なら 1）

    while ((s && i>0) || (!s && i<0)) {
        i *= 2;
        printf("%d\n", i);
    }
    return 0;
}
```

- 2 倍して出力してから条件を調べるので、符号が変わった値（または 0）も出力される
- twice2.c は、入力の符号を `s` に覚えておき、符号が同じ間だけ繰り返す
- int の範囲を超える計算（オーバーフロー）は、C 言語では未定義動作

ex03 解答例: dfrac.c（実数の積と商の切り捨て）

```c
#include <stdio.h>

int main() {
    double x, y;

    while (1) {
        scanf("%lf%lf", &x, &y);
        if (y == 0.0) break;

        printf("%d %d\n", (int)(x*y), (int)(x/y));
    }
    return 0;
}
```

- double を `scanf()` で読むときは `%lf`（`%f` ではない）
- `(int)` への変換は小数点以下を切り捨てる（0 の方向へ丸める）。この問題では $x \geqq 0$、$y > 0$ なので、そのまま使える
- `(int)x*y` と書くと、x だけが先に整数になる。式全体を括弧で囲むこと

ex03 解答例: timesec.c（時間換算）

```c
#include <stdio.h>

int main() {
  int sec;
  scanf("%d", &sec);

  int s, m, h;
  s = sec%60;
  sec /= 60;
  m = sec%60;
  sec /= 60;
  h = sec;

  printf("%d:%d:%d\n", h, m, s);
  return 0;
}
```

- 60 で割った余りが秒、商をさらに 60 で割った余りが分、残りが時間
- C99 以降、整数の `/` は 0 の方向へ切り捨て、`%` の結果は左の値と同じ符号になる。そのため負の秒数も、場合分けなしで `0:-1:-1` のように出力できる

ex03 解答例: factorize.c（素因数分解）

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    int k = 2;
    while (n > 1) {
        if (n % k == 0) {
            n /= k;
            printf("%d\n", k);
        } else {
            k++;
        }
    }
    return 0;
}
```

- 割り切れる間は同じ k で割り続け、割り切れなくなったら k を 1 増やす
- 小さい素因数で割り尽くしてから進むので、合成数の k では割り切れない。素数かどうかを調べる必要はない
- n が 2 未満なら、何も出力せずに終わる

ex03 解答例: factorize2.c（素因数分解2）

```c
#include <stdio.h>
// INT_MAX 2147483647

int main() {
    int n;
    scanf("%d", &n);

    int p = 2;
    int kp = 0;

    while (n > 1) {
        if (n % p == 0) {
            n /= p;
            kp++;
        } else {
            if (kp > 0) {
                printf("%d ** %d\n", p, kp);
                kp = 0;
            }
            if (p == 2) p++;
            else p += 2;
        }
    }
    if (kp > 0) printf("%d ** %d\n", p, kp);

    return 0;
}
```

- 割った回数を `kp` で数える。割り切れなくなったら `p ** kp` を出力し、次の p へ進む
- 最後の素因数は、割り切ったところで n が 1 になりループを抜ける。ループの後の `printf()` を忘れないこと
- 2 の次は奇数だけを調べる（`p += 2`）。調べる回数がおよそ半分になる