プログラミング及び実習2

第 4 回課題（ex04）の解答例

佐野 彰

ex04 解答例: decimal.c（実数の整数部と小数部）

```c
#include <stdio.h>

int main() {
    double x;
    int n, p;

    scanf("%lf", &x);
    n = x;
    p = (x-n)*10000;
    printf("%d %04d\n", n, p);

    return 0;
}
```

- `n = x;` のように double を int に代入すると、小数部が切り捨てられて整数部が残る
- `(x-n)*10000` を int に代入すると、小数点以下 5 桁目以降が切り捨てられる
- `%04d` は 4 桁に満たないとき先頭を 0 で埋める（`0625`、`0000`）

ex04 解答例: decimal.c の注意（浮動小数点の誤差）

この解答例に `1.15` を入力すると、`1 1500` ではなく `1 1499` と表示されます。

```c
double x = 1.15;
printf("%.20f\n", x);           // 1.14999999999999991118
printf("%.15f\n", (x-1)*10000); // 1499.999999999999091 → int にすると 1499
```

- double は値を 2 進数で表すので、`1.15` のような 10 進数の小数の多くは正確に表せない。わずかに小さい値として記憶されると、切り捨てで 1 小さくなる
- `2.3` も同じく `2 2999` になる。`0.0625`（= 1/16）のように 2 進数で割り切れる値は正確に表せる
- aiJudge のテストには、この誤差が表に出る入力を入れていない
- 浮動小数点数の誤差と、その扱い方は、2 年生の数値計算の科目で詳しく学ぶ

ex04 解答例: char.c（文字整数型）

```c
#include <stdio.h>

int main() {
    int a;

    while (1) {
        scanf("%d", &a);
        printf("%d", a);
        if (a < 33 || 126 < a)
            break;
        else
            printf(" %c\n", a);
    }
    printf("\n");
    return 0;
}
```

- `%c` に int の値を渡すと、その文字コードの文字が表示される（65 なら `A`）
- 整数は先に出力し、範囲外なら `break` する。最後の改行はループの外で出力する

ex04 解答例: sum.c（繰り返し処理）

```c
#include <stdio.h>

int main() {
    int i, n;
    int sum = 0;
    scanf("%d", &n);

    for (i=1; i<=n; i++) {
        sum += i;
        printf("%d\n", sum);
    }
    return 0;
}
```

- `sum` を 0 で初期化し、ループの中で足しながら毎回出力する
- n が 1 未満ならループは 1 回も回らず、何も出力しない。この場合も `return 0;` で終える

ex04 解答例: trig.c（三角関数と繰り返し処理の中断）

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define PI 3.14159265358979323846

int main() {
    int a;

    do {
        scanf("%d", &a);
        if (abs(a) >= 360)
            continue;

        double rad = a*PI/180;
        printf("%f %f\n", sin(rad), cos(rad));
    } while (a != 0);

    return 0;
}
```

- `sin()` ・ `cos()` の引数はラジアン。度は `a*PI/180` で変換する
- `PI` は `#define` で定義する。`M_PI` は標準 C に含まれないので使わない
- `do` 〜 `while` の中の `continue` は、条件 `a != 0` の判定に進む

ex04 解答例: log.c（対数関数）

```c
#include <stdio.h>
#include <math.h>

int main() {
    double a, b;

    do {
        scanf("%lf%lf", &a, &b);
        if (a == 1.0)
            break;
        else if (a <= 0 || b <= 0)
            continue;

        printf("%.8f\n", log(b)/log(a));
    } while (1);

    return 0;
}
```

- 底の変換公式により `log(b)/log(a)` を計算する（`log()` は自然対数）
- a が 1 かどうかを先に調べる。`1 0` のように b が 0 以下でも、a が 1 ならすぐに終了する

ex04 解答例: comp3.c（3つの整数の比較）

```c
#include <stdio.h>

int main() {
  int a, b, c;

  while (1) {
    scanf("%d%d%d", &a, &b, &c);

    if ( a == b && b == c) {
      if (a == 0) break;
      printf("すべて同値\n");
    }
    else if (a == b || b == c || a == c)
      printf("2つが同値で1つ異なる\n");
    else
      printf("すべて異なる\n");
  }

  return 0;
}
```

- 「すべて同値」を先に調べると、次の条件は「どれか 2 つが等しい」だけで済む
- 終了の判定（3 つとも 0）は、すべて同値の場合の中で `a == 0` を調べればよい

ex04 解答例: maclaurin.c（マクローリン展開）

```c
#include <stdio.h>

int main() {
  int m;
  double x;
  scanf("%d%lf", &m, &x);

  int k, fac;
  double pow, sum;

  for (k=0; k<=m; k++) {
    if (k==0)
      fac = pow = sum = 1.0;
    else {
      fac *= k;
      pow *= x;
      sum += pow/fac;
    }
    printf("%d %.10f\n", k, sum);
  }

  return 0;
}
```

- k! と x の k 乗は、前の値に k や x を掛けて更新する。毎回はじめから計算しなくてよい
- k = 0 では `fac` ・ `pow` ・ `sum` をすべて 1 にする（0! = 1 のため）
- m が負ならループは回らず、何も出力しない
- 変数名 `pow` は math.h の関数 `pow()` と同じ名前なので、math.h を読み込むプログラムでは避ける

ex04 解答例: nsphere.c（n次元ユークリッド球の体積）

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define PI 3.14159265358979323846

int main() {
  int n, k;
  double r;

  scanf("%d%lf", &n, &r);
  if (n <= 0 || r <= 0) return 0;

  k = n/2;

  double fact_k=1;
  int i;
  for (i=0; i<k; i++) {
    fact_k *= i+1;
  }
  double v;
  if (n%2 == 0) {
    v = pow(PI, k)/fact_k * pow(r, 2*k);
  } else {
    double fact_k2=fact_k;
    for(i=k; i<2*k+1; i++) {
      fact_k2 *= i+1;
    }
    v = 2*fact_k*pow(4*PI,k) / fact_k2 * pow(r, 2*k+1);
  }

  printf("%f\n", v);
  return 0;
}
```

ex04 解答例: nsphere.c（考え方）

前のスライドのプログラムは、次の式をそのまま計算しています（k = n/2）。

```
n = 2k   のとき  V = π^k / k! × r^(2k)
n = 2k+1 のとき  V = 2 × k! × (4π)^k / (2k+1)! × r^(2k+1)
```

- `k = n/2` は、n が偶数でも奇数でも正しい k になる（整数の割り算は切り捨て）
- 階乗は int では 13! で桁あふれするので、double で計算する
- (2k+1)! は、k! に k+1 から 2k+1 までを順に掛けて求める
- `pow(x, y)` は x の y 乗（math.h）。`PI` は `#define` で定義する