# プログラミング及び実習2

第 5 回課題（ex05）の解答例

佐野 彰

# ex05 解答例: sqrsum.c（平方根の和）

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

int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    if (n <= 0 || m <= 0 || n > m) return 0;

    double sqrsum = 0.;
    for (int k=n; k<=m; k++) {
        sqrsum += sqrt(k);
        printf("%.6f\n", sqrsum);
    }
    return 0;
}
```

- `sqrt()` を使うには `#include <math.h>` が要る（環境によってはコンパイル時に `-lm` も要る）
- 和の変数は `double` にし、`0.` で初期化しておくこと
- 途中の和を毎回出すので、`printf()` はループの中に置く

# ex05 解答例: revers.c（配列変数への入出力）

```c
#include <stdio.h>
#define N 10

int main() {
    int i;
    int a[N];

    for (i=0; i<N; i++)
        scanf("%d", &a[i]);

    for (i=N-1; i>=0; i--)
        printf("%d\n", a[i]);

    return 0;
}
```

- 添字は `0` から `N-1` まで。逆順のループは `i=N-1` から始めて `i>=0` まで回す
- `scanf()` では、配列の要素にも `&a[i]` のように `&` を付けること
- 要素数を `#define` で定数にしておくと、変えるときに直すのは 1 か所で済む

# ex05 解答例: fizzbuzz.c（FizzBuzz）

```c
#include <stdio.h>

int main() {
  int n;
  scanf("%d", &n);
  if (n < 1) return 0;

  for (int i=1; i<=n; i++) {
    if (i%3 == 0 && i%5 == 0) printf("FizzBuzz\n");
    else if (i%3 == 0) printf("Fizz\n");
    else if (i%5 == 0) printf("Buzz\n");
    else printf("%d\n", i);
  }

  return 0;
}
```

- 「3 の倍数かつ 5 の倍数」の判定を最初に置くこと。`i%3 == 0` を先に調べると、15 でも `Fizz` と出てしまう
- 最初の条件は `i%15 == 0` と書いてもよい

# ex05 解答例: oddeven.c（奇数と偶数の分割表示）

```c
#include <stdio.h>

#define N 99

int main() {
    int n;
    int a[N];

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

    for (int i=0; i<n; i++)
        scanf("%d", &a[i]);

    for (int i=0; i<n; i++)
        if (a[i]%2 != 0)
            printf("%d\n", a[i]);
    for (int i=0; i<n; i++)
        if (a[i]%2 == 0)
            printf("%d\n", a[i]);

    return 0;
}
```

- 配列は n の上限（99）の大きさで宣言し、範囲外の n は配列を使う前にはじく
- 奇数の判定は `a[i]%2 != 0` にすること
- `a[i]%2 == 1` では、負の奇数を見落とす（`-3 % 2` は `-1` になる）
- 配列を 2 回たどり、1 回目で奇数、2 回目で偶数を出す

# ex05 解答例: onoff.c（オン・オフ状態の保持）

```c
#include <stdio.h>

#define N 10

int main() {
    int a[N];
    int sw;

    for (int i=0; i<N; i++)
        a[i] = 0;

    do {
        for (int i=0; i<N; i++)
            printf("%d ", a[i]);
        printf("\n");

        scanf("%d", &sw);
        if (sw < 0 || N <= sw) break;

        a[sw] = a[sw] ? 0 : 1;
    } while (1);

    return 0;
}
```

- 配列は、すべての要素を 0（オフ）にしてから使うこと
- 表示してから入力を読むので、初期状態が最初に表示される
- `a[sw]` を書き換える前に、番号が範囲内か調べること（`a[-1]` や `a[10]` は配列の範囲外）
- `a[sw] ? 0 : 1` は、`a[sw]` が 0 以外なら 0、0 なら 1 になる
- `a[sw] = 1 - a[sw];` や `if` 文で書いてもよい

# ex05 解答例: kuku.c（掛け算九九表）

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    if (n < 1 || n > 9) return 0;

    printf("    ");
    for (int i=1; i<=n; i++)
        printf("%3d", i);
    printf("\n\n");

    for (int i=1; i<=n; i++) {
        printf("%3d ", i);
        for (int j=1; j<=n; j++)
            printf("%3d", i*j);
        printf("\n");
    }
    return 0;
}
```

- 1 行目の後の空行は `printf("\n\n")` で出す
- 2 行目以降は、行頭の `%3d `（空白 1 つ付き）と積の `%3d`（空白なし）を使い分ける
- 行の改行は、内側のループが終わった後に 1 回だけ出すこと

# ex05 解答例: elecharge.c（電気料金）

```c
#include <stdio.h>

#define BASE 522.58
#define UP1  20.21
#define UP2  25.61
#define UP3  28.59
#define LIMIT0 15
#define LIMIT1 120
#define LIMIT2 300
```

```c
int main() {
  int w;
  double p;

  scanf("%d", &w);
  if (w <= 0) return 0;

  do {
    p = BASE;
    if (w <= LIMIT0) break;

    p += ((w < LIMIT1 ? w : LIMIT1) - LIMIT0) * UP1;
    if (w <= LIMIT1) break;

    p += ((w < LIMIT2 ? w : LIMIT2) - LIMIT1) * UP2;
    if (w <= LIMIT2) break;

    p += (w-LIMIT2) * UP3;
    break;
  } while(0);

  printf("%.2f\n", p);
  return 0;
}
```

# ex05 解答例: elecharge.c の考え方

基本料金から始めて、段階ごとに、その段階に入る分の電力量だけを足していきます。
121kWh なら次のようになります。

```
522.58 + (120 − 15) × 20.21 + (121 − 120) × 25.61 = 2670.24
```

- `(w < LIMIT1 ? w : LIMIT1)` は、`w` と `LIMIT1` の小さい方。その段階の上限で頭打ちにする
- `do { ... } while(0);` は 1 回だけ実行するブロックで、`break` で途中から抜けるために使っている。`if` 〜 `else if` で書いてもよい
- 単価や境界の値を `#define` にしておくと、単価が変わったときに直すのはそこだけで済む