情報領域演習第二 C 演習 (クラス 3) レポート

学籍番号: 1810678
名前: 山田朔也

2019 年 6 月 7 日

演習問題

問 1　まず、作成したソースコードを、Listing1 に示す。

Listing 1　問 1 のソースコード

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void dump_byte(unsigned int n) {
5       int buf[8];
6       // 下位から順に２進数にして配列に格納
7       for(int i = 0; i < 8; i++) {
8           buf[i] = n % 2;
9           n = n / 2;
10      }
11      // 変換結果を上位から順に出力
12      for(int i = 7; i >= 0; i--) {
13          if(buf[i] == 0) {
14              printf("0");
15          } else {
16              printf("1");
17          }
18      }
19  }
20
21  void dump_bytes(unsigned char cp[], int nbyte) {
22      for(int i = 0; i<nbyte; i++) {
23          dump_byte(cp[nbyte-i-1]); // little endian -> 逆順
24          // dump_byte(cp[i]); // big endian -> 番地順
25          printf(" ");
26      }
27      printf("\n");
28  }
```

```
29
30      int main(int ac, char *av[]) {
31          float f; // 単精度浮動小数点数(32 bit)
32          double d; // 倍精度浮動小数点数(64 bit)
33
34          // av[1]: プログラム実行の第一引数の文字列
35          // atof: 引数で指定した値を倍精度浮動小数点数に変換
36          f = atof(av[1]); // 代入の際に単精度に変換される
37          d = atof(av[1]); // 倍精度そのまま
38
39          dump_bytes((unsigned char *)&f, sizeof f);
40          dump_bytes((unsigned char *)&d, sizeof d);
41
42          exit(0);
43      }
```

また、これの実行結果は以下の Listing2 ようになる

Listing 2　問 1 の実行結果

```
1   [y1810678@red99 c2]$ ./a.out 1.0
2   00111111 10000000 00000000 00000000
3   00111111 11110000 00000000 00000000 00000000 00000000
       00000000 00000000
4   [y1810678@red99 c2]$ ./a.out -1.0
5   10111111 10000000 00000000 00000000
6   10111111 11110000 00000000 00000000 00000000 00000000
       00000000 00000000
7   [y1810678@red99 c2]$ ./a.out 6.0
8   01000000 11000000 00000000 00000000
9   01000000 00011000 00000000 00000000 00000000 00000000
       00000000 00000000
```

問 2　まず、作成したソースコードを、Listing3 に示す。

Listing 3　問 2 のソースコード

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <stdbool.h>
4   int main(int ac, char *av[]) {
5   // ここから (-1)^0 * 1.1001 * の計算 2^-4
6     int a[] = {1,0,0,1};
7     float f = 1;
8     for(int i = 0; i < 4; i++) {
9         if(a[i] == 1) {
10            float temp = 1;
```

```
            for(int j = 0; j < i+1; j++) {
                temp /= 2;
            }
            f += temp;
        }
    }
    for(int i = 0; i < 4; i++) {
        f /= 2;
    }
    printf("%.10f\n", f);

// ------------------------------------------------------
// ここから (-1)^0 * 1.10011001 * の計算 2^-4
    int b[] = {1,0,0,1,1,0,0,1};
    double d = 1;
    for(int i = 0; i < 8; i++) {
        if(b[i] == 1) {
            double temp = 1;
            for(int j = 0; j < i+1; j++) {
                temp /= 2;
            }
            d += temp;
        }
    }
    for(int i = 0; i < 8; i++) {
        d /= 2;
    }
    printf("%.10lf\n", d);
    return 0;
}
```

これは、2 進数の小数点以下の部分をそれぞれ計算し、足し合わせた後に、必要回数分 2 で割っているものである。また、これの実行結果は以下の Listing4 ようになる

Listing 4　問 2 の実行結果

```
[y1810678@red99 c2]$ ./a.out
0.0976562500
0.0062408447
```

問 3　まず、作成したソースコードを、Listing5 に示す。

Listing 5　問 3 のソースコード

```
#include <stdio.h>

int main(void) {
```

```
4           double i = 2, j = 3;
5           double x = i / j + j / i;
6           printf("%.10lf\n", x);
7           return 0;
8       }
```

これは、割る数割られる数の両方を実数に変更して計算しているものである。また、printf の指定方法を変更して 10 桁まで表示するようにしている。また、これの実行結果は以下の Listing6 ようになる

Listing 6　問 3 の実行結果

```
1       [y1810678@red99 c2]$ ./a.out
2       2.1666666667
```

問 4　まず、作成したソースコードを、Listing7 に示す。

Listing 7　問 4 のソースコード

```
1       #include <stdio.h>
2
3       int main(void) {
4           float x, y;
5           x = 777777.7;
6           y = 1.111111;
7           y *= 10;
8           x += y;
9           printf("%f\n", x);
10          return 0;
11      }
```

これは、10 回足すのではなく、 1 度 y に 10 を掛けてから足している。これでもとのコードよりは精度が上がり計算できる。また、これの実行結果は以下の Listing8 ようになる

Listing 8　問 4 の実行結果

```
1       [y1810678@red99 c2]$ ./a.out
2       777788.812500
```

問 5　まず、作成したソースコードを、Listing9 に示す。

Listing 9　問 5 のソースコード

```
1       #include <stdio.h>
2       #include <stdlib.h>
3
4       void show_byte(unsigned char x) {
5           for(int i = 7; i >= 0; i--) {
```

```
6               printf("%d", (x>>i) & 1);
7           }
8           printf("\n");
9       }
10
11      int main(void) {
12          show_byte(0x6a);
13          show_byte(0x0f);
14          show_byte((0x6a)|(0x0f));
15          show_byte((0x6a)&(0x0f));
16          show_byte(~(0x6a));
17          return 0;
18      }
```

これは、関数 show_byte に「$(6a)_{16}$」、「$(0f)_{16}$」、「２つの数の論理和」、「論理積」、「$(6a)_{16}$ のビット反転させたもの」を引数として渡すことで、それぞれの２進数表現を表示させている。また、これの実行結果は以下の Listing10 ようになる

Listing 10　問 5 の実行結果

```
1       [y1810678@red99 c2]$ ./a.out
2       01101010
3       00001111
4       01101111
5       00001010
6       10010101
```

問 6　まず、作成したソースコードを、Listing11 に示す。

Listing 11　問 6 のソースコード

```
1       #include <stdio.h>
2       #include <stdlib.h>
3
4       void show_byte(unsigned char x) {
5           for(int i = 7; i >= 0; i--) {
6               printf("%d", (x>>i) & 1);
7           }
8           printf("\n");
9       }
10
11      int main(void) {
12          show_byte((0x11)<<1);
13          show_byte((0x11)<<2);
14          show_byte((0x11)>>1);
15          return 0;
16      }
```

これは、関数 show_byte に $(11)_{16}$ をそれぞれ左に 1 ビットシフトしたもの、左に 2 ビットシフトしたもの、右に 1 ビットシフトしたものを引数として渡すことで、それぞれの 2 進数表現を表示させている。また、これの実行結果は以下の Listing12 ようになる

Listing 12　問 6 の実行結果

```
[y1810678@red99 c2]$ ./a.out
00100010
01000100
00001000
```

宿題

問題 1　まず、作成したソースコードを、Listing13 に示す。

Listing 13　問 1 のソースコード

```
#include <stdio.h>

int main(void) {
    float a, b, c, d, s;
    a = 100001.0;
    b = 0.123456;
    c = 0.111111;
    d = -100000.0;
    s = 0.0;
    s += a;
    s += d;
    s += b;
    s += c;
    printf("%f\n", s);
}
```

これは、先に a と b を計算することで、桁落ちなどを防いでいる。また、これの実行結果は以下の Listing14 ようになる

Listing 14　問 1 の実行結果

```
[y1810678@red99 c2]$ ./a.out
1.234567
```

問題 2　まず、作成したソースコードを、Listing15 に示す。

Listing 15　問 2 のソースコード

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

```

```
4     void show_byte(unsigned char x) {
5         for(int i = 7; i >= 0; i--) {
6             printf("%d", (x>>i) & 1);
7         }
8         printf("\n");
9     }
10
11    int main(void) {
12        show_byte((0x6a)&(0xF0));
13        return 0;
14    }
```

これは、$(6a)_{16}$ と $(11110000)_2$ との論理積をとっているものである。そのため、末尾 4 ビットが 0 になる。また、これの実行結果は以下の Listing16 ようになる

Listing 16　問 2 の実行結果

```
1     [y1810678@red99 c2]$ ./a.out
2     01100000
```