CS1001A.10
习题课02

SA21011004
郭雨轩

实验03-T1

```
#include <math.h>
#define EPS 1e-6
int feq(float a, float b) {
    return fabs(a - b) < EPS; // 正确
}
```

```
a == b; // 错误
```

浮点数是否相等的比较要考察两数做差的绝对值小于EPS

实验03-T2

```
void case_1(int score) {
    switch (score / 5)
    // assume score >= 85
    {
    case 18:
        printf("4.0");
        break;
    case 17:
        printf("3.7");
        break;
    default:
        printf("4.3");
        break;
    }
}
```

```
void case_2(int score) {
    switch (score)
    // assume score >= 85
    {
    case 95 ... 100:
        printf("4.3");
        break;
    case 90 ... 94:
        printf("4.0");
        break;
    default:
        printf("3.7");
        break;
    }
}
```

错误点：（忽略考虑了100分是4.3的情况……）

实验03-Bonus

```c
float int2float(int i) {
    int sign;
    int exp;
    int frac;
    int result;
    if (i > 0) {
        sign = 0 << 31;
    } else if (i < 0) {
        sign = 1 << 31;
        i = -i;
    } else {
        result = 0;
        return *(float *)(&result);
    }
    exp = 0;
    for (int tmp = i; tmp != 1; tmp /= 2) {
        exp++; // 计算该整数的二进制科学计数法的指数大小
    }
    frac = i & ~((0x1) << (exp)); // 整数只会产生规约形式的浮点数，需要把最高位的1置0
    frac <<= (23 - exp); // 注意左移的位数与exp相关
    exp += 127;
    exp <<= 23;
    result = (sign | exp | frac); // 使用按位或运算拼接三部分
    return *(float *)(&result);
}
```

作业02-T2

```
#define SWAP(a, b) {\
    char __tmp = (a);\
    (a) = (b);\
    (b) = __tmp;\
}
void reverse_1(char s[], int a, int b) {
    for (int i = 0; i <= (b - a) / 2; ++i) {
        SWAP(s[a + i], s[b - i]);
    }
}
void reverse_2(char s[], int a, int b) {
    for (int i = a; i <= (a + b) / 2; ++i) {
        SWAP(s[i], s[a + b - i]);
    }
}
```

错误点:

1. 循环下标错误

2. 函数调用的方式不正确、传参数不正确

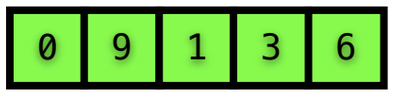
3. 错误增加返回值

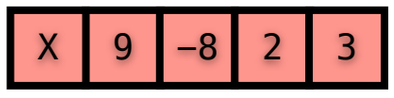
4. reverse函数调用时参数范围不正确

作业02-T3&4

```c
void interval_add1(int s[], int l, int r, int len) {
    s[l]++;
    if (r < len) {
        s[r + 1]--;
    }
}
```

何时需要更新右端：

Original seq: 

, Diff seq: 

需要考虑边界情况

实验06-Bonus

```
#define I 10 // 10个状态
#define J 100 // 100种输入
int transition[I][J];
void (*output_func[I])(void );
void init_state_machine() {
    initial "transition" "output_func"
}
int move(int current_state, int input) {
    return transition[current_state][input];
}
void output(int current_state) {
    (*(output_func[current_state]))();
}
```

状态机可以抽象为函数 $move : \mathrm{S} \times \mathrm{I} \rightarrow \mathrm{S}$和 $output : S \rightarrow void$，可以使用多维数组来存储状态转移的信息，输出函数仅与当前状态有关。

补充：数组与指针

```c
#include <stdio.h>

int main(void) {
    int a = 10; // correct
    int b[20]; // correct
    int c[] = {1, 3, 2}; // ?
    int *d = {1, 2, 3}; // ?
}
```

```c
#include <stdio.h>

int main(void) {
    char a = 'a'; // correct
    // char b = "b"; // ?
    char c[20] = "asdas"; // correct
    char d[] = "dadsadas"; // rwdata
    char *e = "asdasdas"; // ? rodata
}
```

补充：数组与指针

```c
#include <stdio.h>

int main(void) {
    int a[4] = {9, 1, 2, 3};
    printf("%d\n", *a); // 9
    printf("%d\n", a[0]); // 9
    printf("%d\n", *(a + 3)); // 3
}
```

```c
#include <stdio.h>

void func1(int *a, int b) {
    a[2] = 10;
    b = -1; // 如果改成 *(&b) = -1 ?;
}
void func2(int a[], int *b) {
    a[3] = -1;
    b[0] = 233;// b = 233 ?
}
int main(void) {
    int a[4] = {9, 1, 2, 3};
    int b = 10;
    func1(a, b);
    printf("a[2]: %d, b: %d\n", a[2], b);
    func2(a, &b);
    printf("a[3]: %d, b: %d\n", a[3], b);
}
```