



Testing | HW 2

Выполнил: Васюков Александр, avvasiukov@edu.hse.ru, @vasyukov_al

Github: <https://github.com/vasyukov1/hse-testing/tree/main/hw2>

1. Неправильная инициализация переменной `r` приводит к постоянному результату `0` и неверному ответу при введении неположительных значений второго аргумента (12:20 09.02.26)

1. Состояние системы до вызывавшего ошибку воздействия:

```
public int pow(int a, int b) {  
    int r = 0;  
  
    while(b > 0) {  
        if((b&1) != 0) r *= a;  
        r *= r;  
        b >>= 1;  
    }  
  
    return r;  
}
```

2. Описание произведенного воздействия:

Вызов функции `pow(2, 2)`, `pow(2, 0)` и `pow(2, -1)`.

3. Указание на фактическое возвращаемое значение / состояние после воздействия:

Вернулся ответ 0.

4. Указание на ожидаемое возвращаемое значение / состояние после воздействия:

Ожидались ответы 4, 1, 1 так как $2^2 = 4$, а при неположительном втором аргументе должна возвращаться 1, но данный код всегда будет возвращать 0, потому что r инициализирован нулём, а потом умножается сам на себя, то есть

$$0 * 0 = 0.$$

5. Ссылка на нарушенное требования:

Нарушено требование:

- 1 "Предусловие тривиально, т.е. метод должен работать для всех целочисленных значений своих параметров";
- 2a "при нулевом значении второго аргумента и любом значении первого должен возвращаться результат 1";
- 2b "при отрицательных значениях второго аргумента и любом значении первого должен возвращаться результат 1".

Описание изменений предпринятых для исправления ошибки:

Инициализация переменной r единицей: int r = 1. А также исправлен код-стиль:

```
public int pow(int a, int b) {  
    int r = 1;  
  
    while (b > 0) {  
        if ((b & 1) != 0) {  
            r *= a;  
        }  
        r *= r;  
        b >>= 1;  
    }  
  
    return r;  
}
```

2. Неверный алгоритм возведения в степень (12:35 09.02.26)

- Состояние системы до вызывавшего ошибку воздействия:

```
public int pow(int a, int b) {  
    int r = 1;  
  
    while (b > 0) {  
        if ((b & 1) != 0) {  
            r *= a;  
        }  
        r *= r;  
        b >>= 1;  
    }  
  
    return r;  
}
```

- Описание произведенного воздействия:

Вызов функции `pow(2, 5)`.

- Указание на фактическое возвращаемое значение / состояние после воздействия:

Вернулся ответ `1024`.

- Указание на ожидаемое возвращаемое значение / состояние после воздействия:

Ожидался ответ `32`, так как $2^5 = 32$.

- Ссылка на нарушенное требования:

Нарушен пункт 1 “Предусловие тривиально, т.е. метод должен работать для всех целочисленных значений своих параметров” — неверная реализация диахотомического алгоритма.

Описание изменений предпринятых для исправления ошибки:

Добавлена переменная `int base = a` для сохранения изначального числа, которое возводится в степень.

```
public int pow(int a, int b) {  
    int r = 1;  
    int base = a;  
  
    while (b > 0) {  
        if ((b & 1) != 0) {  
            r *= base;  
        }  
        base *= base;  
        b >>= 1;  
    }  
  
    return r;  
}
```

3. Неверное поведение при переполнении (14:50 09.02.26)

1. Состояние системы до вызывавшего ошибку воздействия:

```
public int pow(int a, int b) {  
    int r = 1;  
    int base = a;  
  
    while (b > 0) {  
        if ((b & 1) != 0) {  
            r *= base;  
        }  
        base *= base;  
        b >>= 1;  
    }  
}
```

```
    return r;
}
```

2. Описание произведенного воздействия:

Вызов функции `pow(3, 100)`.

3. Указание на фактическое возвращаемое значение / состояние после воздействия:

Вернулся ответ `-818408495`.

4. Указание на ожидаемое возвращаемое значение / состояние после воздействия:

Ожидался ответ `1329075153`, так как по требованию необходимо брать значения по модулю `2^31`.

5. Ссылка на нарушенное требования:

Нарушен пункт 2с "При переполнении (т.е. если точный результат возвведения

в степень превосходит по абсолютной величине `2^31-1`) возвращается результат возвведения в степень по модулю `2^31`".

Описание изменений предпринятых для исправления ошибки:

Изменён тип переменных с `int` на `long`, добавлена переменная

`final long MOD = 1L << 31`, которая используется в цикле после умножения `r` и `base`.

```
public int pow(int a, int b) {
    long r = 1L;
    long base = a;
    final long MOD = 1L << 31;

    while (b > 0) {
        if ((b & 1) != 0) {
            r = (r * base) % MOD;
        }
        base = (base * base) % MOD;
```

```
    b >= 1;  
}  
  
return (int) r;  
}
```