针对所谓宇宙最快的矩阵转置方法的一些测试和思考

最原始版本的转置算法就不贴上来了，直接看一下测试代码：

```
template<typename T>
void Matrix<T>::test4MatrixTranspose()
{
    std::cout << "--------------------initialize matrix a-------------------
-" << std::endl;
    Matrix<int> a(3, 5, 5);
    std::cout << "a._t:    " << a._t << std::endl << a << endl;

    std::cout << "--------------------a.transpose() ==> b-------------------
-" << std::endl;
    Matrix<int> b = a.transpose();
    std::cout << "a._t:    " << a._t << std::endl << a << endl;
    std::cout << "b._t:    " << b._t << std::endl << b << endl;

    std::cout << "--------------------b.transpose() ==> c-------------------
-" << std::endl;
    Matrix<int> c = b.transpose();
    std::cout << "b._t:    " << b._t << std::endl << b << endl;
    std::cout << "c._t:    " << c._t << std::endl << c << endl;
}
```

我们加上断点以后执行结果如下：

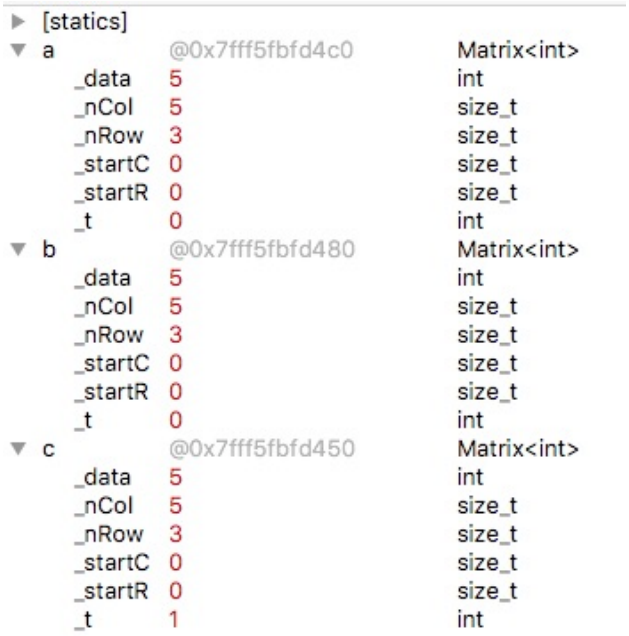
```
--------------------initialize matrix a--------------------
a._t:    1
5       5       5       5       5
5       5       5       5       5
5       5       5       5       5

--------------------a.transpose() ==> b--------------------
a._t:    0
5       5       5
5       5       5
5       5       5
5       5       5
5       5       5

b._t:    1
5       5       5       5       5
5       5       5       5       5
5       5       5       5       5

--------------------b.transpose() ==> c--------------------
b._t:    0
5       5       5
5       5       5
5       5       5
5       5       5
5       5       5

c._t:    1
5       5       5       5       5
5       5       5       5       5
5       5       5       5       5
```



很明显有大问题在里面，至少我们可以看到，当`a`转置完了赋值给`b`，`b._t`的值为默认的1，但是矩阵`b`和`a`初始值竟然一毛一样，`b`经过转置以后才勉为其难将`b._t`变为0而且此时跟`a`转置完的结果才一样，至于`c`，重复上次操作，自己感受吧。

所以之前一直以为转置出现了问题其实不准确，估计是构造函数出了问题。

但是构造函数那么多，我怎么知道该改那个？初步估计可能是那个`swap`函数有问题，**但是怎么改我还没想好，也不想去想**。