1 测试程序的设计思路

首先本次测试我们需要测试以下功能的正确性：

1. constructor, copy constructor, move constructor, deconstructor
2. copy assignment, move assignment
3. 前后插入, 删除
4. front, back
5. ++ / –
6. 解引用 (*)
7. == / !=
8. 不同模板类

测试方法分别如下：

1. constructor: 初始化一个初始值为 1，2 的链表 b，打印 b 看是否符合预期。

2. copy constructor: 通过拷贝 b 构造一个链表 c。打印 c 看是否符合预期。再修改 b 的 front 和 back 后再次打印 c。若 c 无变化则正确。

3. copy assignment: 先初始化一个链表 a，再将 b 拷贝赋值给 a。打印 a 和 b。再修改 b，若 a 无变化则正确。

4. move constructor: 通过 a 移动构造一个链表 d。打印 d 的值，并查看 a 是否为 empty。若 a 的值成功赋给了 e 且 a 为空则正确。

5. move assignment: 将构造换为赋值后同上。

6. deconstructor: 用 valgrind 检查是否存在内存泄漏。

7. push_front/back, pop_front/back: 在修改 b 时使用这四个函数并检查结果是否符合预期。

8. front, back: 用 front, back 输出 push_front/back 的结果。

9. *, ++, –: 将 b.begin() 赋值给一个 iterator it，分别输出 $*it++$, $*++it$, $*--it$, $*it--$，并在每次执行完后输出 *it。观察是否符合预期。

10. 模板类: 初始化一个模板类为 std::string 的链表 f，push_front 插入”runis”, push_back 插入”cierra”，打印 f 若输出”cierra runis” 则正确。

11. != / ==: 由于!= 等于 == 取反，只需测试 == 的正确性。在 3. 中将 b 拷贝赋值给 a 后，输出 $a.begin() == b.begin()$。由于 iterator 的 == 输出的是是否指向同一 node，所以虽然此时二者 node 的值相等，结果仍应该为 0。接着测试 $b.begin() == b.begin()$ 应该输出 1。最后，给一个 iterator it1 赋值为 b.begin()，将 b.begin() 修改后测试 $it1 == b.begin()$，应该为 1。

2 测试的结果

测试结果一切正常。

我用 valgrind 进行测试，发现没有发生内存泄露。