作业里常见的问题：

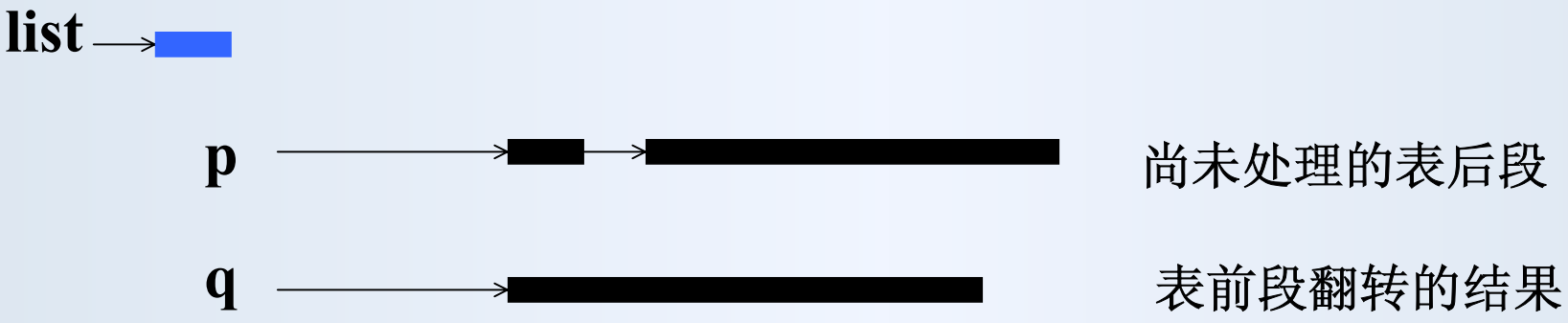
- 作业里的一些程序不能顺利编译运行，可能提交方式不对。以后用到网页中提供的文件，请统一用原文件名
- 有些同学对头文件的理解不对，在其中定义函数等
- 有些同学定义常量**MAX**太小，导致数组越界，运行出错
- 插入删除操作没有先判断可行性（下标是否在合法范围？链表是否为空？等等）
- 指针操作之前不判断是否为空指针（**NULL**）
- 在条件判断中用“=”（应该用“==”）
- 函数定义每写返回值类型，函数调用的参数前面写类型
- 在有返回值的函数里缺 **return** 语句

进一步提高程序的质量：

- 有些同学的程序格式太乱。应该用良好的格式写程序，各种编程环境都为此提供了很好的支持（以后将适当扣分）
- 判断函数（如判断是否空表），最好用预定义的常量 **TRUE** 和 **FALSE**（提高程序的可读性）
- 程序的输出格式，应该有一点设计考虑。应考虑如何使输出比较容易读。在输出语句里适当地使用 **\t**，**\n** 等
- 在程序里适当的地方空行，加入适当的注释等等
- 选择合适的变量名和函数名，选择合适的文件名

表翻转：

对表最容易做的是前端操作：取下一个元素或加上一个元素



假定 p 指向要翻转的表，q、t 是两个表指针：

```
for (q = NULL; p != NULL; ) {
    t = p;  p = t->link; /* 从未处理的表前段取下一结点 */
    t->link = q;  q = t; /* 把该结点连到翻转后表段的前端 */
}
/* 现在 q 指向翻转后的表 */
```

```
void reverse(LinkList list) { /* list 是带有头结点的表 */
    LinkList p = list->link, q = NULL, t;
    /* p 指向待翻转的表结点链，q 指向空表 */
    while ( p != NULL ) {
        t = p;
        p = p->link;
        t->link = q;
        q = t;
    }
    list->link = q;
}
```

循环不变关系：

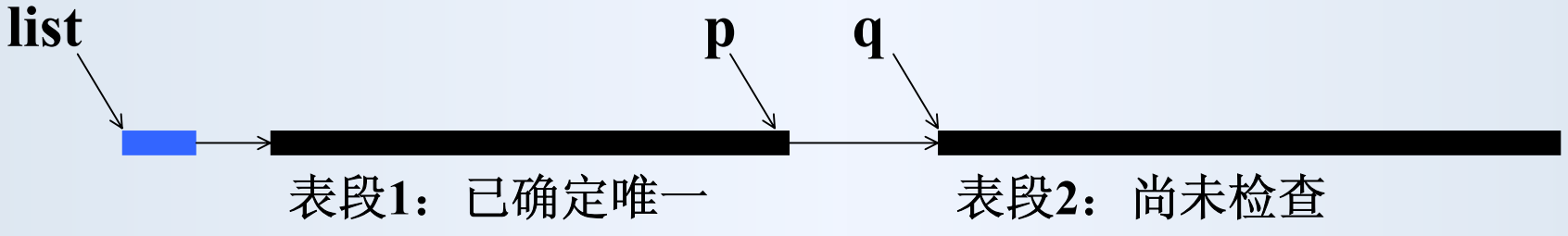
把 q 所指的表段翻转回去，连到 p 所指表段之前，得到的就是初始的待翻转表段

删除表中重复结点：

表 **list**

从 **list->link** 到 **p** 的结点为已确定唯一的元素，**q = p->link**

这就是“循环不变式”



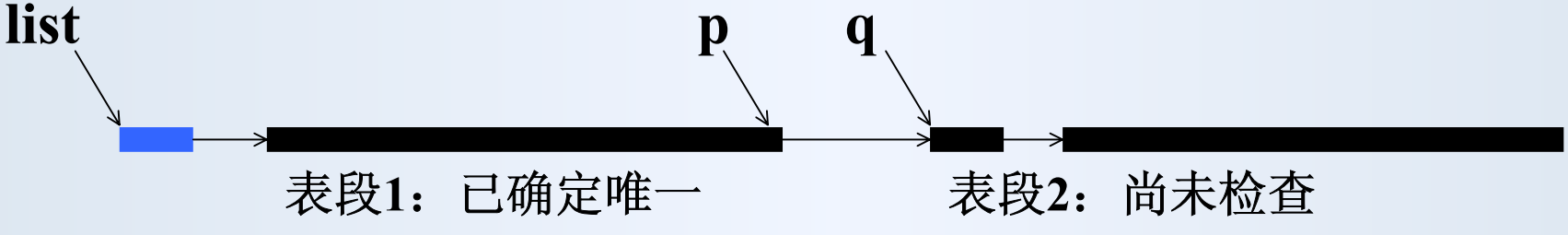
初始：表段1（从 **list->link** 到 **p**）为空，表段2 （从 **q** 到表结束）包含所有实际结点

p = list; q = p->link;

表段2不空则处理没完成，应继续循环（条件：**q != NULL**）

循环的每一步应该缩短表段2，处理其第一个结点

循环中检查 q 所指结点是否在表段1中出现，如果未出现就延伸表段1（表段2缩短），否则就删除（表段2也缩短）。



需要用一个内层循环完成检查，而后分别情况进行处理

注意“循环不变式”（invariant）

```
void unique (LinkList list) {
   LinkList p = list, q = list->link, t;
   while (q != NULL) {
      for (t = list->link; t != q && t->info != q->info; t = t->link)
            ;  /* 空循环体 */
      if (t == q)
         p = q; /* 该数据没出现过，接受 */
      else { /* q 所指结点的数据已在前面出现 */
         p->link = q->link; /* 将 q 所指结点摘下删除 */
         free(q);
      }
      q = p->link;  /* 无论如何，总让 q 指向 p 的下一结点 */
   }
}
```

写程序前应首先把问题想清楚。空表等特殊情况最好也能统一处理（常可做到）。请检查自己的作业，看能否改善