项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 沈星宇

学 号： 1951576

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

1. **分析**
2. **应用背景**

设某银行有A，B两个业务窗口，且处理业务的速度不一样，其中A窗口处理速度是B窗口的2倍----即当A窗口每处理完2个顾客时，B窗口处理完1个顾客。给定到达银行的顾客序列，请按照业务完成的顺序输出顾客序列。假定不考虑顾客信后到达的时间间隔，并且当不同窗口同时处理完2个顾客时，A窗口的顾客优先输出。

1. **项目功能要求**

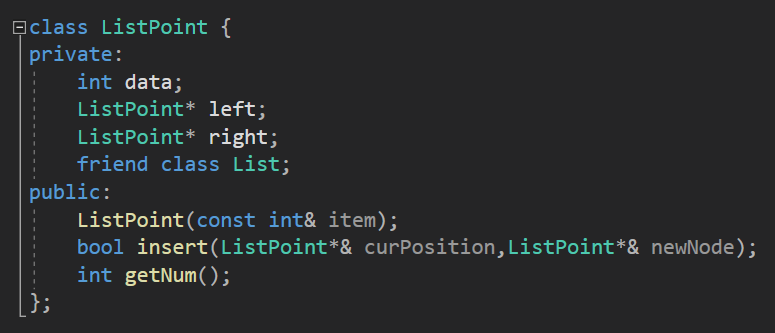
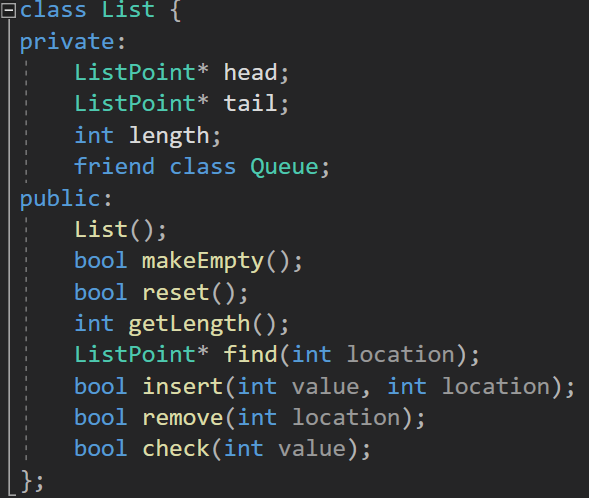
1、输入说明：输入为一行正整数，其中第一数字N（N<=1000）为顾客总数，后面跟着N位顾客的编号。编号为奇数的顾客需要到A窗口办理业务，为偶数的顾客则去B窗口。数字间以空格分隔。

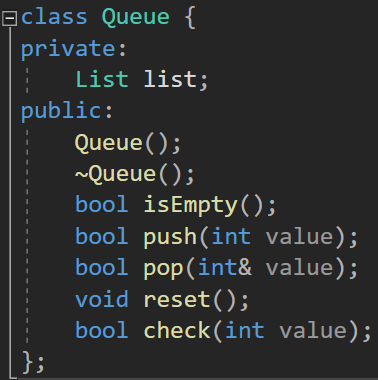
2、输出说明：按照业务处理完成的顺序输出顾客的编号。数字键以空格分隔，但是最后一个编号不能有多余的空格。

3、测试用例：

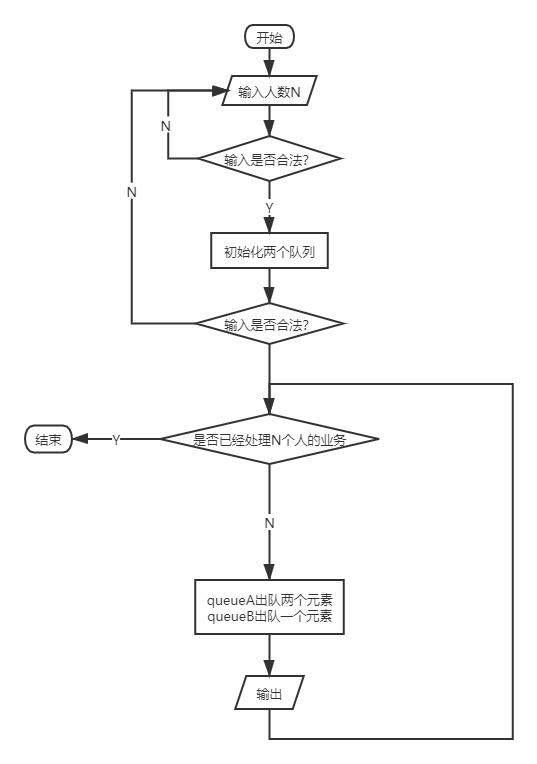


1. **设计**
2. **数据结构设计**

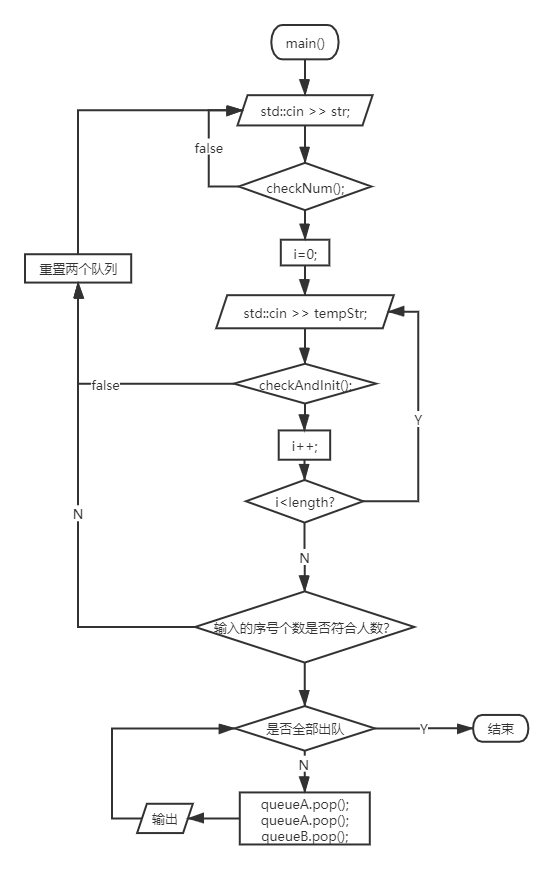
**、**



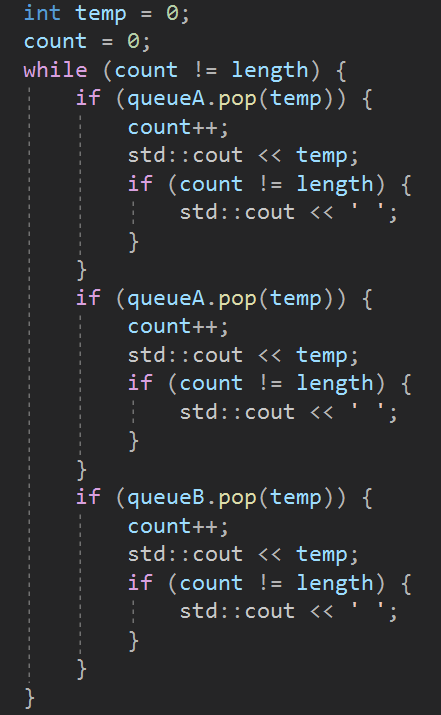
1. ListPoint是结点的结构，内部包含了存储的数字data和指针left、right，为了锻炼写双链表的能力才写成了双链表，其实本项目单链表即可实现
2. List是链表的结构，里面包含了常规的操作的函数，为了使List能够访问ListPoint的成员，故将其在ListPoint类里写作了友元。
3. Queue是队列的结构，里面包含了一个链表（链式的队列），为了使Queue能够访问List的成员，故将其再List类里面写作了友元。
4. **程序流程设计**



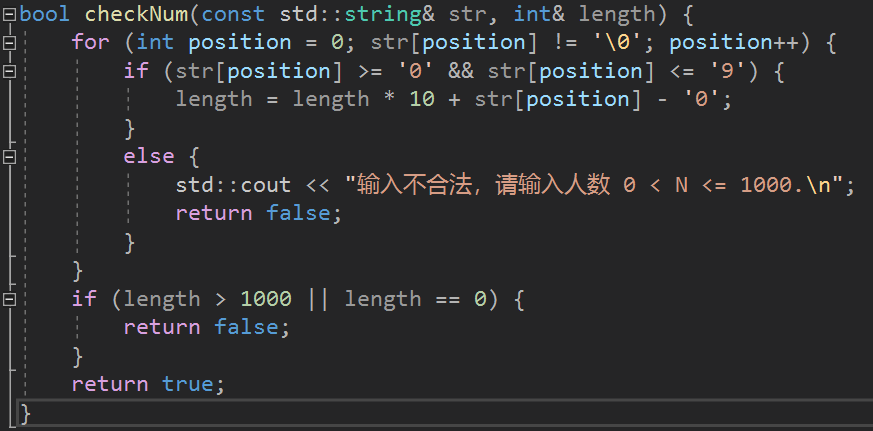
1. 首先输入的是一串数字序列，但我们先对第一个数字，即人数N进行合法性检测，非正整数输入均会被要求重新输入
2. 然后循环读入N个数字，即序号，注意第N个之后的数字不会被读入。读入数字的同时会根据它的奇偶性分别压入queueA或queueB，同时会对数字进行合法性检测，如果读到非正整数或者读到重复的号码，都会被要求重新输入
3. 循环让队列中的元素出队，每次循环queueA出队两个元素，queueB出队一个元素，一边出队一边输出
4. **实现**
5. **main()函数的内部逻辑**

****

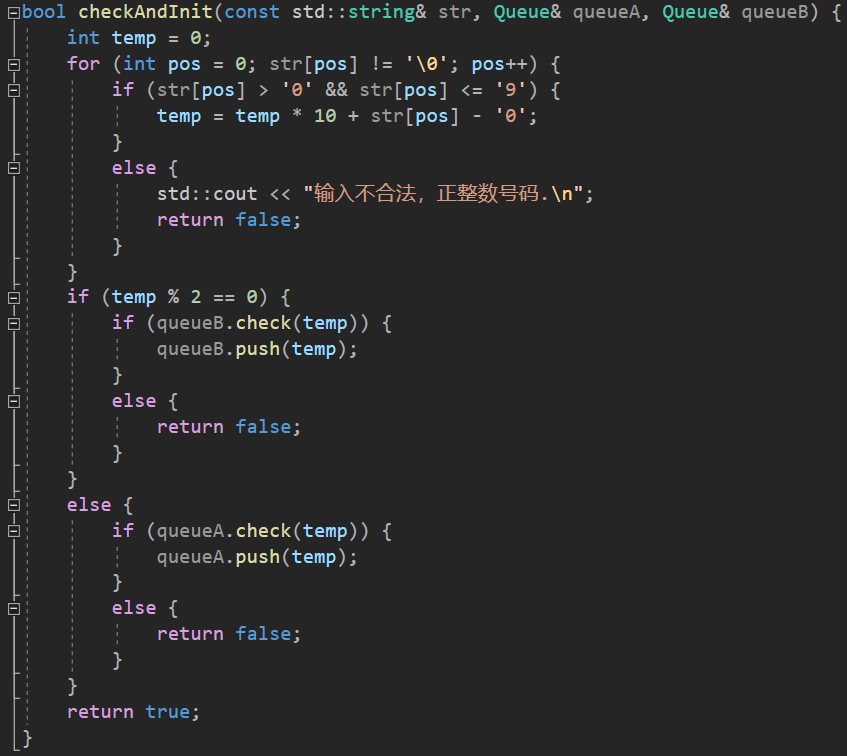
1. main()函数会首先读入第一个数字str（人数），并调用checkNum();函数对读入的字符串进行合法性检测，还通过传入int& length的形式将转化成int类型的人数传出。如果输入不合法则会要求重新输入。
2. 接着连续读入length个数字，并调用checkAndInit();函数对读入的字符串进行合法性检测（包括不能有重复的数字出现），并根据奇偶将其分别压入队列queueA和queueB当中。如果输入不合法则会要求重新输入。
3. 结束输入后循环让队列中的元素出队，每次循环queueA出队两个元素，queueB出队一个元素，一边出队一边输出

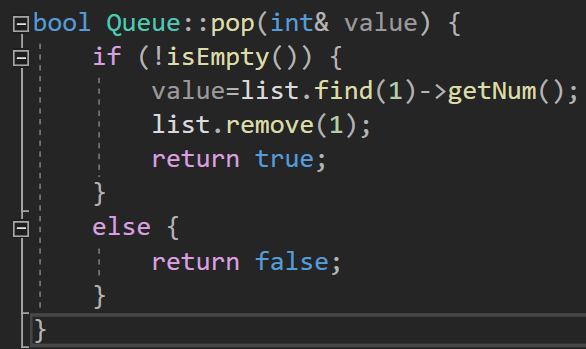


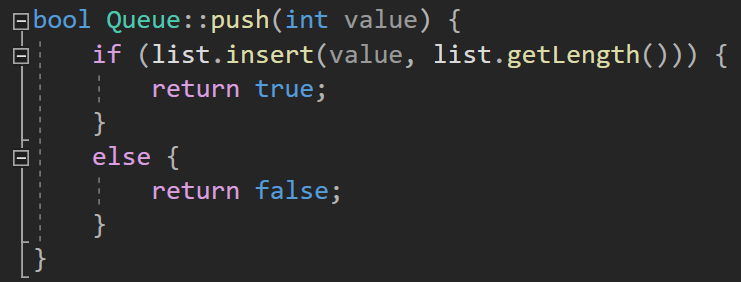
1. **checkNum()的实现**



1. 对于输入的字符串，进行逐位的判断和操作，用str[position] >= '0' && str[position] <= '9'来判断是否输入了非法字符，将其转化为int类型的length
2. 判断输入的数字是否超过1000
3. **checkAndInit()的实现**



1. 对于输入的字符串，进行逐位的判断和操作，用str[position] >= '0' && str[position] <= '9'来判断是否输入了非法字符，将其转化为int类型的temp
2. 调用Queue::check();来查看temp是否在前面的序列种已经出现过，若已经出现则说明该序列不合法，直接中止要求重新输入。根据temp的奇偶将其用Queue::push();函数压入队列。
3. **Queue::push();和Queue::pop();的实现**



由于是链式的队列，所以push()操作就是在链表的末尾插入元素，pop()操作就是将链表的首元素给删除。

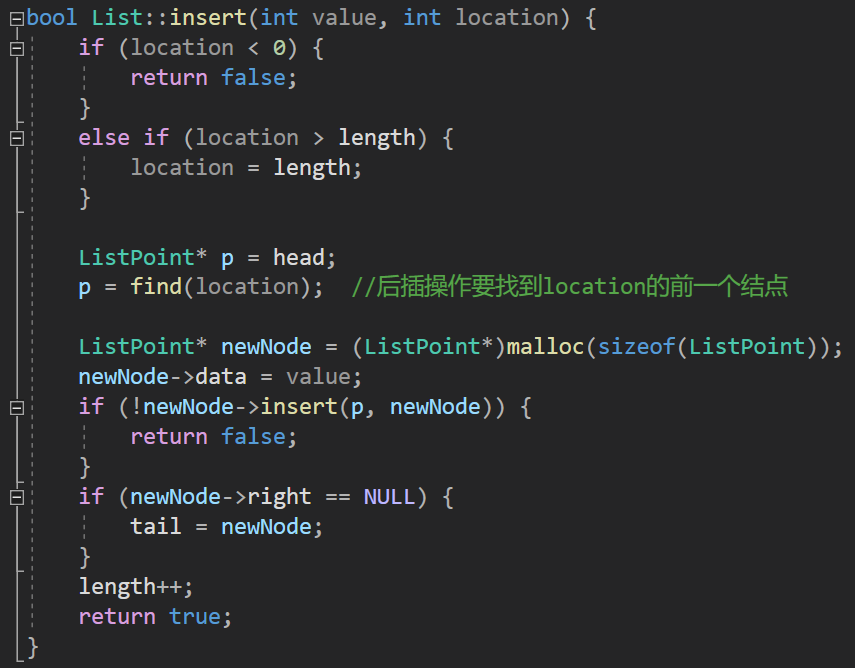
1. **List::insert()的实现**

insert()函数其实是按位后插，即在location的后面插入一个结点。

（1） 首先会对location的合法性进行检测（虽然本项目按照顺序插入在队尾不会涉及到这个内容，但是为了安全性和整体性仍然进行检测）。禁止在第0位及第0位之前插入结点；在超出链表的最后一个结点的时候在最后一个结点插入。

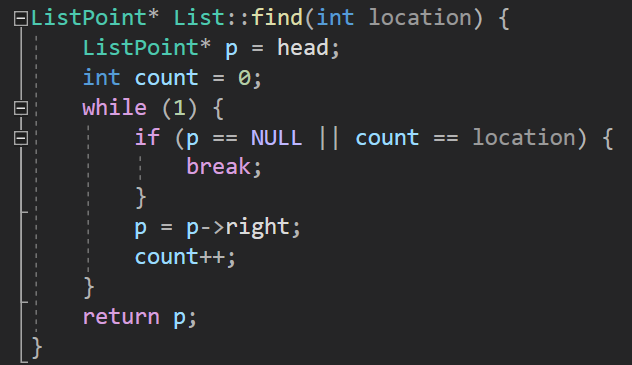
（2）然后调用find()函数去查找到location位置的结点，用malloc为新的结点动态分配一个内存，修改新节点newNode的data并且调用newNode的insert()函数将它的指针链接上去。

（3）调整尾指针，使链表的长度增加。



1. **List::find()的实现**

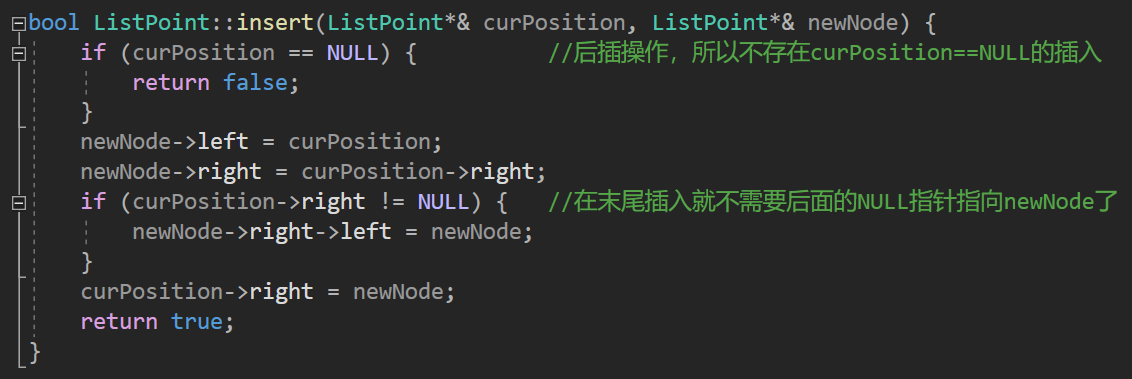
按位查找，用一个指针从第一个结点指向最后一个结点，该函数返回所寻找位置的结点指针。



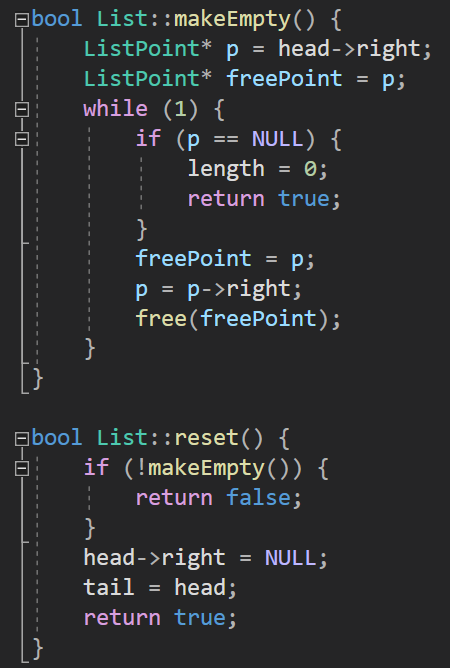
1. **ListPoint::insert()的实现**

后插操作要不存在curPositon==NULL的情况，即不在空指针后面插入内容。

把结点用指针链接起来。因为最后一个结点的right指针指向的是NULL，如果是在链表的末尾插入结点，那么不需要将newNode之后的结点（其实是空指针）的left指针指向newNode，因为空指针不存在left指针。



1. **List::reset()和List::makeEmpty()的实现**

（1）在程序接收到不合法的输入时，程序会将链表清空，并且请求用户重新输入数字序列，直到输入的序列合法为止。实现这一部分功能的是reset()和makeEmpty()函数。

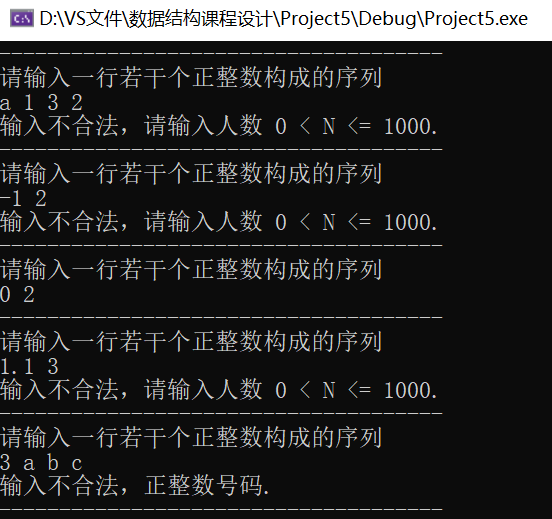
（2）makeEmpty()是释放内存的作用，在main()函数的末尾、程序结束时也会被调用，防止碎片化内存块的出现。它通过指针p去访问链表的每一个结点，将每一个结点的内存释放，然后重置链表的长度length = 0。

（3）对于reset()函数，用于清空重置链表，内部会先调用makeEmpty()释放内存，然后将头指针的right指针指向NULL（防止访问脏数据），将尾指针指向头结点。

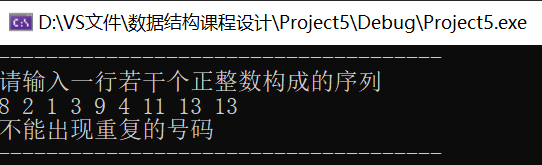
1. **测试**

1、 **初始化表格的合法性检测**

（1） 输入非正整数、浮点数和字符要求重新输入



**2、不允许出现重复的号码**



**3、 一般情况**

