**实验二 栈与队列的应用**

PB17050941 李喆昊

1. 问题描述

题目1：简单的行编辑器（要求实现对文本文件的编辑）

“简单的行编辑器”抽象定义如下：

文本：由0个或多个字符构成的单行序列。这些字符的ASCII码在闭区间[32, 126]内，也就是说，这些字符均为可见字符或空格；如果这段文本为空，我们就说这个文本编辑器是空的。

光标：在一段文本中用于指示位置的标记，可以位于文本的第一个字符之前，文本的最后一个字符之后或文本的某两个相邻字符之间，光标初始位置为0。

文本编辑器：为一个可以对一段文本和该文本中的一个光标进行如下七条操作的程序。：

1.  Move k：将光标移动到第k个字符之后，如果k=0则将光标移到文本第一个字符之前。

2.  Insert n （换行） S：在光标后插入长度为n的字符串S，光标位置不变，n≥1。

3.  Delete n：删除光标后的n个字符，光标位置不变，n≥1。

4.  Rotate n：反转光标后的n个字符，光标位置不变，n≥1。

5.  Get：输出此时程序内保存的文本

6.  Prev：光标前移一个字符。

7.  Next：光标后移一个字符。

编写一个程序要求：

1.  建立一个空的文本编辑器。

2.  从输入文件中读入一些操作指令并执行。

3.  对所有执行过的Get操作，将指定的内容写入输出文件。

输入格式：输入文件中第一行是指令条数N，以下是需要执行的N个操作。除了回车符之外，输入文件的所有字符的ASCII码都在闭区间[32, 126]内。且行尾没有空格。

输出格式：依次对应输入文件中每条Get指令的输出，不得有任何多余的字符。

输入输出样例：

|  |  |
| --- | --- |
| 输入样例 | 输出样例 |
| 12  Insert 13  Balanced eert  Get  Move 2  Delete 5  Get  Next  Insert 6  editor  Move 0  Get  Move 10  Rotate 4  Get | Balanced eert  Bad eert  Badeditor eert  Badeditor tree |

说明：

1.  各种操作总数不超过50000个。

2.  所有Insert插入的字符数之和不超过100000。

3.  Delete操作、Rotate操作执行时光标后必然有足够的字符。Move、Prev、Next操作不会把光标移动到非法位置。

4.  输入文件没有错误。

题目2：括号配对检验（不考虑括号优先级）

假设一个表达式有英文字母（大、小写）、数字、四则运算符（+，-，\*，/）和左右小括号、中括号、大括号构成，以“@”作为表达式的结束符。请编写一个程序检查表达式中的左右大中小括号是否匹配，若匹配，则返回“YES”；否则返回“NO”。

输入格式：

输入文件中第一行是表达式数目N

之后是需要进行括号配对检测的N个表达式。

输出格式：

N行输出分别对应输入的N行表达式，每行都为“YES”或“NO”

输入输出样例：

|  |  |
| --- | --- |
| 输入样例 | 输出样例 |
| 3  2\*[(x+y)/(1-x)]@  (25+x)\*(a\*(a+b+b)}@  {1+2[c-d(7/9)}@ | YES  NO  NO |

说明：

表达式长度小于100000，左括号少于10000个。

题目3：迷宫求解（要求实现随机迷宫、最短路径的提取）

有一个m\*n格的迷宫(表示有m行、n列)，其中有可走的点也有不可走的点，我们用1表示可以走，0表示不可以走。现在要你编程找出最短的道路，要求所走的路中没有重复的点，走时只能沿着上下左右四个方向。如果没有道路，则输出-1。

输入格式：

第一行是两个数m，n(1<m，n<100)

接下来是m行n列由1和0组成的数据

最后两行是起始点和结束点的坐标，输入坐标时先行后列。

输出格式：

一条最短的路径，描述一个点时用(x，y)的形式，除开始点外，其他的都要用“->”表示方向。

如果没有一条可行的路则输出-1。

输入输出样例：

|  |  |
| --- | --- |
| 输入样例 | 输出样例 |
| 5 6  1 0 0 1 0 1  1 1 1 1 1 1  0 0 1 1 1 0  1 1 1 1 1 0  1 1 1 0 1 1  1 1  5 6 | (1,1)->(2,1)->(2,2)->(2,3)->(2,4)->(2,5)->(3,5)->(4,5)->(5,5)->(5,6) |

说明：

答案可能不唯一，输出其中任意一条可行的最短路径即可。

题目4：银行业务模拟（具体要求：参考习题集中P100的实验2.6）

客户业务分为两种：

第一种是申请从银行得到一笔资金，即取款或借款；

第二种是向银行投入一笔资金，即存款或还款。

银行有两个服务窗口，相应地有两个队列。客户到达银行后先排第一个队，处理每个客户业务时，如果属于第一种，且申请额超出银行现存资金总额而得不到满足，则立刻排入第二个队等候直至满足时才离开银行;否则业务处理完后立刻离开银行,每接待完一个第二种业务的客户，则顺序检查和处理(如果可能)第二个队列中的客户，对能满足的申请者予以满足,不能满足者重新排到第二个队列的队尾。注意，在此检查过程中，一旦银行资金总额少于或等于刚才第一个队列中最后一个客户(第二种业务)被接待之前的数额，或者本次已将第二个队列检查或处理了一遍，就停止检查(因为此时已不可能还有能满足者)转而继续接待第一个队列的客户。任何时刻都只开一个窗口。假设检查不需要时间，营业时间结束时所有客户立即离开银行。

写一个上述银行业务的事件驱动模拟系统，通过模拟方法求出客户在银行内逗留的平均时间。

输入格式：

第一行输入三个数N、total、close\_time、average\_time，分别表示来银行的总人数、银行开始营业时拥有的款额、今天预计的营业时长和客户交易时长

之后的N行每行输入两个数a、b，第一个数a为客户办理的款额，用负值和正值分别表示第一类和第二类业务。第二个数b为客户来到银行的时间

输出格式：

前N行输出分别对应每个客户的等候时长

第N+1行输出为客户等候平均时长，结果去尾法保留整数即可

输入输出样例：

|  |  |
| --- | --- |
| 输入样例 | 输出样例 |
| 4 10000 600 10  -2000 0  -11000 10  -10000 30  2000 50 | 0  590  30  0  155 |

样例解释：

开始时银行有存款10000元，客户交易时长10

1号客户0时到达银行，于10时办理完手续离开，此时银行有存款8000

2号客户10时到达银行，此时银行内存款不足，进入队列2

3号客户30时到达银行，此时银行内存款不足，进入队列2

4号客户50时到达银行，于60时办理完手续离开，此时银行有存款10000元。

这时扫描队列2，队列2中第一个客户2号客户的要求仍然不满足，2号客户出队再入队；此时队列2中第一个客户3号客户满足要求，因此3号客户于60时开始办理手续，于70时办理完毕离开银行，等待时间为60-30=30，此时银行有存款0元。

然后直到银行营业时间600时之前2号客户都没有离开银行，等待时间600-10=590。

说明：

输入数据保证客户的来银行的时间按升序排列，并保证银行关闭营业之后不会再有人来到银行。

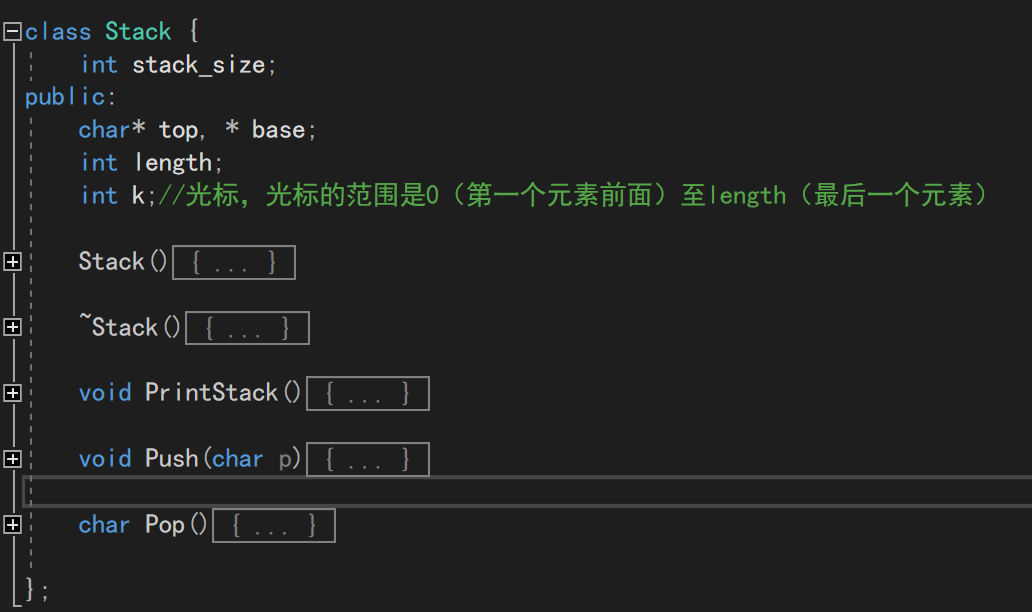
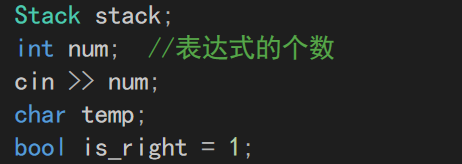
1. 算法描述
2. 数据结构描述

题目1，2使用栈，题目3，4使用队列。逻辑结构为线性结构。

**主要变量**：

**题目1 文本编辑器**

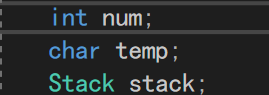
①栈类定义

②num记录输入的指令的数目，temp记录指令的第一个字母。

**题目2 括号匹配**

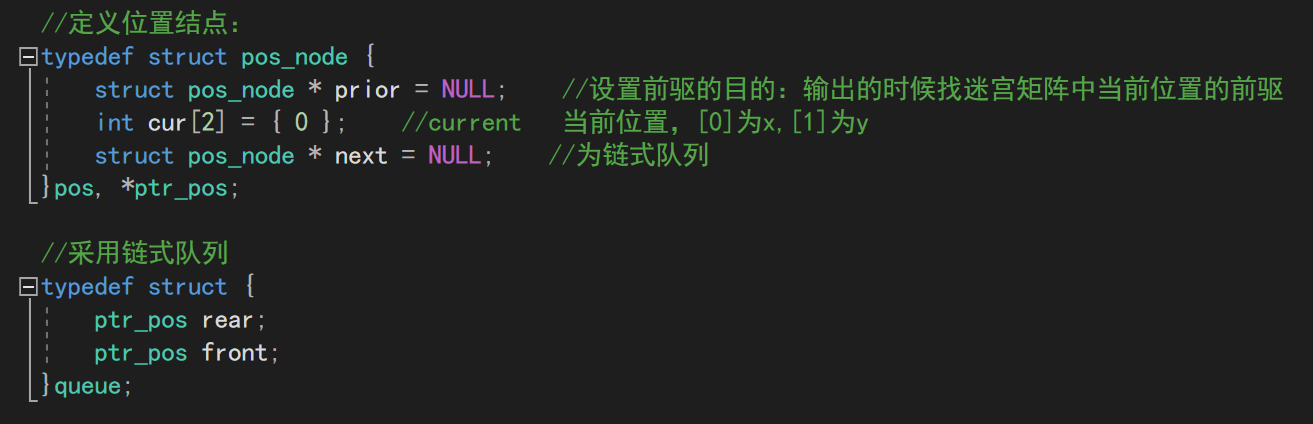
栈类定义与题目1相同



num记录表达式的数目，temp作为表达式中字符的临时记录

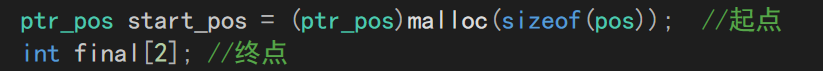
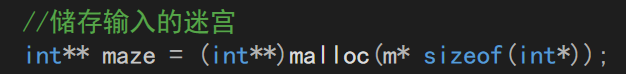
**题目3 迷宫求解**

采用链式队列，主要结构体的定义如下



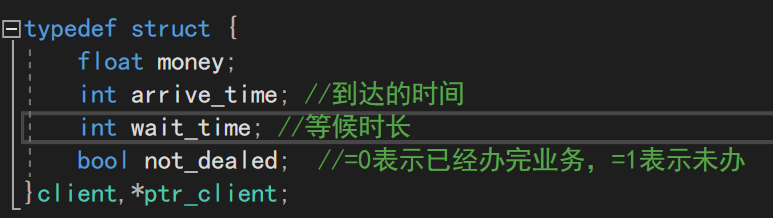
m,n分别是迷宫的行数和列数



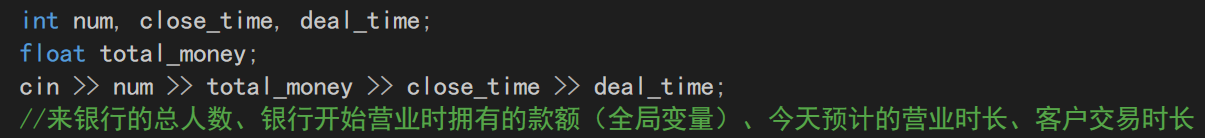


**题目4 银行模拟**

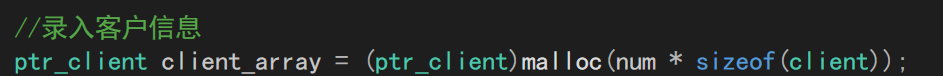
①银行客户结构体的定义



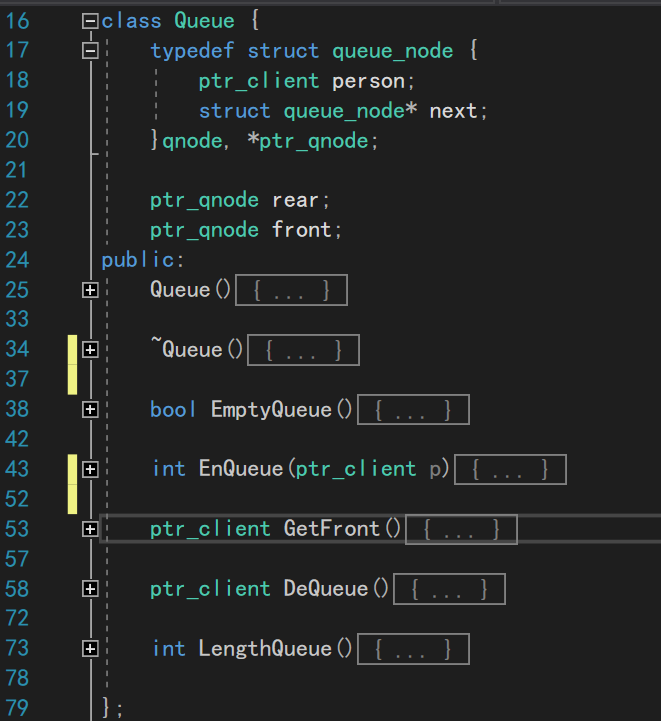
②



③存储输入的客户信息的数组

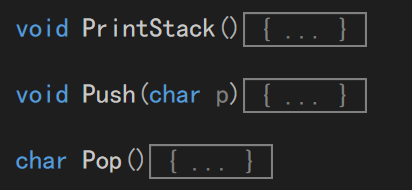


④队列类定义如下



1. 程序结构描述
2. **题目1**

主要的函数就是栈类定义中的成员函数。

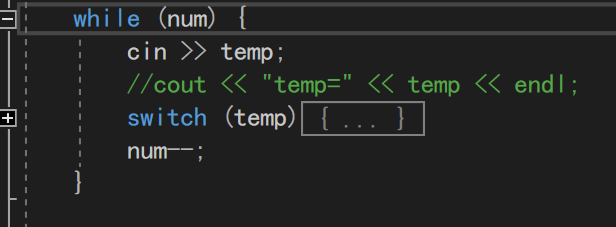


PrintStack( ): 按一定顺序打印栈中所有元素

Push( p ): char型元素p入栈

Pop( ) : 栈顶元素出栈

另外，main函数的主体是一个while循环，功能是接收并判断当前输入的指令



switch( temp )语句的功能就是通过判断输入的指令的第一个字母来区分不同指令。

1. **题目2**

主要的栈成员函数同题目1

题目2main函数的主体也是一个while循环，用于读取表达式并检查正确性。

1. **题目3**



功能：初始化队列que，成功返回1，否则返回0



功能：元素pos入队列que，成功返回1，否则返回0



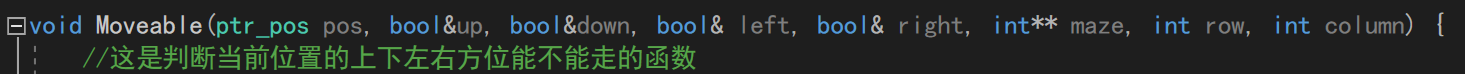
功能：对头元素出队列que，并通过返回值返回指向该元素结点的指针



功能：以（x, y）二维坐标形式打印p指向的位置结点的信息



功能：最终迷宫路径结果的打印输出，p为路径的起点，final为路径终点





1. **题目4**



功能：处理业务二

接口：que2：队列2，total\_money:银行总钱数，time：时间，deal\_time：每位客户交易的时间



功能：处理业务一，如有不能满足的客户，将其放至队列二队尾

接口：que1、que2：队列1、2，total\_money:银行总钱数，time：时间，deal\_time：每位客户交易的时间

三．调试分析

（1）测试数据使用的是助教提供的测试数据。

（2）程序调试中遇到的问题：

① 题目1 文本编辑器中，需要对文本进行Rotate即反转操作，由于位移量计算错误导致结果不对。

解决方法：使用一些具体的例子直观地发现了错误并修改。

② 题目 3 迷宫求解最后打印输出时，使用‘\b’消除最后的“->”符号，但是在测试程序上不正确。

解决方法：将使用’\b‘换成使用一个循环。

四． 算法时空分析

1. 题目1

主要操作的时间复杂度：插入、删除、反转：都需要移动栈中光标后的元素，故为O(n)

1. 题目2

判断的时间规模与表达式中括号的数目成正比，故为O(n)

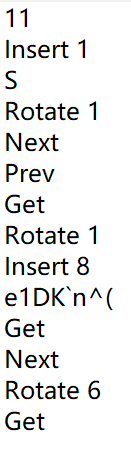
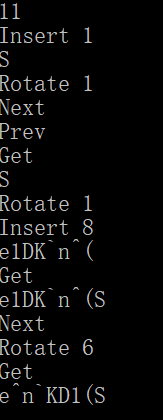
1. 题目3

采用广度优先遍历，为O（m\*n）m与n为迷宫的行数和列数

1. 题目4

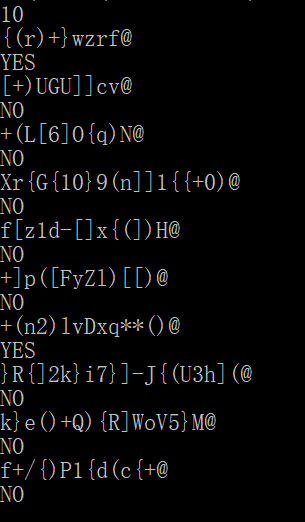
时间规模与客户的数量成正比，为O（n）

1. 测试结果及分析
2. 题目1



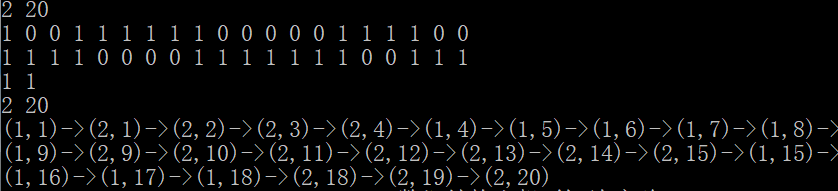
结果正确。

1. 题目2



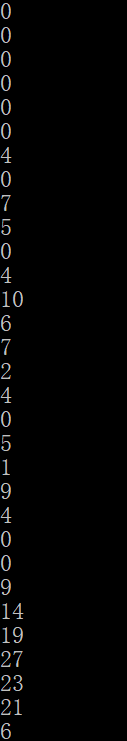
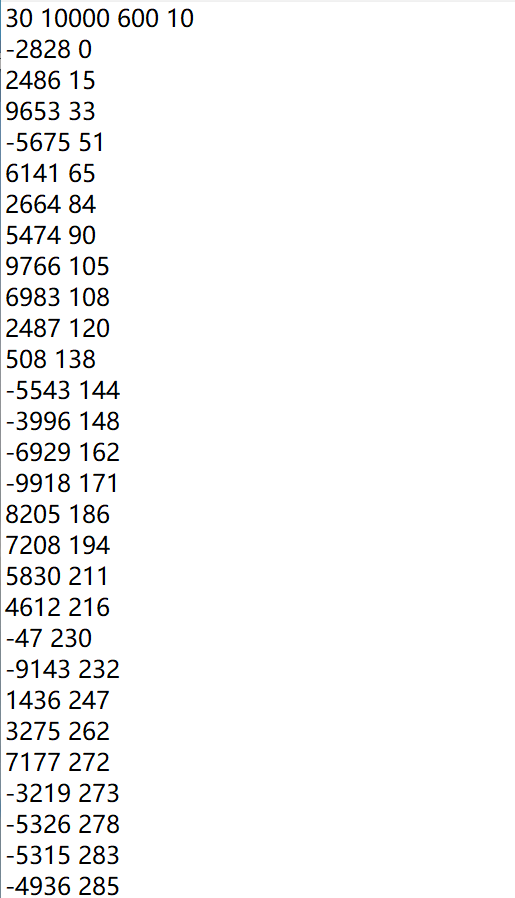
结果正确。

1. 题目3



结果正确。

1. 题目4



结果正确。

1. 实验体会和收获
2. 增进了对栈和队列的理解，而且我在本次实验中使用了class，增进了对抽象数据类型的认识。
3. 本次实验题目内容比较丰富，较好地锻炼了编程及调试能力。