**2020秋季学期**

**数据结构实验报告**

**实验二 栈和队列实验**

**班级：\_\_计实验19 \_**

**学号：\_19101130120\_**

**姓名：\_\_\_黄周杰\_\_\_**

|  |
| --- |
| **评语：**  **日期：** |

**1、**停车场

**需求分析：**（包括对问题的理解，解决问题的策略、方法描述）

在数据结构上，用栈来表示车辆的出入，来保存车辆的顺序信息，然后现在有两个操作，对应着不同的状态。操作一，车来：1.当停车场的车不到上限的时候，这辆车，进入栈一，并在此时记录他的进入时间。2.当停车场的车到达上限的时候，这辆车，进入队列。

操作二，车辆离开：1.当这辆车是在停车场的时候，这辆车出栈，并记录下时间，算车费，输出，然后将这辆车之后的车前挪，然后如果便道有车，则将便道中第一辆车进栈，然后将入栈时间记录下来。2.当这辆车在便道的时候只需要直接退出即可，将前面的往后最后放，并将其后面的车往前挪。

**系统设计：**（包括数据结构定义、抽象出基本操作描述、主程序模块处理过程描述）

**typedef** struct SqStack**{**

int **\***id\_b1**;** //这四项用于储存双向栈的头指针和尾指针

int **\***id\_t1**;** //这个栈用于id

int **\***time\_b**;** //这个栈用于储存时间

int **\***time\_t**;**

**}**SqStack**;**

栈

**typedef** struct QNode**{**

int id**;**

struct QNode **\***next**;**

**}**QNode**,** **\***QueuePtr**;**

**typedef** struct **{**

QueuePtr front**;**

QueuePtr rear**;**

**}**LinkQueue**;**

队列

**基本操作：**

void InitStack**(**SqStack **&**S**)**

操作结果：创建了一个空栈S。

int find**(**SqStack S**,**int id**)**

初始条件：栈S不为空

操作结果：返回车辆所在地方，返回0时，表示在停车场，返回1时，表示在便道。

void Push**(**int side**,**SqStack **&**S**,**int id**,**int time**)**

初始条件：栈S不为空

操作结果：插入id和time，分别作为两个栈的栈顶元素。

void InitQueue**(**LinkQueue **&**Q**)**

操作结果：创建了一个队列Q。

int Pop**(**int side**,**SqStack **&**S**,**int id**)**

初始条件：栈S存在

操作结果：将车号为id的车辆移除栈，并将其余车辆往前挪

void output**(**SqStack S**,**LinkQueue Q**,**int n**)**

初始条件：栈S,队列Q存在

操作结果：按顺序输出S和Q的车辆编号

void EnQueue**(**LinkQueue **&**Q**,** int id**)**

初始条件：队列Q存在

操作结果：将id放到Q的队尾

int DeQueue**(**LinkQueue **&**Q**,**int id**)**

初始条件：队列Q存在

操作结果：将车号为id的车辆移出队列，并挪动车辆。

void input**(**int n**,** float m**)**

操作结果：完成了将上诉所有操作的有序结合，完成了题设要求。

**基本操作的实现：**（对各基本操作实现的描述）

void InitStack**(**SqStack **&**S**)**

**{**

S**.**id\_b1 **=** **(**int **\*)**malloc**(**200 **\*** **sizeof(**int**));**

S**.**id\_t1 **=** S**.**id\_b1**;**

S**.**time\_b **=** **(**int **\*)**malloc**(**200 **\*** **sizeof(**int**));**

S**.**time\_t **=** S**.**time\_b**;**

**}**

InitStack

**描述：**初始化了栈

int find**(**SqStack S**,**int id**)**

**{**

int **\***p **=** S**.**id\_b1**;**

**while(**p **!=** S**.**id\_t1**)** //遍历查找对应的ID

**{**

**if(\***p **==** id**)**

**{**

**return** 0**;**

**}**

p**++;**

**}**

**return** 1**;**

**}**

find

**描述：**通过遍历栈，来判断这辆车是否在停车场，如果不在，则说明在便道。

Push

void Push**(**int side**,**SqStack **&**S**,**int id**,**int time**)**

**{**

**\***S**.**id\_t1 **=** id**;**

**\***S**.**time\_t **=** time**;**

S**.**id\_t1**++;**

S**.**time\_t**++;**

**}**

**描述：**将id和time信息放入栈S的顶部，并让两个指针向上移一步。

void InitQueue**(**LinkQueue **&**Q**){**

Q**.**front **=** **(**QNode**\*)**malloc**(sizeof(**QNode**));**

Q**.**rear **=** Q**.**front**;**

Q**.**front**->**next **=** 0**;**

Q**.**rear**->**next **=**0**;**

**}**

InitQueue

**描述：**创建一个新的队列。

void output**(**SqStack S**,**LinkQueue Q**,**int n**)**

**{**

int **\***p **=** S**.**id\_b1**;**

**while(**p**!=**S**.**id\_t1**)**

**{**

cout**<<\***p**<<**' '**;**

p**++;**

**}**

cout**<<**endl**;**

QueuePtr fp **=** Q**.**front**;**

fp **=** fp**->**next**;**

**while(**n**--)**

**{**

cout**<<**fp**->**id**<<**' '**;**

fp **=** fp**->**next**;**

**}**

cout**<<**endl**;**

**}**

Output

**描述：**遍历栈S，遍历队列Q，输出各自的车辆顺序。

Pop

**描述**：根据id找到相应的位置，然后将后一个元素，依此向前赋值，直到栈顶。

int Pop**(**int side**,**SqStack **&**S**,**int id**)** //根据ID来出栈

**{**

int **\***p**,\***time**,\***fp**,\***t**;**

int ans**;**

p **=** S**.**id\_b1**;**

t **=** S**.**time\_b**;**

**while(\***p **!=** id**)**//找到对应位置 **{**

p**++;**

t**++;**

**}**

ans **=** **\***t**;**

fp **=** p**;**//将车前挪

fp**++;**

**while(**fp **!=** S**.**id\_t1**)** **{**

**\***p **=** **\***fp**;**

fp**++;**

p**++;**

**}**

fp **=** t**;**

fp**++;**

**while(**fp **!=** S**.**time\_t**)** //时间前挪 **{**

**\***t **=** **\***fp**;**

fp**++;**

t**++;**

**}**

S**.**id\_t1**--;**

S**.**time\_t**--;**

**return** ans**；**

**}**

EnQueue

void EnQueue**(**LinkQueue **&**Q**,** int id**)**

**{**

QueuePtr p **=** **(**QueuePtr**)**malloc**(sizeof(**QNode**));**

p**->**id **=** id**;**

p**->**next **=** **NULL;**

Q**.**rear**->**next **=** p**;**

Q**.**rear **=** Q**.**rear**->**next**;**

**}**

**描述：**创建一个新的空间，并将其赋值，将其放在队列的末尾

int DeQueue**(**LinkQueue **&**Q**,**int id**)**

**{**

**if** **(**id **==** 0**)**

**{**

QueuePtr p **=** Q**.**front**->**next**;**

id **=** p**->**id**;**

Q**.**front**->**next **=** p**->**next**;**

free**(**p**);**

**return** id**;**

**}**

**else**

**{**

QueuePtr p **=** Q**.**front**->**next**;**

QueuePtr fp **=** Q**.**front**;**

**while(**p**->**id **!=** id**)**

**{**

p**=**p**->**next**;**

fp **=** fp**->**next**;**

**}**

Q**.**rear**->**next **=** Q**.**front**->**next**;**

Q**.**rear **=** fp**;**

fp**->**next **=** 0**;**

Q**.**front**->**next **=** p**->**next**;**

**}**

**}**

DeQueue

**描述：**假如ID为0，直接将队列前面的车辆出队，并且返回id。假如为其他，就找到对应位置，再将其出队，并将队列重置。

**{**

int move**,**id**,**time**;**

int count1 **=** 0**,**count2 **=** 0**;**//用于记录车的数量

int side**,**l\_time**;**

LinkQueue Q**;**

SqStack S**;**

InitQueue**(**Q**);**

InitStack**(**S**);**

cin**>>**move**>>**id**>>**time**;**

**while(!(**move **==** 0 **&&** id **==** 0 **&&** time **==** 0**))**

**{**

**if(**move **==** 0**)**

**{**

**if(**count1 **<** n**)**//当停车场还有位置

**{**

Push**(**0**,**S**,**id**,**time**);**

count1**++;**

**}**

**else**//当停车场没位置

**{**

EnQueue**(**Q**,**id**);**//往队列插入

count2**++;**

**}**

**}**

**else**

**{**

side **=** find**(**S**,**id**);**

**if(**side **==** 0**)**

**{**

l\_time **=** Pop**(**0**,**S**,**id**);**//返回时间

cout**<<**id**<<**' '**<<**setiosflags**(**ios**::**fixed**)<<**setprecision**(**2**)<<(**time **-** l\_time**)** **\*** m**<<**endl**;**

count1**--;**

**if(**count2**)**

**{**

id **=** DeQueue**(**Q**,**0**);**//提出队列的头

Push**(**0**,**S**,**id**,**time**);**

count2**--;**

count1**++;**

**}**

**}**

**else**

**{**

DeQueue**(**Q**,**id**);**//提出队列的位置

count2**--;**

**}**

**}**

cin**>>**move**>>**id**>>**time**;**

**}**

output**(**S**,**Q**,**count2**);**

**}**

input

count1**--;**

**if(**count2**)**

**{**

id **=** DeQueue**(**Q**,**0**);**//提出队列的头

Push**(**0**,**S**,**id**,**time**);**

count2**--;**

count1**++;**

**}**

**}**

**else**

**{**

DeQueue**(**Q**,**id**);**//提出队列的位置

count2**--;**

**}**

**}**

cin**>>**move**>>**id**>>**time**;**

**}**

output**(**S**,**Q**,**count2**);**

**}**

**描述：**根据需求分析写的步骤，将各个基础操作排列起来，达到算法目的。

**测试结果：**（输入的测试数据及运行结果、正确性、在线测试情况）

2  1.5

0  1  5

0  2  10

1  1  15

0  3  20

0  4  25

0  5  30

0  6  35

1  2  40

0  7  45

1  6  50

0  0  0

输入数据

运行结果

1  15.00

2  45.00

3  4

7  5

正确性：完全正确

在线测试情况：完全正确

完全正确

**调试分析：**（包括调试过程中对原设计的修改，以及遇到的问题和解决的方法）

遇到的问题：

无法正确的循环队列。

解决方法：

将队列中有多少个元素，传入输出函数中，固定循环次数。

2、C程序括号匹配

**需求分析：**（包括对问题的理解，解决问题的策略、方法描述）

这道题需要我们逐字符读取，根据不同的情况，进行处理。由于是要进行匹配，所以我们使用栈来进行处理这道题。当我们遇到 ( 和 { 的时候，我们将其入栈，并用另一个栈存，当第一个栈中没有元素的时候，记录下这个括号的位置，当我们遇到) 和 }的时候，我们将其和栈顶的字符进行比较，如果匹配，则出栈，不匹配则输出符号以及行数，该程序结束。如果遇到 / 继续读取，如果是 /，则直接跳过该行，如果是 \* ，继续读取，直到遇到\*，然后判断下一个是否是/，之后再正常读取。 文件到头时，则判断栈是否为空，为空则输出，不为空，则输出对应符号以及行数 |、。

**系统设计：**（包括数据结构定义、抽象出基本操作描述、主程序模块处理过程描述）

void InitStack**(**SqStack **&**S**)**

**{**

S**.**base **=** **(**char **\*)**malloc**(**200 **\*** **sizeof(**char**))**

S**.**top **=** S**.**base**;**

**}**

数据结构定义

基本操作：

void InitStack**(**SqStack **&**S**)**

操作结果：创建了一个空栈

void Push**(**SqStack **&**S**,**char e**)**

初始条件：栈S已经存在

操作结果：插入元素e为新的栈顶元素

void Pop**(**SqStack **&**S**)**

初始条件：栈S已经存在

操作结果：删除S的栈顶元素

int input**(**SqStack **&**S1**,**SqStack **&**S2**)**

初始条件：栈S1，S2已经存在

操作结果：输出答案

过程描述：

先是打开文件，然后再逐行读取文件，遍历每个字符，并记录下这一行的行数，根据每一个字符做出不同的操作，当为 （ ， ），{ ，}时，做出入栈操作，当遇到“，/，\*，记录下该信息，做出跳过，或是找对应的\*/的操作。读取完后，判断栈内是否还有元素，还有的话，就输出题设要求，如果没有就输出所有括号。

**基本操作的实现：**（对各基本操作实现的描述）

void InitStack**(**SqStack **&**S**)**

**{**

S**.**base **=** **(**char **\*)**malloc**(**200 **\*** **sizeof(**char**));**

S**.**top **=** S**.**base**;**

**}**

InitStack

**描述：**构造一个空栈S

void Push**(**SqStack **&**S**,**char e**){**

**\***S**.**top **=** e**;**

S**.**top**++;**

**}**

Push

**描述：**插入e为新的栈顶元素

void Pop**(**SqStack **&**S**){**

S**.**top**--;**

**}**

Pop

**描述：**删除S的栈顶元素

int input**(**SqStack **&**S1**,**SqStack **&**S2**)**

**{**

ifstream infile**;**

infile**.**open**(**"example.c"**);**//打开文件夹

string data**;**

int len**,**i**,**floor **=** 0**;** //len用于记录字符串长度，i用于记录字符串位置

char c**;** //floor用于记录当前行数,l用于记录line的空位置

int line**,**count **=** 0**;** //count 用于判断是否需要找对应字符

**while(!**infile**.**eof**())**

**{**

floor**++;**

getline**(**infile**,**data**);** //读取整行

len **=** data**.**length**();**

**for(**i **=** 0**;** i **<** len **;** i**++)**

**{**

**if(**count **==** 1**&&**data**[**i**]** **==** '\*' **&&** data**[**i**+**1**]** **==** '/'**)** //判断是否需要找\*/

**{**

count **=** 0**;**

i**++;**

**}**

**else** **if(**count **==** 2 **&&** data**[**i**]** **==** '"'**)**

**{** //判断是否需要找“

count **=** 0**;**

**continue;**

**}**

**else** **if(**count**)**

**{**

**continue;**

**}**

c **=** data**[**i**];**

**if(**c **==** '(' **||** c **==** '{'**)**

**{** //如果是（ 和 { 入栈

**if(**S1**.**top **==** S1**.**base**)**

**{**

line **=** floor**;**

**}**

Push**(**S1**,**c**);**

Push**(**S2**,**c**);**

**}**

**else** **if(**c **==** ')' **||** c **==** '}'**)**

**{**

**if(**c **==** ')'**)**

**{**

char **\***p **=** S1**.**top**;**

p**--;**

**if(\***p **==** '('**)**

**{**

Pop**(**S1**);**

Push**(**S2**,**c**);**

**continue;**

**}**

**}**

**else** **if(**c **==** '}'**)**

**{**

char **\***p **=** S1**.**top**;**

p**--;**

**if(\***p **==** '{'**)**

**{**

Pop**(**S1**);**

Push**(**S2**,**c**);**

**continue;**

**}**

**}**

cout**<<**"without maching '"**<<**c**<<**"' at line "**<<**floor**;**

**return** 0**;**

**}**

**else** **if(**c **==**'/'**)**

**{**

i**++;**

**if(**data**[**i**]** **==** '/'**)**

**{**

i **=** len**-**1**;**

**}**

**if(**data**[**i**]** **==** '\*'**)**

**{**

count **=** 1**;**

**}**

**}**

**else** **if(**c **==** '"'**)**

**{**

count **=** 2**;**

**}**

**}**

**}**

**if(**S1**.**top **!=** S1**.**base**)**

**{**

cout**<<**"without maching '"**<<\***S1**.**base**<<**"' at line "**<<**line**;**

**}**

**else**

**{**

char **\***p **=** S2**.**base**;**

**while(**p **!=** S2**.**top**)**

**{**

cout**<<\***p**;**

p**++;**

**}**

**}**

**}**

Push**(**S1**,**c**);**

Push**(**S2**,**c**);**

**}**

**else** **if(**c **==** ')' **||** c **==** '}'**)**

**{** //如果是） 和 } ，判断

**if(**c **==** ')'**)**

**{**

char **\***p **=** S1**.**top**;**

p**--;**

**if(\***p **==** '('**)**

**{**

Pop**(**S1**);**

Push**(**S2**,**c**);**

**continue;**

**}**

**}**

**else** **if(**c **==** '}'**)**

**{**

char **\***p **=** S1**.**top**;**

p**--;**

**if(\***p **==** '{'**)**

**{**

Pop**(**S1**);**

Push**(**S2**,**c**);**

**continue;**

**}**

**}**

//如果不满足则输出

cout**<<**"without maching '"**<<**c**<<**"' at line "**<<**floor**;**

**return** 0**;**

**}**

**else** **if(**c **==**'/'**)**

**{**

i**++;**

**if(**data**[**i**]** **==** '/'**)**

**{**

i **=** len**-**1**;**

**}**

**if(**data**[**i**]** **==** '\*'**)**

**{**

count **=** 1**;**

**}**

**}**

**else** **if(**c **==** '"'**)**

**{**

count **=** 2**;**

**}**

**}**

**}**

**if(**S1**.**top **!=** S1**.**base**)**

**{**

cout**<<**"without maching '"**<<\***S1**.**base**<<**"' at line "**<<**line**;**

**}**

**else**

**{**

char **\***p **=** S2**.**base**;**

**while(**p **!=** S2**.**top**)**

**{**

cout**<<\***p**;**

p**++;**

**}**

**}**

**}**

input

i **=** len**-**1**;**

**}**

**if(**data**[**i**]** **==** '\*'**)**

**{**

count **=** 1**;**

**}**

**}**

**else** **if(**c **==** '"'**)**

**{**

count **=** 2**;**

**}**

**}**

**}**

**if(**S1**.**top **!=** S1**.**base**)**

**{** /判断S1栈是否有元素

cout**<<**"without maching '"**<<\***S1**.**base**<<**"' at line "**<<**line**;**

**}**

**else**

**{**

char **\***p **=** S2**.**base**;**

**while(**p **!=** S2**.**top**)**

**{**

cout**<<\***p**;**

p**++;**

**}**

**}**

**}**

**描述：**输入S1,S2，并完成了题设要求

**测试结果：**

#include<stdio.h>

int main(){

printf("{ hello world }\n"); // }

)

输入数据

without maching ')' at line 4

结果

准确性：完全正确

在线测试：完全正确

**调试分析：**（包括调试过程中对原设计的修改，以及遇到的问题和解决的方法）

**遇到的问题：**

当输入以下数据时，无法读取到），原因在于没有处理”符号

#include<stdio.h>

int main(){

printf("{ hello world }\n"); // }

)

**解决方法：**

用一个变量去储存遇到这个符号的信息，然后根据这个信息，去决定找不着下一个“符号。