

**数据结构课程实验报告**

**（2）**

**姓名：王秋实**

**学号：2135070824**

**专业：计科**

**完成日期：2023/3/19**

**目录**

[目录 - 1 -](#_Toc434170429)

[1 设计要求 - 2 -](#_Toc434170430)

[2 程序功能框图 - 2 -](#_Toc434170431)

[3 数据结构说明 - 2 -](#_Toc434170432)

[4 重要算法核心代码 - 2 -](#_Toc434170433)

[5 测试运行界面 - 2 -](#_Toc434170434)

[6 完整源程序 - 3 -](#_Toc434170435)

**数据结构实验二（堆栈与队列）**

1. **堆栈应用(1)**

【问题描述】

输入任一表达式，“#”为表达式的结束符，试写一判断表达式中圆括号（“(”与“)”）是否配对的算法。用C语言实现。

【测试情况】

Input the expression string ended with '#' (length≤80):

(a+((a+b)\*c)-d/c)\*e#

Matched

Input the expression string ended with '#' (length≤80):

((a+((a+b)\*c)-d/c)\*e#

Unmatched

1. **堆栈应用(2)**

【问题描述】

算术表达式求值问题：约定表达式中只包含加减乘除4种算术运算（表达式中允许出现圆括号），所有运算对象均为简单变量，输入表达式的结束符为“#”，按回车键后即可得到运算结果。用C语言实现。

1. **队列应用**

【问题描述】

对顺序循环队列，将自然数按序入队、出队。具体的操作是：队列未满时，入队、入队、出队（即做连续两次入队操作之后，做一次出队操作），输出出队元素的值；队列满时，执行连续的出队操作，输出出队元素的值（应与队列未满时所输出的有不同标识），直至队列为空。编写算法实现以上操作。用C语言实现。

【测试情况】

假设顺序循环队列使用的数组大小为20

运行程序得到的实际输出如下：

1\*2\*3\*4\*5\*6\*7\*8\*9\*10\*11\*12\*13\*14\*15\*16\*17\*18\*19#20#21#22#23#24#25#26#27#28#29#30#31#32#33#34#35#36#37#

**2、程序功能框图**

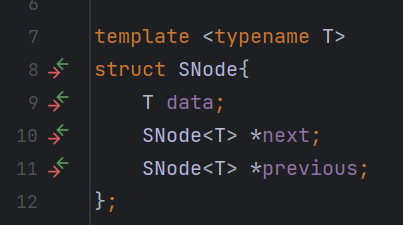
1、用双向循环链栈实现了圆括号的匹配

2、用两个堆栈：操作数栈和运算符栈实现了中缀表达式求值的问题

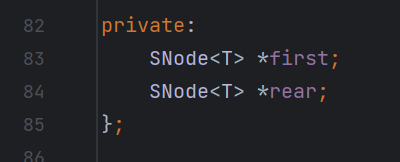
3、用顺序队列实现了规定操做

**3、数据结构说明**

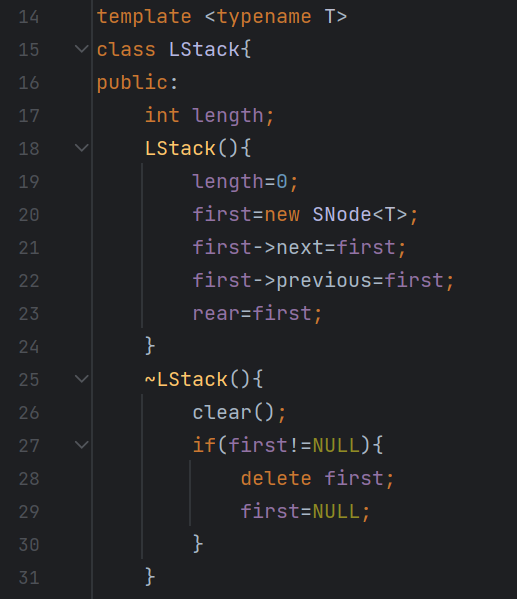
1、链栈



链表节点



私有变量：链栈头结点，链栈尾节点

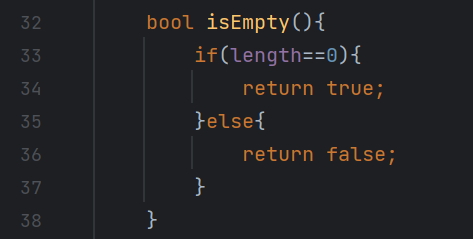


构造方法：

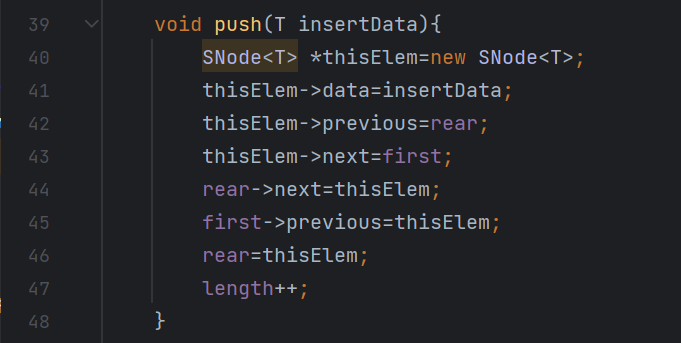
初始化各参量

析构方法：

先调用clear()删除所有节点，然后再删除头结点



判空操作

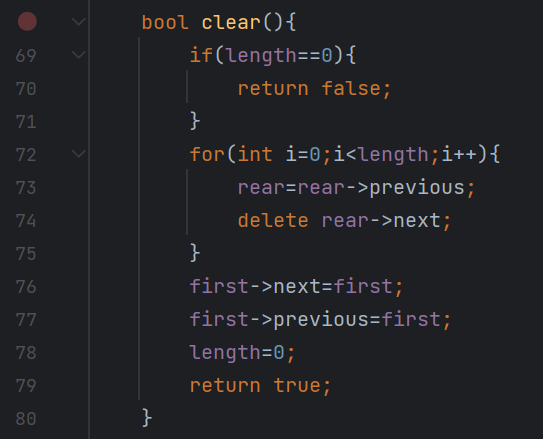


入栈操作，只需更改头尾节点指针即可



查看栈顶元素操作：只返回栈顶元素而不删除

出栈操作：删除并弹出栈顶元素



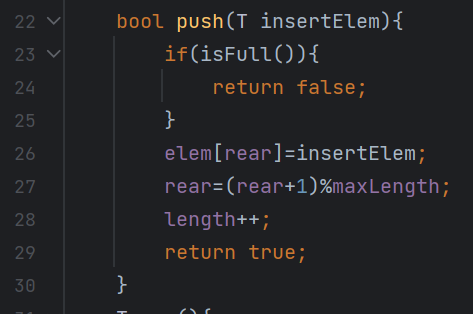
清空操作：

从栈尾到栈首依次删除元素

2、顺序队列



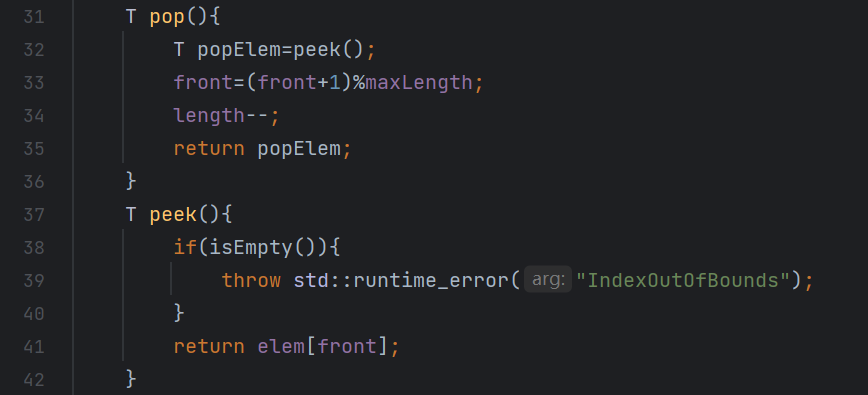
私有变量和构造方法

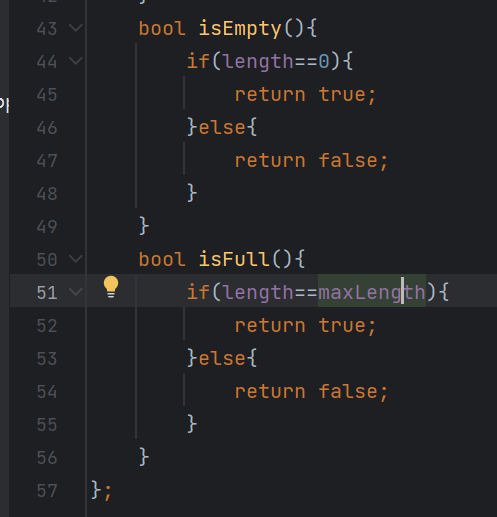


查看队首元素操作

出队操作

入队操作

****

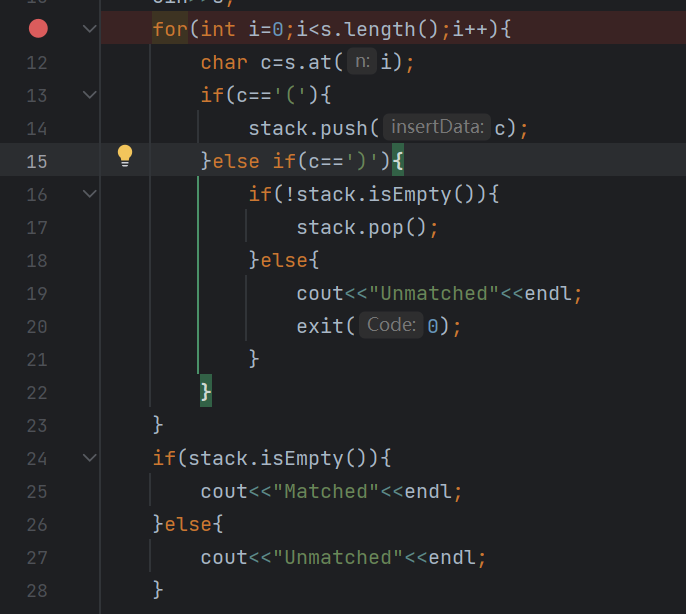
****

判空和判满操作

**4、重要算法核心代码**

1、括号匹配：

遇到左括号就入栈，遇到右括号就出栈。遇到右括号时栈空或读完所有字符后栈不空，说明不匹配，否则匹配

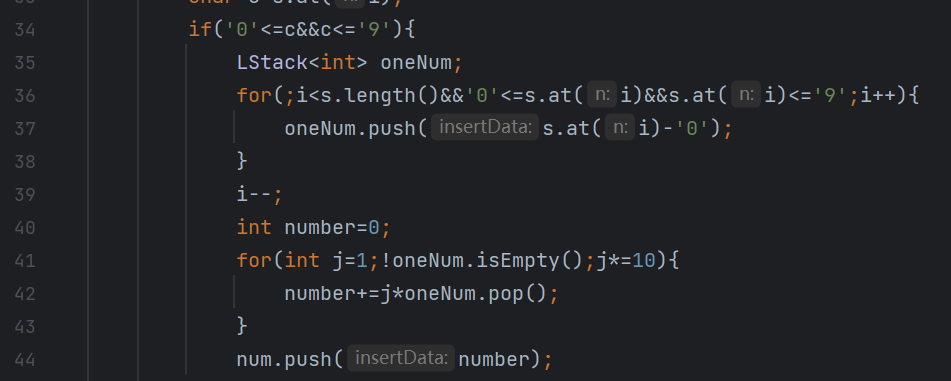


2、表达式求值：

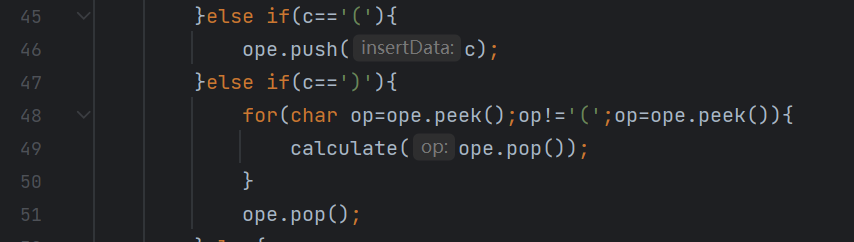
设置两个栈，操作数栈和运算符栈

遵循以下原则：

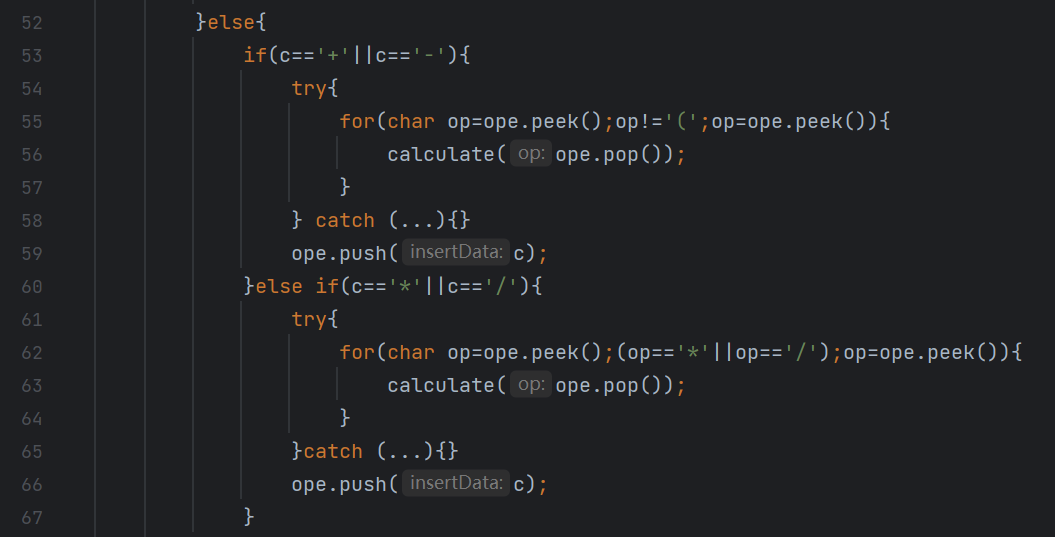
1. 遇到操作数，压入操作数栈中，此操作需要另一个栈读取多位操作数



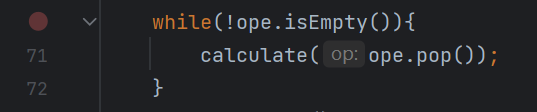
2、遇到左括号，直接压入运算符栈；遇到右括号，依次弹出运算符，被弹出的运算符立即生效，即弹出两个操作数计算出结果后便再次压入操作数栈，直至弹出左括号为止



3、遇到运算符，需依次弹出所有优先级大于或等于该运算符的所有运算符并立即生效，直至遇到左括号或者优先级低于该运算符的运算符。



4、最后弹出所有剩余在栈中的运算符并立即生效



1. 队列弹出

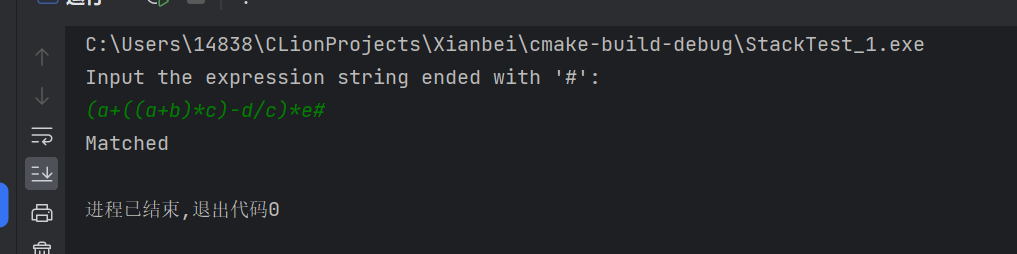
每做一次操作都检查一遍队列是否满，如果没满则执行下一个操作，如果已满

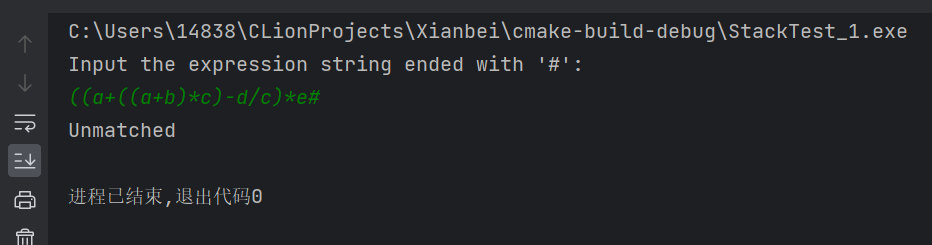
则弹出所有元素并停止



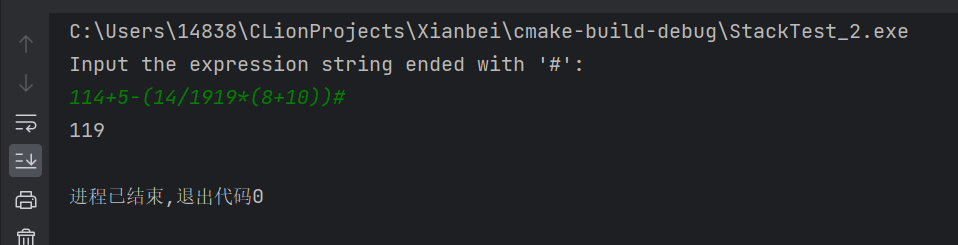
**5、测试运行界面**

1、括号匹配

****

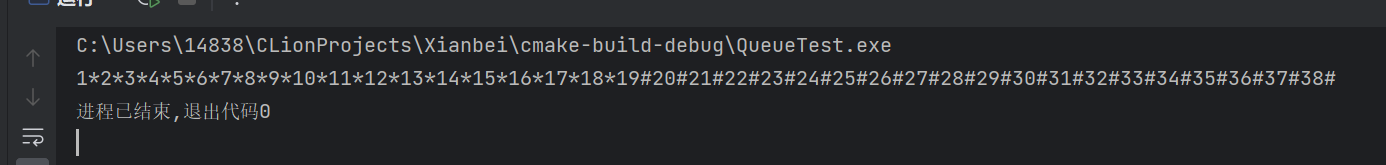


2、表达式求值



1. 队列操作

由于此队列单列一个length变量判空和判满，所以满队列时元素个数为20个，即没有浪费任何一个数组空间



**6、完整源程序**

**链栈头文件：**

#pragma once

#ifndef XIANBEI\_LSTACK\_H

#define XIANBEI\_LSTACK\_H

#include "iostream"

#include "exception"

template <typename T>

struct SNode{

T data;

SNode<T> \*next;

SNode<T> \*previous;

};

template <typename T>

class LStack{

public:

int length;

LStack(){

length=0;

first=new SNode<T>;

first->next=first;

first->previous=first;

rear=first;

}

~LStack(){

clear();

if(first!=NULL){

delete first;

first=NULL;

}

}

bool isEmpty(){

if(length==0){

return true;

}else{

return false;

}

}

void push(T insertData){

SNode<T> \*thisElem=new SNode<T>;

thisElem->data=insertData;

thisElem->previous=rear;

thisElem->next=first;

rear->next=thisElem;

first->previous=thisElem;

rear=thisElem;

length++;

}

T peek(){

if(length==0){

throw std::runtime\_error("IndexOutOfBounds");

}

return rear->data;

}

T pop(){

if(length==0){

throw std::runtime\_error("IndexOutOfBounds");

}

SNode<T> \*popElem=rear;

T popData=rear->data;

rear=popElem->previous;

rear->next=first;

first->previous=rear;

delete popElem;

length--;

return popData;

}

bool clear(){

if(length==0){

return false;

}

for(int i=0;i<length;i++){

rear=rear->previous;

delete rear->next;

}

first->next=first;

first->previous=first;

length=0;

return true;

}

private:

SNode<T> \*first;

SNode<T> \*rear;

};

#endif //XIANBEI\_LSTACK\_H

**顺序队列头文件：**

#ifndef XIANBEI\_AQUEUE\_H

#define XIANBEI\_AQUEUE\_H

#include "iostream"

#include "exception"

template <typename T>

class AQueue{ //都什么年代了，还用传统顺序队列

private:

const static int maxLength=20;

T elem[maxLength];

int front;

int rear;

public:

int length;

AQueue(){

front=0;

rear=0;

length=0;

}

bool push(T insertElem){

if(isFull()){

return false;

}

elem[rear]=insertElem;

rear=(rear+1)%maxLength;

length++;

return true;

}

T pop(){

T popElem=peek();

front=(front+1)%maxLength;

length--;

return popElem;

}

T peek(){

if(isEmpty()){

throw std::runtime\_error("IndexOutOfBounds");

}

return elem[front];

}

bool isEmpty(){

if(length==0){

return true;

}else{

return false;

}

}

bool isFull(){

if(length==maxLength){

return true;

}else{

return false;

}

}

};

#endif //XIANBEI\_AQUEUE\_H

**括号匹配源程序：**

#include "iostream"

#include "string"

#include "LStack.h"

using namespace std;

int main(){

LStack<char> stack;

cout<<"Input the expression string ended with '#':"<<endl;

string s;

cin>>s;

for(int i=0;i<s.length();i++){

char c=s.at(i);

if(c=='('){

stack.push(c);

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

if(!stack.isEmpty()){

stack.pop();

}else{

cout<<"Unmatched"<<endl;

exit(0);

}

}

}

if(stack.isEmpty()){

cout<<"Matched"<<endl;

}else{

cout<<"Unmatched"<<endl;

}

}

**表达式求值源程序：**

#include "iostream"

#include "string"

#include "LStack.h"

using namespace std;

LStack<int> num;

LStack<char> ope;

void calculate(char op){

int b=num.pop();

int a=num.pop();

switch (op) {

case '+':

num.push(a+b);

break;

case '-':

num.push(a-b);

break;

case '\*':

num.push(a\*b);

break;

case '/':

num.push(a/b);

break;

}

}

int main(){

cout<<"Input the expression string ended with '#':"<<endl;

string s;

cin>>s;

for(int i=0;i<s.length();i++){

char c=s.at(i);

if('0'<=c&&c<='9'){

LStack<int> oneNum;

for(;i<s.length()&&'0'<=s.at(i)&&s.at(i)<='9';i++){

oneNum.push(s.at(i)-'0');

}

i--;

int number=0;

for(int j=1;!oneNum.isEmpty();j\*=10){

number+=j\*oneNum.pop();

}

num.push(number);

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

ope.push(c);

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

for(char op=ope.peek();op!='(';op=ope.peek()){

calculate(ope.pop());

}

ope.pop();

}else{

if(c=='+'||c=='-'){

try{

for(char op=ope.peek();op!='(';op=ope.peek()){

calculate(ope.pop());

}

} catch (...){}

ope.push(c);

}else if(c=='\*'||c=='/'){

try{

for(char op=ope.peek();(op=='\*'||op=='/');op=ope.peek()){

calculate(ope.pop());

}

}catch (...){}

ope.push(c);

}

}

}

while(!ope.isEmpty()){

calculate(ope.pop());

}

cout<<num.pop()<<endl;

}

**队列操作源程序：**

#include "iostream"

#include "AQueue.h"

using namespace std;

int main(){

AQueue<int> queue;

for(int i=1;;i++){

queue.push(i);

if(queue.isFull()){

while(!queue.isEmpty()){

cout<<queue.pop()<<"#";

}

break;

}else{

queue.push(++i);

}

if(queue.isFull()){

while(!queue.isEmpty()){

cout<<queue.pop()<<"#";

}

break;

}else{

cout<<queue.pop()<<"\*";

}

}

}