栈的模板

1. 用栈来实现中缀表达式转化为后缀表达式

思路：

创建栈

从左向右顺序获取中缀表达式

a.数字直接输出

b.运算符

情况一：遇到左括号直接入栈，遇到右括号将栈中左括号之后入栈的运算符全部弹栈输出，同时左括号出栈但是不输出。

情况二：遇到乘号和除号直接入栈，直到遇到优先级比它更低的运算符，依次弹栈。

情况三：遇到加号和减号，如果此时栈空，则直接入栈，否则，将栈中优先级高的运算符依次弹栈（注意：加号和减号属于同一个优先级，所以也依次弹栈）直到栈空或则遇到左括号为止，停止弹栈。（因为左括号要匹配右括号时才弹出）。

情况四：获取完后，将栈中剩余的运算符号依次弹栈输出

#include<iostream>

#include"max.h"

#include<cstring>

#include"math.h"

#include"vector"

#include"map"

#include <unordered\_set>

#include <algorithm>

#include <cstdlib>

#include <sstream>

using namespace std;

//下面的main函数请勿修改

#include <stdio.h>

#include <stdlib.h>

// 自己定义需要的栈结构，及栈基本操作函数，假设操作数都是整数

struct stack

{

   int top;

   int tops;

   int \*element;

   char \*elements;

   int maxsize;

};

/\*在此定义并完成第一关函数，参考main要求\*/

typedef struct stack\*  Stack ;

struct stack\* creat(){

   struct stack\* L=(struct stack\*)malloc(sizeof(struct stack));

L->top=0;

L->element=(int \*)malloc(sizeof(int)\*100);

L->elements=(char\*)malloc(sizeof(char)\*100);

L->maxsize=100;

return L;

}

int empty(struct stack\*L){

   if(L->top==0){

      return 1;

   }

   return 0;

}

int empty\_ch(struct stack\*L){

   if(L->tops==0){

      return 1;

   }

   return 0;

}

void push\_int(int val,struct stack\*L){

    if(L->top>=L->maxsize) return;

    L->element[L->top++]=val;

    return;

}

void push\_ch(char val,struct stack\*L){

    if(L->tops>=L->maxsize) return;

    L->elements[L->tops++]=val;

    return;

}

char pop\_char(struct stack\*L){

   char ans;

   if(L->tops-1<0){

      return ' ';

   }

   ans=L->elements[--L->tops];

   return ans;;

}

char front(Stack L){

   char ans;

   ans=L->elements[L->tops-1];

   return ans;

}

int get\_xue(char ch){

   int ans=0;

   switch(ch){

  case '\*': ans=2;

           break;

   case '/': ans=2;

           break;

   case '+': ans=1;

           break;

   case '-': ans=1;

           break;

    default: ans=0;

    break;

   }

   return ans;

}

Stack  inToPost(char \*expression)

{

   //在此处填写代码，完成中缀表达式转换为后缀表达式并输出

   /\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/

   Stack L1=creat();

   Stack L2=creat();

   for(int i=0;expression[i]!='\0';i++){

      if(expression[i]=='('){

         push\_ch(expression[i],L1);

      }

      if(expression[i]=='/'||expression[i]=='\*'||expression[i]=='+'||expression[i]=='-'||expression[i]==')'){

          push\_ch(' ',L2);

       if(empty\_ch(L1)||front( L1)=='('){

          cout << "op"<<endl;

          push\_ch(expression[i],L1);

       }

      else if(get\_xue( expression[i])>get\_xue(front( L1))){

          push\_ch(expression[i],L1);

       }

       else if(expression[i]==')'){

           while(front(L1)!='('){

              char p=pop\_char(L1);

              push\_ch(p,L2);

           }

           pop\_char(L1);

        }

      else  if(get\_xue( expression[i])<=get\_xue( front( L1))){

       while(get\_xue( expression[i])<=get\_xue( front(L1) ))  {

               cout << "ok"<<endl;

               char p=pop\_char(L1);

              push\_ch(p,L2);

               push\_ch(' ',L2);

               if(empty\_ch(L1)){

                  break;

               }

           }

           push\_ch(expression[i],L1);

       }

      }

      else if( expression[i]<='9'&&expression[i]>='0'){

          push\_ch(expression[i],L2);

      }

   }

   if(!empty\_ch(L1)){

      while(!empty\_ch(L1)){

         char p=pop\_char(L1);

         push\_ch(' ',L2);

         push\_ch(p,L2);

      }

   }

   return L2;

   /\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/

}

//print函数用于输出后缀表达式，参数是 inToPost的返回值

void print(Stack s)

{

 for(int i=0;i<s->tops;i++){

   cout << s->elements[i];

 }

}

int main(void){

  char express[80];

    gets(express);

   // printf("%s",express);

   Stack s = inToPost(express);

    print(s);

}

int pop(struct stack\*L){//出数栈

   int ans;

   if(L->top-1<0){

      return 0;

   }

   ans=L->element[--L->top];

   return ans;;

}

c语言后缀表达式求值

int calExp(char \*express)

{

   //在此处填写代码，完成表达式求值并输出

   /\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/

    Stack L=inToPost(express);

    print(L);

   Stack Lp=creat();

   for(int i=0;i<L->tops;i++){

      if(L->elements[i]>='0'&&L->elements[i]<='9'){

         int sum=0;

         while(L->elements[i]>='0'&&L->elements[i]<='9'&&i<L->tops){

           sum=sum\*10+(L->elements[i]-'0');

           i++;

         }

          cout <<"sum:"<<sum<<endl;

         push\_int(sum,Lp);

         i--;

      }

      else{

         if(L->elements[i]!=' '){

         int a1=pop(Lp);

         cout <<a1<<endl;

         int a2=pop(Lp);

         cout <<a2<<endl;

         int p=0;

         switch(L->elements[i]){

            case '+':p=a2+a1;

            break;

             case '-':p=a2-a1;

            break;

             case '\*':p=a2\*a1;

            break;

             case '/':p=a2/a1;

            break;

            default : break;

         }

         push\_int(p,Lp);

         }

      }

   }

   return  pop(Lp);

   /\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/

}