/\*\*

\* z2h.c 中缀表达式转后缀表达式

\*

\* @author Darbuly 970073804@qq.com

\* @copylight 2018-2019 DBL

\*/

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#define STACK\_INIT\_SIZE 20

#define STACKINCREMENT 10

#define OK 1

#define ERROR 0

#define SOVERFLOW -1

#define MAXBUFFER 10

typedef char ElemType;

typedef int Status;

typedef struct

{

ElemType \*base;

ElemType \*top;

int stacksize;

}sqStack;

Status InitStack(sqStack \*s)

{

s->base = (ElemType \*)malloc(STACK\_INIT\_SIZE\*sizeof(ElemType));

if(!s->base)return ERROR;

s->top = s->base;

s->stacksize = STACK\_INIT\_SIZE;

return OK;

}

Status Push(sqStack \*s,ElemType e)

{

if(s->top - s->base >=s->stacksize)

{

s->base = (ElemType \*)realloc(s->base,(s->stacksize+STACKINCREMENT)\*sizeof(ElemType));

if(!s->base)return SOVERFLOW;

s->top = s->base+s->stacksize;

s->stacksize +=STACKINCREMENT;

}

\*(s->top++)=e;

return OK;

}

Status Pop(sqStack \*s,ElemType \*e)

{

if(s->top ==s->base)return ERROR;

\*e = \*(--s->top);

return OK;

}

int StackLen(sqStack s)

{

return (s.top - s.base);

}

int main()

{

sqStack s;

char c,str[MAXBUFFER];

int i=0,len;

ElemType d;

InitStack(&s);

printf("Please enter your expression and end by #:\n");

scanf("%c",&c);

while(c!='#')

{

while(isdigit(c)|| c=='.')

{

str[i++]=c;

str[i]='\0';

if(i>=MAXBUFFER)

{

printf("\nThe number you entered is overflow!\n");

return -1;

}

scanf("%c",&c);

if(!isdigit(c))

{

printf("%s",str);

i=0;

break;

}

}

switch(c)

{

case '(':

Push(&s,c);

break;

case ')':

Pop(&s,&d);

printf("%c",d);

while(d!='(')

{

Pop(&s,&d);

if(d=='(')break;

printf("%c",d);

}

break;

case '+':

case '-':

if(\*(s.top-1)=='\*'||\*(s.top-1)=='/')

{

Pop(&s,&d);

printf("%c",d);

if(\*(s.top-1)=='+'||\*(s.top-1)=='-')

{

Pop(&s,&d);

printf("%c",d);

}

}

Push(&s,c);

break;

case '\*':

case '/':

Push(&s,c);

break;

}

if(c=='#')break;

scanf("%c",&c);

}

len = StackLen(s);

// printf("\nThe len is :%d\n",len);

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

{

Pop(&s,&d);

printf("%c",d);

}

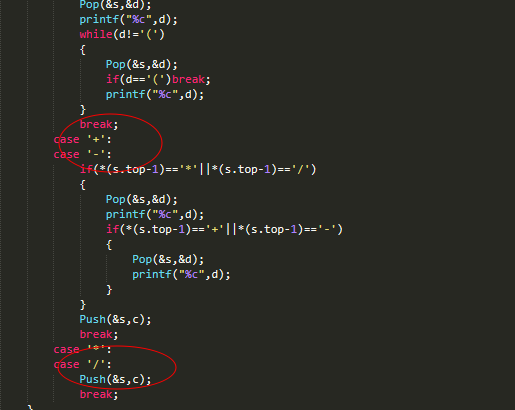
printf("\n");

return 0;

}

算法复杂度：O(n)

1. 总结：
   1. 同类合并，



1. 引发问题：编程者如何解决这里的#遗漏问题？

