

## Experiment topic: Application of stack - arithmetic expression evaluation

Experiment purpose :

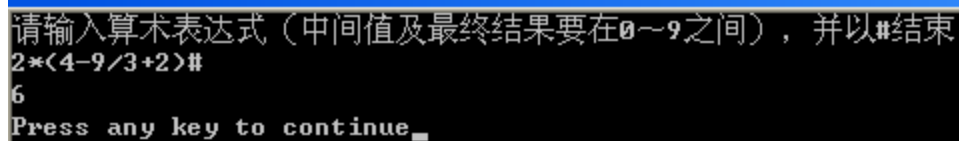
1. Master the definition and implementation of the stack;
2. Master the method of using the stack to solve arithmetic expressions.

Experiment content:

By modifying and improving Algorithm 3.4 in the textbook, the algorithm for evaluating arithmetic expressions is realized by using the stack. The implementation process of several functions called in Algorithm 3.4 is given:

- (1) Function In(c): determine whether c is an operator;
- (2) Function Precede(t1,t2): determine the priority of operators t1 and t2;
- (3) Function Operate(a,theta,b): Perform binary operation theta on a and b.

When the program is running, input a legal arithmetic expression (the intermediate value and the final result should be between 0 and 9, including addition, subtraction, multiplication, division and parentheses), and the corresponding calculation result can be output. As shown below:



```
请输入算术表达式 (中间值及最终结果要在0~9之间), 并以#结束
2*(4-9/3+2)#
6
Press any key to continue_
```

### CODE CalculateAPP.cpp

```
#include "StdAfx.h"
#include "stack.h"
```

```
Status In(SElemType c)
{ // Determine whether c is an operator
switch(c)
{
case '+':
case '-':
case '*':
case '/':
case '(':
case ')':
case '#':
return OK;
break;
default:
return ERROR;
}
}
```

```
SElemType Precede(SElemType t1, SElemType t2)
{ //According to Table 3.1 of the teaching material, judge the priority relationship of the two operators
SElemType f;
switch(t2)
{
case '+':
case '-':
```

```

        if(t1=='|' || t1=='#')
f='<';
else
f='>';
break;
    case '*':
case '/':
        if(t1=='*' || t1=='/' || t1=='')
            f='>';
        else
            f='<';
        break;
    case '(':
        if(t1=='')
            return ERROR;
        else
            f='<';
        break;
    case ')':
        if(t1=='(')
            f='=';
        else if(t1=='#')
            return ERROR;
        else
            f='>';
        break;
    case '#':
        if(t1=='#')
            f='=';
        else if(t1=='(')
            return ERROR;
        else
            f='>';
        break;
    }
return f;
}

```

```

SElemType Operate(SElemType a, SElemType theta, SElemType b)
{
    SElemType c;
    a=a-48;
    b=b-48;
    switch(theta)
    {
    case '+':
        c=a+b+48;
    break;
    case '-':
        c=a-b+48;
        break;
    case '*':
        c=a*b+48;
        break;
    case '/':
        c=a/b+48;
        break;
    }
}

```

```

return c;
}

char EvaluteExpression()
{ //Operator finite algorithm for arithmetic expression evaluation, let OPTR and OPND be operator stack and
operator stack respectively
SqStack OPND, OPTR;
    InitStack(OPND); //Initialize the OPND stack
    InitStack(OPTR); //Initialize the OPTR stack
    Push(OPTR,'#'); //Push the expression start character "#" into the OPTR stack
    char ch;
    scanf("%c",&ch);
    while(ch!='#' || GetTop(OPTR)!='#') //The expression has not been scanned or the top element of OPTR is
not "#"
    {
        if(!In(ch))
        {
            Push(OPND,ch); //ch is not an operator, enter OPND
            scanf("%c",&ch);
        }
        else
        {
            switch(Precede(GetTop(OPTR),ch)) //Comparing the top element of OPTR with the priority of ch
            {
                case '<':
                    Push(OPTR,ch); //The current character ch is pushed into the OPTR stack, and the next
character is read
                    scanf("%c",&ch);
                    break;
                case '>':
                    char a,b,theta;
                    Pop(OPTR,theta); //Pop the operator on the top of the OPTR stack
                    Pop(OPND,b); //Pop the two operands at the top of the OPND stack
                    Pop(OPND,a);
                    Push(OPND,Operate(a,theta,b)); //Press the operation result into the OPND stack
                    break;
                case '=': //OPTR's stack top element is "(" and ch is ")"
                    char x;
                    Pop(OPTR,x); //Pop up the "(" at the top of the OPTR stack, and read the next character ch
                    scanf("%c",&ch);
                    break;
            } //switch
        } //while
        return GetTop(OPND); //OPND stack top element is the expression evaluation result
    }
}

int main()
{
    char c;
    printf("Please enter the arithmetic expression and end with #.\n");
    c = EvaluteExpression();
    printf("%c\n",c);
    return 0;
}

```

## Stack.cpp

```

#include "StdAfx.h"
#include "stack.h"

```

```

Status InitStack(SqStack &S)
{ // Construct an empty stack
    S.base=new SElemType[MAXSIZE]; //Allocate an array space with a maximum capacity of MAXSIZE for the
sequence stack
    if(!S.base)
        exit(OVERFLOW); //Storage allocation failed
    S.top=S.base; //top is initially base, empty stack
    S.stacksize=MAXSIZE; //stacksize is set to the maximum capacity of the stack MAXSIZE
    return OK;
}

Status Push(SqStack &S,SElemType e)
{ // Insert element e as the new top element of the stack
    if(S.top-S.base==S.stacksize)
        return ERROR; //The stack is full
    *S.top++=e; //Element e is pushed onto the top of the stack, and the top pointer of the stack is added
    return OK;
}

Status Pop(SqStack &S, SElemType &e)
{ //Delete the top element of S, return its value with e
    if(S.top==S.base)
        return ERROR; //Stack is empty
    e=*--S.top; //The top pointer of the stack is decremented by 1, and the top element of the stack is assigned
to e
    return OK;
}

SElemType GetTop(SqStack S)
{ //Return the top element of S, without modifying the top pointer
    if(S.top!=S.base) //Stack is not empty
        return *(S.top-1); //Return the value of the top element of the stack, the top pointer of the stack
remains unchanged
}

```

## StdAfx.cpp

```

// stdafx.cpp : source file that includes just the standard includes
//    caculator.pch will be the pre-compiled header
//    stdafx.obj will contain the pre-compiled type information

```

```

#include "StdAfx.h"

```

```

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file

```

## StdAfx.h

```

// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

```

```

#if !defined(AFX_STDAFX_H__FE446F83_4E49_471D_A0A8_9A52C21A1196__INCLUDED_)
#define AFX_STDAFX_H__FE446F83_4E49_471D_A0A8_9A52C21A1196__INCLUDED_

```

```

#if _MSC_VER > 1000
#pragma once

```

```

#endif // _MSC_VER > 1000
#define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
#include <stdio.h>
#include <stdlib.h>
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define MAXSIZE 100
typedef int Status;
typedef char SElemType;

// TODO: reference additional headers your program requires here
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.
#endif // !defined(AFX_STDAFX_H__FE446F83_4E49_471D_A0A8_9A52C21A1196__INCLUDED_)

```

## Stack.h

```

// chain structure of sequence table
typedef struct
{
    SElemType *base;
    SElemType *top;
    int stacksize;
}SqStack;

Status InitStack(SqStack &S); //Stack initialization
Status Push(SqStack &S,SElemType e); //Push into the stack
Status Pop(SqStack &S,SElemType &e); // pop the stack
SElemType GetTop(SqStack S); //Get the top element of the stack
char EvaluateExpression(); //Operator finite algorithm for arithmetic expression evaluation
Status In(SElemType c); //Determine whether the read character is an operator function
SElemType Precede(SElemType t1,SElemType t2); //Function to determine the limited relationship between the
top element of the operator stack and the read operator
SElemType Operate(SElemType a, SElemType theta, SElemType b);
// perform binary operations

```

Test Results:



```

C:\Users\THCMAZJ\Desktop\1实验\实验2 ...
请输入算术表达式, 并以#结束.
2*(4-9/3+2) #
6
Press any key to continue

```



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\THCMAZJ\Desktop\1实验\实验2 ...". The window has a black background with white text. The text inside the window is as follows:

```
请输入算术表达式, 并以#结束.  
3*4-(3+2/1)#  
7  
Press any key to continue
```

The text is displayed in a monospaced font. The prompt "请输入算术表达式, 并以#结束." is followed by the input "3\*4-(3+2/1)#". The result "7" is displayed on the next line. The prompt "Press any key to continue" is displayed on the third line. The window has a standard Windows title bar with a minimize button, a maximize button, and a close button.