



北京師範大學
BEIJING NORMAL UNIVERSITY

一、实验要求

1. 上机之前应做好充分准备, 认真思考所需的上机题目, 提高上机效率。
2. 独立上机输入和调试自己所编的程序, 切忌抄袭、拷贝他人程序。
3. 上机结束后, 整理出实验报告。书写报告时, 重点放在实验的方法、思路以及总结反思上, 以达到巩固课堂学习、提高动手能力的目的。

二、实验内容

中缀表达式求值问题

三、实验步骤 (写出问题分析或者算法思路)

建立两个栈, 一个操作数栈 OPND 存放数字和运算结果, 一个运算符栈 OPTR 存放运算符, 通过比较 isp 和 icp 来分类进行操作, 在下一个字符为数字时, 通过向后延续判断, 再配合 while 结构可以达到任意位数数字压栈:
定义以下函数将功能分类:

```
void initDataStack(DataStack& D);  
  
void initOperatorStack(OperatorStack& O);  
  
//栈的初始化
```

```
bool DataPush(DataStack& D, double x);

bool DataPop(DataStack& D, double& x);

//OPND 的退栈进栈

bool OperatorPush(OperatorStack& O, char ch);

bool OperatorPop(OperatorStack& O, char& ch);

//OPTR 的进栈退栈

int isp(char op);

int icp(char op);

double calculate(char op, double x1, double x2);

//计算函数
```

四、程序清单（源程序代码等）

```
#include <stdio.h>

#include <stdlib.h>

#define stkSize 30

typedef struct
{
    double elem[stkSize];
    int top;
}DataStack;

typedef struct
{
    char elem[stkSize];
    int top;
}OperatorStack;
```

```
void initDataStack(DataStack& D);
void initOperatorStack(OperatorStack& O);
bool DataPush(DataStack& D, double x);
bool DataPop(DataStack& D, double& x);
bool OperatorPush(OperatorStack& O, char ch);
bool OperatorPop(OperatorStack& O, char& ch);
int isp(char op);
int icp(char op);
double calculate(char op, double x1, double x2);

int main()
{
    char expression[stkSize];
    double x1, x2, result, finalresult;
    char ch, ope, ch1;
    int i=0;
    DataStack OPND;
    OperatorStack OPTR;
    initDataStack(OPND);
    initOperatorStack(OPTR);
    printf("请输入表达式（任意位数数字），一定以#结束：");
    gets(expression);
    OperatorPush(OPTR, '#');
    ch=expression[i];
    while(ch!='#' || OPTR.elem[OPTR.top]!='#')
    {
        if(ch>='0' && ch<='9')
        {
            if(expression[i+1]>='0' && expression[i+1]<='9')
            {
```

```
double
newnum=(ch-'0')*10+(expression[i+1]-'0');
int nextposi=i+2;

while(expression[nextposi]>='0'&&expression[nextposi]<='9')
{

newnum=newnum*10+(expression[nextposi]-'0');
    nextposi++;
}
DataPush(OPND, newnum);
i=nextposi;
ch=expression[i];
}
else
{DataPush(OPND, 1.0*(ch-'0'));
ch=expression[++i];
}
}
else
{ ch1=OPTR.elem[OPTR.top];
if(icp(ch)>isp(ch1))
{
OperatorPush(OPTR, ch);
ch=expression[++i];
}
else if(icp(ch)<isp(ch1))
{
DataPop(OPND, x2);
```

```
        DataPop(OPND, x1);
        OperatorPop(OPTR, ope);
        result=calculate(ope, x1, x2);
        DataPush(OPND, result);

    }
    else if(icp(ch)==isp(ch1)&&ch==' ')
    {
        OperatorPop(OPTR, ope);
        ch=expression[++i];
    }

}

}

finalresult=OPND.elem[OPND.top];
printf("\n 结果是: %.2lf", finalresult);
return 0;
}

void initDataStack(DataStack& D)
{
    D.top=-1;
}

void initOperatorStack(OperatorStack& O)
{
    O.top=-1;
}

bool DataPush(DataStack& D, double x)
```

```
{
    D.elem[++D.top]=x;
    return true;
}

bool DataPop(DataStack& D, double& x)
{
    if(D.top==-1)
    {
        printf("栈为空, 无法退栈\n");
        return false;
    }
    x=D.elem[D.top--];
    return true;
}

bool OperatorPush(OperatorStack& O, char ch)
{
    O.elem[++O.top]=ch;
    return true;
}

bool OperatorPop(OperatorStack& O, char& ch)
{
    if(O.top==-1)
    {
        printf("栈为空, 无法退栈\n");
        return false;
    }
    ch=O.elem[O.top--];
    return true;
}
```


```
int isp(char op)
{
    switch(op)
    {
        case '#': return 0;break;
        case '(': return 1;break;
        case '*': return 5;break;
        case '/': return 5;break;
        case '+': return 3;break;
        case '-': return 3;break;
        case ')': return 6;break;
    }
}
```

```
int icp(char op)
{
    switch(op)
    {
        case '#': return 0;break;
        case '(': return 6;break;
        case '*': return 4;break;
        case '/': return 4;break;
        case '+': return 2;break;
        case '-': return 2;break;
        case ')': return 1;break;
    }
}
```

```
double calculate(char op,double x1,double x2)
```

```
{  
    switch(op)  
    {  
        case '+': return x1+x2;break;  
        case '-': return x1-x2;break;  
        case '*': return x1*x2;break;  
        case '/': return x1/x2;break;  
    }  
}
```

五、运行结果（程序运行时的结果说明或运行截图等）

 C:\Users\xx\Desktop\homework\main.exe

请输入表达式（任意位数数字），一定以#结束：10000+5*(15-3)#

结果是:10060.00

Process returned 0 (0x0) execution time : 24.940 s

Press any key to continue.

六、总结（实验中遇到的问题、取得的经验、感想等）

这个问题最难的是如何设计出不同运算符的栈内优先级和栈外优先级，但是这个我是借鉴书上已经设计好的，再者是如何去掉 10 以内数字的限制，一开始觉得很麻烦，但是在把 10 以内数字的情况完成后，发现只需要修改一点点就能够达到要求了。