哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称：数据结构与算法分析

课程类型：限选课程

实验项目名称：线性结构及其应用

实验题目： 中缀表达式向后缀表达式的转化

班级：1303101

学号：1130310128

姓名：杨尚斌

设计成绩	报告成绩	指导老师
		张岩

一、 实验目的

表达式求值是实现程序设计语言的基本问题之一，也是栈的应用的一个典型例子。设计一个程序，演示用算符优先发对算术表达式求值的过程。

（1）. 从键盘输入一个语法正确的中缀表达式，显示并保存该表达式

（2）. 利用栈结构，把上述（中缀）表达式转化为后缀表达式，并显示栈的状态变化过程和所得到的后缀表达式。

（3）. 利用栈结构，对上述后缀表达式进行求值，并显示栈的状态变化过程和最终结果。

二、 实验要求及实验环境

操作系统：Windows7 64bit sp1

编译环境：CodeBlocks

Mingw32-g++

三、设计思想（本程序中的用到的所有数据类型的定义，主程序的流程图及各程序模块之间的调用关系）

1．逻辑设计

Stack//定义栈的结构

Pop//出栈操作，并且返回栈顶元素

Peek//返回栈顶元素

Push//返回栈顶元素

Makenull//栈的初始化操作

Judge//判断运算符的优先级别，返回一个 int

Change//程序由前缀转为后缀的核心模块

Start//转为后缀表达式后计算的核心模块

Main//主函数

编码思路：

1. 向后缀表达式的转化

1．输入表达式——以 char 类型的数组储存，在 main 函数里面储存在 a 数组里面，同时定义数组 b，用于储存后面得到的后缀表达式。

2. 正式的转化——首先定义一个栈 r，初始化。然后 while 循环，用于扫描 middle（即 main 函数里面的 a）数组，遇到#字符后退出。然后对扫描进的字符进行判断。

如果扫描到的是空，则继续扫描

如果扫描到的是（，则让（进 r，然后开始扫描下一个

如果扫描到的是），则查询栈 r 中的栈顶元素，用到定于到的 peek 操作。如果 r 的栈顶不是（，即（与）之间有算术运算，则让栈顶元素出栈，并且把栈顶元素存在数组 last 里面，一直做循环，知道 r 栈中的栈顶元素是（，则跳出循环，pop 掉栈顶的(，然后开始扫描下一位。

如果扫描到的是运算符，则取 r 栈中的栈顶元素，让此操作符与栈顶元素操作符进行优先级的比较，调用 judge 函数即可按照返回的 int 得知优先级。如果栈顶元素的优先级大于等于刚扫描的运算符的优先级，则把栈顶的运算符存在 last 数组里面并且 pop 掉栈顶的运算符，然后继续取栈顶运算符，做相同的操作，直到栈顶元素的优先级小于之前扫描到的运算符，然后把刚才扫描到的运算符压入栈 r 中，开始继续扫描 middle 中的下一位。

如果扫描到的字符 0 到 9 之外的，并且不是.，则抛出错误，终止程序的运行。

如果扫描到的是 0 到 9 之间的数或者是.。则把扫描到的数存在 last 里面，继续扫描下一位。

如果扫描到的是#，停止扫描，顺便把储存在栈里面的符号放在数组 last 里面。

（注：数字与运算符之间的区分是通过空格来区分的，所以每当扫描下一种类型的值的时候，都在在 last 数组里面插入一个空格。本功能有 change 函数完成）

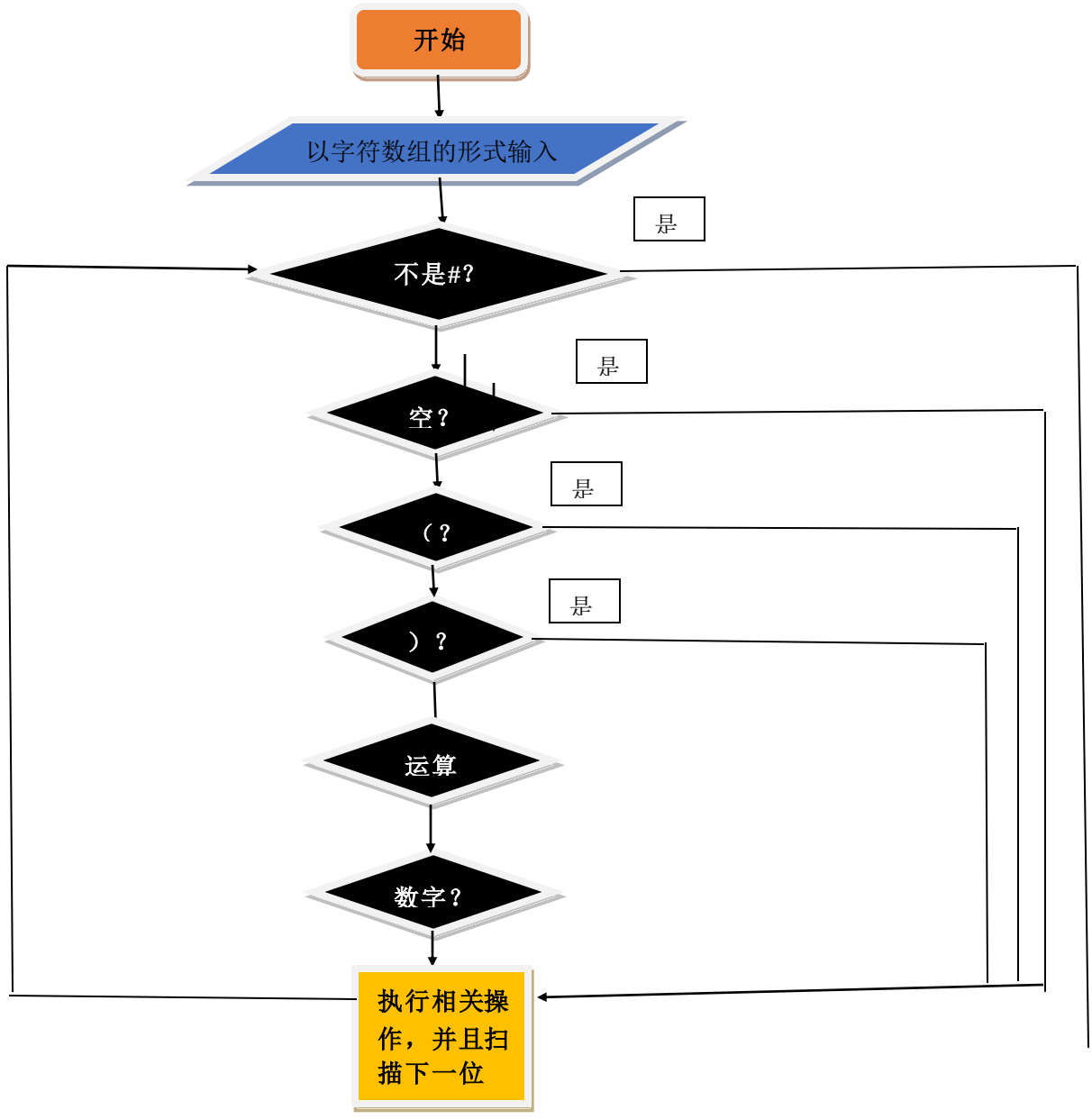
2. 转化为后缀表达式后的求值计算

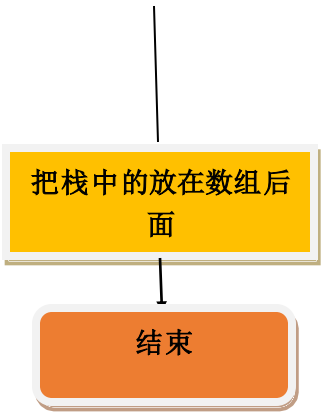
后缀表达式经过 change 函数的转化都存在了数组里面，由于后缀表达式相对来说计算机很容易计算，所以后缀表达式的计算也是相对简单的。

构建栈 s，用来储存临时的数字。

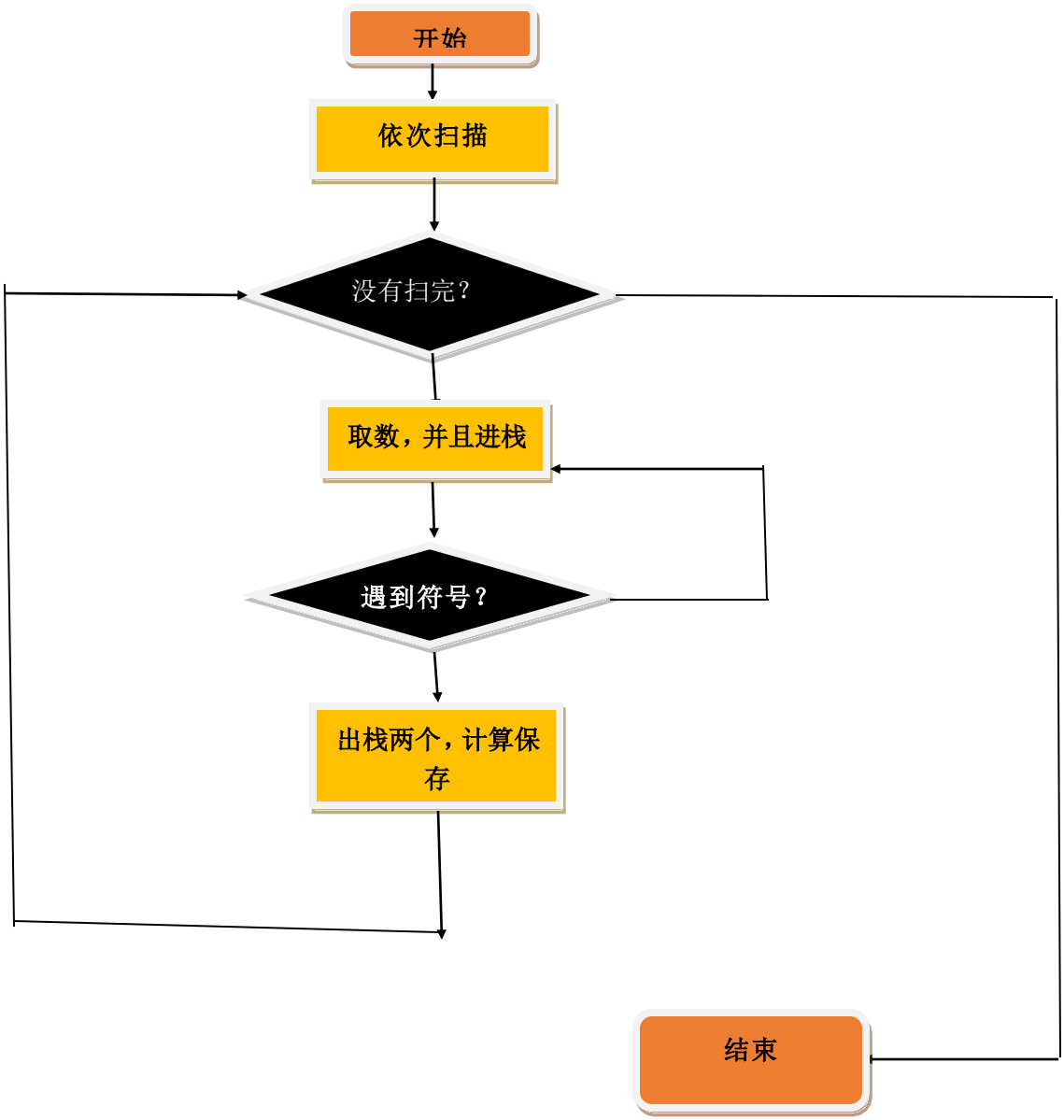
依次扫描 last 中的元素，遇到空格跳过继续扫描，遇到数字入栈。遇到符号的时候栈里面一定存放着两个数字，而两个数字之间要进行的操作为刚扫描的那个运算符要执行的操作。

2． 物理设计



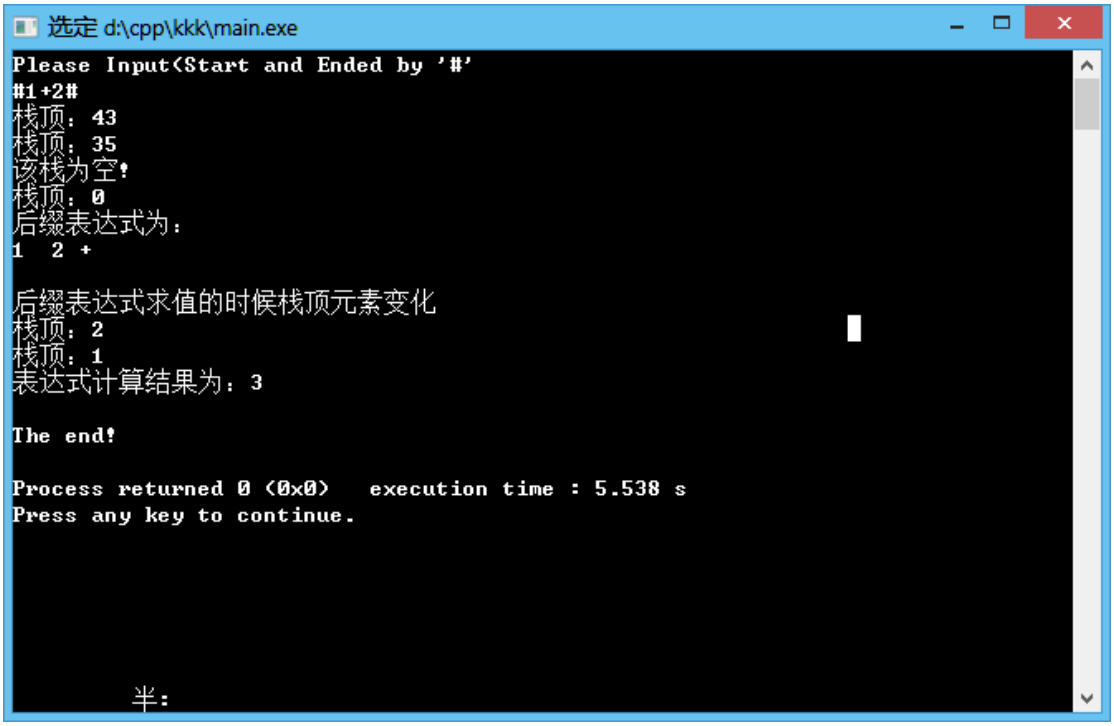


（注：判断框执行是后的内容已经在逻辑设计里面说明）



三、 测试结果

1.#1+2#


选定 d:\cpp\kkk\main.exe
Please Input(Start and Ended by '#'
#1+2#
栈顶：43
栈顶：35
该栈为空!
栈顶：0
后缀表达式为：
1 2 +
后缀表达式求值的时候栈顶元素变化
栈顶：2
栈顶：1
表达式计算结果为：3
The end!
Process returned 0 (0x0) execution time : 5.538 s
Press any key to continue.
半：

2.#1*2#


d:\cpp\kkk\main.exe
Please Input(Start and Ended by '#'
#1*2#
栈顶：42
栈顶：35
该栈为空!
栈顶：0
后缀表达式为：
1 2 *
后缀表达式求值的时候栈顶元素变化
栈顶：2
栈顶：1
表达式计算结果为：2
The end!
Process returned 0 (0x0) execution time : 5.103 s
Press any key to continue.
半：

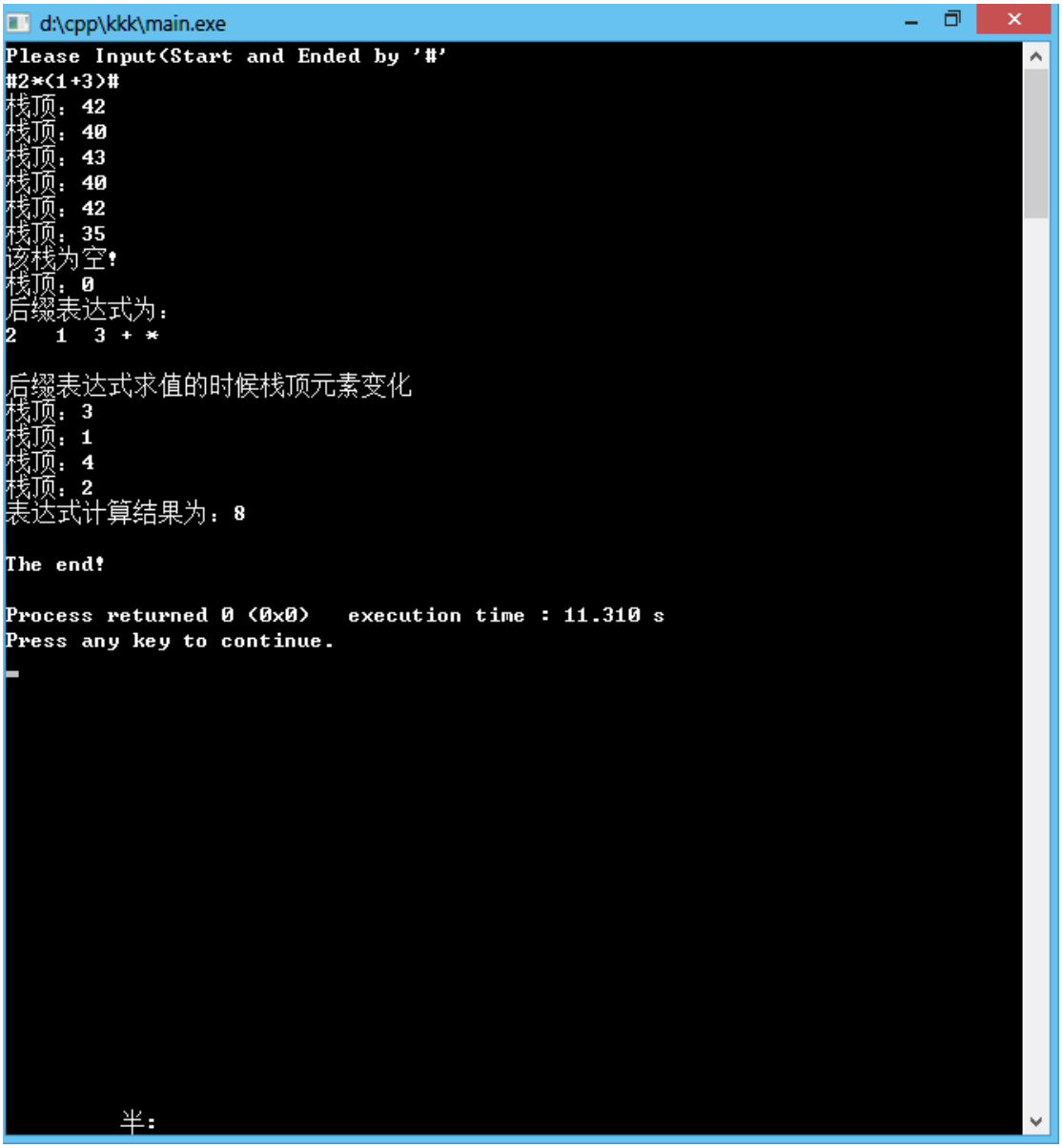
3.#1+2+3#


d:\cpp\kkk\main.exe
Please Input(Start and Ended by '#'
#1+2+3#
栈顶：43
栈顶：35
栈顶：43
栈顶：35
该栈为空!
栈顶：0
后缀表达式为：
1 2 + 3 +
后缀表达式求值的时候栈顶元素变化
栈顶：2
栈顶：1
栈顶：3
栈顶：3
表达式计算结果为：6
The end!
Process returned 0 (0x0) execution time : 5.930 s
Press any key to continue.
半：

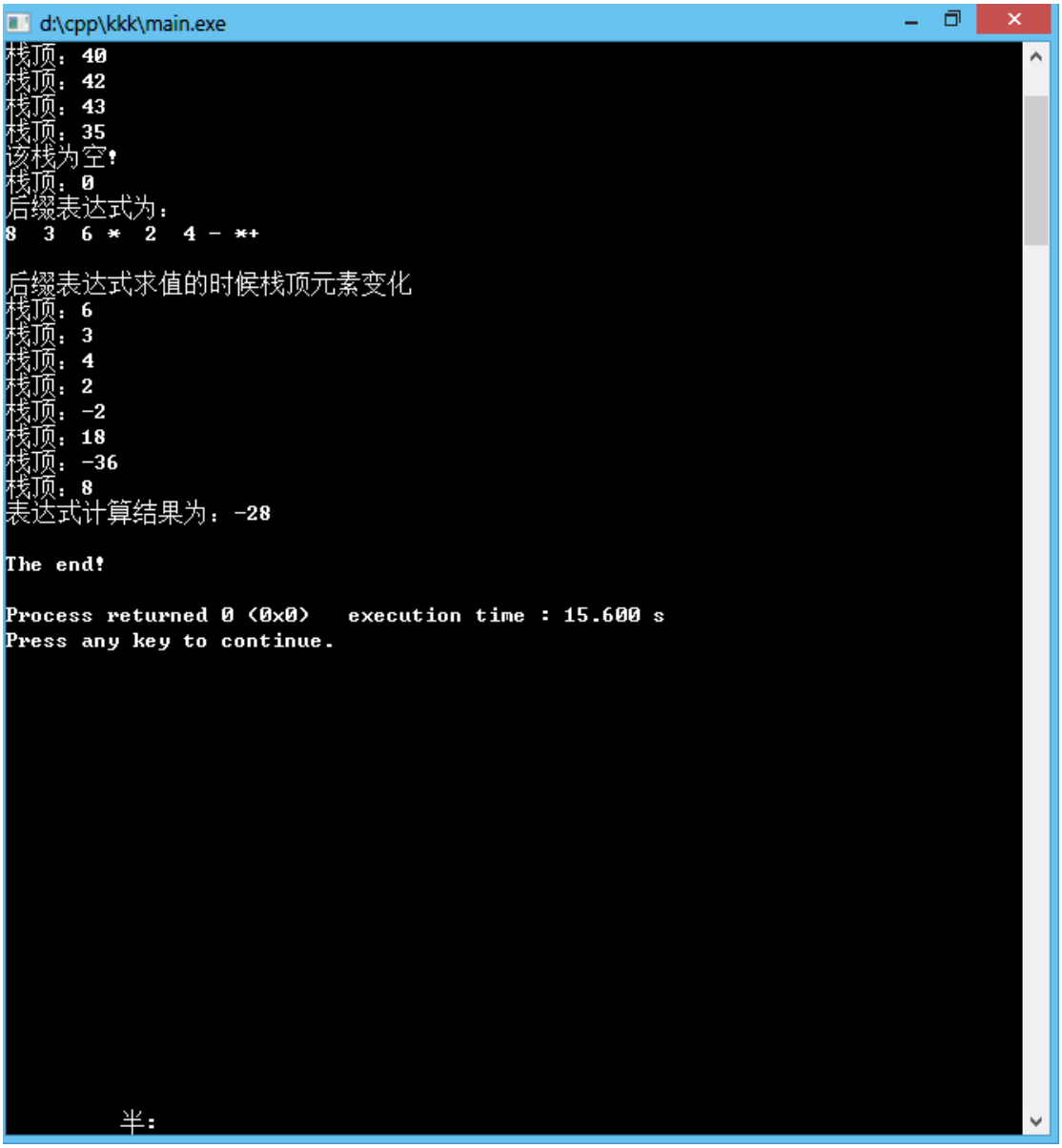
4.#1+2*3#


d:\cpp\kkk\main.exe
Please Input(Start and Ended by '#'
#1+2*3#
栈顶：43
栈顶：42
栈顶：43
栈顶：35
该栈为空!
栈顶：0
后缀表达式为：
1 2 3 **
后缀表达式求值的时候栈顶元素变化
栈顶：3
栈顶：2
栈顶：6
栈顶：1
表达式计算结果为：7
The end!
Process returned 0 (0x0) execution time : 5.101 s
Press any key to continue.
半：

5.#2*（1+3）#


d:\cpp\kkk\main.exe
Please Input(Start and Ended by '#'
#2*(1+3)#
栈顶：42
栈顶：40
栈顶：43
栈顶：40
栈顶：42
栈顶：35
该栈为空!
栈顶：0
后缀表达式为：
2 1 3 + *
后缀表达式求值的时候栈顶元素变化
栈顶：3
栈顶：1
栈顶：4
栈顶：2
表达式计算结果为：8
The end!
Process returned 0 (0x0) execution time : 11.310 s
Press any key to continue.
半：

6.#8+3*6*(2-4)#



五、系统不足与经验体会

系统不足： 由于采用了数组栈，所有产生了两个很是不好的缺点：

1. 当需要的栈比较小时，由于之前就定义了数组栈的长度，所以浪费了多余的空间。
2. 当需要的栈比较大的时候，会超出栈的范围。

改进方案：采用指针栈，动态分配内存的大小。

经验体会：之前没有接触过栈的实际应用，在栈没有熟练掌握的情况下使用了数组栈，并且在调试的初级阶段由于对栈的概念把握不是很深刻造成了很多逻辑上的错误。在以后的学习过程中，会尝试使用指针栈来合理的分配内存空间，刚好的提高代码效率。

六、附录：源代码（带注释）

.CPP 文件

```
#include <iostream>
#include<string.h>
#include <stdlib.h>
#include<math.h>
using namespace std;
#define Maxsize 100000
typedef float ElemType;
/*
*定义栈与栈的操作
*栈：Stack
*初始化：makenull
*入栈：push
*出栈并返回栈顶：pop
*返回栈顶：peek
*/
struct Stack
{
   ElemType s[Maxsize];
   int top;
};
void makenull(Stack&  S)//清空栈中数据
{
   S.top=-1;
}
void push(Stack &S,ElemType  item)//将数据压入栈
{
   if(S.top==Maxsize-1)
      cout<<"该栈已满!"<<endl;
   else
   {
      S.s[++S.top]=item;
   }
```

```
}
ElemType pop(Stack& S)//读取栈顶元素，并删除栈顶元素
{
    if(S.top==-1)
        exit(1);
    else
    {
        return S.s[S.top--];
    }
}
ElemType peek(Stack& S)//读取栈顶元素
{
    if(S.top==-1)
    {
        cout<<"该栈为空!"<<endl;
    }
    return S.s[S.top];
}
/*
*运算符优先级的判断
* +-   return 1
*  x/%   return 2
*  ^    return 3
*   ( #   return 0
*  等级越高，优先级越高
*/
int judge(char op)//运算符的优先级比较
{
    switch(op)
    {
    case '+':
    case '-':
        return 1;
    case '*':
    case '/':
    case '%':
        return 2;
    case '^':
        return 3;
    case '(':
    case '#':
    default:
        return 0;
    }
```

```
}
/*
*中缀转后缀的核心函数，实现了中缀向后缀的转化
*
*Stack r 用来存放运算符
*数存在 middle 的数组中
*last 数组里面存因为括号而在括号之间出现的临时运算符
*/
void change(char* middle,char* last)
{
    Stack r;
    makenull(r);
    //将第一个标志'#'储存在栈底，作为一个标志
    push(r,middle[0]);
    int i=1,j=0;
    char ch=middle[i];
    while(ch!='#')
    {
        if(ch==' ')
        {
            i++;
            ch=middle[i];
        }
        else if(ch=='(')
        {
            push(r,ch);
            cout << "栈顶：" << peek(r) << endl;
            i++;
            ch=middle[i];
        }
        else if(ch==')')
        {
            //发现)，找与他匹配的(
            while(peek(r)!='(')
            {
                //pop 函数删除栈顶并且返回栈顶元素
                last[j]=pop(r);
                j++;
                cout << "栈顶：" << peek(r) << endl;
            }
            //删除(
            pop(r);
            cout << "栈顶：" << peek(r) << endl;
            i++;
```

```
        ch=middle[i];
    }
    //ch 是运算符的情况下
    else if(ch=='+'||ch=='-'||ch=='*'||ch=='/'||ch=='%'||ch=='^')
    {
        char w=peek(r);
        //运算符的比较
        while(judge(w)>=judge(ch))
        {
            last[j]=w;
            j++;
            pop(r);
            cout << "栈顶：" << peek(r) << endl;
            w=peek(r);
        }
        push(r,ch);
        cout << "栈顶：" << peek(r) << endl;
        i++;
        ch=middle[i];
    }
    //排除是小数
    else if((ch<'0'||ch>'9')&&ch!='.')
    {
        cout<<"输入错误"<<endl;
        exit(1);
    }
    else
        while((ch>='0'&&ch<='9')||ch=='.')
        {
            last[j]=ch;
            j++;
            i++;
            ch=middle[i];
        }
    last[j]=' ';
    j++;
}
ch=pop(r);
cout << "栈顶：" << peek(r) << endl;
while(ch!='#')
{
    if(ch=='(')
    {
        cout<<"表达式错误!"<<endl;
```

```
            exit(1);
        }
        else
        {
            last[j]=ch;
            j++;
            ch=pop(r);
            cout << "栈顶：" << peek(r) << endl;
        }
    }
    last[j]='\0';
    j++;
}
/*
*对后缀表达式进行求值计算
*传参：str，即后缀表达式
*/
double start(char *str)//利用后缀表达式求值
{
    Stack s;
    //初始化置空栈
    makenull(s);
    double x,y;
    double a,b;
    cout << "后缀表达式求值的时候栈顶元素变化" << endl;
    int i=0;
    while(str[i])
    {
        if(str[i]==' ')
        {
            i++;
            continue;
        }
        switch(str[i])
        {
        case '+':
            cout << "栈顶：" << peek(s) << endl;
            a=pop(s);
            cout << "栈顶：" << peek(s) << endl;
            b=pop(s);
            x=a+b;
            i++;
            break;
        case '-':
```

```
        cout << "栈顶：" << peek(s) << endl;
        a=pop(s);
        cout << "栈顶：" << peek(s) << endl;
        b=pop(s);
        x=b-a;
        i++;
        break;
    case '*':
        cout << "栈顶：" << peek(s) << endl;
        a=pop(s);
        cout << "栈顶：" << peek(s) << endl;
        b=pop(s);
        x=b*a;
        i++;
        break;
    case '/':
        cout << "栈顶：" << peek(s) << endl;
        a=pop(s);
        cout << "栈顶：" << peek(s) << endl;
        b=pop(s);
        if(a!=0.0)
        {
            x=b/a;
        }
        else
        {
            cout<<"除数不能为0！"<<endl;
            exit(1);
        }
        i++;
        break;
    case '%':
        cout << "栈顶：" << peek(s) << endl;
        a=pop(s);
        cout << "栈顶：" << peek(s) << endl;
        b=pop(s);
        x=(double)((int)b%(int)a);
        i++;
        break;
    case '^':
        cout << "栈顶：" << peek(s) << endl;
        a=pop(s);
        cout << "栈顶：" << peek(s) << endl;
        b=pop(s);
```

```
            x=pow(b,a);
            i++;
            break;
        default:
            //处理小数
            x=0;
            while(str[i]>='0'&&str[i]<='9')
            {
                x=x*10+str[i]-'0';
                i++;
            }
            if(str[i]=='.')
            {
                i++;
                y=0;
                double j=10.0;
                while(str[i]>='0'&&str[i]<='9')
                {
                    y=y+(str[i]-'0')/j;//y保存数值的小数部分
                    i++;
                    j*=10;
                }
                x+=y;
            }
        }
        //压栈
        push(s,x);
    }
    if(s.top==-1)
    {
        cout<<"该栈为空!"<<endl;
        exit(1);
    }
    else
    {
        x=pop(s);
        cout << "表达式计算结果为：" << x << endl;
        cout<<endl;

    }
    return 0;
}
int  main()
{
```

```
    Stack s;
    //初始化栈
    makenull(s);
    char a[100];
    cout << "Please Input(Start and Ended by '#'" << endl;
    cin >> a;
    char b[100];
    change(a,b);
    cout<<"后缀表达式为："<<endl;
    cout<<b<<endl;
    cout<<endl;
    start(b);
    cout << "The end!" << endl;
    return 0;
}
```