**C++ 语 言 程 序 设 计**

实

验

报

告

实 验 二

姓名： 胡志烽

学号： 210310119

班级： 机械1班

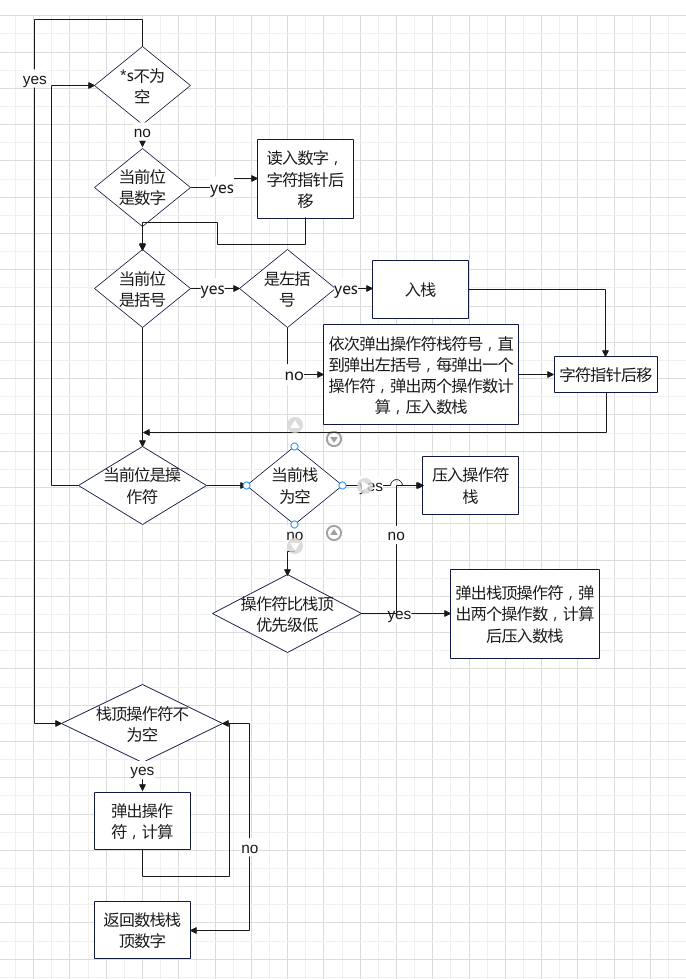
**一 实验项目**

1. 熟悉C++程序设计
2. 掌握C++基本输入输出方法
3. 掌握C++中string类型的使用方法
4. 实现模板栈功能
5. 实现表达式中求值

**二 实验原理**

1. 说明表达式求值的方法（给出算法的**流程图**与简要说明）
2. 给出程序的源代码，说明关键代码的操作含义，给出运行结果

1. 流程图：



简要说明：

首先输入字符串，将字符串传入读取函数，首先判断当前位是否是数字，如果是的话将数字压入数栈，继续读取。然后判断当前位是否是括号，如果是，而且是左括号的话入操作符栈，如果是右括号一次弹出操作符栈括号，每次弹出一个操作符都对两个操作数进行计算，直到弹出的字符是左括号为止

然后判断读入的是否是操作符，如果读入的是乘除操作符直接入栈，如果读入的是加减而栈顶的是乘除的话先计算乘除

#include <iostream>

#include <string>

using namespace std;

template <class T> class Stack;

inline int GetPriority(char s);

double Calculate(double b, char opter, double a);

template <class T>

class Stack

{

private:

T elem;

Stack\* next;

public:

bool is\_empty =false; //如果是底部则为true

void Push(T newelem);

T Pop();

inline T GetTop(){

return elem;

}

void ReadBracket(char\* &s, Stack<double> \*&numhead);

void ReadOpter(char \* &s, Stack<double> \*&numhead);

void ReadNumber(char\* &s, Stack<double> \*&head);

void Calc\_Push(Stack<char>\*opterhead);

};

template<>

void Stack<double>::Calc\_Push(Stack<char>\*opterhead) //弹出两个操作数和一个操作符然后入栈

{

double num1 = Pop();

double num2 = Pop();

char opter = opterhead->Pop();

Push(Calculate(num2, opter, num1)); //要让后弹出的数字作为第一个参数

}

template<class T>

void Stack<T>::Push(T newelem)

{

if (is\_empty)

{

elem = newelem;

is\_empty = false;

}

else

{

Stack<T>\* newnode = new Stack<T>;

T tempelem = elem;

Stack<T>\* temppos = next;

elem = newelem;

next = newnode;

newnode -> next = temppos;

newnode -> elem = tempelem;

}

}

template<class T>

T Stack<T>::Pop()

{

if (is\_empty) //如果栈已经是空的那么再Pop会抛出错误

{

cout << "匹配错误" << endl;

exit(-1);

}

double retelem = elem;

if (next == NULL)

{

is\_empty = true;

}

else

{

elem = next -> elem;

next = next -> next;

}

return retelem;

}

template<>

void Stack<double>::ReadNumber(char\* &s, Stack<double>\* &head) //使用引用以便直接移动字符指针而不引入二级指针

{

int len = 0;

char tempstr[10];

while ('0' <= \*s && \*s <= '9')

{

tempstr[len] = \*s;

len++;

s++;

}

if (len == 0)return;

int num = std::stoi(tempstr);

if (is\_empty)

{

elem = num;

is\_empty = false;

}

else

Push((double)num);

}

template <>

void Stack<char>::ReadBracket(char\* &s, Stack<double> \*&numhead)

{

if (\*s == '(')

{

Push(\*s);

s++;

}

if (\*s == ')')

{

while (GetTop() != '(' && !is\_empty) // 因为已经判断条件里已经有Pop所以循环体里面不能再出现Pop, 使用GetTop()

{

numhead->Calc\_Push(this);

}

if (is\_empty)

{

cout<< "匹配错误！" << endl;

exit(-1);

}

else

Pop();

s++;

}

}

template<>

void Stack<char>::ReadOpter(char \* &s,Stack<double> \*&numhead)

{

if (\*s == '\*' || \*s == '/' || \*s == '+' || \*s == '-')

{

if (is\_empty) //如果当前是空栈的话

{

elem = \*s;

is\_empty = false;

}

else if (GetPriority(\*s) < GetPriority(GetTop())) // \*/优先级>+- 如果入栈加减而栈顶是乘除则需要先计算乘除,反之不需要

{

numhead->Calc\_Push(this);

Push(\*s);

}

else

{

Push(\*s);

}

s++;

}

}

inline int GetPriority(char s) //返回优先级

{

if (s == '\*' || s == '/')return 1;

if (s == '+' || s == '-')return 0;

if (s == '(' || s == ')') return -1;

}

double Calculate(double b, char opter, double a) //计算b theta a

{

switch (opter)

{

case '+':

return b + a;

case '-':

return b - a;

case '\*':

return b \* a;

case '/':

return b / a;

default:

break;

}

}

double Read(char \*s, Stack<double> \*numhead, Stack<char> \*opterhead)

{

while (\*s != '\0')

{

numhead->ReadNumber(s, numhead);

opterhead->ReadBracket(s, numhead);

opterhead->ReadOpter(s, numhead);

}

while (!opterhead->is\_empty) //如果操作符栈不为空

{

numhead->Calc\_Push(opterhead);

}

return numhead->GetTop();

}

int main()

{

Stack<double> \*numhead = new Stack<double>;

Stack<char> \*opterhead = new Stack<char>;

numhead->is\_empty = true;

opterhead->is\_empty = true;

char str[100] ;

cout << "Input!" << endl;

cin >> str;

double retnum = Read(str, numhead, opterhead);

cout << retnum << endl;

}

**三 实验总结与建议**

（总结实验实施过程，说明实验过程中遇到的问题与解决方案；提出实验环节的建议）

在实验过程中遇到的问题有：

1. 要确定操作符的优先级问题
2. 模板栈的链表实现，是否引入头节点（如果不用头节点的话判断是否为空栈会比较麻烦）
3. 括号匹配问题
4. 如果遇到非法输入怎么判断

解决方案：

1. 引入判断操作符优先级的函数，对每个操作符制定优先级度
2. 不引入头节点，引入布尔类型的变量判断是否为空
3. 用实验推荐的方法，遇到右括号依次弹出操作符进行计算，直到弹出左括号
4. 非法输入最后操作符或操作数栈一定会有多余的元素，进行运算时会请求空栈继续弹出，那么就在模板函数Pop里定义如果请求空栈弹出元素那么就报错。