**C++ 语 言 程 序 设 计**

实

验

报

告

实 验 一

姓名： 胡志烽

学号： 210310119

班级： 机械1班

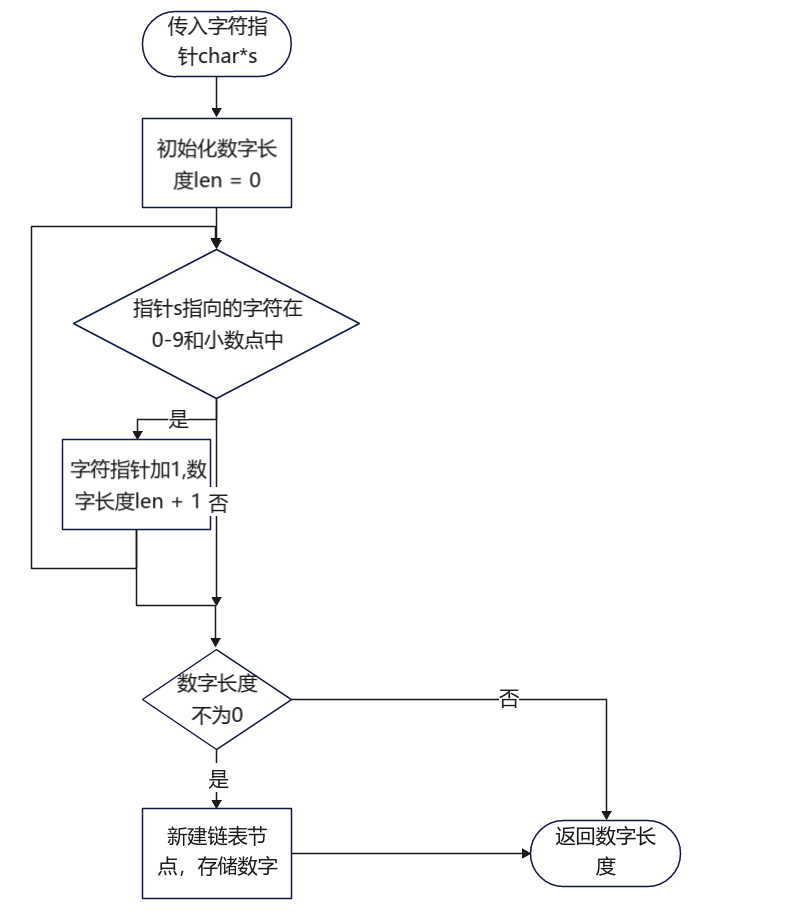
**一 实验项目**

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. 实现字符栈功能
5. 实现表达式中数值与操作符的识别
6. 实现表达式中括号匹配的判断

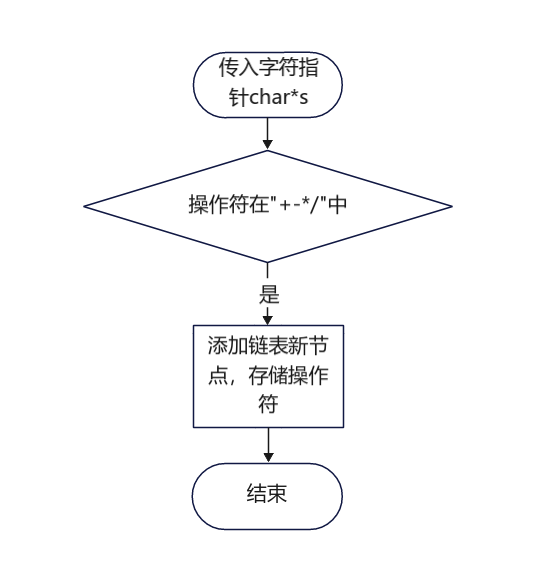
**二 实验原理**

1. 说明字符串表达式中数值提取与操作符识别的方法（给出算法的**流程图**与简要说明）

数值提取方法：



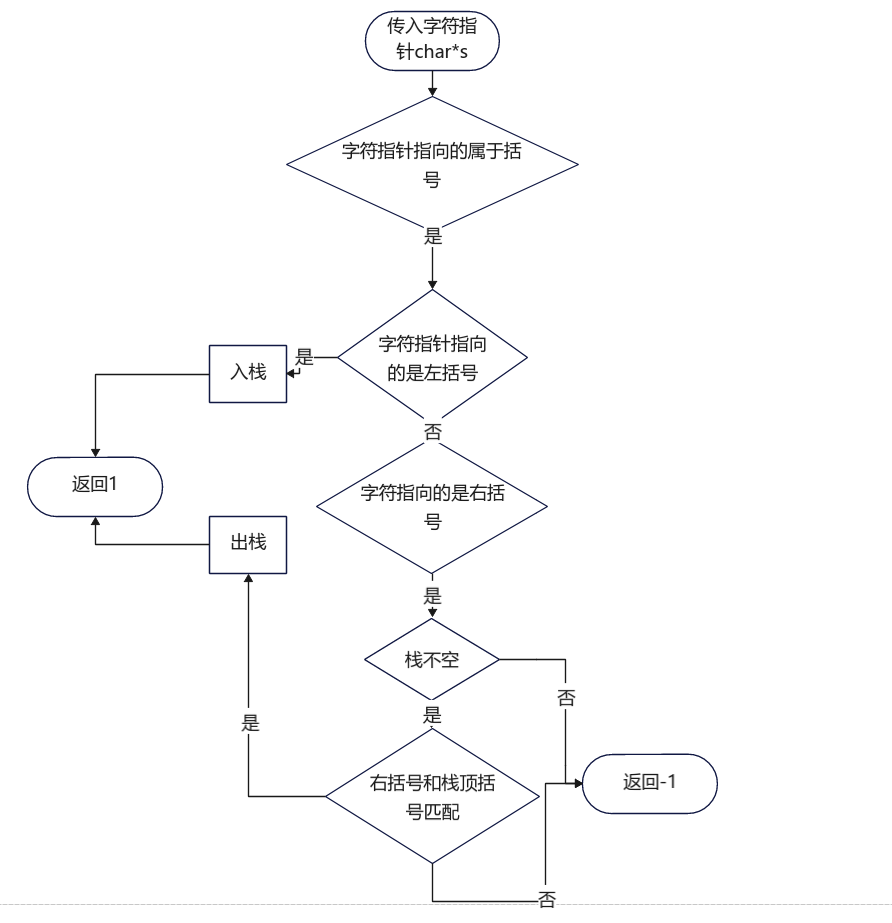
操作符识别：



首先传入字符指针，如果字符首位属于数字，那么字符指针后移直到不是数字的字符为止，记录数字的长度，新建节点，存储数字。

然后字符指针到不是数字的字符那一位，判断是否属于操作符，如果是，那么新建节点，存储操作符

1. 说明表达式中括号匹配的判别方法（给出算法**流程图**与简要说明）



首先传入字符指针，判断字符是否属于括号，如果是，首先判断栈是否为空并且字符是否是左括号

如果是那就入栈，如果不是那么出栈

然后判断字符指针是否右括号，如果属于，然后判断栈是否为空，如果不空进行匹配，如果成功匹配那出栈，不成功匹配错误返回-1

1. 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

#include<iostream>

using namespace std;

typedef char\* ElemType;

class stringnode

{

private:

stringnode\* next;

public:

ElemType name = NULL;

char type; //如果为0就是操作符，为1就是操作数

stringnode\* NewNode()

{

stringnode\* next = (stringnode\*)malloc(sizeof(stringnode));

return next;

}

void PrintNode()

{

if (type == '0')

cout << "操作符" << name <<endl;

else

cout << "操作数" << name <<endl;

}

};

class bracketstack

{

private:

char stackarray[50] ={};

int tail = 0;

public:

int Gettail()

{

return tail;

}

void Pop()

{

tail--;

}

void Push(char bracket)

{

stackarray[tail] = bracket;

tail++;

}

int Judge(char bracket) //判断入栈的括号是否匹配

{

if (bracket == '('||bracket == '[' || bracket == '{') //如果

{

Push(bracket);

return 1;

}

else

{

if (tail == 0)

return -1;

if (

((bracket == '}' || bracket == ']') && bracket == stackarray [tail-1] + 2)

||(bracket == ')' && bracket == stackarray[tail-1]+1)) //如果要入栈的元素栈顶的元素匹配，那就弹出栈顶元素 （通过ASCII码值判断，右括号=左括号+1,大右括号 = 左大括号 + 2

{

Pop();

return 1;

}

else

return -1; //如果不匹配返回-1;

}

}

};

int Readnumber(char\* s, stringnode \* const node)

{

int len = 0; //数字的长度

char\*head = s;

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

{

len ++;

s++;

}

if (len != 0)

{

node->name = (char\*)malloc(sizeof(char)\*len);

int i;

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

node->name[i] = \*(head + i);

node->type = '1';

}

return len;

}

void Read(char \*s, stringnode \*node, bracketstack \* const stack)

{

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

{

int len = Readnumber(s,node);

if (len != 0)

{

s+= len;

node->type = '1';

node->PrintNode();

node = node->NewNode();

}

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

{

node->type = '0';

node->name = (char\*)malloc(sizeof(char));

node->name[0] = \*s;

node->PrintNode();

node = node->NewNode();

s++;

}

if (\*s == '(' || \*s == ')' || \*s == '[' || \*s == ']' || \*s == '{' || \*s == '}')

{

node->type = '0';

node->name = (char\*)malloc(sizeof(char));

node->name[0] = \*s;

node->PrintNode();

if (stack->Judge(\*s) != 1)

{

cout << "匹配错误";

return;

}

node = node->NewNode();

s++;

}

}

if (stack->Gettail()!= 0)

cout << "匹配错误";

}

int main()

{

char\* string = (char\*)malloc(sizeof(char)\*200);

stringnode \*node = (stringnode\*)malloc(sizeof(stringnode));

bracketstack \*stack = (bracketstack\*)malloc(sizeof(bracketstack));

cout << "Input number" << endl;

cin >> string;

Read(string, node, stack);

}

**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）

首先我在纸上画出大致的流程图，然后根据流程图写出实现代码。

遇到的问题有：不熟悉C++语法，类概念，如何使用指针等，这些我通过网上的知识自学了解了相关用法，其次大括号和中括号的左右括号的ASCII码值并不相邻，刚开始遇到这个bug想了很久才想到上网查，改正过来就完成了。

我建议首先了解栈的相关知识,然后先画出代码执行的流程图，依照流程图完成。