

《数 据 结 构》

**实验报告**

|  |  |
| --- | --- |
| **项目名称：** | 栈的实验--栈的简单应用 |
| **班 级：** |  |
| **学生姓名:** |  |
| **学生学号：** |  |
| **指导教师：** |  |
| **成 绩：** |  |

20 年 月 日

成都信息工程大学 计算机学院

1. **编程实现对给定的一组括号序列判断其是否匹配正确？**
2. **实现功能描述**

括号运算是表达式运算中的一个重要部分，不考虑具体运算仅指括号的正确匹配时，具有如下特征：

（1）它必须成对出现，如“(”“)”是一对，“[”与“]”是一对；

（2）出现时有严格的左右关系；

（3）可以以嵌套的方式同时出现多组多括号，但必须是包含式嵌套，不允许交叉式嵌套。比如“()”、“[([][])]”这样是正确的，“[(])”或“([()))”或 “(()]”是不正确的。

（4）将处理的括号扩展为针对“()”“[]”“{}”三类。

1. **方案比较与选择**

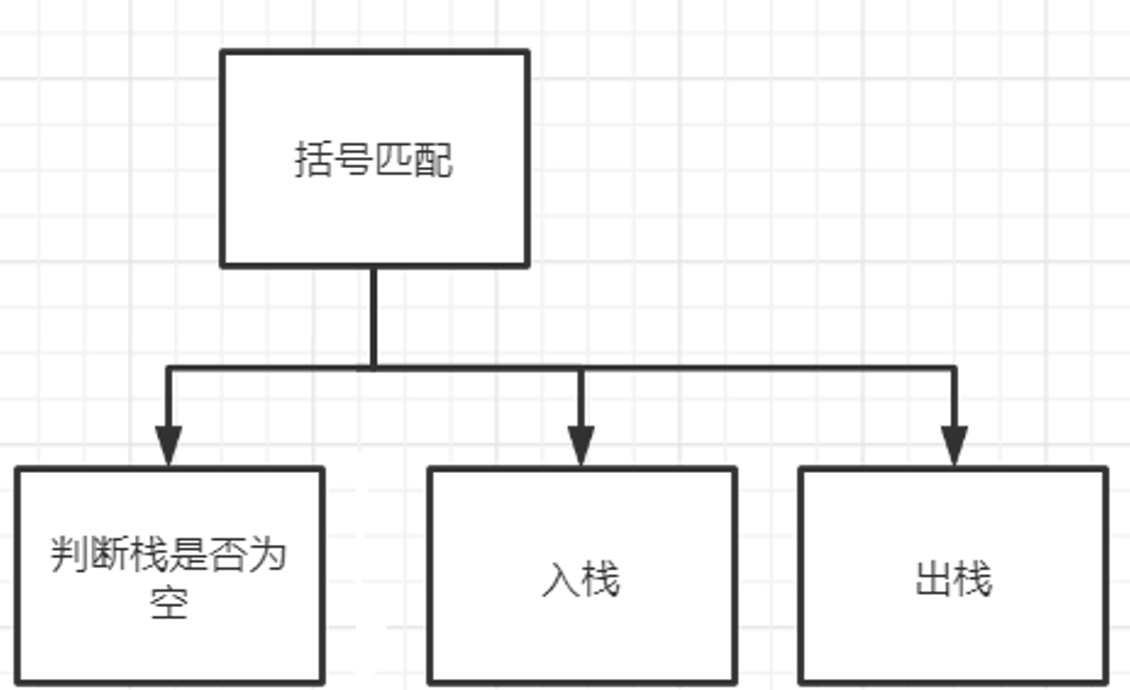
链式栈，动态分配内存空间，节省空间。

顺序栈，便于实现，且不用频繁申请内存空间，节省时间。

为了开发方便，我选择C++标准模板库stack

1. **设计算法描述**

功能模块化：



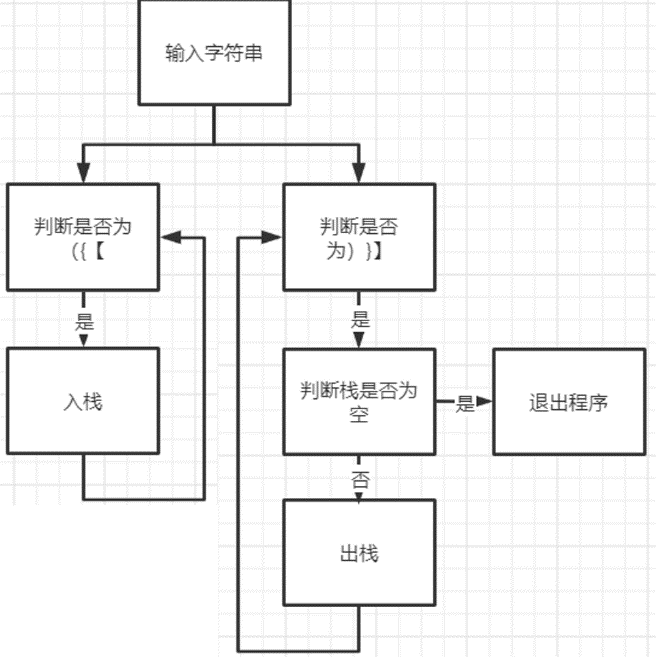
相关子函数：

push();

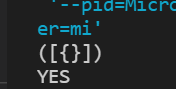
empty();

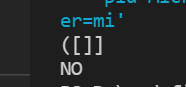
pop();

流程图：



1. **完整程序：**
2. #include<bits/stdc++.h>
3. using namespace std;
4. stack<char> s;
5. char str[10000];
6. int main()
7. {
8. std::ios::sync\_with\_stdio(0);
9. std::cin.tie(0);
10. std::cout.tie(0);
11. cin>>str;
12. int len=strlen(str);
13. int ans=1;
14. for(int i=0;i<len;++i)
15. {
16. if(str[i]=='(' || str[i]=='[' || str[i]=='{')
17. s.push(str[i]);
18. else if(!s.empty())
19. {
20. if(str[i]==')' && s.top()=='(')
21. s.pop();
22. else if(str[i]==']' && s.top()=='[')
23. s.pop();
24. else if(str[i]=='}' && s.top()=='{')
25. s.pop();
26. else
27. {
28. ans=0;
29. break;
30. }
31. }
32. else
33. {
34. ans=0;
35. break;
36. }
37. }
38. cout<<(ans?"YES":"NO");
39. return 0;
40. }
41. **实验结果测试与分析：**





**6.思考及学习心得：**

栈的特点是先入后出。常见应用是处理符号匹配（编译器会用到）。数制转化（r进制转换为其他进制）。中缀表达式计算（人类习惯数学表达式是中缀表达式，而计算机计算需要后缀表达式）。递归（最基础斐波那契数计算，到dfs，到gcd，到线段树，以及各种各样的算法）（虽然所有递归都可以处理为循环，但是递归来写，逻辑会更清晰）

1. **编程实现一个简单的行编辑功能：用户可以输入一行内容，并可进行简易编辑。**
2. **实现功能描述**

（1）遇到输入部分内容有误时操作退格符“#”表示前一位无效；

（2）“@”表示之前的内容均无效。

**2、方案比较与选择**

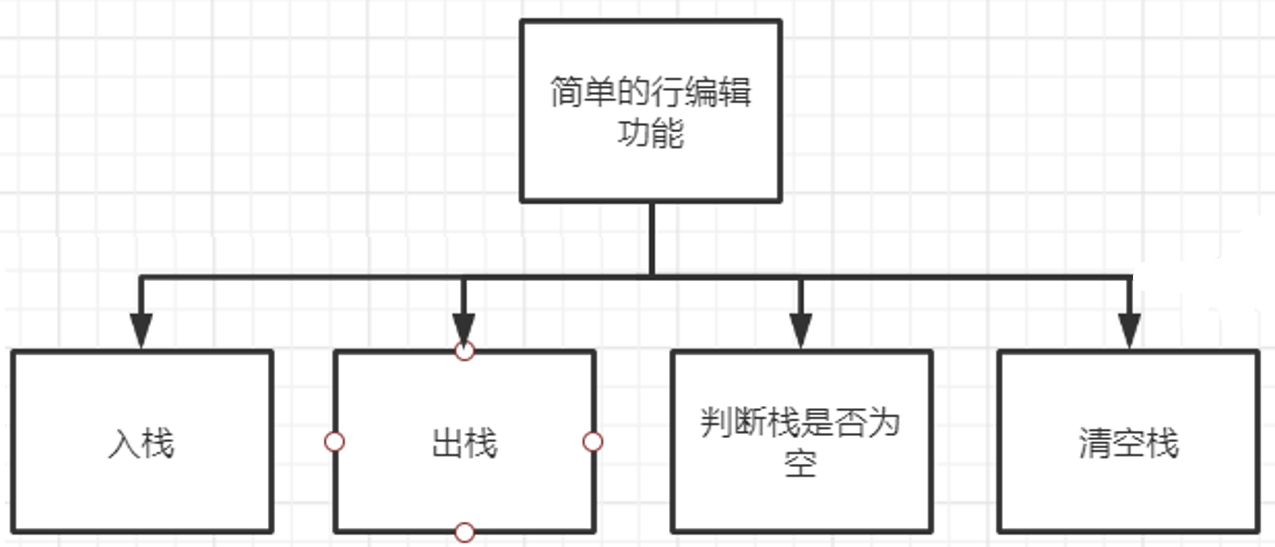
链式栈，动态分配内存空间，节省空间。

顺序栈，便于实现，且不用频繁申请内存空间，节省时间。

为了开发方便，我选择C++标准模板库stack

**3、设计算法描述**

功能模块化：



部分子函数定义：

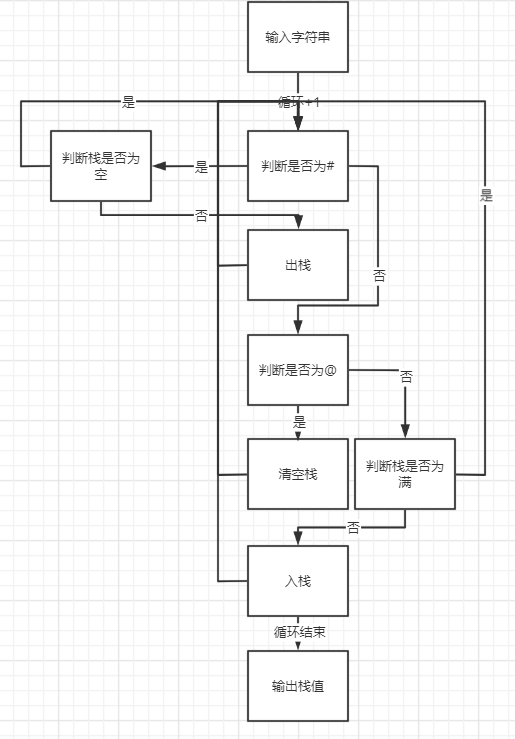
push();

pop():

empty();

clear();

流程图：



1. **完整程序：**
2. #include<bits/stdc++.h>
3. using namespace std;
4. stack<char> s;
5. stack<char> ans;
6. char str[10000];
7. int main()
8. {
9. std::ios::sync\_with\_stdio(0);
10. std::cin.tie(0);
11. std::cout.tie(0);
12. cin>>str;
13. int len=strlen(str);
14. for(int i=0;i<len;++i)
15. {
16. if(str[i]=='#')
17. {
18. if(!s.empty())
19. s.pop();
20. }
21. else if(str[i]=='@')
22. {
23. while(!s.empty())
24. s.pop();
25. }
26. else
27. s.push(str[i]);
28. }
29. while(!s.empty())
30. {
31. ans.push(s.top());
32. s.pop();
33. }
34. while(!ans.empty())
35. {
36. cout<<ans.top();
37. ans.pop();
38. }
39. return 0;
40. }
41. **实验结果测试与分析：**



**三、思考题：**

1.对于这两个应用，除了用栈实现外，还可以用什么样的数据结构实现此功能?给出基本的方法描述。

线性表，模拟栈操作即可。

队列，暴力，把前面的元素出队再入队，就可以操作最后一个元素。

2.对于这两个应用，如果你对栈的基本操作（如压栈、出栈等）采用的是顺序栈（链式栈），那是否可以用链式栈（顺序栈）?谁更优?给出原因分析。

我采用的是顺序栈，可以采用链表栈。顺序栈的优势是查询速度非常快，但是增删改慢；链表栈的优势是增删改快，但是查询慢。并且在链表栈不需要判断栈是否为满，更方便快捷。

1. 对于这两个应用，你还能想到什么实现算法?给出主要描述，并与栈方法进行全方位对比。

想不到