**数据结构实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号-姓名 |  | 实验时间 | 2023年 10月 13日 |
| 诚信声明 | 手写，签名 | | |
| 实验题目 | 括号匹配问题 | | |
| 实验过程中遇到的主要问题 | 在遇到右括号出栈的时候要注意判断是否栈为空，为空则意味着右括号比左括号更多，是不匹配的情况，如果不判断将会出错。 | | |
| 实验小结 | 这次实验实现了括号匹配算法，通过扫描表达式并使用栈来进行匹配操作，判断给定表达式中的括号是否匹配。实验过程中，主要可能出错的地方包括出栈时栈为空的情况的处理以及对表达式中可能出现的其他字符的处理。为了解决这些问题，我采取了以下策略：   1. 当遇到左括号时，将其入栈。 2. 当遇到右括号时，判断栈是否为空，为空代表不匹配。再判断栈顶的左括号是否与之匹配。如果匹配，继续处理。否则，表明括号不匹配，返回false。 3. 如果在遍历完所有字符后，栈不为空，则表明存在未匹配的左括号，返回false；否则，返回true表示括号匹配。 | | |
| 数据结构  （自定义数据类型） | #define MAXLEN 120  typedef char SElemType;  struct Stack {  SElemType base[MAXLEN];  int top;  }; | | |
| 主要算法  （或算法说明） | 括号匹配算法：   1. 初始化一个空栈。 2. 对于表达式中的每个字符c：   a. 如果c是左括号（'('、'['、'{'），将c入栈。  b. 如果c是右括号（')'、']'、'}'）：  i. 如果栈为空，表明右括号多于左括号，返回括号不匹配。  ii. 否则，将栈顶元素取出并与c进行匹配：  - 如果栈顶元素与c匹配，继续遍历下一个字符。  - 如果栈顶元素与c不匹配，返回括号不匹配。  c. 如果c是其他字符，继续遍历下一个字符。   1. 如果遍历完所有字符后，栈不为空，返回括号不匹配；否则，返回括号匹配。 | | |

**数据结构实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号-姓名 |  | 实验时间 | 2023年 10月 13日 |
| 诚信声明 | 手写，签名 | | |
| 实验题目 | 迷宫问题 | | |
| 实验过程中遇到的主要问题 | 使用非递归算法求解迷宫，需要使用栈来模拟回溯过程来找到通路。在设计算法时，正确处理回溯和方向选择的逻辑比较复杂。 | | |
| 实验小结 | 本次实验中，我学会了链栈类型的实现和使用，并且实现了非递归求解迷宫程序。在实验过程中，我进一步理解了栈的概念和应用，以及如何利用栈来解决回溯问题。同时，我也加深了对迷宫问题的理解和解决思路。 | | |
| 数据结构  （自定义数据类型） | struct Pos {  int x, y; // x为行，y为列  };  struct Path {  Pos pos;  int direction; // 1，2，3，4分别是下右上左  };  typedef Path SElemType;  struct SNode {  SElemType data;  SNode \*next;  };  typedef SNode \*Stack; | | |
| 主要算法  （或算法说明） | 非递归求解迷宫通路算法：   1. 创建一个栈对象，用于存储路径信息。 2. 将入口位置入栈，并标记为已访问。 3. 当栈不为空时，重复以下步骤：    1. 获取栈顶元素的位置和方向。    2. 如果当前位置是出口，则找到一条通路，结束算法。    3. 否则，按照当前位置和方向尝试下一个位置，判断是否可行：       1. 如果下一个位置可行，则将其入栈，并标记为已访问。       2. 如果下一个位置不可行，且当前位置的所有方向还没有都尝试过，就按照逆时针方向依次尝试下一个位置，将当前位置的方向进行更改       3. 如果下一个位置不可行，且当前位置的所有方向都尝试过，则将当前位置出栈。 4. 如果栈为空，表示没有找到通路，算法结束。 | | |