数据结构

实验三：堆栈应用括号匹配实验

深圳大学电子与信息工程学院 周飞

实验三 堆栈应用括号匹配实验

一、实验目的

- 掌握堆栈的基本原理
- 掌握堆栈的存储结构
- 掌握堆栈的进栈、出栈、判断栈空的实现方法
- 掌握应用堆栈实现括号匹配的原理和实现方法

3 实验三 堆栈应用括号匹配实验

二、实验要求

- 熟悉C++语言编程
- 熟练使用C++语言实现堆栈的进栈Push、出栈Pop、判断栈空等操作
- 熟练使用堆栈实现括号匹配算法

实验三 堆栈应用括号匹配实验

三、实验内容

1、问题描述

- 一个算术表达式中包括圆括号、方括号和花括号三种形式的括号
- 编程实现判别表达式中括号是否正确匹配的算法

实验三 堆栈应用括号匹配实验

三、实验内容

2、算法

- 顺序扫描算术表达式
- 若算术表达式扫描完成，此时如果栈空，则正确返回(0)；如果栈未空，说明左括号多于右括号，返回(-3)
- 从算术表达式中取出一个字符，如果是左括号（ ‘(‘或 ‘[‘或 ‘{ ‘），则让该括号进栈(PUSH)
- 如果是右括号（ ‘) ‘或 ‘] ‘或 ‘} ‘）：

(1)、如果栈为空，则说明右括号多于左括号，返回(-2)

(2)、如果栈不为空，则从栈顶弹出(POP)一个括号： 若括号匹配，则转1继续进行判断；否则，说明左右括号配对次序不正确，返回(-1)

实验三 堆栈应用括号匹配实验

三、实验内容

3、输入

- 第1行：样本个数，假设为n。
- 第2到n+1行，每一行是一个样本（算术表达式串，[不能有空格]），共n个测试样本。

实验三 堆栈应用括号匹配实验

三、实验内容

4、输入样本

4

{[(1+2)*3]-1}

{[(1+2]*3)-1}

(1+2)*3)-1}

{[(1+2)*3-1]

8 实验三 堆栈应用括号匹配实验

三、实验内容

5、输出

- 共有n行，每一行是一个测试结果，有四种结果：
- 0：左右括号匹配正确　　{[(1+2)*3]-1}
- -1：左右括号配对次序不正确　{[(1+2]*3)-1}
- -2：右括号多于左括号　　(1+2)*3)-1}
- -3：左括号多于右括号　　{[(1+2)*3-1]

实验三 堆栈应用括号匹配实验

三、实验内容

6、输出样本

0

-1

-2

-3

实验三 堆栈应用括号匹配实验

四、实验步骤

1、堆栈的定义

2、初始化堆栈

3、进栈

4、出栈

5、判断栈空

6、括号匹配

实验三 堆栈应用括号匹配实验

四、实验步骤

1、堆栈的定义

- 定义一个堆栈结构，其存储结构包含栈底指针base和栈顶指针top

```
#define MAXSTACKSIZE    100  //栈存储空间最大长度
struct SqStack {
    char  base[MAXSTACKSIZE];//栈底指针，也是栈的基址
    char *top;               //栈顶指针
};
SqStack MBStack;
```

实验三 堆栈应用括号匹配实验

四、实验步骤

2、初始化堆栈

- 如果栈不存在(base==NULL)，则返回错误ERROR
- 将栈顶指针top指向栈底base
- 返回CORRECT

```
int InitStack(SqStack &S)
{
    if (S.base == NULL) return(ERROR);
    S.top = S.base;      // 初始化堆栈(清空堆栈)
    return(CORRECT);
}
```

实验三 堆栈应用括号匹配实验

四、实验步骤

3、进栈

- 如果栈顶(top)超出范围，返回ERROR
- 将新数据(e)插入栈顶指定位置(top)上
- 栈顶指针(top)加1
- 返回CORRECT

```
int Push(SqStack &S, char e)      // 向栈中放入数据[进栈]
{
    if ((S.top-S.base)  >= MAXSTACKSIZE) return(ERROR);
        :
    return(CORRECT);
}
```

实验三 堆栈应用括号匹配实验

四、实验步骤

4、出栈

- 如果栈为空，则返回ERROR
- 栈顶指针(top)减1
- 从栈顶指针指向位置，取一个数据，并放入变量e中
- 返回CORRECT

```
int Pop(SqStack &S, char &e)     // 从栈中取数据[弹栈]
{   if (S.top <= S.base) return(ERROR);
            :
    return(CORRECT);
}
```

实验三 堆栈应用括号匹配实验

四、实验步骤

5、判断栈空

- 如果栈为空，返回ERROR
- 否则返回CORRECT

```
int StackEmpty(SqStack &S)
{
    if (S.top <= S.base) return(ERROR);
    return(CORRECT);
}
```

实验三 堆栈应用括号匹配实验

四、实验步骤

6、括号匹配

```
int MatchBracket(SqStack &S, char *BracketString)// 判断括号是否匹配
{  int  i;  char C, sC;
   InitStack(S);                                  // 清空堆栈
   for (i=0; i<strlen(BracketString); i++) {
       C = BracketString[i];
       if ((C == '(') || (C == '[') || (C == '{')) Push(S, C);
       if ((C == ')') || (C == ']') || (C == '}')) {
          if (StackEmpty(S) == ERROR) return(…);
          Pop(S, sC);
          if ((C == ')') && (sC != '(')) return(…);
          if ((C == ']') && (sC != '[')) return(…);
          if ((C == '}') && (sC != '{')) return(…);}}
   if (StackEmpty(S) != ERROR) return(…);
   return(…); }
```

实验三 堆栈应用括号匹配实验

四、实验步骤

7、主程序

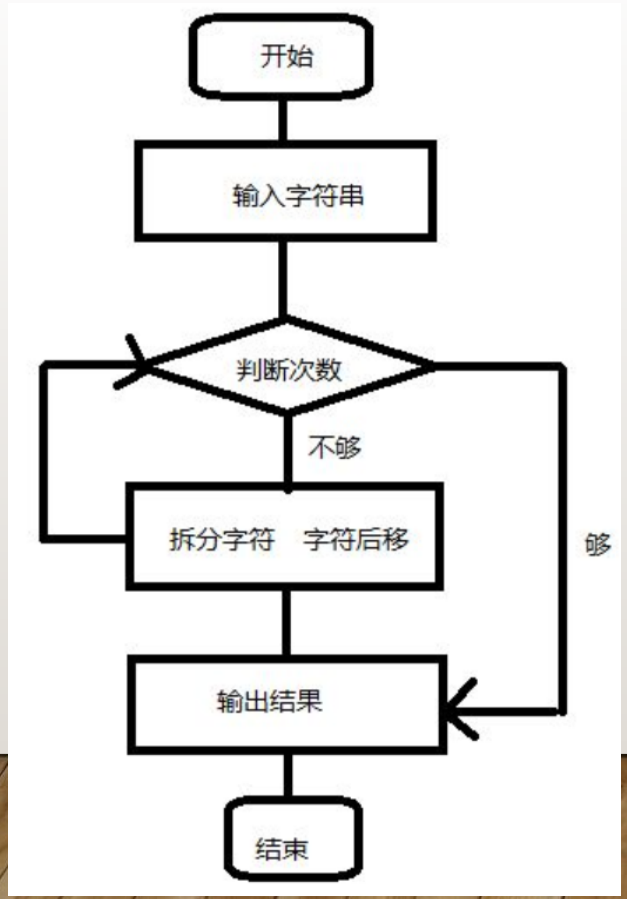
```
int main(int argc, char* argv[])
{
   int  i, SampleNum;
   char BracketString[MAXSTACKSIZE];

   cin >> SampleNum;                    //输入样本数目
   for (i=0; i<SampleNum; i++) {
       cin >> BracketString;            //输入样本数据（一行字符串）
       cout << MatchBracket(MBStack, BracketString) << endl;
   }
   return 0;
}
```

实验三 堆栈应用括号匹配实验

五、程序流程图

- 流程图



实验三 堆栈应用括号匹配实验

六、程序清单