本节内容

栈的应用

——括号匹配

括号匹配问题

```
void test() {
    int a[10][10];
    int x = 10*(20*(1+1)-(3-2));
    printf("加油！奥利给！";
}
```

Expected ')'
to match this '('

```
void test() {
    int a[10][10];
    int x = 10*(20*(1+1)-(3-2));
    printf("加油！奥利给！");
}
```

每一个 单身的人 得看透

想爱 就别怕伤痛

找一个 最爱的 深爱的

想爱的 亲爱的人 来告别单身

又常常羡慕

别人成双入对

括号匹配问题

(((( ))))

① ② ③ ④ ④ ③ ② ①

最后出现的左括号最先被匹配（LIFO）

可用“栈”实现该特性

((( )) ( ))

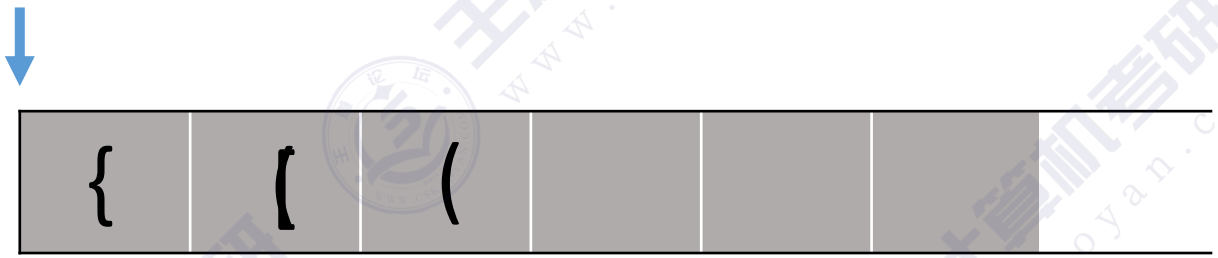
① ② ③ ③ ② ④ ④ ①

每出现一个右括号，就“消耗”一个左括号

出栈

算法演示

栈底



{ (( )) [ ] }

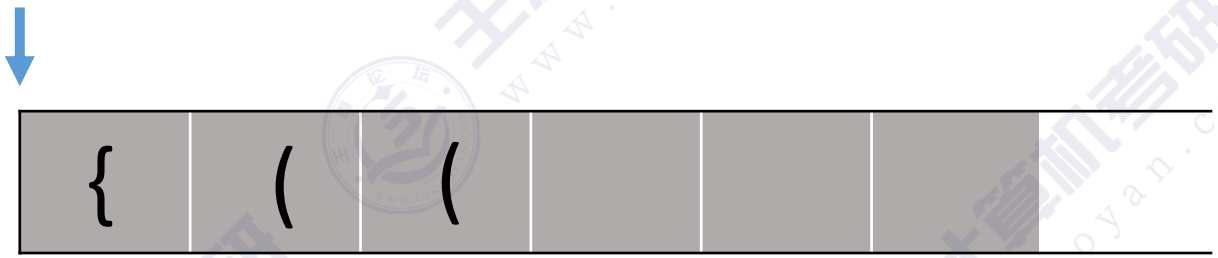
① ② ③ ③ ② ④ ④ ①

所有括号都能两两配对

遇到左括号就入栈
遇到右括号，就“消耗”一个左括号

算法演示

栈底



{ (( )] [ ] }

① ② ③ ③ ②

当前扫描到的右括号
与栈顶左括号不匹配

遇到左括号就入栈
遇到右括号，就“消耗”一个左括号

算法演示

栈底

{ (( )) }] ( )

① ② ③ ③ ② ①

扫描到右括号且栈空，——右括号单身

遇到左括号就入栈
遇到右括号，就“消耗”一个左括号

算法演示

栈底

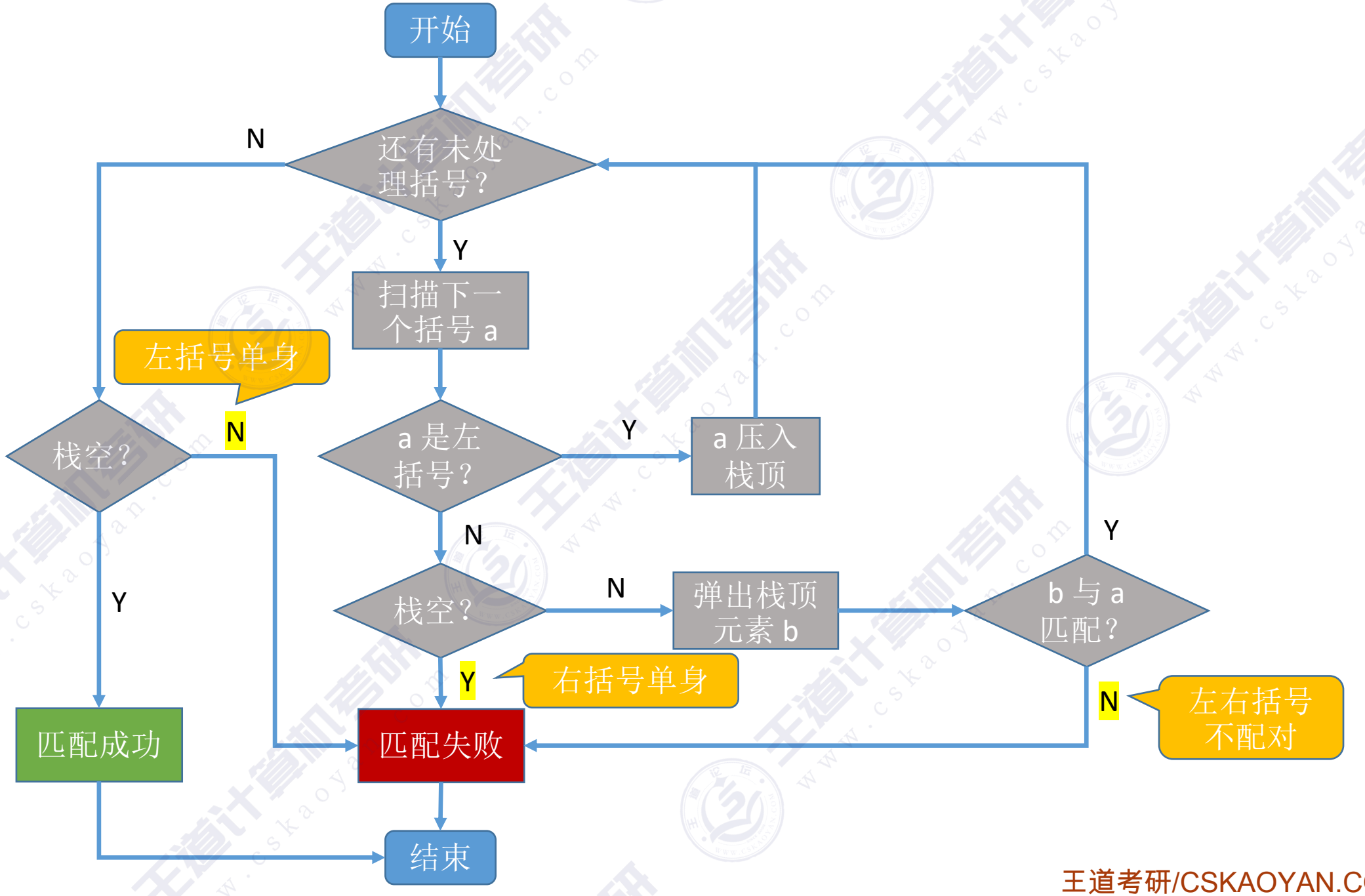
处理完所有括号后，栈非空——左括号单身

{ { (( )) [ ] }

① ② ③ ④ ④ ③ ⑤ ⑤ ②

遇到左括号就入栈

遇到右括号，就“消耗”一个左括号



算法实现

```
bool bracketCheck(char str[], int length) {
    SqStack S;
    InitStack(S);    //初始化一个栈
    for (int i=0; i<length; i++){
        if (str[i]=='(' || str[i]=='[' || str[i]=='{'){
            Push(S, str[i]);    //扫描到左括号，入栈
        } else {
            if (StackEmpty(S))  //扫描到右括号，且当前栈空
                return false;   //匹配失败

            char topElem;
            Pop(S, topElem);    //栈顶元素出栈
            if(str[i]==')' && topElem!='(')
                return false;
            if(str[i]==']' && topElem!='[')
                return false;
            if(str[i]=='}' && topElem!='{')
                return false;
        }
    }
    return StackEmpty(S);   //检索完全部括号后，栈空说明匹配成功
}
```

万一存满了可咋整？

可用链栈！

```
#define MaxSize 10              //定义栈中元素的最大个数
typedef struct{
    char data[MaxSize];         //静态数组存放栈中元素
    int top;                    //栈顶指针
} SqStack;
```

考试中可直接使用基本操作，建议简要说明接口

```
//初始化栈
void InitStack(SqStack &S)

//判断栈是否为空
bool StackEmpty(SqStack S)

//新元素入栈
bool Push(SqStack &S,char x)

//栈顶元素出栈，用x返回
bool Pop(SqStack &S,char &x)
```

练习：不要使用基本操作，动手实现完整代码

知识回顾与重要考点

用栈实现括号匹配：
依次扫描所有字符，遇到左括号入栈，遇到右括号则弹出栈顶元素检查是否匹配。

匹配失败情况：
①左括号单身②右括号单身③左右括号不匹配