

三步程序设计法

乔海燕

中山大学计算机学院

2015 年 9 月 7 日

三步程序设计法：第一步，用自然语言描述算法；第二步，进一步细化和精确化算法，并用伪代码描述算法；第三步，用程序设计语言描述算法，完成代码编写。

下面以解决判断括号匹配问题为例说明三步法。

问题

给定一个字符串序列，包含了括号(,),[],等三种括号和其他字符，请判断其中括号是否匹配。例如，”int main()return 0;”是匹配的，而”void foo(int x[]”则不匹配。

第一步：叙述算法

下面是判断括号匹配的标准算法：

输入：一个字符序列

输出：括号是否匹配

括号匹配算法：

1. 准备空栈；
2. 对于每个输入符号c重复下列步骤：
3. 如果c是左括号，则将其压栈；
4. 如果c是右括号，检查它是否与栈顶符匹配：
5. 如果匹配，则弹出栈顶符；
6. 如果不匹配，则报告错误，算法终止；
7. 最后，输入读完时，如果栈空，则匹配成功，
8. 否则，匹配失败。

这里使用了对齐的方式来表达算法的结构：缩进表示前一个命令的继续，对齐表示命令顺序执行。例如，7是在2-6循环结束后执行的，5-6 属于命令4的继

续等。

第二步：伪代码算法

假定已有三个判定函数：left(c)判断c是否左括号，right(c)判断c是否右括号，match(a,b)判断a与b是否匹配。下面是细化后的伪代码算法。

```
Stack S; //S是空栈
for every input character c {
    if (left(c))
        S.push(c);
    else if (right(c)) {
        if (!S.empty() and match(S.top(), c))
            S.pop();
        else
            return "fail";
    }
}
if (S.empty())
    return "ok";
else
    return "fail";
```

第三步：程序

最后，用程序设计语言表达算法。

```
bool left(char c){
    return c=='(' || c=='[' || c=='{' ;
}
bool right(char c){
    return c==')' || c==']' || c == '}';
}
bool match(char a, char b){
    return (a=='(' && b==')') || (a=='[' && b==']') || (a=='{' && b=='}');
}

bool match(string cs){

    stack<char> S; //初始化空栈S;
    for (size_t i=0; i<cs.size();i++){
        if (left(cs[i]))
            S.push(cs[i]);
        else if (right(cs[i])) {
```

```

        if (!S.empty() && match(S.top(),cs[i]))
            S.pop();
        else
            return false;
    }
}
if (S.empty())
    return true;
else
    return false;
}

```

检验程序

最后，设计一段测试代码检验程序：

```

int main(int argc, char* argv[]){
    if (argc <=1){
        cout <<"usage: match string"<<endl;
        return 1;
    }

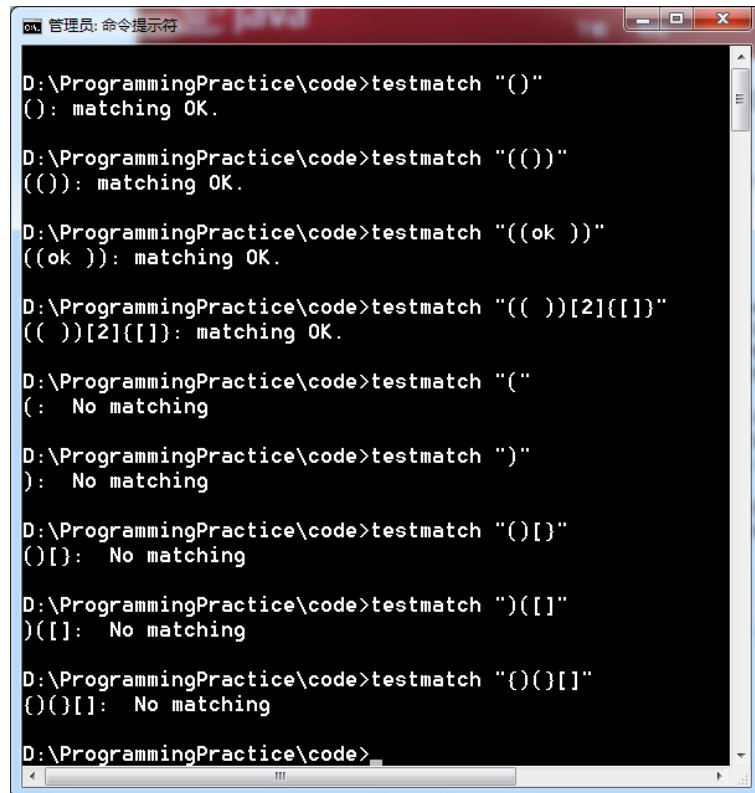
    string cs = argv[1]; //用户输入的字符串

    bool b = match(cs);
    if(b)
        cout << cs <<": matching OK."<<endl;
    else
        cout <<cs <<": No matching"<<endl;

    return 0;
}

```

这里使用了主函数参数：用户在可执行程序后面输入字符串，程序将该串作为输入，判断其中括号是否匹配，见图1。



```
D:\ProgrammingPractice\code>testmatch "()"
(): matching OK.

D:\ProgrammingPractice\code>testmatch "()"
(): matching OK.

D:\ProgrammingPractice\code>testmatch "((ok ))"
((ok )): matching OK.

D:\ProgrammingPractice\code>testmatch "(( ))[2]([)]"
(( ))[2]([)]: matching OK.

D:\ProgrammingPractice\code>testmatch "("
(: No matching

D:\ProgrammingPractice\code>testmatch ")"
): No matching

D:\ProgrammingPractice\code>testmatch "()[]"
()[]: No matching

D:\ProgrammingPractice\code>testmatch ")([]"
)([]: No matching

D:\ProgrammingPractice\code>testmatch "()()[]"
()()[]: No matching

D:\ProgrammingPractice\code>
```

图 1 测试括号匹配程序