養天地正氣 法古今完人

栈的应用（一）
括号匹配

括号匹配

- 问题定义：
  - 任务：检查输入的文本中括号是否正确匹配
  - 限定：
    - 仅考虑：(), [], {}这三类括号
    - 读入一行字符，将忽略括号外的其他所有符号
    - 例如：例如：**{3+2*[(4+2/3)+3]*2+3/4}*4+2**
      - 特点：最后出现的左括号先与遇到的右括号匹配

括号匹配

- 示例：
  - {3*[A+(b*cd)]}
  - 3*A+(b*cd]+5
  - 3*A+(b*cd)dfg]
  - {3*A+(b*cd)+5

括号匹配

- 算法思想：
  - 初始化一个栈
  - 循环读入字符x，分情况讨论，直到读完所有字符：
    - X不是括号：当作普通字符，忽略；
    - X是左括号：入栈
    - X是右括号：
      - 判断栈是否空，若为空，则缺左括号，不匹配
      - 不空，则弹出栈顶左括号，进行匹配；
  - 进行尾部处理：输入完毕，检测此时栈是否空
    - 空：正确匹配
    - 不空：缺右括号，不匹配

括号匹配

```
#include “Stack.h”

int main( )
/* Post: The program has notified the user of any bracket
mismatch in the standard input file.
Uses: The classStack . */
{ Stack openings;
char symbol;
bool is_matched = true;
while (is_matched && (symbol = cin.get( )) != '\n') {
      if (symbol == '{' || symbol == '(' || symbol == '[')
            openings.push(symbol);
if (symbol == '}' || symbol == ')' || symbol == ']') {
```

```
if (openings.empty( )) {
    cout << "Unmatched closing bracket " << symbol << " detected."  << endl;
    is_matched = false;}
else {
   char match;
   openings.top(match);
   openings.pop( );
   is_matched = (symbol == '}' && match == '{')
|| (symbol == ')' && match == '(') || (symbol == ']' && match == '[');
if (!is_matched)   cout << "Bad match " << match << symbol << endl;
}
}}
if (!openings.empty( ))
    cout << "Unmatched opening bracket(s) detected." << endl;
}
```


```
 bool   is_matched(char *s)
{
Stack openings;
char symbol;
bool is_matched = true;
int i=0;
while (s[i]!= '\0') {
if (s[i] == '(' || s[i] == '[‘ || s[i] == ‘{‘ )
          openings.push(s[i]);
if (s[i] == ')' || s[i] == ']‘|| s[i] == ‘{‘ ) {
         if (openings.empty( ))
                  return  false;
         else {
```


```
char match;
openings.top(match);
openings.pop( );
is_matched= ((s[i] == ')' && match == '(') || (s[i]
== ']' && match == '[‘ ) || (s[i] == ‘}' && match
== ‘{'));
If (!is_matched) return false;
}
}
i++;
}
if (!openings.empty( ))
 return false;
return true;
}
```