C++方向编程题答案

第七周

day41

> 题目ID：847 五子棋
>
> 链接：https://www.nowcoder.com/questionTerminal/a811535fed784ea492b63622c28c75c5
>
> **【题目解析】**
>
> 该题是对五子棋赢的检测，五子棋赢的规则：如果当前位置有连在一起的五颗棋子为同一种，则为赢，每取一个位置，检测该位置上、下、左、右以及对角线为是否为统一中棋子，是输出YES，否则获取下一个位置继续检测，如果棋盘中没有连在一起的五颗棋子则输出NO
>
> **【解题思路】**
>
> 结构设计：dir代表当前位置的8个方向，其中上下、左右、左上右下、右上左下为必须放在一起检测。
>
> 获取一个棋盘，按照行列检测棋盘中的每个位置，当拿到一个位置后，按照以下步骤进行操作：
>
> 1. 以该位置为中心，依次检测该位置的上下、左右、左上右下、右上左下，比如左上
> 2. 从该位置开始向上检测，找出连在一起的同种棋子个数，再向下检测同种棋子的个数并累计，注意在检测时，中心位置统计了两次，上下统计完时，需要给结果减去1
> 3. 按照2统计完上下、左右、左上右下、右上左下各个方向，找出最大的同种棋子个数
> 4. 检测3中统计出的最大同种棋子个数，如果大于等于5，输出YSE，否则取下一个位置继续1
> 5. 如果所有的位置检测完，没有超过5个的相同棋子，则输出NO

```
/*
    https://blog.csdn.net/qq_28051453/article/details/51843130
*/
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

#define N 20

int count(string table[], char ch, int x, int y)
{
    int maxc = 0;
    int dir[4][2][2] = { {{ -1,0 },{ 1,0 }},{{ 0,-1 },{ 0,1 }},{{ -1,-1 },{ 1,1 }},{{ -1,1
},{ 1,-1 }} };
    for (int i = 0; i < 4; ++i) // 四种方向
    {
        int c = 0;
        for (int j = 0; j < 2; ++j)  // 两个小方向
        {
            int nx = x, ny = y;
            while (nx >= 0 && nx < N && ny >= 0 && ny < N && table[nx][ny] == ch)

            {
```

```
                nx += dir[i][j][0];
                ny += dir[i][j][1];
                ++c;
            }
        }
        maxc = (maxc > c ? maxc : c);
    }
    return maxc-1; //统计两个方向（如横向的左右两个方向）的时候，当前棋子被计算了两次
}

bool solve(string table[])
{
    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
        {
            if (table[i][j] == '*' || table[i][j] == '+')
                if (count(table, table[i][j], i, j) >= 5)
                    return true;
        }
    }
    return false;
}

int main()
{
    string table[N];
    while (cin >> table[0])
    {
        for (int i = 1; i < N; ++i) cin >> table[i];
        cout << (solve(table) ? "Yes" : "No") << endl;
    }
    return 0;
}
```


题目ID：794 Emacs计算器

链接：https://www.nowcoder.com/questionTerminal/1a92fbc771a54feb9eb5bd9b3ff2d0a9

【题目解析】

逆波兰表达式(后缀表达式)求值，需要借助栈，具体参见解题思路

【解题思路】

循环输入，获取逆波兰表达式，然后进行以下补助，直到测试完所有的测试用例：

1. 遇到数字字符串，将该数字字符串转化为数字然后入栈。
2. 遇到操作符时，从栈顶取两个数字，然后进行该运算符所对应运算，完成好将结果入栈，注意：先取到的数字为运算符的右操作数。
3. 继续1和2，直到处理完所有的字符串，最终栈顶元素即为所要结果。

```
#include<iostream>
#include<stack>
```

```cpp
#include<string>
#include<vector>
using namespace std;
int main(){
    int n;
    int a,b,ret;
    while(cin>>n){
        if(n==0)
            continue;
        vector<string> s(n);
        stack<int> st;
        for(int i=0;i<n;i++){
            cin>>s[i];
        }
        for(int i=0;i<n;i++){
            if(s[i][0]>='0'&&s[i][0]<='9'){
                st.push(atoi(s[i].data()));
            }
            else{
                a=st.top();
                st.pop();
                b=st.top();
                st.pop();
                switch(s[i][0]){
                    case '+': st.push(a+b);break;
                    case '-': st.push(b-a);break;
                    case '*': st.push(a*b);break;
                    case '/': st.push(b/a);break;
                }
            }
        }
        cout<<st.top()<<endl;
    }
    return 0;
}
```