4.正则表达式(计算器)

4.1.认识正则表达式

正则表达式是一种描述字符序列的方法 默认的正则表达式语言是'ECMAScript'.

4.2.正则表达式组件库

4.2.1.变量类型

> regex --> 表示有一个正则表达式的类
>
> smatch --> 存储字符串匹配结果

4.2.2.常用组件

> `R"(...)"` --> 表示原始字符串字面量,允许字符串中包含特殊字符而无需转义
>
> `(... | ... | ...)` --> 表示匹配,捕获并分组`(\+|\-)`
>
> `[...]` --> 表示匹配`[abc][+-][a-z][A-Za-z]`
>
> `(?:...)` --> 表示非捕获分组
>
> 例如 `regex r(R"((\+|\-)?\d+(\.\d+)?)");` 存储一个可以匹配数字的正则表达式

4.2.3.特殊符号

> `^...` -->在`[]`内表示非,在`[]`外表示开始
>
> `...$` -->表示结束
>
> `\...` --> 转义符号 `[+-]`无需转义,`(\+|\-)`有`|`需转义
>
> `-` --> 范围运算符
>
> `...` --> 匹配一个式子
>
> `...?` --> 匹配0次或1次
>
> `...+` --> 匹配一个或多个式子
>
> `...*` --> 匹配零个或多个式子

4.2.4.常用字符类

> `.` --> 匹配除回车任意字符
>
> `\s` --> 匹配空白字符(空格/换行等)
>
> `\d` --> `[\d]=[0-9]`
>
> `\w` --> `[\w]=[a-zA-Z0-9_]`

> [:space:] --> [[:space:]]=[\s]
>
> [:digit:] --> [[:digit:]]=[0-9]
>
> [:alpha:] --> [[:alpha:]]=[a-zA-Z]
>
> [:lower:] --> [[:lower:]]=[a-z]
>
> [:upper:] --> [[:upper:]]=[A-Z]

4.2.5.正则表达式常用函数

> regex_match(seq,r) --> 确定字符序列seq与regex对象r是否匹配,匹配则返回true
>
> regex_search(seq,m,r) --> 寻找字符序列seq中与regex对象r匹配的子串,找得到则返回true,同时把匹配结果存在smatch对象m中
>
> regex_replace(seq,r,fmt) --> 寻找字符序列seq中与regex对象r匹配的子串,使用fmt来替代输出,返回一个替换后的字符串

4.2.6.示例

```
#include<iostream>
#include<regex>
#include<string>
using namespace std;

const double Limit=10000.0;
class Calculator
{
private:
    static const regex formula; // 正则表达式对象,用于匹配输入的表达式
    void calculate(double a,double b,char op); // 计算函数
public:
    bool preprocess(string expression); // 预处理输入的表达式,包括语法检查和调用计算函数

};

const regex Calculator::formula(R"(\s*([+-]?\d+(?:\.\d+)?)\s*([+\-*/])\s*([+-]?\d+(?:\.\d+)?)\s*)"); // 匹配形如"1+1"或"1 + 1"的表达式

bool Calculator::preprocess(string expression)
{
    {
        smatch parts;
        if(!regex_match(expression,parts,formula)) // 如果表达式不符合正则表达式的规则
        {
            cout<<"Expression error!"<<'\n';
            return 0;
        }
        double a,b;
```

```
        char op;
        a=stod(parts[1].str());
        op=parts[2].str()[0];
        b=stod(parts[3].str());
        if(a<-Limit || a>Limit || b<-Limit || b>Limit) // 如果数字超出范围
        {
            cout<<"Number out of range!"<<'\n';
            return 0;
        }
        if(op=='/' && b==0) // 如果除数为零
        {
            cout<<"Division by zero!"<<'\n';
            return 0;
        }
        calculate(a,b,op);
        return 1;
    }
}
void Calculator::calculate(double a,double b,char op)
{
    double result;
    switch(op)
    {
        case '+':
            result=a+b;
            break;
        case '-':
            result=a-b;
            break;
        case '*':
            result=a*b;
            break;
        default:
            result=a/b;
            break;
    }
    cout<<a<<" "<<op<<" "<<b<<" = "<<result<<'\n';
}

int main()
{
    cout<<"请用户输入单项数学式(输入'exit'退出)\n";
    Calculator calc;
    while(1)
    {
        string expression;
        getline(cin,expression);
        if(expression=="exit")
            break;
        if(!calc.preprocess(expression))
            cout<<"\n请重新输入\n";
        else
            cout<<"\n请继续输入\n";
    }
```

```
    return 0;
}
```