

```

1  /*****
2  EGRE 531: Multi-threaded Programming
3  Lab 1
4  Programmed by: Luis Barquero
5  Purpose: Program will calculate the value of an RPN expression.
6  *****/
7  #include <iostream>
8  #include <stack>
9  #include "Lab1.h"
10 #include <string.h>
11 #include <stdlib.h>
12 #include <stdbool.h>
13 #include <algorithm>
14 using namespace std;
15
16 Calculator::Calculator() //Default constructor.
17 {
18
19 };
20
21 void Calculator::add(double x, double y) //function that will add 2 numbers.
22 {
23     this -> answer = x + y; //this -> answer to access private member answer. Adds up
24     two numbers.
25 };
26
27 void Calculator::add(double x) //function that will add a number to previously marked
28 answer.
29 {
30     stack<double> answer_stack;
31     answer_stack.push(this -> answer); //places the answer at the previous answer at
32     the top of the stack.
33     answer_stack.pop(); //pops out the answer at the top of the stack.
34     this -> answer = this -> answer + x; //performs the + operation and stores the
35     answer in this -> answer.
36     answer_stack.push(this -> answer); //pushes this -> answer back into the stack.
37 };
38
39 void Calculator::sub(double x) // function that will subtract a number from a previous
40 answer.
41 {
42     stack<double> answer_stack;
43     answer_stack.push(this -> answer); //places the answer at the previous answer at
44     the top of the stack.
45     answer_stack.pop(); //pops out the answer at the top of the stack.
46     this -> answer = this -> answer - x; //performs the - operation and stores the
47     answer in this -> answer.
48     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
49 };
50
51 void Calculator::sub(double x, double y) // function that will subtract a number from a
52 previous answer.
53 {
54     stack<double> answer_stack;
55     answer_stack.push(this -> answer); //places the answer at the previous answer at
56     the top of the stack.
57     answer_stack.pop(); //pops out the answer at the top of the stack.
58     this -> answer = x - x; //performs the - operation and stores the answer in this ->
59     answer.
60     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
61 };
62
63 void Calculator::div(double x, double y) // function that will subtract a number from a
64 previous answer.
65 {
66     stack<double> answer_stack;
67     answer_stack.push(this -> answer); //places the answer at the previous answer at
68     the top of the stack.
69     answer_stack.pop(); //pops out the answer at the top of the stack.
70     this -> answer = x / y; //performs the / operation and stores the answer in this ->
71     answer.
72     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
73 };
74
75 void Calculator::mul(double x, double y) // function that will multiply a number from a
76 previous answer.
77 {
78     stack<double> answer_stack;
79     answer_stack.push(this -> answer); //places the answer at the previous answer at
80     the top of the stack.
81     answer_stack.pop(); //pops out the answer at the top of the stack.
82     this -> answer = x * y; //performs the * operation and stores the answer in this ->
83     answer.
84     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
85 };
86
87 void Calculator::mod(double x, double y) // function that will calculate the modulus of a
88 number from a previous answer.
89 {
90     stack<double> answer_stack;
91     answer_stack.push(this -> answer); //places the answer at the previous answer at
92     the top of the stack.
93     answer_stack.pop(); //pops out the answer at the top of the stack.
94     this -> answer = x % y; //performs the % operation and stores the answer in this ->
95     answer.
96     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
97 };
98
99 void Calculator::exp(double x, double y) // function that will calculate the exponential
100 of a number from a previous answer.
101 {
102     stack<double> answer_stack;
103     answer_stack.push(this -> answer); //places the answer at the previous answer at
104     the top of the stack.
105     answer_stack.pop(); //pops out the answer at the top of the stack.
106     this -> answer = exp(x * y); //performs the exp operation and stores the answer in
107     this -> answer.
108     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
109 };
110
111 void Calculator::log(double x, double y) // function that will calculate the logarithm of
112 a number from a previous answer.
113 {
114     stack<double> answer_stack;
115     answer_stack.push(this -> answer); //places the answer at the previous answer at
116     the top of the stack.
117     answer_stack.pop(); //pops out the answer at the top of the stack.
118     this -> answer = log(x * y); //performs the log operation and stores the answer in
119     this -> answer.
120     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
121 };
122
123 void Calculator::sqrt(double x, double y) // function that will calculate the square root
124 of a number from a previous answer.
125 {
126     stack<double> answer_stack;
127     answer_stack.push(this -> answer); //places the answer at the previous answer at
128     the top of the stack.
129     answer_stack.pop(); //pops out the answer at the top of the stack.
130     this -> answer = sqrt(x * y); //performs the sqrt operation and stores the answer in
131     this -> answer.
132     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
133 };
134
135 void Calculator::pow(double x, double y) // function that will calculate the power of a
136 number from a previous answer.
137 {
138     stack<double> answer_stack;
139     answer_stack.push(this -> answer); //places the answer at the previous answer at
140     the top of the stack.
141     answer_stack.pop(); //pops out the answer at the top of the stack.
142     this -> answer = pow(x * y, this -> answer); //performs the pow operation and stores
143     the answer in this -> answer.
144     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
145 };
146
147 void Calculator::sin(double x, double y) // function that will calculate the sine of a
148 number from a previous answer.
149 {
150     stack<double> answer_stack;
151     answer_stack.push(this -> answer); //places the answer at the previous answer at
152     the top of the stack.
153     answer_stack.pop(); //pops out the answer at the top of the stack.
154     this -> answer = sin(x * y); //performs the sin operation and stores the answer in
155     this -> answer.
156     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
157 };
158
159 void Calculator::cos(double x, double y) // function that will calculate the cosine of a
160 number from a previous answer.
161 {
162     stack<double> answer_stack;
163     answer_stack.push(this -> answer); //places the answer at the previous answer at
164     the top of the stack.
165     answer_stack.pop(); //pops out the answer at the top of the stack.
166     this -> answer = cos(x * y); //performs the cos operation and stores the answer in
167     this -> answer.
168     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
169 };
170
171 void Calculator::tan(double x, double y) // function that will calculate the tangent of a
172 number from a previous answer.
173 {
174     stack<double> answer_stack;
175     answer_stack.push(this -> answer); //places the answer at the previous answer at
176     the top of the stack.
177     answer_stack.pop(); //pops out the answer at the top of the stack.
178     this -> answer = tan(x * y); //performs the tan operation and stores the answer in
179     this -> answer.
180     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
181 };
182
183 void Calculator::cot(double x, double y) // function that will calculate the cotangent of
184 a number from a previous answer.
185 {
186     stack<double> answer_stack;
187     answer_stack.push(this -> answer); //places the answer at the previous answer at
188     the top of the stack.
189     answer_stack.pop(); //pops out the answer at the top of the stack.
190     this -> answer = 1 / tan(x * y); //performs the cot operation and stores the answer
191     in this -> answer.
192     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
193 };
194
195 void Calculator::sec(double x, double y) // function that will calculate the secant of a
196 number from a previous answer.
197 {
198     stack<double> answer_stack;
199     answer_stack.push(this -> answer); //places the answer at the previous answer at
200     the top of the stack.
201     answer_stack.pop(); //pops out the answer at the top of the stack.
202     this -> answer = 1 / cos(x * y); //performs the sec operation and stores the answer
203     in this -> answer.
204     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
205 };
206
207 void Calculator::csc(double x, double y) // function that will calculate the cosecant of
208 a number from a previous answer.
209 {
210     stack<double> answer_stack;
211     answer_stack.push(this -> answer); //places the answer at the previous answer at
212     the top of the stack.
213     answer_stack.pop(); //pops out the answer at the top of the stack.
214     this -> answer = 1 / sin(x * y); //performs the csc operation and stores the answer
215     in this -> answer.
216     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
217 };
218
219 void Calculator::log10(double x, double y) // function that will calculate the base 10
220 logarithm of a number from a previous answer.
221 {
222     stack<double> answer_stack;
223     answer_stack.push(this -> answer); //places the answer at the previous answer at
224     the top of the stack.
225     answer_stack.pop(); //pops out the answer at the top of the stack.
226     this -> answer = log10(x * y); //performs the log10 operation and stores the answer
227     in this -> answer.
228     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
229 };
230
231 void Calculator::log2(double x, double y) // function that will calculate the base 2
232 logarithm of a number from a previous answer.
233 {
234     stack<double> answer_stack;
235     answer_stack.push(this -> answer); //places the answer at the previous answer at
236     the top of the stack.
237     answer_stack.pop(); //pops out the answer at the top of the stack.
238     this -> answer = log2(x * y); //performs the log2 operation and stores the answer
239     in this -> answer.
240     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
241 };
242
243 void Calculator::logn(double x, double y, double n) // function that will calculate the
244 logarithm of a number from a previous answer.
245 {
246     stack<double> answer_stack;
247     answer_stack.push(this -> answer); //places the answer at the previous answer at
248     the top of the stack.
249     answer_stack.pop(); //pops out the answer at the top of the stack.
250     this -> answer = logn(x * y, n); //performs the logn operation and stores the answer
251     in this -> answer.
252     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
253 };
254
255 void Calculator::exp2(double x, double y) // function that will calculate the exponential
256 of a number from a previous answer.
257 {
258     stack<double> answer_stack;
259     answer_stack.push(this -> answer); //places the answer at the previous answer at
260     the top of the stack.
261     answer_stack.pop(); //pops out the answer at the top of the stack.
262     this -> answer = exp2(x * y); //performs the exp2 operation and stores the answer in
263     this -> answer.
264     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
265 };
266
267 void Calculator::exp10(double x, double y) // function that will calculate the exponential
268 of a number from a previous answer.
269 {
270     stack<double> answer_stack;
271     answer_stack.push(this -> answer); //places the answer at the previous answer at
272     the top of the stack.
273     answer_stack.pop(); //pops out the answer at the top of the stack.
274     this -> answer = exp10(x * y); //performs the exp10 operation and stores the answer
275     in this -> answer.
276     answer_stack.push(this-> answer);
```

```

58     answer_stack.pop(); //pops out the answer at the top of the stack.
59     this -> answer = x / y; //performs the - operation and stores the answer in this ->
    answer.
60     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
61 };
62
63 void Calculator::mult(double x, double y) // function that will subtract a number from
    a previous answer.
64 {
65     stack<double> answer_stack;
66     answer_stack.push(this -> answer); //places the answer at the previous answer at
    the top of the stack.
67     answer_stack.pop(); //pops out the answer at the top of the stack.
68     this -> answer = x * y; //performs the - operation and stores the answer in this ->
    answer.
69     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
70 };
71
72 void Calculator::mult(double x) // function that will multiply a number to a previous
    answer.
73 {
74     stack<double> answer_stack;
75     answer_stack.push(this -> answer); //places the answer at the previous answer at
    the top of the stack.
76     answer_stack.pop(); //pops out the answer at the top of the stack.
77     this -> answer = this -> answer * x; //performs the * operation and stores the
    answer in this -> answer.
78     answer_stack.push(this-> answer); //pushes this -> answer back into the stack.
79 };
80
81 void Calculator::clear() //function that clears the inputs.
82 {
83     answer_stack.push(0); //pushes this -> answer into the stack.
84     cout << "CLEARED ANSWER" << endl;
85 };
86
87 void Calculator::enter(double x) //function that enters a number to the registry.
88 {
89     answer_stack.push(x); //places this -> answer at the top of the stack.
90 };
91
92 void Calculator::div(double x) // function that divides the answer and the number x.
93 {
94     stack<double> answer_stack;
95     answer_stack.push(this -> answer); //pushes this -> answer into the stack.
96     answer_stack.pop(); //pops out the answer at the top of the stack.
97     this -> answer = (this -> answer) / (x); //performs the / operator and stores the
    answer in this -> answer.
98     answer_stack.push(this-> answer); // pushes this -> answer into the stack.
99 };
100
101 void Calculator:: prt() //print function.
102 {
103     cout << "ANSWER: " << answer_stack.top() << endl;
104     cout << "\n";
105 };
106
107
108 void Calculator:: mult()
109 {
110     double a = 0;
111     double b = 0;
112     a = answer_stack.top(); // sets a = to the top of the stack.
113     answer_stack.pop();//pops out a;
114     b = answer_stack.top(); // sets b = to the top of the stack.
115     answer_stack.pop(); //pops out the top of the stack.
116     answer_stack.push(a*b); // pushes the expression a*b.;
117     cout << "\n" << b << " * " << a << endl;
118 };

```

```

119
120 void Calculator:: add()
121 {
122     double a = 0;
123     double b = 0;
124     a = answer_stack.top(); // sets a = to the top of the stack.
125     answer_stack.pop(); //pops out a;
126     b = answer_stack.top(); // sets b = to the top of the stack.
127     answer_stack.push(a+b); // pushes the expression a+b.
128     cout << "\n" << b << " + " << a << endl;
129 };
130
131 void Calculator:: sub()
132 {
133     double a = 0;
134     double b = 0;
135     a = answer_stack.top(); // sets a = to the top of the stack.
136     answer_stack.pop(); //pops out a;
137     b = answer_stack.top(); // sets b = to the top of the stack.
138     answer_stack.push(b-a); // pushes the expression b-a.
139     cout << "\n" << b << " - " << a << endl;
140 };
141
142 void Calculator:: div()
143 {
144     double a = 0;
145     double b = 0;
146     a = answer_stack.top(); // sets a = to the top of the stack.
147     answer_stack.pop(); //pops out a;
148     b = answer_stack.top(); // sets b = to the top of the stack.
149     answer_stack.pop(); //pops out b;
150     answer_stack.push(b/a); // pushes the expression b/a.
151     cout << "\n" << b << " / " << a << endl;
152     if (a == 0) // condition where if a == 0, it will output an error message, since
        dividing by zero is not allowed.
153     {
154         cout << "Error." << endl;
155         exit(1);
156     }
157 };

```