

```

1 #include <iostream>
2 #include "operand.h"
3 #include "stack.h"
4 #include <math.h>
5
6 using namespace std;
7
8 #ifndef expr_float_h
9 #define expr_float_h
10
11 int expr_float(){
12
13     cout << "Enter a Numeric Expression ( May include integers,(),*,/,%,^,-,+ ).";
14     while(true){
15
16         int MAXLEN(200);
17         char* raw(new char[MAXLEN]);           // creating a char Array to
18         cout << "\n[float]> ";                 // store user input
19
20         cin.getline( raw , MAXLEN-1 , '\n' ); // taking input from user
21         if(!strlen(raw,MAXLEN)){              // Quit if no input
22             return 0;
23         }
24
25         stack<double> postfix;                // creating a double stack
26         stack<operand> opd;                   // creating an operand stack
27
28         opd.push_back(operand(-1,'('));       // Pushing an opening bracket
29
30         bool error(0);                        // an error flag
31
32         /* Following loop converts Expression to
33          * postfix and calculates it: */
34         for( int i=0, iflag(0), dflag(0); i<=strlen(raw) ; ++i ){
35
36             //1. For a Literal
37             if((int)(raw[i])-48 >= 0 && (int)(raw[i])-48 <= 9 || raw[i]=='.' ){
38                 if(iflag && !dflag){
39                     if(!(raw[i]=='.')){
40                         double a =(float)((int)(raw[i])-48) + postfix.top() * 10;
41                         postfix.pop_out();
42                         dflag=0;
43                         postfix.push_back(a);
44                     }
45                     else if(raw[i]=='.' ){
46                         dflag=1;
47                     }
48                 }
49             }
50             else if(iflag && dflag){
51                 if(!(raw[i]=='.')){
52                     double a =(float)((int)(raw[i])-48)/(pow(10,dflag++))
53                     + postfix.top();
54                     postfix.pop_out();
55                     postfix.push_back(a);
56                 }
57                 else if(raw[i]=='.' ){
58                     error=1;
59                     cout << "-> Invalid String" << endl;
60                     break;
61                 }
62             }
63             else if(!iflag && !dflag){
64                 iflag=1;
65                 int a =(int)(raw[i])-48;
66                 postfix.push_back(a);
67             }
68             else if(!iflag && dflag){

```

```

69         error=1;
70         cout << "-> Invalid String" << endl;
71         break;
72     }
73 }
74
75 //2. For an Operand
76 else if(raw[i] == '(' || raw[i] == ')' ||
77         raw[i] == '*' || raw[i] == '/' ||
78         raw[i] == '%' || raw[i] == '-' ||
79         raw[i] == '+' || raw[i] == '^' ||
80         raw[i] == ' ' || raw[i] == '\\0'){
81     iflag=0;dfalg=0;
82     int poco;
83     switch(raw[i]){
84         case '+':case '-':poco=1;break;
85         case '*':case '/':case '%':poco=2;break;
86         case '^':poco=3;break;
87         case ')':poco=-2;break;
88         case '(':poco=-1;break;
89         default: poco=0;break;
90     }
91     operand dob(poco,raw[i]);
92     if( (dob > 0 && dob >= opd.top()) || dob == -1 ){
93         opd.push_back(dob);
94     }
95     else if( dob > 0 && dob < opd.top()){
96         operand popped(opd.top().p, opd.top().o);
97         while(dob < popped){
98             opd.pop_out();
99             double b = postfix.get(postfix.size());
100             double a = postfix.get(postfix.size()-1);
101             double r(1);
102             postfix.pop_out();
103             postfix.pop_out();
104             switch(popped.o){
105                 case '+':r=a+b;break;
106                 case '-':r=a-b;break;
107                 case '*':r=a*b;break;
108                 case '/':r=a/b;break;
109                 case '%':r=fmod(a,b);break;
110                 case '^':r=pow(a,b);break;
111                 default: r=a+b;break;
112             }
113             postfix.push_back(r);
114             popped(opd.top().p, opd.top().o);
115         }
116         opd.push_back(dob);
117     }
118     else if(dob == -2 || dob.o == '\\0'){
119         operand popped(opd.top().p, opd.top().o);
120         while(popped != -1){
121             opd.pop_out();
122             double b = postfix.get(postfix.size());
123             double a = postfix.get(postfix.size()-1);
124             double r(1);
125             postfix.pop_out();
126             postfix.pop_out();
127             switch(popped.o){
128                 case '+':r=a+b;break;
129                 case '-':r=a-b;break;
130                 case '*':r=a*b;break;
131                 case '/':r=a/b;break;
132                 case '%':r=fmod(a,b);break;
133                 case '^':r=pow(a,b);break;
134                 default: r=a+b;break;
135             }
136             postfix.push_back(r);

```

```

137             popped(opd.top().p, opd.top().o);
138         }
139         opd.pop_out();
140     }
141
142     }
143     //else
144     else{
145         error=1;
146         cout << "-> Invalid String" << endl;
147         break;
148     }
149 }
150 if(!error)
151     cout << "=> Answer: " << postfix.top();
152 postfix.clear();
153 opd.clear();
154 cout << endl ;
155
156 }
157 return 0;
158 }
159
160 #endif

```