

A Simple Calculator

12112609 刘一桢

1. Code Explanation

1.1 Struct BigNum

1.1.1 BigNum is struct to store big num having basic parameters.

```
28 struct BigNum{           //a struct to store big num
29     int size;             //size of a[N]
30     bool pos;             //if the value is positive
31     int a[N];             //store value bit by bit like 0 1 2 3 equal to 3210
32     ll e;
```

1.1.2 Defined function optimize can adjust size and a[n] to delete the pre and suf zeros. **And if it is all zeros, which means its value is 0, we just manipulate it manually.**

```
55 void optimize(){ //adjust size a[n] to delete the pre and suf zero
56     while(a[size- 1]== 0){
57         size--;
58         if(size== 0){
59             break;
60         }
61     }
62     int num= 0;
63     while(a[num]== 0&& num< size){
64         num++;
65     }
66     For(i, num, size- 1){
67         a[i- num]= a[i];
68     }
69     e+= num;
70     size-= num;
71
72     if(size== 0){ //avoid there is null in a[n]
73         pos= 1;
74         a[size++]= 0;
75         e= 0;
76     }
```

1.1.3 Reloaded operator * which return a new

BigNum equal to BigNum1 and BigNum2.

```
BigNum operator* (const BigNum o){ //define * operation
    BigNum ans;
    ans.size= size+ o.size;
    ans.pos= !(pos^ o.pos);
    ans.e= e+ o.e;
    For(i, 0, ans.size- 1){
        ans.a[i]= 0;
    }
    For(i, 0, size- 1){
        For(j, 0, o.size- 1){
            ans.a[i+ j]+= a[i]* o.a[j];
        }
    }
    For(i, 1, ans.size- 1){
        ans.a[i]+= ans.a[i- 1]/ 10;
        ans.a[i- 1]%= 10;
    }
    ans.optimize();
    return ans;
}
```

1.1.4 Defined function read can initialize the parameters to equal to given string s (the input must be valid).

```
79 void read(string s){ //initialize BigNum with string s (input must be valid)
80     //find the length of s
81     int len= s.length();
82     //find the 'e''s position
83     int ePos= -1;
84     int n= s.length();
85     For(i, 0, n- 1){
86         if(s[i]== 'e'){
87             ePos= i;
88         }
89     }
90     //find if it is positive or not
91     pos= s[0]!='-';
92     //determine e from give string's suffix if having
93     if(ePos== -1){
94         e= 0;
95     }else {
96         e= readString(s.substr(ePos+ 1, len- ePos- 1) );
97     }
```

```

97     }
98     //determine size a[n], adjust e form string's prefix
99     len= (ePos== -1)? len: ePos; //omit the exxx part
100    size= 0;
101    int dotPos= -1;
102    For(i, 0, len- 1){
103        if(s[i]== '-'){
104            continue;
105        }
106        if(s[i]== '.'){
107            dotPos= i;
108            continue;
109        }
110        a[size++]= (s[i]- '0');
111    }
112    if(dotPos!= -1){//update e
113        e-= len- 1- dotPos;
114    }

```

```

115    For(i, 1, size/ 2){ //reverse a[n]
116        swap(a[i- 1], a[size- i]);
117    }
118
119    optimize();//delete pre and suf zero
120 }

```

1.1.5 Defined function print can print the value in the format like 213e12321 or just 21300 or or 0.0123. **Only when the normal print is too long, we print it with 'e' format like 21300000000000 into 213e10, also like 0.00000000000213 into 213e-13.**

```

122 void print(){ //print the value
123     if(!pos){
124         cout<< '-';
125     }
126
127     if(e>= 0&& e<= 5){ //like 123000
128         Ford(i, size- 1, 0){
129             cout<< a[i];
130         }
131         For(i, 1, e){
132             cout<< 0;
133         }
134     }else{
135         if(-e>= size&& -e<= size+ 5){ //like 0.0123
136             cout<< "0.";
137             For(i, 1, -e- size){
138                 cout<< 0;
139             }
140             Ford(i, size- 1, 0){
141                 cout<< a[i];
142             }
143         }else if(-e< size&& -e> 0){ //like 1.23
144             Ford(i, size- 1, -e){
145                 cout<< a[i];
146             }
147             cout<< '.';
148             Ford(i, -e- 1, 0){
149                 cout<< a[i];
150             }
151         }else{ //like 123e123
152             Ford(i, size- 1, 0){
153                 cout<< a[i];
154             }
155             cout<< 'e'<< e;
156         }
157     }
158 }
159 };

```

1.1.5 A functional function readString, just used to assign value to e, can transform a string-like long long integer into long long integer type.

```

11 ll readString(string s){ //turn string into Long Long Integer
12     ll n= 0, b= 1;
13     int len= s.length();
14     int i= 0;
15     char c= s[0];
16     while(!isdigit(c) ){
17         if(c== '-') b= -1;
18         c= s[++i];
19     }
20     while(isdigit(c) ){
21         n= n* 10;
22         n+= c- 48;
23         c= s[++i];
24     }
25     return n* b;
26 }

```

1.2 Valid judgements

1.2.1 Defined function isPureNum can return true **IFF** all char of given string are digits.

```

161 bool isPureNum(string s){ //return true IFF all char of given string are digits
162     int len= s.length();
163     //boundary1 length= 0;
164     if(len== 0){
165         return false;
166     }
167     //boundary2 all is digit
168     int flag= true;
169     For(i, 0, len- 1){
170         flag&= isdigit(s[i]);
171     }
172     if(flag){
173         return true;
174     }else{
175         return false;
176     }
177 }

```

1.2.2 Defined function isNum can return true **IFF** string is in '-123' or '-1.23'-like format.

```

179 bool isNum(string s){ //return true IFF string is in '-123' or '-1.23'-like format
180     if(s[0]== '-'){ //omit the '-' in 0 index;
181         s= s.substr(1, s.length()- 1);
182     }
183
184     int len= s.length();
185     For(i, 0, len- 1){ //if there is one '.', then true IFF two parts are purenum(all is digit)
186         if(s[i]== '.'){
187             return isPureNum(s.substr(0, i) )&& isPureNum(s.substr(i+ 1, len- i) );
188         }
189     }
190
191     //no '.', then true IFF this is purenum(all is digit)
192     return isPureNum(s);
193 }

```

1.2.3 Defined function isValid can return true **IFF** string is in '-123' or '-1.23' or '-1.23e-12'-like format.

```
202 bool isValid(string s){ //check if s is valid
203     int len= s.length();
204     int eNum= 0;
205     int ePos= -1;
206     For(i, 0, len- 1){
207         if(s[i]== 'e'){ //if number of 'e' >0 return false;
208             eNum++;
209             ePos= i;
210             if(eNum> 1){
211                 return false;
212             }
213         }
214     }
215
216     if(eNum== 0){ //if no 'e' return if s is num, if not return if two parts are nums
217         return isNum(s);
218     }else{
219         return isNum(s.substr(0, ePos) )&& isIntNum(s.substr(ePos+ 1, len- ePos- 1) );
220     }
221 }
```

1.2.4 Defined function isIntNum can return true **IFF** string is in '-123'-like integer format.

```
195 bool isIntNum(string s){//return true IFF string is in '-123'-like format
196     if(s[0]== '-'){ //omit the '-' in 0 index;
197         s= s.substr(1, s.length()- 1);
198     }
199     return isPureNum(s);
200 }
```

1.3 Main

For two string, check if it is valid. Then transform two string into BigNum. Then multiply this two BigNum and print the result BigNum.

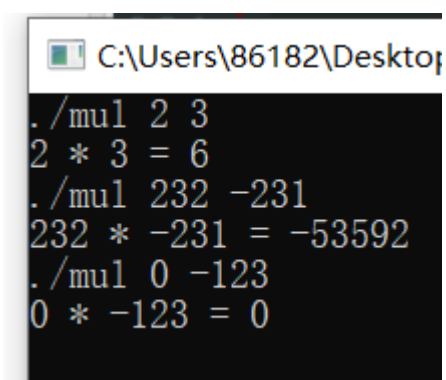
```

223 int main(int argc, char* argv[])
224 {
225     string s1, s2;
226     if(argc < 3){
227         cout<< "please input two string!"<< endl;
228         return 0;
229     }
230     s1= argv[1];
231     s2= argv[2];
232     if(!isvalid(s1)|| !isvalid(s2) ){ //determine if it is valid num
233         cout<< "The input cannot be interpret as numbers!"<< endl;
234         return 0;
235     }
236     BigNum bn1, bn2, bn3;
237     bn1.read(s1);
238     bn2.read(s2);
239     bn3= bn1* bn2;
240     cout<< s1;
241     cout<< " * ";
242     cout<< s2;
243     cout<< " = ";
244     bn3.print();
245     cout<< endl;
246
247     return 0;
248 }

```

2. Requirements

2.1 When you run the program as follows, it will output the expression and the result. The two numbers should be input through the command line arguments. If the two numbers are integers, the program will multiply them in integer format.



```

C:\Users\86182\Desktop> ./mul 2 3
2 * 3 = 6
C:\Users\86182\Desktop> ./mul 232 -231
232 * -231 = -53592
C:\Users\86182\Desktop> ./mul 0 -123
0 * -123 = 0

```

2.2 If the input contains some non-integer numbers, the program will try to interpret the input as floatingpoint numbers.

(it can also identify 0002 wisely)

```
C:\Users\86182\Desktop\sundry workshop\source.  
./mul 3.1416 2  
3.1416 * 2 = 6.2832  
./mul 3.1415 2.0e-2  
3.1415 * 2.0e-2 = 0.06283  
./mul 2e123211 3.123e-123211  
2e123211 * 3.123e-123211 = 6.246  
./mul 0002e0000000 00.03e123  
0002e0000000 * 00.03e123 = 6e121
```

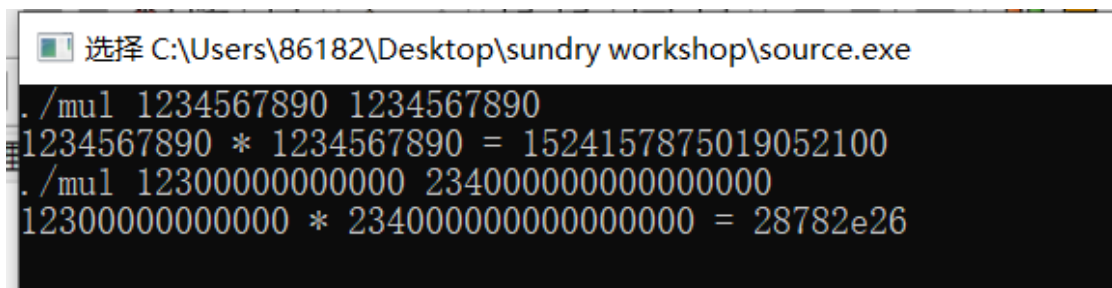
2.3 It can tell that the input is not a number.

return true **IFF in format (-)123(.)123(e(-)123).**

```
C:\Users\86182\Desktop\sundry workshop\source.exe  
./mul a 2  
The input cannot be interpret as numbers!  
./mul 12e 2  
The input cannot be interpret as numbers!  
./mul -11-2e-1 2  
The input cannot be interpret as numbers!  
./mul -1.e1 2  
The input cannot be interpret as numbers!  
./mul 1.-1e1 2  
The input cannot be interpret as numbers!  
./mul 1.1.1e1 2  
The input cannot be interpret as numbers!  
./mul 1e1.1 2  
The input cannot be interpret as numbers!
```

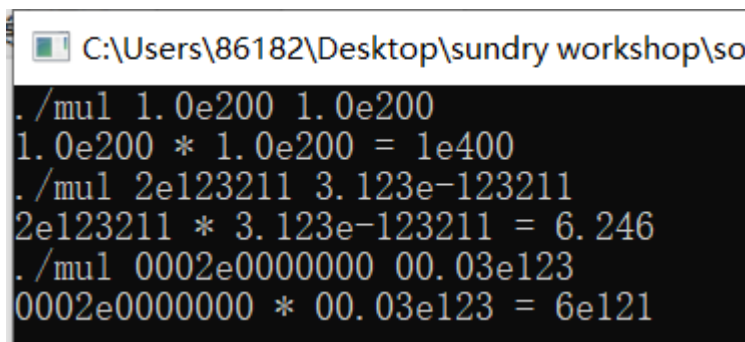
2.4 If you input some big integers, what will happen?
Please describe some possible solutions, and try to

implement them.



```
选择 C:\Users\86182\Desktop\sundry workshop\source.exe
./mul 1234567890 1234567890
1234567890 * 1234567890 = 1524157875019052100
./mul 1230000000000000 23400000000000000
1230000000000000 * 23400000000000000 = 28782e26
```

2.5 If you input some big floating-point numbers, what will happen? Please describe some possible solutions, and try to implement them.



```
C:\Users\86182\Desktop\sundry workshop\so
./mul 1.0e200 1.0e200
1.0e200 * 1.0e200 = 1e400
./mul 2e123211 3.123e-123211
2e123211 * 3.123e-123211 = 6.246
./mul 0002e0000000 00.03e123
0002e0000000 * 00.03e123 = 6e121
```

2.6 Some others which can improve the program. like the highlight of 2.2 and 2.3.

We can enlarge the parameter range in BigNum. Like the size of a[n]. also we can use a array to store the number of e. but in practice is unnecessary.

In the main function. We can check if there is two or more string. If not, print "please input two string!".