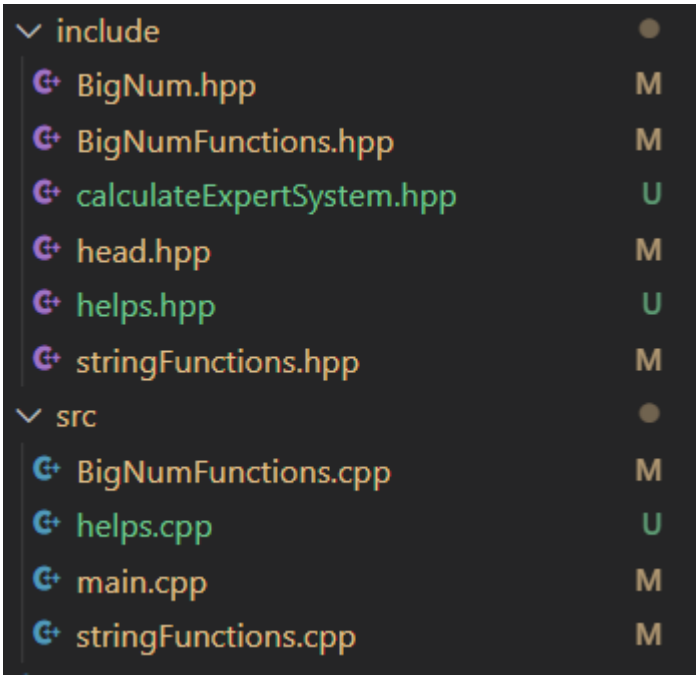
A Simple Calculator

12112609 刘一郴

1. Code Explanation

1.1 Struct Overview



Helps.cpp is to output some instructional statement to help user;

BigNum is a structure to store big number;

Head.cpp is some typedef and define;

StringFunctions.cpp provide some useful **tool** to manage string;

BigNumFunctions.cpp provide some useful **tool** related to BigNum;

calculateExperSystem.cpp in a structure

calculateExperSystem which **work as ExperSystem** to receive statements and queries and react to answer the value of expression or give some warning.

1.2 BigNum.hpp

1.2.1 Some parameters

N = 1e5; BigNum can store big number like 123e123 , -123e-123

```
8  struct BigNum{    //a struct to store big num
9      int size;     //size of a[N]
10     bool pos;     //if the value is positive
11     int a[N];     //store value bit by bit like 0 1 2 3 equal to 3210 ; index from 0
12     ll e;         //store the number of e
13
```

1.2.2 reload the basical operator

Reload the basical operator for BigNum to **ease** following coding work

```
32  BigNum operator / (BigNum o){//define / operation···
76
77  BigNum operator - (BigNum o){//define - operation···
81
82  bool operator >= (BigNum o){//define >= operation···
85
86  bool operator == (BigNum o){ //define == operation···
89
90  bool operator > (BigNum o){ //define > operation···
100
101 BigNum operator + (BigNum o){ //define + operation···
150
151 BigNum operator * (const BigNum o){  //define * operation···
```

1.2.3 Some useful tool

In which sqrt() is by **binary search algorithm**

```
162 >   ll toLongLong(){ //change into LongLong···
171
172 >   void sqrt(){//define sqrt operation···
212
213 >   void carryBits(){  // carry the bits···
223
224 >   void approximate(ll n){ //approximate this BigNum into E(n)···
244
245 >   bool absCompare(BigNum * po){ //return abs(this) > abs(o)···
265
266 >   void addSufZerofromE(ll n){ //pick n zeros from e, append n zeros to a[];···
284
285 >   void optimizeZeros(){ //adjust size and a[n] to delete the pre and suf zeros···
308
```

1.3 BigNumFunctions.cpp

1.3.1 BigNum Function

```
3 > BigNum sqrt(BigNum o){ //define sqrt operation···
7
8 > BigNum pow(BigNum o, ll n){ //define pow operation for BigNum^(longlong)···
30
31 > pair<bool , string> takeOffBigNumFromString(string s , BigNum * bn){ //take of the first BigNum from string ; return true and change
61
62 > BigNum abs(BigNum o){ //define the abs operation···
```

In which pow is in **quick pow method**

1.3.2 Function takeOffBigNumFromString

take off the first BigNum from string ; return true and change the string **IFF** success; bn is the BigNum takenoff from s.

```cpp
pair<bool , string> takeOffBigNumFromString(string s , BigNum * bn){ //take off the first BigNum from string ; return true and chang
    s = removePreSufSpace(s);
    int len = s.length();
    int pos = len;
    For(i, 0, len - 1){
        char ch = s.at(i);
        if(ch == '-' && i == 0){
            continue;
        }
        if(ch == '-' && i > 0 && s.at(i - 1) == 'e' ){
            continue;
        }
        if(ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '(' || ch == ')' || ch == ','){
            pos = i;
            break;
        }
    }

    string bigNumString = s.substr(0, pos - 0);
    bigNumString = removePreSufSpace(bigNumString);

    pair<bool , string> ans;

    if(!isvalid(bigNumString) ){    //if bigNumString not valid
        ans.first = false;
        ans.second = s;
        return ans;
    }

    ans.first = true;
    ans.second = s.substr(pos, len- pos);
    *bn = BigNum(bigNumString);
    return ans;
}
```

1.4 calculateExpertSystem.cpp

1.4.1 parameters

```cpp
struct calculateExpertSystem{
    map< string , BigNum > vMap;  //variableMap
    map< string , int >    fMap;  //functionMap ; int represents the parameters number of funciton
    vector<Node > v;              //vector to store suffix expression
```

1.4.2 initialize

Insert all function name in to fMap

```cpp
    calculateExpertSystem(){   //initial fMap
        addFunction("sqrt" , 1);
        addFunction("pow" , 2);
        addFunction("abs" , 1);
    }
```

1.4.3 tool function

To add and search vMap and fMap

```
306 >   void addVariable(string variableName , string value ){  //add variable into variableMap···
309
310 >   void addVariable(string variableName , BigNum bn ){  //add variable into variableMap···
313
314 >   bool existVariable(string variableName){ //check if variable exists···
317
318 >   BigNum VariableValue(string variableName){//return the value of variable···
323
324 >   void addFunction(string functionName , int parameterNum ){ //add function into functionMap···
327
328 >   bool existFunction(string functionName){ //check if function exists···
331
332 >   int parameterNumberOfFunction(string functionName){ //return the parameter number of a funciton···
```

1.4.4 string Function

take of the first Function from string ; return true and change the string IFF success; bn is the value of the takenoff Function.

take of the first Variable from string ; return true and change the string **IFF** success; bn is the value of the takenoff Variable.

```
156 >   pair<bool , string> takeOffFunctionFromString(string s , BigNum * bn){ //take of the first Function from string ; return true an
242
243 >   pair<bool , string> takeOffVariableFromString(string s , BigNum * bn){ //take of the first Variable from string ; return true an
```

1.4.5 statement handle function

```
129 >   pair<bool , BigNum> ValueOfExpression(string s){ //return True and the value of expression s; if expression invalid return False
143 >   void handleStatement(string s){ //handle the  input statement like calculate or declare a variable···
104 >   bool handleVariableDeclaration(string s){ ///handle the  input statement which declare a variable···
```

1.4.6 calculate function

for an expression , turn it in to **suffix expression** firtly , then calculate the suffix expression

```
39 >    bool processToSuffix(string s){ //nifix expression into postfix expression···
278 >   pair<bool , BigNum> calc() {       //calculate the suffix expression ; if success return ture···
```

1.5 head.hpp

Some typedef and definition to **ease** following

coding work

```cpp
// some parameters and includes
#ifndef _HEAD_HPP
#define _HEAD_HPP

#include<iostream>
#include<string>
using namespace std;
typedef long long ll;
#define ri register int
#define For(i, a, b) for(ri i= a;i<= b;i++)
#define Ford(i, a, b) for(ri i= a;i>= b;i--)
#define N 100000      //maximum size of BigNum a[N]
#define DivisionPrecision -4  //the precison of '/' operation
#define SqrtPrecision -4  //the precison of sqrt operation
#define ApproximationPrecision -1 //approximationPrecision

#endif
```

1.6 helps.cpp

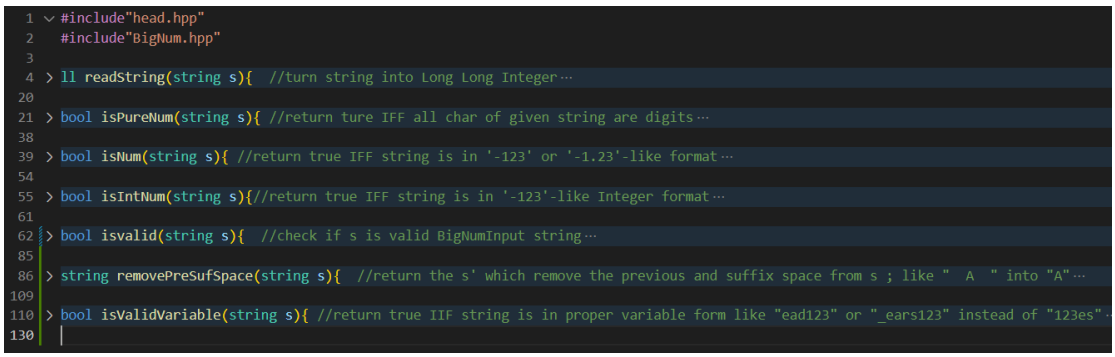
Helps.cpp is to output some instructional statement to help user;

```cpp
// the help list
#include"head.hpp"

void printHelpList(){ // print help list
    cout<< "-------------------------------------------------------------------------"<< endl;
    cout<< "Variable declaration format : variableName = expression ; like \" a = 3 + (3 * 4) \" , \" a = 3 * b \""<< endl;
    cout<< "'+' '-' '*' '-' '(' ')' can be used"<< endl;
    cout<< "you can declare a number like \"12e-123\""<< endl;
    cout<< "Input \"help\" to check help list"<<endl;
    cout<< "Input \"exit\" to exit calculator"<<endl;
    cout<< "-------------------------------------------------------------------------"<< endl;
    cout<< "Fllowing functions can be used"<< endl;
    cout<< "sqrt(x)      ------- the sqrt of x     ; x must be positive number, or warning!"<< endl;
    cout<< "pow(a, b)    ------- the power : a ^ b ; a must NOT be 0, or Warning!"<< endl;
    cout<< "abs(a)       ------- the absolute value of a"<< endl;
    cout<< "-------------------------------------------------------------------------"<< endl;
}
```

1.7 stringFunctions.cpp

Includes all string management tool functions to **ease** string operation



3.Requirements

3.1 When you run your program and input an express in a line as follows, it can output the correct results. The operator precedence (order of operations) should be correct.

```
2 + 3
5
5 + 2 * 3
11
3 + 2 / 3 * 6 - 1
6
```

3.2 Use parentheses to enforce the priorities

```
(5 + 2) * 3
21
( (1 + 1 ) / 2 * 4) - 2
2
```

3.3 Variables can be defined as follows

```
x = 3
y = 6
z = (x + y) * 2
x + y * 2 - z
-3
```

```
_x_xx12 = 1
_y_xx12 = 2 * _x_xx12
_y_xx12
2
```

3.4 Some math functions can be supported

Sqrt(x) pow(a, b) abs(x)

```
a = 4
sqrt(a) + 1
3
sqrt(3.4)
1.8
pow(a, 2)
64
abs( -3.4)
3.4
sqrt(abs(4))
2
```

3.5 It can support arbitrary precision.

```
a = 1e-12
b = 123e0
a + b
123.000000000001
99999999999999999.22222222222 + 1.0
100000000000000000.22222222222
```

3.6 More features can be found in the calculator BC in Unix-like systems. You can visit this page for more information.

Command Line Options

`bc` takes the following options from the command line:

`-h`, `--help`
: Print the usage and exit.

`-l`, `--mathlib`
: Define the standard math library.

`-w`, `--warn`
: Give warnings for extensions to POSIX `bc`.

`-s`, `--standard`
: Process exactly the POSIX `bc` language.

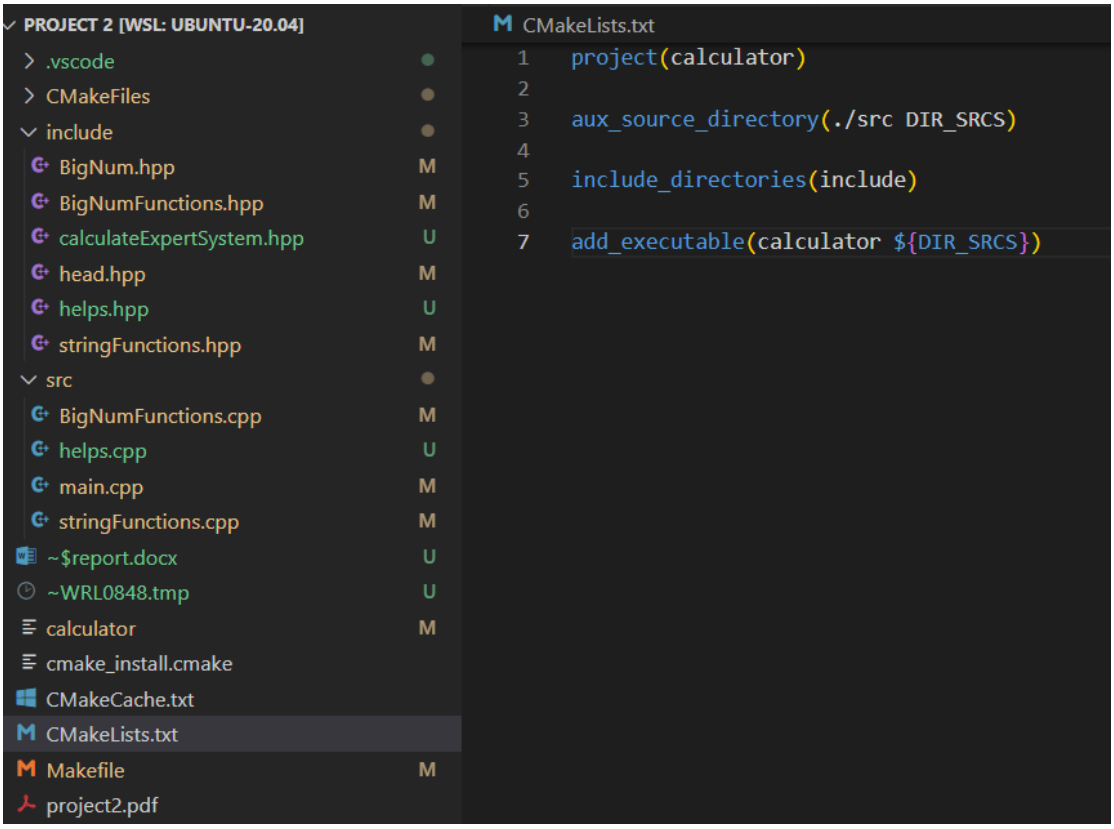
`-q`, `--quiet`
: Do not print the normal GNU `bc` welcome.

`-v`, `--version`
: Print the version number and copyright and quit.

Like BC we also have help command and exit command

```
help
------------------------------------------------------------------------
Variable declaration format : variableName = expression ; like " a = 3 + (3 * 4) " , " a = 3 * b
'+' '-' '*' '-' '(' ')' can be used
you can declare a number like "12e-123"
Input "help" to check help list
Input "exit" to exit calculator
------------------------------------------------------------------------
Fllowing functions can be used
sqrt(x)      ------- the sqrt of x     ; x must be positive number, or warning!
pow(a, b)    ------- the power : a ^ b ; a must NOT be 0, or Warning!
abs(a)       ------- the absolute value of a
------------------------------------------------------------------------
exit
cunlidaniang@LAPTOP-J1V8P7F0:/mnt/d/workshops/coding workshop/c++ workshop/project 2$
```

3.7 Please use CMake to manage the source files if there are several.



3.8 GitHub

https://github.com/cunlidaniang/coding-workshop/tree/master/c%2B%2B%20workshop/project%202

4. HighLight

4.1 can calculate any complicated expression

```
a = sqrt(4)
sqrt(a) / pow(a , abs(a) )
0.17493267059326171875
```

5.Need to improve

Easy to cause segment fault if recursion depth is too high

Cause by the memory of stack in linux is too low (stack overflow)

```
sqrt( sqrt(4) )
Segmentation fault
```