CS205 C/C++ Programming - Project 2

Name: 钟元吉(Zhong Yuanji)

SID: 12012613

Part 1 - Analysis

> This project to implement a much better calculator than that in the last project, which can support addition, substaction, prodction, division and other math function, also support brackets and assignment statement.

本次项目在前一个项目的基础上，还需实现加、减、除、余数、乘方、阶乘、括号、取整、三角函数、对数等数学计算，以及参数赋值与调用，其中需要考虑的点更多了。在设计这样的计算器程序时，我们需要考虑以下问题：

1. 检查使用者输入的字符串是否合理
2. 如何分析得到的字符串
3. 如何进行除法与乘方的高精度计算
4. 如何将错误位置尽可能准确地告知使用者
5. 如何解决在调用用户自定义变量中错误选择的问题

1. 检查使用者输入的字符串是否合理

考虑到使用者可能错误地输入了无法计算的参数，当我们在进行算式计算时，并不是直接进行读取和计算，而是进行初步的错误检查。首先我们会检查输入的字符串是否为空串或者空格，由于使用者在任意位置输入的空格数量未知，并且为了不增加后续计算的复杂程度，**先将输入字符串中的空格去除。**
C++中字符数组与内存是对应的，所以并不能直接删除空格对应的位置，或者说不能单独释放空格所在的内存。因此我们的思路是用空格的后一个字符代替空格所在的字符，并将后续的字符依次向前移动，从而达到去除空格的目的。

接下来的错误检查将会按照三个方向进行：输入格式错误、数字输入错误、计算错误。

错误类型			
格式错误	1. 赋值变量名错误	2. 没有输入	3. 以等号开始
	4. 存在不合理的字符	5. 左右括号数量不同	6. 数字与操作符数量错误
	7. 参数与数字相连	8. 操作符与操作符相连	8*. 以操作符开始
数字错误	9. 存在至少2个小数点	10. 指数上有小数点	11. 存在至少2个指数符号e
计算错误	12. 除数为零	13. 存在无穷大或非实数	14. 操作符错误

其中赋值变量名的要求为：所有字母只能为数字、字母和下划线，变量的首字母不能是数字，否则会与数字乘上变量冲突，变量名不区分大小写。

存在无穷大或非实数为什么要告知使用者呢？因为例如使用者输入：

```
Input: asin(1.1)
```

此时，我们知道函数`arc sin(x)`在大于1的时候没有定义，因此该表达式会得到`nan`，其实这是一种错误的运算，应当告知使用者。类似的问题还有`log(-1)`是负无穷等等。

```
Please input the expression in the next line: (quit:q)
asin(1.1)
asin(1.1)
^~~~~~  ("a" is invalid)
There might be inf or nan in the expression. Please try again.

Please input the expression in the next line: (quit:q)
log(-1)
log(-1)
^~~~~~  ("l" is invalid)
There might be inf or nan in the expression. Please try again.
```

2. 如何分析得到的字符串

由于得到的字符串中可能出现较多的可能，就算是正确的标准的输入，由于字母、数字与操作符之间会有很多中不一样的组合，因此我们应该在遍历字符串的时候，时刻记录此时的状态。

我们将**当前的状态分为三种类型**：**左侧不是数字或变量、左侧是数字但未结束、左侧是数字但已经结束。**

我们规定：

- 左侧不是数字或变量时，此时左侧为操作符，因此后面只能是数字、变量或函数值，不能是操作符，否则会出现两个操作符相连无法判断的情况。
- 左侧是数字但未结束时，后面可以是任何情况；如果是变量或函数值时，认为是**数字与变量(函数)相乘，但没有写乘号**，因此会自动向操作符数组中补充一个乘号；如果后面不是数字(包括指数符号e与小数点)，则读取前面的数据并记录与数值数组中。
- 左侧是数字但已经结束时，此时左侧为数字、变量或函数值，后面不能为数字(包括小数点，e作为自然对数底数)；如果后面是变量或函数值时，认为是**数字与变量(函数)相乘，但没有写乘号**，因此会自动向操作符数组中补充一个乘号。
- 函数名认为是一种操作符，但函数值认为是变量（上面的变量包括常量**pi和e，不区分大小写**）。

在进行字符串识别时，我们将从以下顺序对比检测：数字、用户定义的变量中较长的、用户定义的变量中较短的、函数名、操作符(减号单独考虑)、常数pi、常数e与指数符号e的区分、减号与负号的区分、小数点、其他错误字符。

在识别分析字符串后即可进行计算(见Part 4问题1)。

3. 如何进行除法与乘方的高精度计算

由于我们需要高精度的浮点数计算，因此我们需要构建整数结构体和浮点数结构体进行逐位存储与运算。考虑到**除以一个数等于乘这个数的倒数**，但是这样计算并没有解决除法问题，仍然需要计算倒数，而且还进行了乘法运算，反而会加大计算时间。因此我们考虑**参考除法的竖式运算**，逐位进行计算，通过两者比较大小与做减法，得到减法次数即为该位结果，以实现除法的高精度计算。

计算除法中，我们会面临一个问题：某些结果可能出现无穷小数的情况，如：1/3，但我们不可能将无穷的结果打印出来，因此程序中除法将计算200位有效数字，避免输出的结果过多的情况。

对于乘方计算，如果指数为整数，我们可以通过逐次累乘的方式进行计算，但是这样效率较低。我们考虑用快速幂的方法计算(见Part 4问题2)。对于指数是浮点数的情况，程序使用数学库中的函数进行计算(可能可以转换成对数、指数用泰勒展开计算)。

这里需要注意：程序中认为0^0=1，**小数点不是识别整数的标准**，例如指数为1.0时，仍然会认为是整数，进而进行整数幂的快速幂计算。

4. 如何将错误位置尽可能准确地告知使用者

当输入的运算式较长时，仅仅告诉使用者出现的问题不利于使用者检查运算式，因此程序在将错误告知使用者时，会尽可能地将错误位置告知使用者。程序中使用数组`error[2]`来存储错误数据，0号位置存储错误类型，1号位置存储错误位置。程序在运算即检查中一旦遇到错误，将立刻记录错误信息，返回打印给使用者。

这里有几个需要考虑的地方：

- 由于程序运行使用者在任意位置输入空格，这将导致记录的错误位置索引值是建立在空格消除后的字符串上的，与原先输入字符串位置不对应，因此需要将空格消除后的字符串重新输出，以告知使用者准确位置。
- 由于除法函数需要统一写在结构体对应的源文件中，与`main`方法所在文件不是同一个，由于除法中需要检查除数是否为零，`main`方法中需要检查输入是否为空或空格，两者均需要对数组`error`的元素赋值，于是需要解决跨文件使用变量的问题(见Part 4问题2)。

5. 如何解决在调用用户自定义变量中错误选择的问题

当使用者定义的变量中存在名称包含关系时，程序往往存在变量调用不确定性。例如：使用者定义了两个变量：`num1`与`num11`，他们存在变量名包含关系，如果不做变化的话，程序在遇到运算式`num11`时，会识别为`num1`和`1`，进而告知使用者错误。但是这并没有错，因为它对应这一个变量名更长的变量。此时程序在定义变量顺序不同时结果不同，这并不是我们想要的结果。

因此，我们应该在变量赋值之后进行一次排序(这里用的是冒泡排序，也可以用其他排序方法)，将变量数组按变量名长度从长到短进行排序，这样将匹配最长即最可能出现的变量，避免选择问题。

Part 2 - Code

由于代码较长，这里仅放置输入字符串的分析部分，其他函数简介请参考源文件`Main.cpp`或函数头文件`FunHead.hpp`，结构体简介请参考函数头文件`StructHead.hpp`，源文件中的注释比较详细。

```
// Analyse function for input string
void analyse(char *str)
{
    // Variables about input string
    int len = strlen(str), Id, lastId = 0;
    int dot = -1, e = -1;
    bool isNum = false, numEnd = false;
    // Variables about calculate stack
    numPt = -1, opPt = -1;
    numSet = new Number[len];
    opSet = new char[len];
    priorSet = new char[len];
    // Analyse every char of input
    for (Id = 0; Id < len && !error[0]; ++Id)
    {
        char now = str[Id];
        // It is not a number char
        if (now < '0' || now > '9')
        {
            bool cmp = false;
            // Compare to variables name
            for (int j = 0; j <= varId; ++j)
                if (strncasecmp(&(str[Id]), varName[j], strlen(varName[j])) == 0)
                {
                    addNumBVar;
                    numSet[++numPt] = varVal[j].copy();
                    Id += strlen(varName[j]) - 1;
                    goto NEXT_CHAR;
                }
            // Compare to functions name
            for (int j = 0; j < 12; ++j)
                if (strncasecmp(&(str[Id]), FUN_STR[j], strlen(FUN_STR[j])) == 0)
                {
                    addNumBVar;
                    isNum = false;
                    addOp(j, 5);
                    Id += strlen(FUN_STR[j]) - 1;
                    goto NEXT_CHAR;
```

```
                }
            // Compare to operators name
            for (int j = 0; j < 8 && !cmp; ++j)
                if (now == OP_SG[j])
                {
                    // '(' is special
                    if (now == '(')
                    {
                        addNumBVar;
                        isNum = false;
                    }
                    else
                    {
                        addNumBOp;
                        if (now == ')')
                            isNum = true, numEnd = true, str[Id] = ')';
                    }
                    addOp(now, OP_PRIOR[j]);
                    goto NEXT_CHAR;
                }
            // Compare to pi
            if ((now == 'p' || now == 'P') && (str[Id + 1] == 'i' || str[Id + 1]
== 'I'))
            {
                addNumBVar;
                ++Id;
                numSet[++numPt] = constPi;
                goto NEXT_CHAR;
            }
            // Compare to 'e' / 'E'
            if (now == 'e' || now == 'E')
            {
                if (Id == 0 || str[Id - 1] == '\0' || str[Id - 1] == ')' ||
(str[Id + 1] != '-' && (str[Id + 1] < '0' || str[Id + 1] > '9')))
                {
                    addNumBVar;
                    numSet[++numPt] = constE;
                    str[Id] = '0';
                }
                else if (e >= 0)
                {
                    error[0] = 11, error[1] = Id;
                    return;
                }
                else
                    e = Id - lastId;
                goto NEXT_CHAR;
            } // Compare to '-'
            if (now == '-')
            {
                // As a minus signal
                if (Id != 0 && str[Id - 1] != '\0' && str[Id - 1] != 'e')
                {
                    addNumBOp;
                    addOp('-', 1);
```

```
                }
                // As an oppsite signal
                else if (!isNum)
                {
                    isNum = true, numEnd = false, lastId = Id;
                    dot = -1, e = -1;
                }
                goto NEXT_CHAR;
            } // Compare to '.'
            if (now == '.')
            {
                // Check if more than one dots or float exponent
                if (dot >= 0 || e >= 0)
                {
                    error[0] = 9 + (e >= 0), error[1] = Id;
                    return;
                }
                if (!isNum)
                    isNum = true, lastId = Id, e = -1;
                dot = Id - lastId;
                goto NEXT_CHAR;
            }
            // There is invalid char in the input
            error[0] = 4, error[1] = Id;
            return;
        NEXT_CHAR:
            continue;
        }
        // It is a number char
        else
        {
            // Last char is not number
            if (!isNum)
                lastId = Id, dot = -1, e = -1;
            else if (numEnd)
            {
                error[0] = 7, error[1] = Id;
                return;
            }
            isNum = true;
            numEnd = false;
        }
    }
    // Pick up the Number and finish the rest calculate in the end
    if (isNum && !numEnd)
        addstrNum(&(str[lastId]), dot, e);
    while (opPt >= 0 && priorSet[opPt] >= 0 && !error[0])
        calculate();
    // Save answer
    if (numPt == 0)
    {
        varVal[0].del();
        varVal[0] = numSet[numPt--];
    }
    else
```

```
        error[0] = 6;
    return;
}
```

Part 3 - Result & Verification

Test case #1: 基本要求的实现

注：程序中统一采用科学计数法进行输出

```
Input: cmake . & make
Input: ./Project2.out
Input: 2+3
Output: 5
```

```
Input: 5+2*3
Output: 1.1e1
```

```
Input: (5+2)*3
Output: 2.1e1
```

```
Input: x=3
Output: 3
Input: y=6
Output: 6
Input: x+2*y
Output: 1.5e1
```

```
Input: sqrt(3.0)
Output: 1.73205080756887729366
```

注：下面例子中第一个数包含30个9和10个2

```
Input: 999999999999999999999999999999.2222222222 + 1.0
Output: 1.000000000000000000000000000000022222222222e30
```

```
Consolidate compiler generated dependencies of target Project2.out
-- Generating done
-- Build files have been written to: /mnt/d/VScodeProjects/CppClass
[100%] Built target Project2.out
[3]-  Done                    cmake .
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./Project2.out
Please input the expression in the next line: (quit:q)
2+3
5

Please input the expression in the next line: (quit:q)
5+2*3
1.1e1

Please input the expression in the next line: (quit:q)
(5+2)*3
2.1e1

Please input the expression in the next line: (quit:q)
x=3
3

Please input the expression in the next line: (quit:q)
y=6
6

Please input the expression in the next line: (quit:q)
x+2*y
1.5e1

Please input the expression in the next line: (quit:q)
sqrt(3.0)
1.7320508075688772936

Please input the expression in the next line: (quit:q)
99999999999999999999999999999.2222222222 + 1.0
1.0000000000000000000000000000002222222222e30
```

Test case #2: 对错误输入的判断

对于**错误的具体位置**，程序会在告知错误前标明。

格式错误：

```
Input:
Output: There is no input. Please try again.
```

```
Input: =2+3
Output: The input cannot start with equal sign. Please try again.
```

```
Input: 3a=5-2
Output: The variable name on the left of equal sign is invalid. Please try again.
```

```
Input: 3+4_3
Output: There is invalid char in the input. Please try again.
```

```
Input: (3+4))
Output: The number of '(' and ')' are different. Please try again.
```

```
Input: pi2
Output: The numbers cannot follow a variable. Please try again.
```

```
Input: 2//2
Output: The operator cannot follow a operator. Please try again.
```

```
Input: *5
Output: The input starts with a operator. Please try again.
```

```
Please input the expression in the next line: (quit:q)


^~~~~~  ("" is invalid)
There is no input. Please try again.
Please input the expression in the next line: (quit:q)
=2+3
=2+3
^~~~~~  ("=" is invalid)
The input cannot start with equal sign. Please try again.
Please input the expression in the next line: (quit:q)
3a=5-2
3a=5-2
^~~~~~  ("3" is invalid)
The variable name on the left of equal sign is invalid. Please try again.

Please input the expression in the next line: (quit:q)
3+4_3
3+4_3
   ^~~~~~  ("_" is invalid)
There is invalid char in the input. Please try again.

Please input the expression in the next line: (quit:q)
(3+4))
(3+4))
^~~~~~  ("(" is invalid)
The number of '(' and ')' are different. Please try again.

Please input the expression in the next line: (quit:q)
pi2
pi2
  ^~~~~~  ("2" is invalid)
The numbers cannot follow a variable. Please try again.

Please input the expression in the next line: (quit:q)
2//2
2//2
  ^~~~~~  ("/" is invalid)
The operator cannot follow a operator. Please try again.

Please input the expression in the next line: (quit:q)
*5
*5
^~~~~~  ("*" is invalid)
The input starts with a operator. Please try again.

Please input the expression in the next line: (quit:q)
```

Test case #3: 对错误输入的判断

数字输入错误:

```
Input: 1..2
Output: There are more than one dots in a number. Please try again.
```

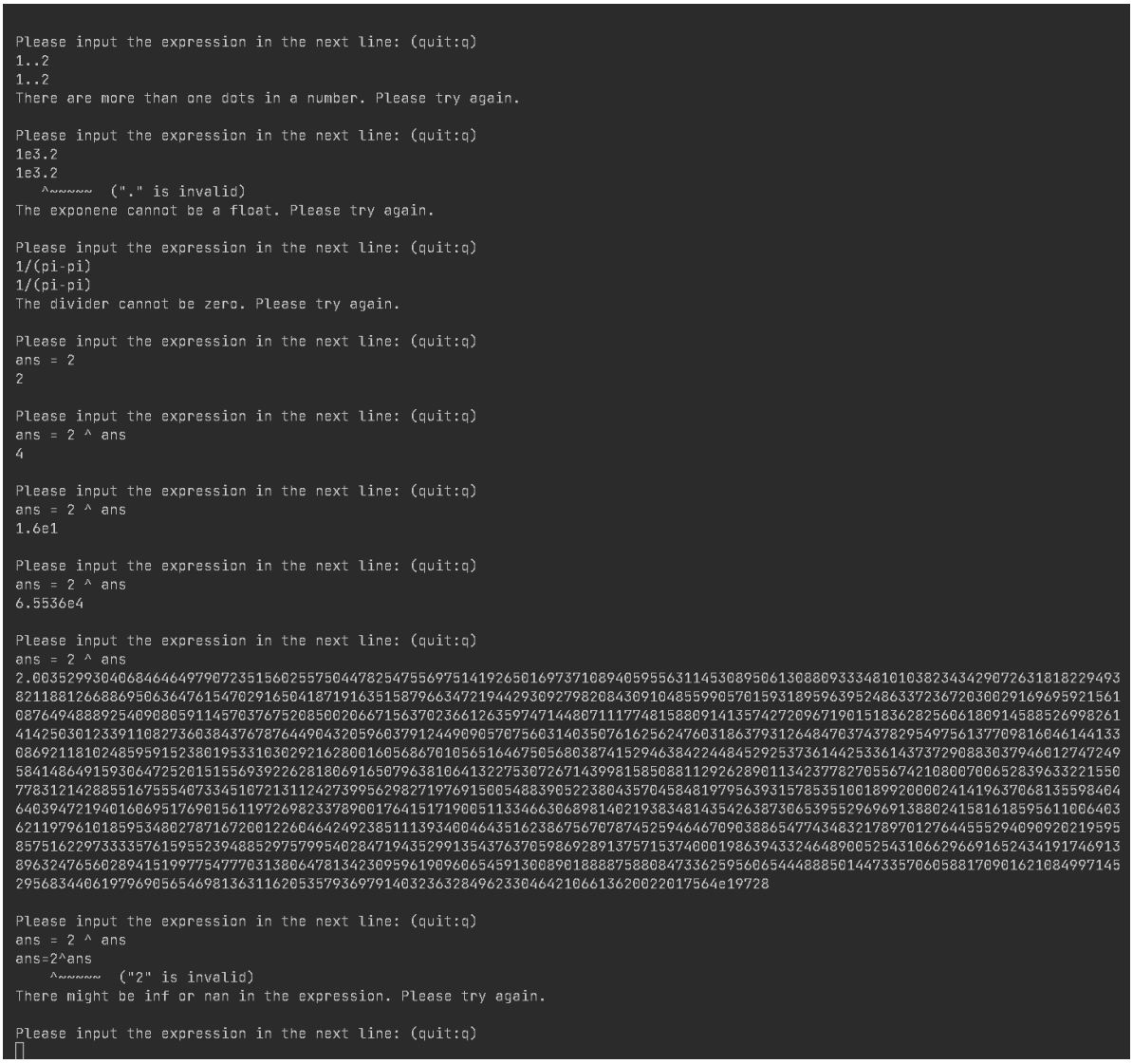
```
Input: 1e3.2
Output: The exponene cannot be a float. Please try again.
```

计算错误：

```
Input: 1/(pi-pi)
Output: The divider cannot be zero. Please try again.
```

先使用`ans = 2`初始化，然后重复进行以下运算：

```
Input: ans = 2 ^ ans
...
Input: ans = 2 ^ ans
Output: There might be inf or nan in the expression. Please try again.
```



```
Please input the expression in the next line: (quit:q)
1..2
1..2
There are more than one dots in a number. Please try again.

Please input the expression in the next line: (quit:q)
1e3.2
1e3.2
   ^~~~~~  ("." is invalid)
The exponene cannot be a float. Please try again.

Please input the expression in the next line: (quit:q)
1/(pi-pi)
1/(pi-pi)
The divider cannot be zero. Please try again.

Please input the expression in the next line: (quit:q)
ans = 2
2

Please input the expression in the next line: (quit:q)
ans = 2 ^ ans
4

Please input the expression in the next line: (quit:q)
ans = 2 ^ ans
1.6e1

Please input the expression in the next line: (quit:q)
ans = 2 ^ ans
6.5536e4

Please input the expression in the next line: (quit:q)
ans = 2 ^ ans
2.0035299304068464649790723515602557504478254755697514192650169737108940595563114530895061308809333481010382343429072631818229493
8211881266886950636476154702916504187191635158796634721944293092798208430910485599057015931895963952486337236720300291696959215610
8764948889254090805911457037675208500206671563702366126359747144807111774815880914135742720967190151836282560618091458852699826141
4142503012339110827360384376787644904320596037912449090570756031403507616256247603186379312648470374378295497561377098160461441330
8692118102485959152380195331030292162800160568670105651646750568038741529463842244845292537361442533614373729088303794601274724958
5841486491593064725201515569392262818069165079638106413227530726714399815850881129262890113423778270556742108007006528396332215500
7783121428855167555407334510721311242739956298271976915005488390522380435704584819795639315785351001899200002414196370681355984040
6403947219401606951769015611972698233789001764151719005113346630689814021938348143542638730653955296969138802415816185956110064030
6211979610185953480278716720012260464249238511139340046435162386756707874525946467090388654774348321789701276445552940909202195950
8575162297333357615955239488529757995402847194352991354376370598692891375715374000198639433246489005254310662966916524341917469130
8963247656028941519977547770313806478134230959619096065459130089018888758808473362595606544488850144733570605881709016210849971450
2956834406197969056546981363116205357936979140323632849623304642106613620022017564e19728

Please input the expression in the next line: (quit:q)
ans = 2 ^ ans
ans=2^ans
    ^~~~~~  ("2" is invalid)
There might be inf or nan in the expression. Please try again.

Please input the expression in the next line: (quit:q)
```

Test case #4: 程序特色功能

具体实现功能参考 README.md

1. 函数中**可以不使用括号**，常数`pi`与`e`有100位小数的精度

    ```
    Input: cos pi
    Output: -1
    ```

```
Input: log e
Output: 1
```

```
Input: floor 2.5
Output: 2
```

```
Input: asin(sin1)
Output: 1
```

2. 数字与(参数、函数)、(参数、函数)与(参数、函数)之间的乘法**可以不写乘号**（但**数字与数字之间的乘号不可省略**）

```
Input: pipi
Output:
9.869604401089358618834490999876151135313699407240790626413349376220044822419
205243001773403718552231307874263580850209166609835428373261595226022618170033
881496242667944434304721741925322332314849321041
```

```
Input: 1e10e (指1*10^10*e)
Output:
2.718281828459045235360287471352662497757247093699959574966967627724076630353
547594571382178525166427 4e10
```

3. 答案储存

```
Input: ans/10 (与之前的结果有关)
Output:
2.718281828459045235360287471352662497757247093699959574966967627724076630353
547594571382178525166427 4e9
```

```
Please input the expression in the next line: (quit:q)
cos pi
-1

Please input the expression in the next line: (quit:q)
log e
1

Please input the expression in the next line: (quit:q)
floor 2.5
2

Please input the expression in the next line: (quit:q)
asin(sin1)
1

Please input the expression in the next line: (quit:q)
pipi
9.869604401089358618834490999876151135313699407240790626413349376220044822419205243001773403718552231307874263580850209166609835428373261595226022618170338814962426679444343047217419253223323148493210 41

Please input the expression in the next line: (quit:q)
1e10e
2.718281828459045235360287471352662497757247093699959574966967627724076630353547594571382178525166427 4e10

Please input the expression in the next line: (quit:q)
ans/10
2.718281828459045235360287471352662497757247093699959574966967627724076630353547594571382178525166427 4e9
```

Test case #5: 快速幂与大数高精度运算

注意：小数点不是识别整数的标准，即下面这个例子输入改为`1.01^365.0`结果一样。

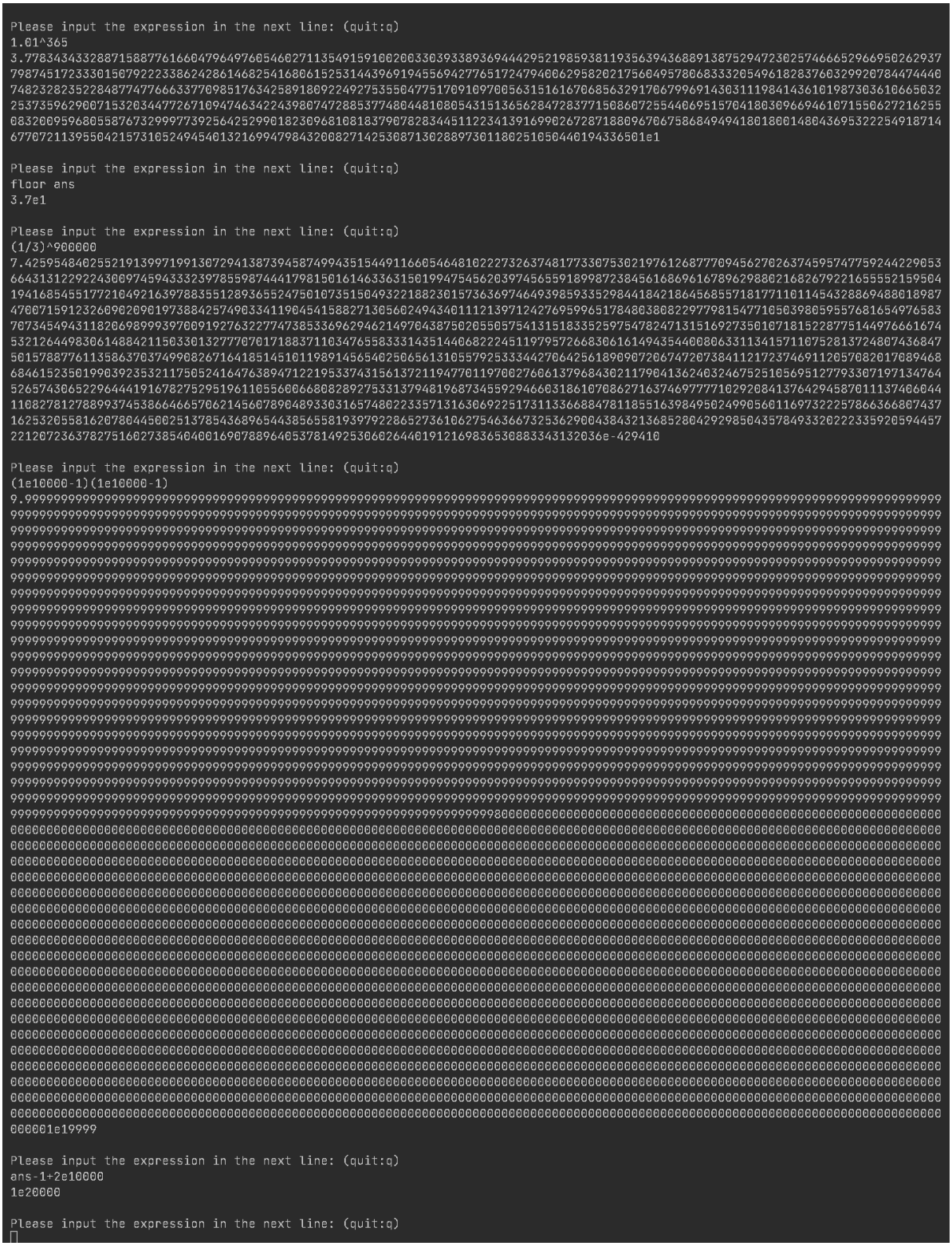
```
Input: 1.01^365
Output:
3.778343433288715887761660479649760546027113549159100200330393389369444295219859381193563943688913875294723025746665296695026293779874517233301507922233862428614682541680615253144396919455694277651724794006295820217560495780683332054961828376032992078447444074823282352284877477666337709851763425891809224927535504775170910970056315161670685632917067996914303111984143610198730361066503225373596290071532034477267109474634224398074728853774804481080543151365628472837715086072554406951570418030966946107155062721625508320095968055876732999773925642529901823096810818379078283445112234139169902672871880967067586849494180180014804369532225491871467707211395504215731052494540132169947984320082714253087130288973011802510504401943365O1e1
```

```
Input: floor ans
Output: 3.7e1
```

```
Input: (1/3)^900000
Output:
7.425954840255219139971991307294138739458749943515449116605464810222732637481773307530219761268777094562702637459574775924 4...e-429410 (输出中共1500位有效数字)
```

```
Input: (1e10000-1)(1e10000-1)
Output: 9.9999999999...9999998000000...0001e19999 (输出中共19999位有效数字)
```

```
Input: ans-1+2e10000
Output: 1e20000
```



注：中间部分的部分9和0被长截图拼合时吞了

Part 4 - Difficulties & Solutions

问题1：如何对带括号的表达式进行运算

如果使用者输入的运算式中含有括号，那么程序不能简单地进行从函数->乘方->乘除模(余数)->加减来运算，如果通过**迭代的方式逐步消除括号**，则这样遍历字符串时会出现重复遍历，导致效率较低。

通过借鉴**中缀表达式转后缀表达式**的方法，我们其实无需完全转化成后缀表达式再进行运算，可以一边转化一边运算，以节省运算时间。

在这种方法中，我们规定：**遇到一个运算符时，将其前面出现的所有不低于该运算符优先值的符号进行计算**，否则记录该符号，最后再从右向左进行运算。其中遇到右括号时，必须运算到出现左括号为止。

例如计算：`1+2*3*(4+5)`，我们需要使用两个数组来分别存储数字和运算符：`numSet[]`，`opSet[]`，当遍历到1时，添加数字1，`numSet[] = {1,}`，接下来如下表：

遍历到	`numSet[]`	`opSet[]`	操作
1	{1,}	{}	
+	{1,}	{'+',}	
2	{1,2}	{'+',}	
*	{1,2}	{'+','*'}	
3	{1,2,3}	{'+','*'}	
*	{1,6}	{'+','*'}	2*3=6
(	{1,6}	{'+','*','('}	
4	{1,6,4}	{'+','*','('}	
+	{1,6,4}	{'+','*','(','+'}	
5	{1,6,4,5}	{'+','*','(','+'}	
)	{1,6,9}	{'+','*'}	4+5=9
持续运算	{1,54}	{'+'}	6*9=54
持续运算	{55}	{}	1+54=55

问题2：如何进行整数幂的快速运算

前面提到如果用逐次乘法进行幂的运算效率太低，因此我们使用**快速幂法**结合**位运算**来计算整数或浮点数的正整数幂。

快速幂方法是指将整数的幂指数按照二进制展开，从后向前判断是否为1，如果是，将底数乘入结果，接着将底数平方，循环上面的操作。其时间复杂度为$O(\log_2 N)$，减少运算时间。

例如计算：a^5，a是浮点数，5=(101)2

底数	指数二进制下的位	结果
a	1	a
a^2	0	a
$(a^2)^2=a^4$	1	$a*a^4=a^5$

注：虽然没有在示例中演示浮点数次幂的运算，但程序允许浮点数次幂的运算

问题3：如何调用不在同一个文件中的变量

前面提到我们需要跨文件调用变量，这里我们使用C++关键字`extern`告诉该文件这个变量`error[]`是**外部变量**，可以直接调用与赋值，在g++编译关联时会将这个变量关联起来，可以防止重复定义该变量。

Part 5 - Summary

本次project实现的计算器较为复杂，在构造`Integer`和`Number`结构体各种方法与实现输入字符串的分析判断中会遇到许多bug，尤其是运算时对左右括号与其他符号的计算方式完全不同。之前我认为左右括号在运算中优先级最高，所以其优先值应该赋给最大的数，但是在运算时规定将其前面出现的所有不低于该优先值的符号进行计算，这往往会使得左括号直接被去除，右括号也起不到作用。于是我考虑将左右括号赋给最小的优先值，左括号不进行操作，右括号遇到相同优先值时去除，同时能检查是否有错误，又能不打破原先的规定。因此我们在遇到困难时，可以往反方向思考，有可能会获得解决困难的思路。另外，在debug时，往往程序中有许多bug夹杂在一起，这时候我们应该使用较为简单的测试数据，逐步运行，时刻关注运行中数据值的变化，进而逐一找出问题所在。

以上是本次Report的所有内容，感谢您的阅读！