CS205 C/ C++ Programming - Project

Name: 钟元吉(Zhong Yuanji)

SID: 12012613

Part 1 - Analysis

> This project is to implement a calculator which can multiply two numbers.

题目要求分析

在设计与实践两个数相乘的计算器时，我们会考虑到许多问题，例如：输入的数字个数不是两个，输入含有无法转换为数字的符号，输入数字中负号在错误位置，输入的数字中含有浮点数，输入数字过大，存储内存过大等等。

输入数字个数判断

在题目中有明确说明被乘的两个数通过命令行输入，所以当命令行参数超过2个或不足2个时，将输出`Please check the number of inputs. It must be 2.`并结束程序。

对错误输入的判断与是否含有浮点数的判断

由于在向命令行输入参数并运行程序时，操作者有可能在无意间输入错误的字符或格式，而错误输入的种类较多，在程序的运行中，我们需要首先对通过命令行输入的字符进行详细的检查，并告知操作者哪一个输入有问题。

首先我们需要排出除了**数字、负号、指数符号、小数点**以外的所有其他字符的输入，如果命令行参数中有这些形式（如第一个参数含有中文字符），将输出`The first input cannot be interpret as numbers! (invalid char)`并结束程序。

其次，我们需要判断负号、指数符号与小数点的输入格式是否正确。在一个正确格式的数字中，**负号**只能出现在数字的开始位置或者科学计数法中的指数的开始位置，因此对于出现在数字之后或连续出现两个及以上的负号等错误情况时，程序会通过命令行告知操作者该参数存在不合理的负号。而**指数符号**在科学计数法中写作`'e'`或`'E'`，只能在浮点数中出现，在一个浮点数中无论哪一种写法都至多只能出现一次，另外作为指数符号，`'e'`或`'E'`均不能出现在每个数字的开始位置。程序中用变量`haveE`来记录输入参数中指数符号的个数，并判断指数符号的位置是否合理，对不合理的情况进行输出提示。**小数点**与指数符号相似，同样在一个浮点数中只能至多出现一次，但小数点可以出现在数字或指数的开始位置，例如.1表示0.1；另外指数不能为小数，例如：我们认为1.1e1.1属于科学计数法的错误写法并在程序中输出`The exponent of the first number cannot be a float number.`详细示例见Part 3.

当任意一个输入参数中含有正确格式的小数点或指数符号，我们就可以认为输入的参数是浮点数，例如1.0和1e1为浮点数。对于不同的数据，程序使用不同的数据结构记录输入参数，当输入均为整数时，程序使用`unsigned short`数组逐位记录（将在Part 4说明原因），当输入有浮点数时，程序用`long double`记录输入数据并进行运算。

乘法的实现

对于输入参数均为整数的情况，程序中使用`unsigned short`数组逐位记录输入参数的数值，例如将输入的两个整数及他们的乘积记为：

$$\sum_{i=0}^{n_1} a_i, \sum_{i=0}^{n_2} b_i, \sum_{i=0}^{n_1+n_2} c_i$$

其中`a_i,b_i,c_i`代表对应整数从右到左的第`i+1`位，若两数的乘积超过`n1+n2+1`位，则认为是`c_n1+n2`包含最左边两位。则三个数（或称数组）之间满足柯西乘积的关系，即：

$$\sum_{i=0}^{n_1+n_2} c_i = (\sum_{i=0}^{n_1} a_i) \cdot (\sum_{i=0}^{n_2} b_i) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} a_i b_j = \sum_{i=0}^{n_1+n_2} \sum_{j=\max(0,i-n_2)}^{\min(i,n_1)} a_j b_{i-j}$$

或者说，只需要遍历`i,j`将`a_i*b_j`加到`c_i+j`上即可。对于逐位得到的乘积结果，还需要进行进位处理，然后再逐位输出同时跳过开始的0.

对于输入参数中含有小数的情况，由于`double`类型存储的数据较少，程序以`long double`的形式读入输入参数，并进行计算，如果出现乘积结果为`Inf`时，程序将告知操作者输入的参数过大。

Part 2 - Code

```cpp
#include <iostream>
#include <string.h>
#include <cmath>
using namespace std;

// The main function
int main(int argc, char const *argv[])
{
    // Check if the number of inputs is 2
    if (argc != 3)
    {
        cout << "Please check the number of inputs. It must be 2." << endl;
        return 0;
    }
    // Check if inputs include float number or not a number (or have more than 1
dot or incorrct minus signal or other char)
    string name[] = {"", "first", "second"};
    int num1len = strlen(argv[1]), num2len = strlen(argv[2]);
    bool haveFloat = false, haveDot, haveE;
    for (short k = 1; k < 3; ++k)
    {
        haveDot = false, haveE = false;
        for (int i = 0; i < strlen(argv[k]); ++i)
        {
            // Check if inputs include float number
            if (~haveFloat && (argv[k][i] == '.' || argv[k][i] == 'e' || argv[k]
[i] == 'E'))
            {
                haveFloat = true;
            }
            // Check if inputs include invalid char
            if (('0' > argv[k][i] || argv[k][i] > '9') && argv[k][i] != '.' &&
argv[k][i] != 'e' && argv[k][i] != 'E' && argv[k][i] != '-')
            {
                cout << "The " << name[k] << " input cannot be interpret as
numbers! (invalid char)" << endl;
                return 0;
            }
            // Check if have more than one dots
            else if (argv[k][i] == '.')
            {
                if (haveE)
                {
                    cout << "The exponent of the " << name[k] << " number cannot
be a float number." << endl;
                    return 0;
                }
                else if (!haveDot)
                {
                    haveDot = true;
                }
                else
                {
                    cout << "There are more than one dots in the " << name[k] <<
" number." << endl;
                    return 0;
                }
            }
            else if (argv[k][i] == 'e' || argv[k][i] == 'E')
            {
                // Check if the first char is 'e' or 'E'
                if (i == 0)
                {
                    cout << "The first char in the " << name[k] << " number
cannot be \'e\' or \'E\'." << endl;
                    return 0;
                }
                // Check if have more than one 'e' or 'E'
                else if (!haveE)
                {
                    haveE = true;
                }
                else
                {
                    cout << "There are more than one \'e\' or \'E\' in the " <<
name[k] << " number." << endl;
                    return 0;
                }
            }
            // Check if have an incorrect minus signal
            else if (argv[k][i] == '-')
            {
                if (i != 0 && argv[k][i - 1] != 'e' && argv[k][i - 1] != 'E')
                {
                    cout << "There is an incorrect minus signal in the " <<
name[k] << " number." << endl;
                    return 0;
                }
            }
        }
    }

    // Divide float and int, compute and output
    if (haveFloat)
    {
        // Read and compute
        // ATTENTION: If the inputs include correct float, but output is "The
inputs float numbers are too big.", that is because
        //              the version of g++ is too low. Please change the "long
double" in next line into "double" and run again.
        long double num1, num2;
        sscanf(argv[1], "%Lf", &num1);
        sscanf(argv[2], "%Lf", &num2);
        long double num3 = num1 * num2;

        // Check if the number is too big and show result
        if (isnan(num3) || isinf(num3))
        {
            cout << "The input float numbers are too big." << endl;
        }
        else
        {
            cout << argv[1] << " * " << argv[2] << " = " << num3 << endl;
        }
    }
    else // Inputs are integers
    {
        // Judge the symbol of product
        char symbol = (argv[1][0] == '-' ? 1 : 0) + (argv[2][0] == '-' ? 1 : 0) %
2;
        // Read num1
        char num1[num1len], num2[num2len];
        for (int i = num1len - 1; i >= 0; --i)
        {
            if (i == 0 && argv[1][0] == '-')
            {
                num1[num1len - 1] = 0;
                break;
            }
            num1[num1len - 1 - i] = argv[1][i] - '0';
        }
        // Read num2
        for (int i = num2len - 1; i >= 0; --i)
        {
            if (i == 0 && argv[2][0] == '-')
            {
                num2[num2len - 1] = 0;
                break;
            }
            num2[num2len - 1 - i] = argv[2][i] - '0';
        }
        // Initial num3 and compute the product
        unsigned short num3[num1len + num2len];
        for (int i = 0; i < num1len + num2len; i++)
            num3[i] = 0;
        for (int i = 0; i < num1len * num2len; i++)
        {
            int x = i % num1len, y = i / num1len;
            num3[x + y] += num1[x] * num2[y];
        }
        for (int i = 0; i < num1len + num2len - 1; i++)
        {
            num3[i + 1] += num3[i] / 10;
            num3[i] %= 10;
        }
        // Show result
        cout << argv[1] << " * " << argv[2] << " = " << (symbol == 1 ? "-" : "");
        char unzero = 0;
        for (int i = num1len + num2len - 1; i >= 0; --i)
        {
            if (unzero != 0)
            {
                cout << num3[i];
            }
            else if (num3[i] != 0)
            {
                cout << num3[i];
                unzero = 1;
            }
        }
        cout << endl;
    }
    return 0;
}
```

Part 3 - Result & Verification

Test case #1: 基本要求的实现

```
Input: g++ ./Project1/source.cpp -o mul
Input: ./mul 2 3
Output: 2 * 3 = 6
```

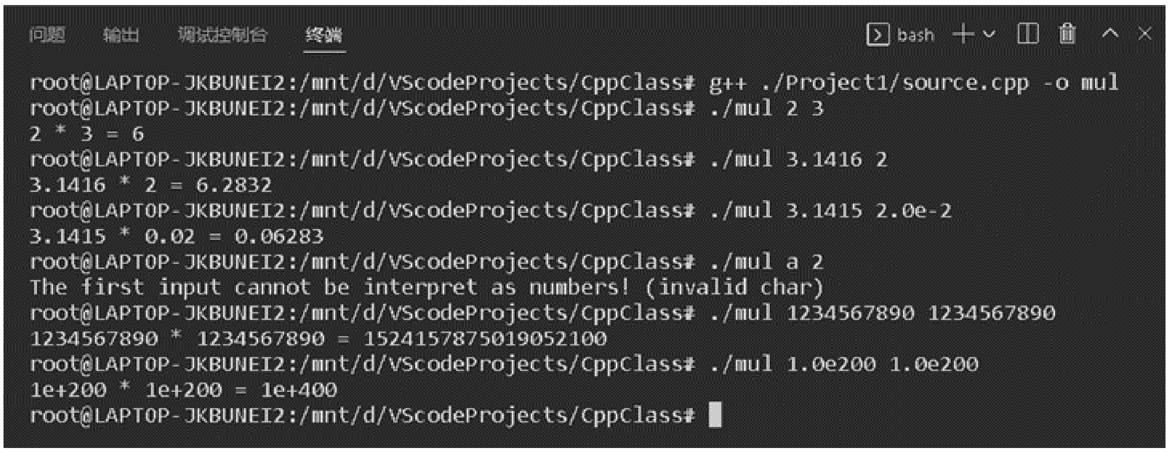
```
Input: ./mul 3.1416 2
Output: 3.1416 * 2 = 6.2832
```

```
Input: ./mul 3.1415 2.0e-2
Output: 3.1415 * 2.0e-2 = 0.06283
```

```
Input: ./mul a 2
Output: The first input cannot be interpret as numbers! (invalid char)
```

```
Input: ./mul 1234567890 1234567890
Output: 1234567890 * 1234567890 = 1524157875019052100
```

```
Input: ./mul 1.0e200 1.0e2002 * 3 = 6
Output: 1.0e200 * 1.0e200 = 1e+400
```



```
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# g++ ./Project1/source.cpp -o mul
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 2 3
2 * 3 = 6
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 3.1416 2
3.1416 * 2 = 6.2832
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 3.1415 2.0e-2
3.1415 * 0.02 = 0.06283
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul a 2
The first input cannot be interpret as numbers! (invalid char)
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 1234567890 1234567890
1234567890 * 1234567890 = 1524157875019052100
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 1.0e200 1.0e200
1e+200 * 1e+200 = 1e+400
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# █
```

Test case #2: 对错误输入的判断

```
Input: ./mul 100000 1 2
Output: Please check the number of inputs. It must be 2.
```

```
Input: ./mul 100000 1,2
Output: The second input cannot be interpret as numbers! (invalid char)
```

```
Input: ./mul 1e10 2.3.e22
Output: There are more than one dots in the second number.
```

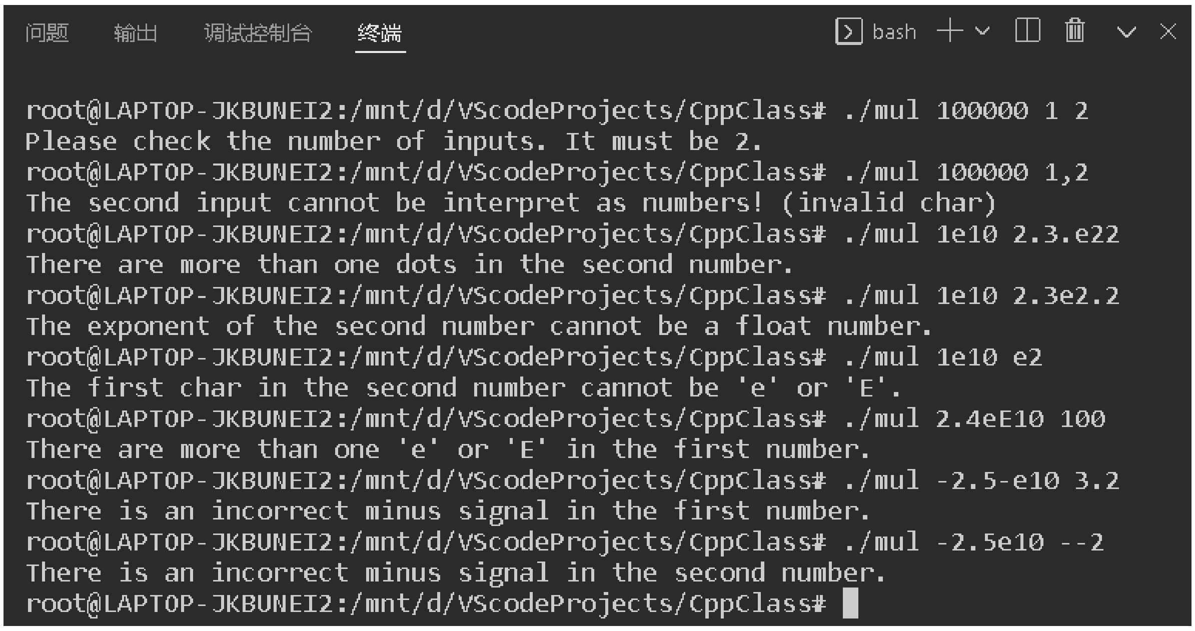
```
Input: ./mul 1e10 2.3e2.2
Output: The exponent of the second number cannot be a float number.
```

```
Input: ./mul 1e10 e2
Output: The first char in the second number cannot be 'e' or 'E'.
```

```
Input: ./mul 2.4eE10 100
Output: There are more than one 'e' or 'E' in the first number.
```

```
Input: ./mul -2.5-e10 3.2
Output: There is an incorrect minus signal in the first number.
```

```
Input: ./mul -2.5e10 --2
Output: There is an incorrect minus signal in the second number.
```



```
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 100000 1 2
Please check the number of inputs. It must be 2.
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 100000 1,2
The second input cannot be interpret as numbers! (invalid char)
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 1e10 2.3.e22
There are more than one dots in the second number.
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 1e10 2.3e2.2
The exponent of the second number cannot be a float number.
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 1e10 e2
The first char in the second number cannot be 'e' or 'E'.
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 2.4eE10 100
There are more than one 'e' or 'E' in the first number.
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul -2.5-e10 3.2
There is an incorrect minus signal in the first number.
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul -2.5e10 --2
There is an incorrect minus signal in the second number.
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# 
```

Test case #3: 正确例子与大数计算

通过python检验，以下输出结果数值均正确。

```
Input: ./mul .5 20
Output: .5 * 20 = 10
```

注：.5属于0.5的一种特殊写法，这里认为是正确输入格式

```
Input: ./mul 123456789 -987654321
Output: 123456789 * -987654321 = -121932631112635269
```

```
Input: ./mul -1111122222333334444455555666666 1111122222333334444455555666666
Output: -1111122222333334444455555666666 * 1111122222333334444455555666666 =
-1234592592962967901296297037028394962961851839506037035555 6
```

```
Input: ./mul 1.234e123 -2.345E345
Output: 1.234e123 * -2.345E345 = -2.89373e+468
```

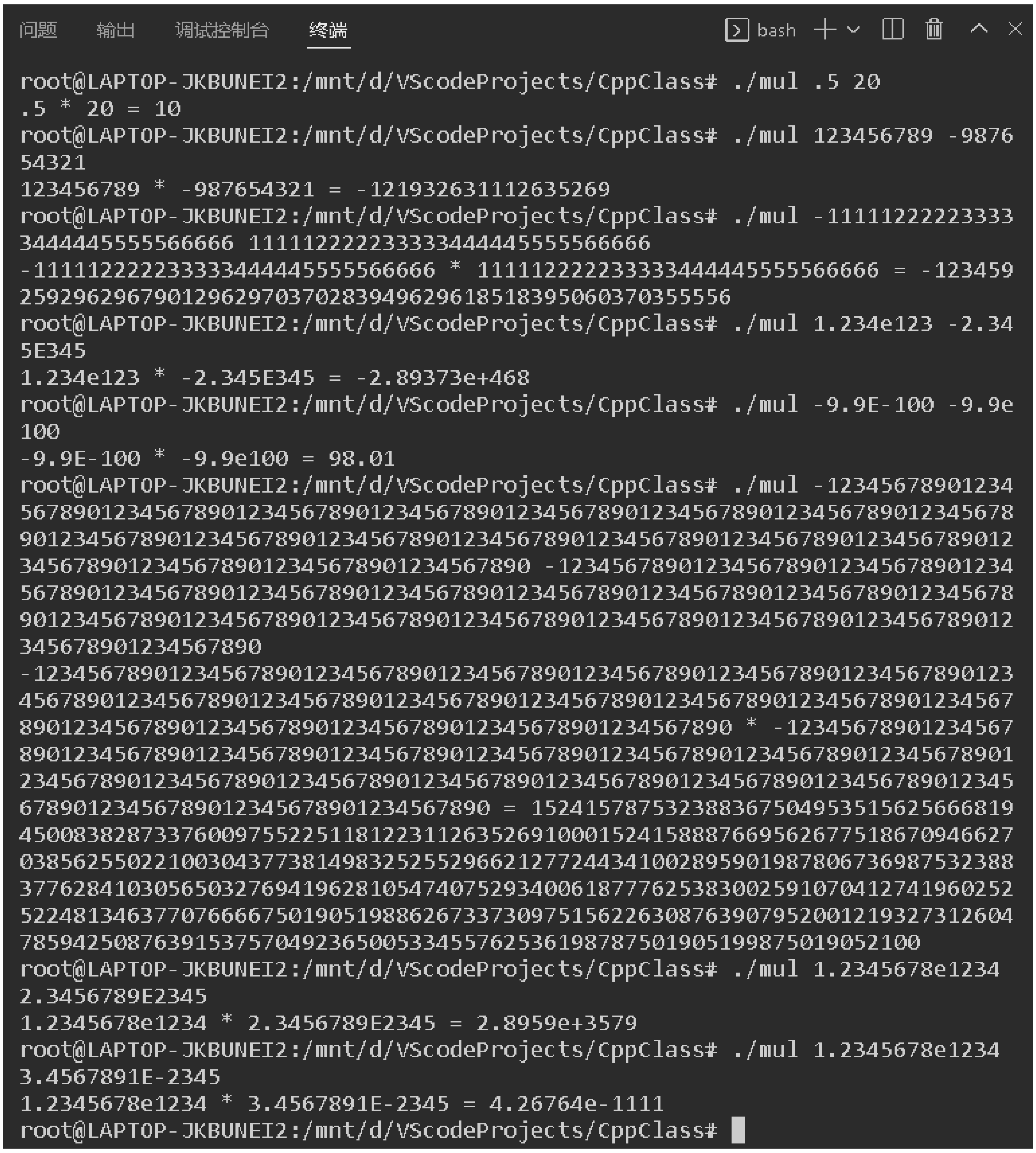
```
Input: ./mul -9.9E-100 -9.9e100
Output: -9.9E-100 * -9.9e100 = 98.01
```

200位*200位整数：

```
Input: ./mul
-12345678901234567890123456789012345678901234567890123456789012345678901234567890
123456789012345678901234567890123456789012345678901234567890123456789012345678901
234567890123456789012345678901234567890
-12345678901234567890123456789012345678901234567890123456789012345678901234567890
123456789012345678901234567890123456789012345678901234567890123456789012345678901
234567890123456789012345678901234567890
Output:
-12345678901234567890123456789012345678901234567890123456789012345678901234567890
123456789012345678901234567890123456789012345678901234567890123456789012345678901
234567890123456789012345678901234567890 *
-12345678901234567890123456789012345678901234567890123456789012345678901234567890
123456789012345678901234567890123456789012345678901234567890123456789012345678901
234567890123456789012345678901234567890 =
152415787532388367504953515625666819450083828733760097552251181223112635269100015
241588876695626775186709466270385625502210030437738149832525529662127724434100289
590198780673698753238837762841030565032769419628105474075293400618777625383002591
070412741960252522481346377076666750190519886267337309751562263087639079520012193
273126047859425087639153757049236500533455762536198787501905199875019052100
```

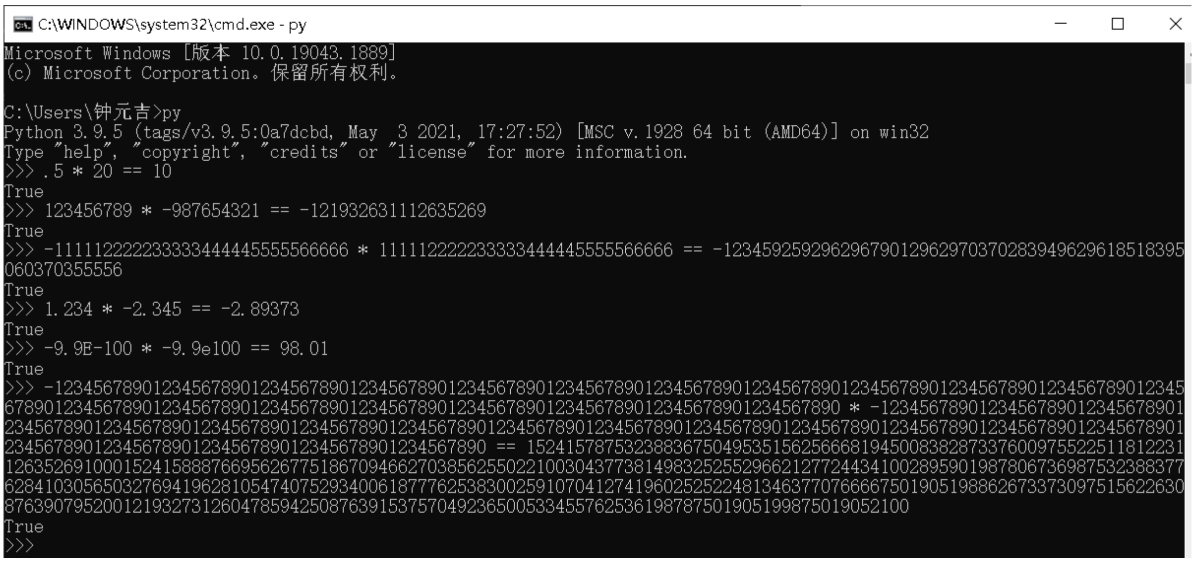
```
Input: ./mul 1.2345678e1234 2.3456789E2345
Output: 1.2345678e1234 * 2.3456789E2345 = 2.8959e+3579
```

```
Input: ./mul 1.2345678e1234 3.4567891E-2345
Output: 1.2345678e1234 * 3.4567891E-2345 = 4.26764e-1111
```



```
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul .5 20
.5 * 20 = 10
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 123456789 -9876
54321
123456789 * -987654321 = -121932631112635269
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul -11111222223333
3444445555566666 1111122222333334444455555666666
-1111122222333334444455555666666 * 1111122222333334444455555666666 = -123459
2592962967901296297037028394962961851839506037035556
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 1.234e123 -2.34
5E345
1.234e123 * -2.345E345 = -2.89373e+468
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul -9.9E-100 -9.9e
100
-9.9E-100 * -9.9e100 = 98.01
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul -12345678901234
5678901234567890123456789012345678901234567890123456789012345678901234567
8901234567890123456789012345678901234567890123456789012345678901234567890
34567890123456789012345678901234567890 -12345678901234567890123456789012345
67890123456789012345678901234567890123456789012345678901234567890123456789
0123456789012345678901234567890123456789012345678901234567890123456789012
345678901234567890
-1234567890123456789012345678901234567890123456789012345678901234567890123
4567890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890 * -12345678901234567
89012345678901234567890123456789012345678901234567890123456789012345678901
23456789012345678901234567890123456789012345678901234567890123456789012345
67890123456789012345678901234567890 = 152415787532388367504953515625666819
45008382873376009755225118122311263526910001524158887669562677518670946627
03856255022100304377381498325255296621277244341002895901987806736987532388
37762841030565032769419628105474075293400618777625383002591070412741960252
52248134637707666675019051988626733730975156226308763907952001219327312604
78594250876391537570492365005334557625361987875019051998750190521000
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 1.2345678e1234
2.3456789E2345
1.2345678e1234 * 2.3456789E2345 = 2.8959e+3579
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass# ./mul 1.2345678e1234
3.4567891E-2345
1.2345678e1234 * 3.4567891E-2345 = 4.26764e-1111
root@LAPTOP-JKBUNEI2:/mnt/d/VScodeProjects/CppClass#
```

用python检验(最后两个例子已超出python运算范围)：



```
Microsoft Windows [版本 10.0.19043.1889]
(c) Microsoft Corporation。保留所有权利。

C:\Users\钟元吉>py
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> .5 * 20 == 10
True
>>> 123456789 * -987654321 == -121932631112635269
True
>>> -1111122222333334444455555666666 * 1111122222333334444455555666666 == -12345925929629679012962970370283949629618518395
060370355556
True
>>> 1.234 * -2.345 == -2.89373
True
>>> -9.9E-100 * -9.9e100 == 98.01
True
>>> -1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
67890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890 * -12345678901234567890123456789012345678901
23456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901
2345678901234567890123456789012345678901234567890 == 152415787532388367504953515625666819450083828733760097552251181223 1
126352691000152415888766956267751867094662703856255022100304377381498325255296621277244341002895901987806736987532388377
628410305650327694196281054740752934006187776253830025910704127419602525224813463770766667501905198862673373097515622630
8763907952001219327312604785942508763915375704923650053345576253619878750190519987501905 2100
True
>>>
```

Part 4 - Difficulties & Solutions

问题1：两个`long long int`数据的乘积可能超过`long long int`范围

对于一个乘法程序，使用`long long int`数据进行整数的乘法只能进行约10$^{19\text{与}10}$19的乘法，而且对于结果的储存而言，一个大小约10^27的数字在C++的数据结构中没有一种能够直接存储。

在解决这个问题时，我想到了将输入的数字截断的方法，对于输入的两个数据在10$^{10\text{至}10}$18范围内时，用两个`int`数据存储输入数据从个位起的每9位，这时将18位与18位数字的乘法分解成了4个9位与9位的乘法，每一步乘法中至多产生18位的结果，刚好可以用`long long int`数据储存，然后进行格式化输出，得到由3个`long long int`数据拼接而成的结果。虽然这个思路解决了问题中的存储计算部分，也就是说此时程序可以进行10^18 * 10^18 = 10^36的整数计算了，但是可以对于超过18位较大的数字来说，程序仍然无法计算，于是这个思路仍有一定的缺陷。

前期思路对应代码（已舍弃）：

```
// Check if the number is too big
if (num1len > 18 || num2len > 18)
{
    cout << "The input integers are too big." << endl;
}
else
{
    long long int num1, num2;
    sscanf(argv[1], "%lli", &num1);
    sscanf(argv[2], "%lli", &num2);
    if (num1len + num2len > 18)
    { // If the product may bigger than 10^18, it could not be stored in long
long int.
        long long num3 = (num1 % 1000000000) * (num2 % 1000000000);
        long long num4 = (num1 / 1000000000) * (num2 % 1000000000) + (num1 %
1000000000) * (num2 / 1000000000);
        long long num5 = (num1 / 1000000000) * (num2 / 1000000000);
        num4 += num3 / 1000000000;
        num3 %= 1000000000;
        num5 += num4 / 1000000000;
        num4 %= 1000000000;
        if (num5 == 0)
        { // If the product is less than 10^18, num5 need not to be shown.
            printf("%lli * %lli = %lli%09lli\n", num1, num2, num4, abs(num3));
        }
        else
        {
            printf("%lli * %lli = %lli%09lli%09lli\n", num1, num2, num5,
abs(num4), abs(num3));
        }
    }
    else
    {
        cout << argv[1] << " * " << argv[2] << " = " << num1 * num2 << endl;
    }
}
```

问题2：输入整数可能超过`long long int`范围

虽然问题1中并没有解决`long long int`数据对输入的整数存储的限制问题，但是通过数字截断的方法，如果将输入的数据逐个截断，并存储于整形数组中，再遍历输入参数对应两个数组`i,j`的每一位及组合，并将对应乘积加到结果数组的第`i+j`位，然后对每一位进行进位操作。通过逐位乘法计算，我们可以有效避免输入整数过大的问题。但是，我们又迎来了新的问题，存储整数数组的数据类型应该选什么呢？

问题3：程序运行效率与存储数据占用内存的平衡与解决方案

对于整数数据运算，如果我们选取`int`,`long`或`long long`来存储整数的每一位，则容易占用内存过大；如果我们选用`char`数组来存储每一位，则在进行各位数字乘积相加时容易超过`char`数据类型的范围，当然如果每次相加时都进行进位，则进位需要遍历循环，会影响程序运行效率。因此程序中选用`short`类型数组来存储数据，考虑到数字的符号可以通过输入数字的第一位字符直接决定，进一步可以将`short`类型数组改为`unsigned short`类型数组来存储数据，这样就做到了程序运行效率与存储数据占用内存之间的平衡。

Part 5 - Summary

虽然本次project是设计是做一个计算器上一个简单功能的实现，但是我们需要考虑许多问题，从对输入数据格式的检测到对计算大数选择数据类型的思考。编程时有时思维会产生局限性，因此更需要对程序的每个部分做出详细的考虑。

以上是本次Report的所有内容，感谢您的阅读！