CS205 C/ C++ Programming -- Project 1 Report

Name: MIAO Fangran

Student ID: 11911819

PART 1 Analysis

在这个问题中，我们会输入两个数，每个数都可能是整数，小数以及科学计数法。 我们需要输入这两个数相乘的结果。

首先**第一个要解决的问题**是确定这两个数是否是上面这些可能的样式。 这里用到的方法是正则表达式。 例如，如下的正则表达式可以匹配所有带正负号或无符号的整数以及小数。

```
"^[+-]?((\\d+\\.?\\d*)|(\\.\\d+))$"
```

上面一段正则匹配可以匹配两种形式，第一种是整数或小数，其中整数可以允许形如`123.`这种形式，另一种是省略写法，如`.123`的形式。 这两种特别的形式都会经常出现在现代的编程语言中，所以我们把这个纳入考虑。

以下的正则可以匹配科学计数法

```
"^[+-]?(([1-9]\\.?)|([1-9]\\.\\d*))[Ee]([+-]?\\d+)$"
```

该正则只会匹配标准形式的科学计数法，即一个数 a 与 10 的 n 次幂相乘的形式。 其中，$1 \leq |a| < 10, n$ 为整数。 其他形如 0.1×10^5, 12×10^5, $9 \times 10^{2.34}$ 的形式均不接受，且会视为不标准的输入。

第二个需要解决的问题是输入的统一化。 即当既输入整（小）数，又输入了科学计数法时，我们需要统一格式，来方便运算。 在这里，我们将所有的输入统一成小数（或整数），即要从科学计数法转化成小数。

解决这一步只需要移动移动小数点即可。 幂指数过大的情况则只需在数字的前方或后方补0即可。

第三个需要解决的问题是大数相乘。 当相乘的两个数和结果在 `long long` 或 `double` 范围内时，可以直接将输入转化成这两种类型然后相乘。 但一旦超出，我们就需要用字符串手动演化一个乘法器。 为了本方法具有普遍性，我们一律使用乘法器进行计算。 乘法器本质是人们手算乘法所需要的步骤，即每一位与另一个数相乘然后累加起来。

第四个需要解决的问题是输出的格式化. 即我们应该避免像 `00001` 或者是 `1.200000` 这种首尾带有很多 0 的情况. 因此, 我们专门写了一个格式化输出的函数, 专门用于格式化输出, 即去掉首尾多余的 0.

PART 2 Code

判断是否是整（小）数或科学计数法

```
bool isNumber(string s) {
    regex r("^[+-]?((\\d+\\.?\\d*)|(\\.\\d+))$");
    return regex_match(s, r);
}

bool isScientific(string s) {
    regex r("^[+-]?(([1-9]\\.?)|([1-9]\\.\\d*))[Ee]([+-]?\\d+)$");
    return regex_match(s, r);
}
```

将科学计数法转化为整（小）数

```
string scientific2Int(string s) {
    smatch results;
    regex r("^([+-]?)(([1-9]\\.?)|([1-9]\\.\\d*))[Ee]([+-]?\\d+)$");
    string part1;
    long long part2;

    if (regex_match(s, results, r)) {
        part1 = results[2];
        part2 = stol(results[5]);
    }

    if (part1.find(".") < part1.length()) {
        part1.erase(1, 1);
    }

    if (part2 < 0) {
        part1.insert(0, -part2, '0');
        part1.insert(1, ".");
    } else {
        if (part2 >= part1.length()) {
            part1.append(part2 - part1.length() + 1, '0');
        } else {
            part1.insert(part2 + 1, ".");
        }
    }

    if (results[1] == "-")
        part1.insert(0, "-");

    return part1;
}
```

大数相乘

```
string multiply(int *a1, int *a2, int a1_length, int a2_length) {
    int *result_int = new int[a1_length + a2_length];
```

```
    memset(result_int, 0, sizeof(*result_int) * (a1_length + a2_length));

    for (long long i = a1_length - 1; i >= 0; i--) {
        int remainder = 0;
        int temp[a2_length + 1] = {0};

        for (int j = a2_length - 1, k = a2_length; j >= 0; j--) {
            int r = a1[i] * a2[j] + remainder;
            temp[k] = r % 10;
            remainder = r / 10;
            k--;
        }
        temp[0] = remainder;

        remainder = 0;
        for (int j = a2_length, k = i + a2_length; j >= 0; j--) {
            int r = result_int[k] + temp[j] + remainder;
            result_int[k] = r % 10;
            remainder = r / 10;
            k--;
        }
    }

    string result_str;

    for (long long i = 0; i < a1_length + a2_length; i++) {
        result_str.append(to_string(result_int[i]));
    }

    return result_str;
}
```

格式化输出

```
string format(string s) {
    long long count = 0;
    for (long long i = 0; i < s.length(); i++) {
        if (s[i] == '0')
            count += 1;
        else
            break;
    }
    if (count == s.length()) return "0";

    long long start_0 = 0;
    for (long long i = 0; i < s.length(); i++) {
        if (s[i] == '0' && s[i + 1] != '.') {
            start_0 += 1;
        } else {
            break;
        }
```

```
        }
        s.erase(0, start_0);

        if (s.find(".") > s.length())
            return s;
        else {
            long long end_0 = 0;
            for (long long i = s.length() - 1; i >= 0; i--) {
                if (s[i] == '0')
                    end_0 += 1;
                else
                    break;
            }

            s.erase(s.length() - end_0, end_0);

            if (s[s.length() - 1] == '.') s.erase(s.length() - 1, 1);

            return s;
        }
    }
```

PART 3 Result and Verification

输入为非数字或非标准输入

```
(base) $ ./mul a 2
First input is Not a Number or Not Standard!
```

```
(base) $ ./mul 20e2 3
First input is Not a Number or Not Standard!
```

```
(base) $ ./mul 2e2 .3.
Second input is Not a Number or Not Standard!
```

```
(base) $ ./mul 2e1.78 3
First input is Not a Number or Not Standard!
```

标准输入样例及结果

```
(base) $ ./mul 2.3455 89
2.3455 * 89 = 208.7495
```

```
(base) $ ./mul +2.3455 -89
+2.3455 * -89 = -208.7495
```

```
(base) $ ./mul +2.1234e-12 -5.66e+11
+2.1234e-12 * -5.66e+11 = -1.2018444
```

```
(base) $ ./mul 0 1
0 * 1 = 0
```

PART 4 Difficulties and Solutions

大部分问题都在第一部分提出与分析, 这里便再进行总结以下.

对于识别是否是整(小)数和科学计数法, 使用了正则表达式, 这样既方便了匹配也方便了提取对应的部分.

对于输入不统一的情况, 一律转化成了整(小)数进行计算, 并且可以用同一个接口进行运算.

对于大数相乘, 使用 `long long` 也会遇到超出最大范围的情况, 因此使用了字符串相乘, 即手写乘法器的方法.

最后对于格式化输出, 我们专门写一个格式化输出的函数进行格式化.