CS205 C/C++ Program Design Project1

Name: 眭和(SUI He)

SID: 12012929

Part 1 - Analysis

At first, we can simply produce a simple program to satisfy the basic requirement:

```cpp
#include <iostream>

using namespace std;
int main()
{
    cout << "Please input two integers" << endl;
    int num1, num2;
    cin >> num1 >> num2;
    int result = num1 * num2;
    cout << num1 << " * " << num2 << " = " << result << endl;
    return 0;
}
```

Use `string` to save the input

However, when the input become larger and larger, we can not save the number by `int`, `long long`. Then I think I can use `string` to save the input number. The maximum length of `string` is large enough for us to save almost all the numbers in our daily life.

Deal with command line arguments

If the numbers are entered from command line arguments, the programe also should handle them. To achieve this, I can judge whether `argc < 3`. If it returns true, it means the user didn't enter two numbers in the command line, then I need to let user input numbers in the program. Else, the program read the two numbers from command line by `num1 = argv[1]` and `num2 = argv[2]`.

Judge whether the input is negative

And then, if I use `string`, I need to judge whether the input number is negative. I can get the first `char` from the number, and judge whether it is equal to the minus sign (`num[0] == '-'`). Besides, I need to have a `bool` variable to save the return value. And if the number is negative, I will use `.substr(1)` to remove the minus sign so that I can easily determine whether there is other non-numeric character in the input.

Check whether the input is legal

At this step, I need to judge whether the input has illegal character. Because I've already remove the minus sign before. If it is a number, then all the digits needs to be number, or it must contains illegal character. Therefore, we need to check every `char` whether it correspond to a number. The char value of 0 to 9 correspond to 48 to 57. Therefore, I only need to see if it is in this range.

```
static bool isDigit(string num)
{
    for (char m : num)
    {
        if (!(m >= 48 && m <= 57))
            return false;
    }
    return true;
}
```

Re-input the wrong number

If the input is illegal, I will output such as "The first input is not correct, please re-input the first integer" to let the user re-input the number. This process will be set in a `while` loop. Only the user input the right format of the numbers will exit the loop. This can promise all the inputs are numbers. It is an important premise of the subsequent calculation.

Remove high digits Zeros

In consideration of the users may input many zeros before the number such as `000001234567890`. If there are many zeros and I do not remove the zeros before it, it will take more time and more space to calculate the result.

Therefore, I search the number from left to right until I find the first non-zero number, and record the position. Then use `.substr()` to remove the before zeros.

Multiplication

This is the most important part of the program. Firstly I created three new arrays. Two of them used to save all the digits of the two input number. And another array used to save the result. The length of the result array is the sum of the length of two numbers. This will guarantee the array is long enough to save the result.

```
int m[num1.length()], n[num2.length()];
int resultMaxLength = num1.length() + num2.length();
int result[resultMaxLength] = {};
for (int i = 0; i < num1.length(); i++)
     m[num1.length() - i - 1] = num1[i] - '0';
for (int j = 0; j < num2.length(); j++)
    n[num2.length() - j - 1] = num2[j] - '0';
```

After that, I need to multiply these two numbers. I used the method we learned in primary school: the first number is multiplied by each digit of the second number and write the result on the appropriate place and then add them together. The following code is the implementation.

```
for (int j = 0; j < num2.length(); j++)
{
    for (int i = 0; i < num1.length(); i++)
    {
        result[i + j] += m[i] * n[j];
        int counter = 0;
        while (result[i + j + counter] > 9)
        {
            result[i + j + 1] += result[i + j] / 10;
            result[i + j] %= 10;
            counter++;
        }
    }
}
```

Print out the result

At last, I need to print out the result. At this time, I need to consider whether the result is positive or negative. If the input numbers have same sign, then the result is positive, else, the result is negative. If the result is negative, we need to print out a `-` first. And then print out the result from higher digit to lower digit. And our program is finish.

Part2 - Code

```
#include <iostream>
#include <string>

using namespace std;
static bool isNegative(string);
static bool isDigit(string);
static string removeHighDigitZeros(string);
static void multiplication(string, string, bool, bool);

int main(int argc, char *argv[])
```

```
{
    string num1, num2;
    if (argc < 3)
    {
        cout << "Please input two integers" << endl;
        cin >> num1 >> num2;
    }
    else
    {
        num1 = argv[1];
        num2 = argv[2];
    }

    bool num1Neg = isNegative(num1);
    bool num2Neg = isNegative(num2);

    // Remove the "-"
    if (num1Neg)
        num1 = num1.substr(1);
    if (num2Neg)
        num2 = num2.substr(1);

    bool num1IsDigit = isDigit(num1);
    bool num2IsDigit = isDigit(num2);

    while (!(num1IsDigit && num2IsDigit))
    {
        if (!num1IsDigit && !num2IsDigit)
        {
            cerr << "The inputs are not correct, please re-input two
integers" << endl;
            cin >> num1 >> num2;
            num1Neg = isNegative(num1);
            num2Neg = isNegative(num2);
            if (num1Neg)
                num1 = num1.substr(1);
            if (num2Neg)
                num2 = num2.substr(1);
            num1IsDigit = isDigit(num1);
            num2IsDigit = isDigit(num2);
        }
        else if (!num1IsDigit)
        {
            cerr << "The first input is not correct, please re-input the
first integer" << endl;
            cin >> num1;
            num1Neg = isNegative(num1);
            if (num1Neg)
                num1 = num1.substr(1);
            num1IsDigit = isDigit(num1);
        }
        else
        {
            cerr << "The second input is not correct, please re-input the
second integer" << endl;
            cin >> num2;
            num2Neg = isNegative(num2);
            if (num2Neg)
```

```
                num2 = num2.substr(1);
            num2IsDigit = isDigit(num2);
        }
    }
    num1 = removeHighDigitZeros(num1);
    num2 = removeHighDigitZeros(num2);

    multiplication(num1, num2, num1Neg, num2Neg);
    return 0;
}

static bool isNegative(string num)
{
    return num[0] == '-';
}

static bool isDigit(string num)
{
    for (char m : num)
    {
        if (!(m >= 48 && m <= 57))
            return false;
    }
    return true;
}

static string removeHighDigitZeros(string num)
{
    if (num.length() == 1)
        return num;
    int flag = 0;
    for (char m : num)
    {
        if (m == '0')
            flag++;
        else
            break;
    }
    if (flag > 0)
        num = num.substr(flag);
    return num;
}

static void multiplication(string num1, string num2, bool num1Neg, bool
num2Neg)
{
    int m[num1.length()], n[num2.length()];
    int resultMaxLength = num1.length() + num2.length();
    int result[resultMaxLength] = {};
    for (int i = 0; i < num1.length(); i++)
        m[num1.length() - i - 1] = num1[i] - '0';
    for (int j = 0; j < num2.length(); j++)
        n[num2.length() - j - 1] = num2[j] - '0';

    //calculate the result by the way people mainly used

    for (int j = 0; j < num2.length(); j++)
    {
```

```
123         for (int i = 0; i < num1.length(); i++)
124         {
125 
126             result[i + j] += m[i] * n[j];
127 
128             int counter = 0;
129             while (result[i + j + counter] > 9)
130             {
131                 result[i + j + 1] += result[i + j] / 10;
132                 result[i + j] %= 10;
133                 counter++;
134             }
135         }
136     }
137 
138     int flag = 0;
139     for (int k = resultMaxLength - 1; k >= 0; k--)
140     {
141         if (result[k] != 0)
142         {
143             flag = k;
144             break;
145         }
146     }
147 
148     // check whether the result is 0
149     bool resultIsZero = true;
150     for (int l = flag; l >= 0; l--)
151     {
152         if (result[l] != 0)
153         {
154             resultIsZero = false;
155             break;
156         }
157     }
158 
159     //print the result
160     cout << "The result is: ";
161     if (num1Neg != num2Neg && !resultIsZero)
162         cout << "-";
163     for (int l = flag; l >= 0; l--)
164         cout << result[l];
165     cout << endl;
166 }
```

Part 3 - Result & Verification

Test case #1: Basic requirement

Input: 2 3

Output: 6

```
Please input two integers
2 3
The result is: 6
```

Test case #2: Both are positive numbers and not larger than max value of `int`

Input: 39408 9344

Output: 368228352

```
Please input two integers
39408 9344
The result is: 368228352
```

Test case #3: Input numbers are larger than `long long`

Input: 17483940392948932819849320 98 8901893740193874510384729134 21

Output: 1556401795379149355919759812305423606913543029632917887258

```
Please input two integers
17483940392948932819849320 98  8901893740193874510384729134 21
The result is: 1556401795379149355919759812305423606913543029632917887258
```

Test case #4: One of the input numbers is negative

Input: 382759 -174839574

Output: -66921420504666

```
Please input two integers
382759  -174839574
The result is: -66921420504666
```

Test case #5: Both of the input numbers are negative

Input: -748928 -27482924

Output: 20582731305472

```
Please input two integers
-748928 -27482924
The result is: 20582731305472
```

Test case #6: The first input is non-integer

Input: 472a893 -71839201

Output: The first input is not correct, please re-input the first integer

Re-input the first integer: -174028

Output: 12502032471628

```
Please input two integers
472a893 -71839201
The first input is not correct, please re-input the first integer
-174028
The result is: 12502032471628
```

Test case #7: The second input is non-integer

Input: 47281 37&234

Output: The second input is not correct, please re-input the second integer

Re-input the second integer: 829129

Output: 39202048249

```
Please input two integers
47281 37&234
The second input is not correct, please re-input the second integer
829129
The result is: 39202048249
```

Test case #8: Both of the inputs are non-integer

Input: -372.7492 38(3824

Output: The inputs are not correct, please re-input two integers

Re-input two integer: 38293 -928462

Output: -35553595366

```
Please input two integers
-372.7492 38(3824
The inputs are not correct, please re-input two integers
38293 -928462
The result is: -35553595366
```

Test case #9: Input non-integer several times

Input: 73829 ^73824!

Output: The second input is not correct, please re-input the second integer

Re-input the second integer: 8374(61738

Output: The second input is not correct, please re-input the second integer

Re-input the second integer: 3829463

Output: 282725423827

```
Please input two integers
73829 ^73824!
The second input is not correct, please re-input the second integer
8374(61738
The second input is not correct, please re-input the second integer
3829463
The result is: 282725423827
```

Test case #10: There are zeros before the number

Input: 000037284 328473

Output: 12246787332

```
Please input two integers
000037284 328473
The result is: 12246787332
```

Test case #11: One of the input is `0`

Input: 0 378247682

Output: 0

```
Please input two integers
0 378247682
The result is: 0
```

Test case #12: One of the input is `-0`

Input: 3782874 -0

Output: 0

```
Please input two integers
3782874 -0
The result is: 0
```

Test case #13: Deal with command line arguments

Input: 292473 -8272947

Output: -2419613627931

```
sui@LAPTOP-2J3CU669:/mnt/d/SUSTech/课程/CS205 CC++程序设计/Project1$ ./mul 292473  -8272947
The result is: -2419613627931
```

Test case #14: Command line argument is non-integer

Input: 37274*93 327309

Output: The first input is not correct, please re-input the first integer

Re-input the first integer: 838492

Output: 274445978028

```
sui@LAPTOP-2J3CU669:/mnt/d/SUSTech/课程/CS205 CC++程序设计/Project1$ ./mul 37274*93 327309
The first input is not correct, please re-input the first integer
838492
The result is: 274445978028
```

Part 4 - Difficulties & Solutions

1. When I deal with the process of multiplication, I should have used the `int[]` arrays. But I unconsciously invoked the `char[]` arrays. This mistake makes the result very large and very strange. I read the code very careful again and find the wrong place and change it right.
2. When dealing with the carry bit. If the number are really large, and if I plus all the digits at the end of the calculation. Some bits may cause over-flow. Therefore, I choose to deal with the carry bit immediately after one digit is calculated. This will prevent the situation mentioned above.

```
1    for (int j = 0; j < num2.length(); j++)
2    {
3        for (int i = 0; i < num1.length(); i++)
4        {
```

```
            result[i + j] += m[i] * n[j];

            int counter = 0;
            while (result[i + j + counter] > 9)
            {
                result[i + j + 1] += result[i + j] / 10;
                result[i + j] %= 10;
                counter++;
            }
        }
    }
```

3. In most cases, the high digit of the result might be zero, when output the result, I should not print these zeros. Therefore, I need to find the first non-zero digit and record it. And when output the result, I can print it from the record location to the lowest digit.
4. I noticed that if user input `-0`, and if another input is positive, the output might be `-0`. This is not the usual way to write `0`. Therefore, I let the program to detect whether the answer is 0. If answer is 0, the `-` won't be printed out.

```
    int flag = 0;
    for (int k = resultMaxLength - 1; k >= 0; k--)
    {
        if (result[k] != 0)
        {
            flag = k;
            break;
        }
    }

    // check whether the result is 0
    bool resultIsZero = true;
    for (int l = flag; l >= 0; l--)
    {
        if (result[l] != 0)
        {
            resultIsZero = false;
            break;
        }
    }
```