

# CS205 C/ C++ Program Design

## Assignment 1

Name: 王奕童

SID: 11910104

### Part 1. Source Code

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#include <ctype.h>

using namespace std;

static inline bool isDigit(string s);
static inline void bigIntegerMultiply(string big1,string big2);
static inline string clearBeforeZero(string numStr);

int main()
{
    cout<<"Please input two integers"<<endl;
    string numA,numB;
    cin >> numA;

    bool aNegative=numA[0]=='-';
    if(aNegative){
        numA=numA.substr(1);
    }
    numA=clearBeforeZero(numA);
    if(!isDigit(numA)){
        cerr << "The first input is not correct. Exit..." <<endl;
        return 0;
    }

    cin >> numB;
    bool bNegative=numB[0]=='-';
    if(bNegative){
        numB=numB.substr(1);
    }
```

```

numB=clearBeforeZero(numB);
if(!isDigit(numB)){
    cerr << "The second input is not correct. Exit..." <<endl;
    return 0;
}

if(aNegative^bNegative){
    cout<<"-";
}
bigIntegerMultiply(numA,numB);

return 0;
}
bool isDigit(string s)
{
    for (int i = 0; i < s.length(); ++i) {
        if(!isdigit(s[i])){ //The function to check if a char can
be transformed into a digital number.
            return false;
        }
    }
    return true;
}

string clearBeforeZero(string numStr)
{
    int noneZero=0;
    for (int i = 0; i < numStr.length(); ++i) {
        if(numStr[i]!='0'){
            noneZero=i;
            break;
        }
    }
    return numStr.substr(noneZero);
}

void bigIntegerMultiply(string big1,string big2)
{
    int intArray1[big1.length()],intArray2[big2.length()];
    for (int i = 0; i < big1.length(); ++i) {
        intArray1[i]=big1[i]-'0';
    }
    for (int i = 0; i < big2.length(); ++i) {
        intArray2[i]=big2[i]-'0';
    }
}

```

```

    }
    int resultArray[big1.length()+big2.length()];
    memset(resultArray,0, sizeof(resultArray));
    for (int i = big1.length()-1; i >= 0; --i) {
        for (int j = big2.length()-1; j >= 0; --j) {
            resultArray[i+j]+=intArray1[i]*intArray2[j];
        }
    }

    for (int k = big1.length()+big2.length()-1; k > 0; --k) {
        int upper=resultArray[k]/10;
        resultArray[k]=resultArray[k]%10;
        resultArray[k-1]+=upper;
    }
    bool isZero= true;
    for (int l = 0; l < big1.length()+big2.length()-1; ++l) {
        if(resultArray[l]!=0){
            isZero= false;
        }
        if(!isZero){
            cout<<resultArray[l];
        }
    }
    if(isZero){
        cout<<"0";
    }
}

```

## Part 2. Result & Verification

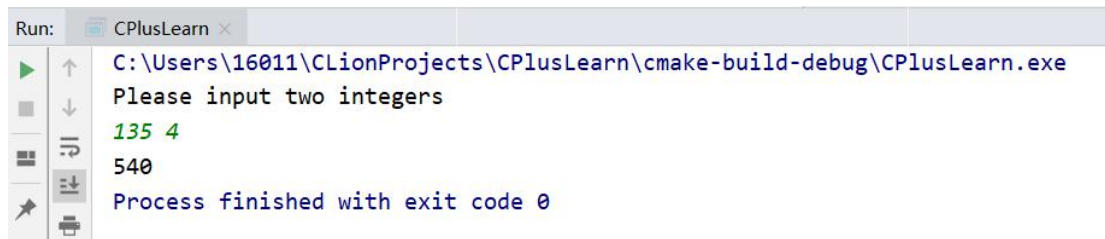
In this part, you should present the result of your program by listing the output of test cases and optionally add a screen-shot of the result.

**【Test case #1】 Both are positive numbers and neither is larger than the max value of int.**

Input: 135 4

Output: 540

Screen-short for case #1:



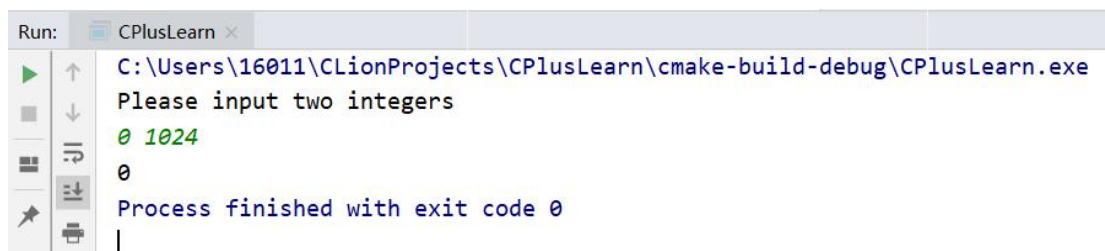
```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
135 4
540
Process finished with exit code 0
```

## 【Test case #2】 One of the numbers is 0.

Input: 0 1024

Output: 0

Screen-short for case #2:



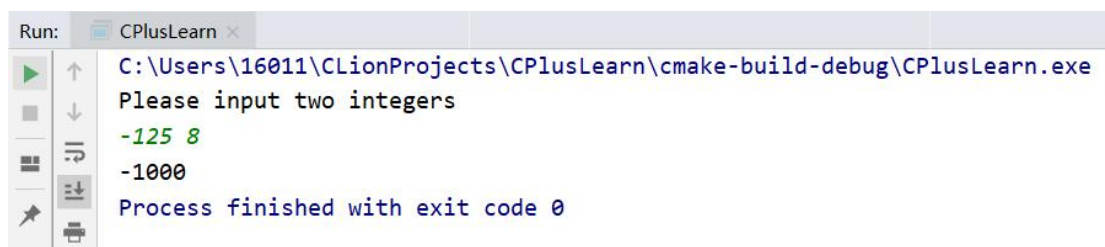
```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
0 1024
0
Process finished with exit code 0
```

## 【Test case #3】 One of the numbers is negative.

Input: -125 8

Output: -1000

Screen-short for case #3:



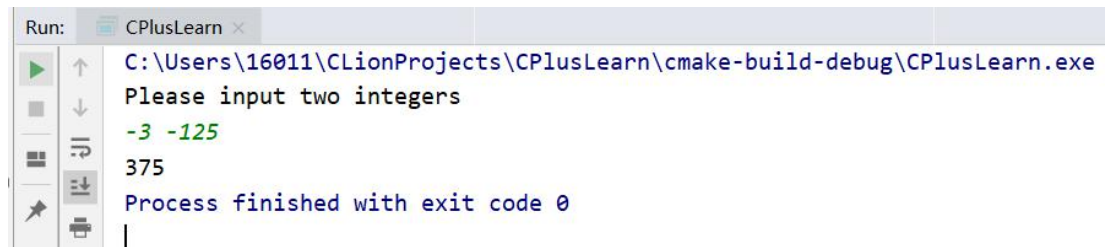
```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
-125 8
-1000
Process finished with exit code 0
```

## 【Test case #4】 Both are negative numbers.

Input: -3 -125

Output: 375

Screen-short for case #4:



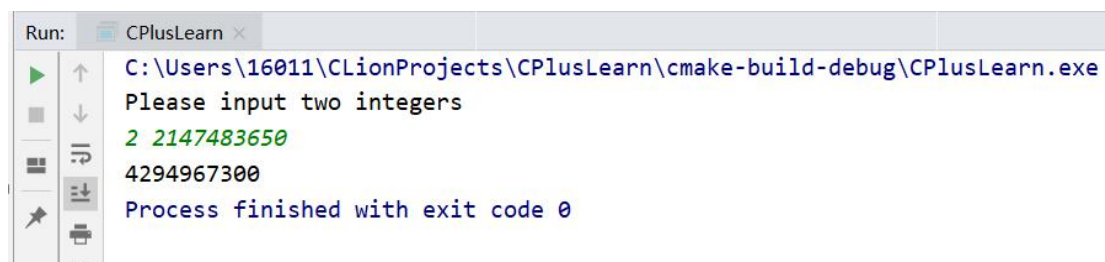
```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
-3 -125
375
Process finished with exit code 0
```

**【Test case #5】 One of the numbers is bigger than the max value of int(2147483647).**

Input: 2 2147483650

Output: 4294967300

Screen-short for case #5:



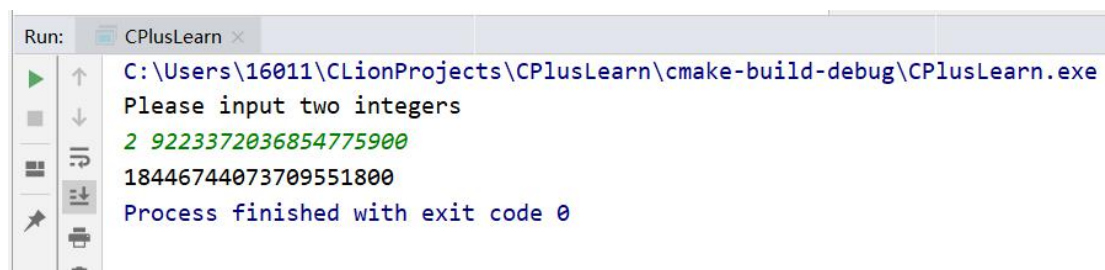
```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
2 2147483650
4294967300
Process finished with exit code 0
```

**【Test case #6】 One of the numbers is bigger than the max value of long long(9223372036854775807).**

Input: 2 9223372036854775900

Output: 184467440737095518004294967300

Screen-short for case #6:



```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
2 9223372036854775900
18446744073709551800
Process finished with exit code 0
```

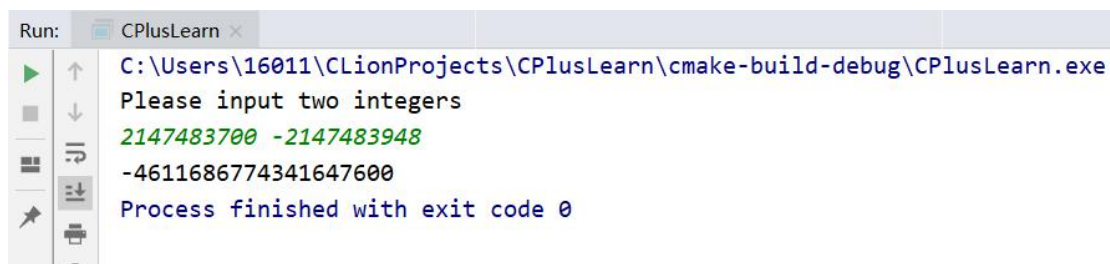
**【Test case #7】 One of the numbers is bigger than the**

max value of int(2147483647), while another is smaller than the min value of int(-2147483648).

Input: 2147483700 -2147483948

Output: -4611686774341647600

Screen-short for case #7:



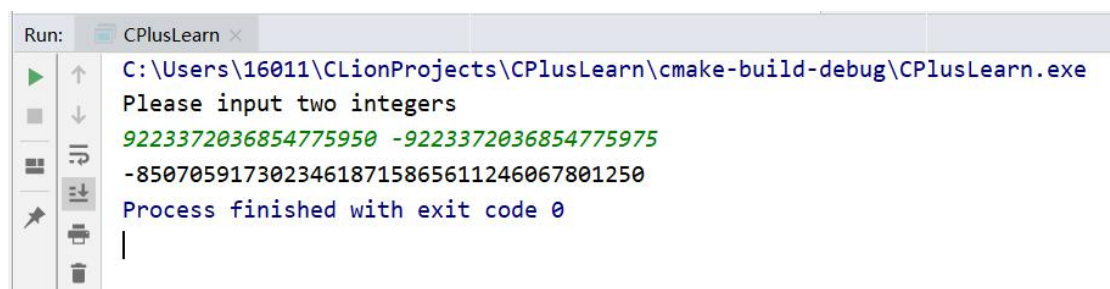
```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
2147483700 -2147483948
-4611686774341647600
Process finished with exit code 0
```

**【Test case #8】** One of the numbers is bigger than the max value of long long(9223372036854775807), while another is smaller than the min value of long long(-9223372036854775808).

Input: 9223372036854775950 -9223372036854775975

Output: -85070591730234618715865611246067801250

Screen-short for case #8:



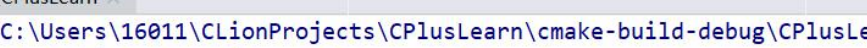
```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
9223372036854775950 -9223372036854775975
-85070591730234618715865611246067801250
Process finished with exit code 0
|
```

**【Test case #9】** The first input is not a digital number.

Input: 123a 2

Output: The first input is not correct. Exit...

Screen-short for case #9:




```
Run: CPlusLearn x
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
123a 2
The first input is not correct. Exit...
Process finished with exit code 0
```

**【 Test case #10 】 The second input is not a digital number.**

Input: -3 456b

Output: The second input is not correct. Exit...

Screen-short for case #10:



```
Run: CPlusLearn.exe
C:\Users\16011\CLionProjects\CPlusLearn\cmake-build-debug\CPlusLearn.exe
Please input two integers
-3 456b
The second input is not correct. Exit...
Process finished with exit code 0
```

**【Test case #11】** There are some zeros before the real value.

Input:

[illegible]

Output: 369

Screen-short for case #11:

[illegible]

## Part 3. Difficulties & Solutions, or others

1.The 1<sup>st</sup> edition code: (only satisfying the first requirement)

```
#include <iostream>
#include <cstring>
#include <math.h>
using namespace std;

int main() {
    cout<<"Please input two integers"<<endl;
    int a, b;
    cin >> a >> b;
    cout << a * b << endl;

    return 0;
}
```

2.After seeing the 2<sup>nd</sup> requirement, I tried to check the input and print some error message if the input is incorrect:

```
#include <iostream>
#include <cstring>
#include <math.h>
using namespace std;

int main() {
    cout << "Please input two integers" << endl;
    int a, b;
    cin >> a;
    if(!cin){
        cerr << "The first input is not correct. Exit..." << endl;
        return 0;
    }
    cin >> b;
    if(!cin){
        cerr<<"The second input is not correct. Exit..."<<endl;
        return 0;
    }
    cout << a * b << endl;
    return 0;
}
```



3. After seeing the 3<sup>rd</sup> requirement, I changed my number type into long long to pass the case:

```
#include <iostream>
#include <cstring>
#include <math.h>
using namespace std;

int main() {
    cout << "Please input two integers" << endl;
    long long a, b;
    cin >> a;
    if(!cin){
        cerr << "The first input is not correct. Exit..." << endl;
        return 0;
    }
    cin >> b;
    if(!cin){
        cerr<<"The second input is not correct. Exit..."<<endl;
        return 0;
    }
    cout << a * b << endl;
    return 0;
}
```

4. In order to calculating much larger numbers, I changed into using string to store the input. What's more, I also design a function to judge if the string is consist of digital number. (Tips: isdigit(int \_C) is a function in ctype.h)

```
static inline bool isDigit(string s);
bool isDigit(string s)
{
    for (int i = 0; i < s.length(); ++i) {
        if(!isdigit(s[i])){ //The function to check if a char can
be transformed into a digital number.
            return false;
        }
    }
    return true;
}
```

5.To calculate, the function called bigIntegerMultiply is also designed. I mainly designed this function based on the calculating method learned in our childhood. The most interesting is that the higher digit of the number, the lower index of char in the string. This is due to the difference between the habits of reading and calculating. In this function, the carry addition is also paid attention to. I used an int array to represent the number.

```
static inline void bigIntegerMultiply(string big1,string big2);
void bigIntegerMultiply(string big1,string big2)
{
    int intArray1[big1.length()],intArray2[big2.length()];
    for (int i = 0; i < big1.length(); ++i) {
        intArray1[i]=big1[i]-'0';
    }
    for (int i = 0; i < big2.length(); ++i) {
        intArray2[i]=big2[i]-'0';
    }
    int resultArray[big1.length()+big2.length()];
    memset(resultArray,0, sizeof(resultArray));
    for (int i = big1.length()-1; i >= 0; --i) {
        for (int j = big2.length()-1; j >= 0; --j) {
            resultArray[i+j]+=intArray1[i]*intArray2[j];
        }
    }

    for (int k = big1.length()+big2.length()-1; k > 0; --k) {
        int upper=resultArray[k]/10;
        resultArray[k]=resultArray[k]%10;
        resultArray[k-1]+=upper;
    }
    bool isZero= true;
    for (int l = 0; l < big1.length()+big2.length()-1; ++l) {
        if(resultArray[l]!=0){
            isZero= false;
        }
        if(!isZero){
            cout<<resultArray[l];
        }
    }
    if(isZero){
        cout<<"0";
    }
}
```

6. After that, I find this design cannot satisfy the need of calculating negative numbers. So in the main() function, I also add some statements to satisfy this requirement. And this problem is solved as well. After modified, the main() function is:

```
int main()
{
    cout<<"Please input two integers"<<endl;
    string numA,numB;
    cin >> numA;

    bool aNegative=numA[0]=='-';
    if(aNegative){
        numA=numA.substr(1);
    }
    if(!isDigit(numA)){
        cerr << "The first input is not correct. Exit..." <<endl;
        return 0;
    }

    cin >> numB;
    bool bNegative=numB[0]=='-';
    if(bNegative){
        numB=numB.substr(1);
    }
    if(!isDigit(numB)){
        cerr << "The second input is not correct. Exit..." <<endl;
        return 0;
    }

    if(aNegative^bNegative){
        cout<<"-";
    }
    bigIntegerMultiply(numA,numB);

    return 0;
}
```

7. Then, I also find that the user may input the numbers that have 0s before the real value. So I designed a function to clear these unnecessary 0s.

```
static inline string clearBeforeZero(string numStr);
string clearBeforeZero(string numStr){
    int noneZero=0;
    for (int i = 0; i < numStr.length(); ++i) {
        if(numStr[i]!='0'){
```

```

        noneZero=i;
        break;
    }
}
return numStr.substr(noneZero);
}

```

8.Finally, the programme can satisfy the requirement of two integers' multiply.

```

#include <stdio.h>
#include <iostream>
#include <string.h>
#include <ctype.h>

using namespace std;

static inline bool isDigit(string s);
static inline void bigIntegerMultiply(string big1,string big2);
static inline string clearBeforeZero(string numStr);

int main()
{
    cout<<"Please input two integers"<<endl;
    string numA,numB;
    cin >> numA;

    bool aNegative=numA[0]=='-';
    if(aNegative){
        numA=numA.substr(1);
    }
    numA=clearBeforeZero(numA);
    if(!isDigit(numA)){
        cerr << "The first input is not correct. Exit..." <<endl;
        return 0;
    }

    cin >> numB;
    bool bNegative=numB[0]=='-';
    if(bNegative){
        numB=numB.substr(1);
    }
    numB=clearBeforeZero(numB);
    if(!isDigit(numB)){
        cerr << "The second input is not correct. Exit..." <<endl;
        return 0;
    }
}

```

```

    }

    if(aNegative^bNegative){
        cout<<"-";
    }
    bigIntegerMultiply(numA,numB);

    return 0;
}
bool isDigit(string s)
{
    for (int i = 0; i < s.length(); ++i) {
        if(!isdigit(s[i])){ //The function to check if a char can
        be transformed into a digital number.
            return false;
        }
    }
    return true;
}

string clearBeforeZero(string numStr)
{
    int noneZero=0;
    for (int i = 0; i < numStr.length(); ++i) {
        if(numStr[i]!='0'){
            noneZero=i;
            break;
        }
    }
    return numStr.substr(noneZero);
}

void bigIntegerMultiply(string big1,string big2)
{
    int intArray1[big1.length()],intArray2[big2.length()];
    for (int i = 0; i < big1.length(); ++i) {
        intArray1[i]=big1[i]-'0';
    }
    for (int i = 0; i < big2.length(); ++i) {
        intArray2[i]=big2[i]-'0';
    }
    int resultArray[big1.length()+big2.length()];
    memset(resultArray,0, sizeof(resultArray));
    for (int i = big1.length()-1; i >= 0; --i) {

```

```

        for (int j = big2.length()-1; j >= 0; --j) {
            resultArray[i+j]+=intArray1[i]*intArray2[j];
        }
    }

    for (int k = big1.length()+big2.length()-1; k > 0; --k) {
        int upper=resultArray[k]/10;
        resultArray[k]=resultArray[k]%10;
        resultArray[k-1]+=upper;
    }
    bool isZero= true;
    for (int l = 0; l < big1.length()+big2.length()-1; ++l) {
        if(resultArray[l]!=0){
            isZero= false;
        }
        if(!isZero){
            cout<<resultArray[l];
        }
    }
    if(isZero){
        cout<<"0";
    }
}

```

### Summary:

Although it is only a simple problem of solving the multiply of 2 integers, I still need great efforts to satisfy every kinds of needs.