第二次作业——长整数加法

1.编译步骤

```
cd build
cmake ..
make
```

2. 运行步骤

项目1：

`inputFile.txt`文件中两行数字可为`001253`,`+0253`,`-1508`,`-00568`等形式的十进制数字

运行指令：

```
./homework_2 ../data/inputFile.txt
```

运行结果

```
 num_str = -21
lyt@dell-g5:~/WorkSpace/sl_homework/cpp_homework/homework_2/build$ ./homework_2 ../data/inputFile.txt
********长整数加法运算********
采用默认输入输出十进制
输入进制： 10 输出进制： 10
********解析文件中两行数字********
Parse the number : -00021 -> -21
Parse the number : +01568 -> 01568
********在10进制下进行加法运算********
num1 = -21
num2 = 1568
num1 + num2 = 1547
********以10进制输出结果********
 num_str = 1547
```

扩展1：

`inputFile.txt`文件中两行数字可为`001a2`,`+04d`,`-1508`,`-00568`等形式的指定输入进制的数字，若`inputFile.txt`中的数字与输入进制不符会提示。

输入的指定进制可以是2~36之间，默认输出进制与输入进制相同

运行指令：

```
./homework_2 ../data/inputFile.txt 16
```

```
lyt@dell-g5:~/WorkSpace/sl_homework/cpp_homework/homework_2/build$ ./homework_2 ../data/inputFile.txt 16
********长整数加法运算********
输入进制： 16 输出进制： 16
********解析文件中两行数字********
Parse the number : -001a2 -> -1a2
Parse the number : +014d -> 014d
********将16进制转换为10进制********
16->10进制： -1a2 -> -418
16->10进制： 014d -> 333
********在10进制下进行加法运算********
num1 = -418
num2 = 333
num1 + num2 = -85
********以16进制输出结果********
 num_str = -55
```

扩展2：

同理，`inputFile.txt`文件中两行数字可为`001a2`,`+04d`,`-1508`,`-00568`等形式的指定输入进制的数字，若`inputFile.txt`中的数字与输入进制不符会提示。

举例如下：

```
lyt@dell-g5:~/WorkSpace/sl_homework/cpp_homework/homework_2/build$ ./homework_2 ../data/inputFile.txt 8 4
********长整数加法运算********
输入进制: 8 输出进制: 4
********解析文件中两行数字********
Parse the number : -001a2 -> 第1个数字无效
Parse the number : +014d -> 第2个数字无效
文件中含有非法数字，请检查文件
```

输入输出的指定进制可以是2~36之间

运行指令：

```
./homework_2 ../data/inputFile.txt 16 4
```

```
lyt@dell-g5:~/WorkSpace/sl_homework/cpp_homework/homework_2/build$ ./homework_2 ../data/inputFile.txt 16 4
********长整数加法运算********
输入进制: 16 输出进制: 4
********解析文件中两行数字********
Parse the number : -001a2 -> -1a2
Parse the number : +014d -> 014d
********将16进制转换为10进制********
16->10进制: -1a2 -> -418
16->10进制: 014d -> 333
********在10进制下进行加法运算********
num1 = -418
num2 = 333
num1 + num2 = -85
********以4进制输出结果********
 num_str = -1111
```

代码中还对`+0`,`0000`,`+000`,`-0000`类似的数字做了处理，均能满足任务需求。

暂存疑问

目前还有点疑惑就是，不知怎么处理溢出这种情况，比如"+1000000000000000000000"这种输入的长数字。