复数计算器

考察内容：异常和文件读写

题目描述

实现一个复数域上的计算器。复数的形式：$a+bi$ 其中表示实部， 表示虚部，为了简单起见，对复数的形式作如下规定：

- 复数的实部和虚部都是int类型， 即使复数的虚部为0/1，在输入或者输出该复数时，它的形式也应该为 $a+0i$ 或者 $a+1i$。
- 当复数的虚部为负数时，在输入或输出该复数时，其形式为 $a-|b|i$, 例如你应当输出$2-3i$，而非 $2+-3i$。
- 对于0这个数而言，其输入输出格式为 $0+0i$。

需要实现的功能如下：

- 复数的基本运算：加减乘除
- 复数表达式求值，输入复数字符串，输出结果。
- 复数运算的时候可能会出现各种异常，请正确处理并抛出异常信息。

MyException.h

MyException.h不需要实现。

```
#ifndef MYEXCEPTION_H
#define MYEXCEPTION_H
#include <string>
class MyException {
public:
    MyException(std::string s) { eMsg = s; }
    void seteMsg(std::string s) { eMsg = s; }
    std::string what() { return eMsg; }
private:
    std::string eMsg;
};

//找不到文件异常
class FileNotFound : public MyException {
public:
    FileNotFound(std::string s) : MyException(s) {}
};

//溢出异常
class OverFlow : public MyException {
public:
    OverFlow(std::string s) : MyException(s) {}
};

//算数运算错误异常
```

```
class ArithmeticError : public MyException {
public:
    ArithmeticError(std::string s) : MyException(s) {}
};


//非法表达式异常；有兴趣可以实现该异常的捕获和处理
/*
class InvalidExpression : public MyException {
public:
    InvalidExpression(std::string s) : MyException(s) {}
};
*/

#endif
```

Complex.h

Complex.h中定义了需要实现的接口，请在Complex.cpp中实现。

```
#ifndef COMPLEX_H
#define COMPLEX_H
#include <string>
#include <vector>
#include "MyException.h"
#include <iostream>
#include <sstream>
#include <climits>
#include <fstream>
using namespace std;
class Complex
{
private:
    int real; // 实部
    int imag; // 虚部
public:
    Complex() = default;
    Complex(int real, int imag) {
        this->real = real;
        this->imag = imag;
    }
    int getReal(){
        return real;
    }
    int getImag(){
        return imag;
    }
    // TODO 重载+，实现复数的相加


    // TODO 重载-，实现复数的相减


    // TODO 重载*，实现负数的相乘


    // TODO 重载/，实现复数的相除
```

```
    // 重载<<，输出格式为a+bi，a和b分别是复数的实部和虚部,没有空格，不需要输出回车
    // 注意:a或b为复数的输出示例，-1-3i
    friend ostream & operator << (ostream & out, const Complex &c);
    // 重载>>，读取a+bi格式的输入，格式同上。
    friend istream & operator >> (istream & in, Complex & c);
};
#endif
```

MathUtils.h

```
#ifndef MATHUTILS_H
#define MATHUTILS_H
#include "Complex.h"
class MathUtils {
public:
    // 接受一个表达式字符串并返回计算结果。例如:calculator("(1+2i)+(2+3i)")的结果为3+5i
    static Complex calculator(string s);
    // 使用函数calculator处理文本文件，输入文件名为 inputFileName，输出文件名为
outputFileName。
    // 每一行为一个表达式，为了简单起见，你在这个函数中只需要处理FileNOTFound异常，并且完成文
件读写和计算的过程
    static void processTextFile(string inputFilename, string outputFilename);
};

#endif
```

注意事项

1. 上面的每个函数都要检测所有可能出现的异常，并抛出异常。针对每种异常，必须抛出对应的异常类，例如溢 出时，必须抛出 Overflow 异常类。
2. 异常的优先级从大到小为： FileNotFound，ArithmeticError， Overflow，即若有 多个异常，先抛出优先级最高的异常。
3. 异常格式如下：

```
Overflow:

•   Complex add overflow!\n

•   Complex minus overflow!\n

•   Complex multiply overflow!\n

•   Complex divide overflow!\n

FileNotFound:

•   "Cannot open " + inputFilename + "\n"

•   "Cannot open " + outputFilename + "\n"
```

```
ArithmeticError:

•   ArithmeticError!\n
```

4. 溢出说明：
   - C++用INT_MIN来表示int能表示的最小值，INT_MAX表示int能表示的最大值，在头文件 中定义。INT_MIN = -2147483648，INT_MAX = 2147483647。
   - int + int：**两个大于0的int相加结果为负数或者两个小于0的int相加结果为正数**，溢出。
   - int - int：**可以转化为int + (-int)来判断**，但是注意如果第二个int为INT_MIN，则取负之后超过INT_MAX，**此时需要特殊判断：任意>=0的整型加上 -INT_MIN均溢出。**
   - int * int：`sum = int1 * int2`，如果`sum/int1 != int2`，溢出。**需要特殊处理**`int1==0`**的情况，以及**`int1==INT_MIN && int2==-1`**的情况。**
   - int / int：由于复数的分母一定大于等于0，因此**只需要判断/0异常，抛出ArithmeticError。**（如果普通的int / int，需要特判 `INT_MIN/-1` 的情况）。
   - 除法计算过程中的乘法出现溢出报乘法溢出异常。
   - 实部虚部的计算均可能出现异常。
5. 输入表达式进行了简化，即输入的字符串一定满足以下条件：
   - 包含的字符有：0 1 2 3 4 5 6 7 8 9 + - * / i ( )， 保证测试样例中一定不会出现其他字符。
   - 复数必须包含在括号内，且一定只有两个复数进行运算，例如：(1+4i)+(3-4i)
   - 运算符一定位于两个复数中间。
   - 输入的实部和虚部一定在int的范围内，只是计算的时候可能出现溢出。
   - 先抛出除零异常，再抛出溢出异常。
6. 在函数 processFile 中：
   - 输入输出文件都以文本格式储存，每一行为一个合法的复数计算表达式，你需要调用calculator函数计算 返回值，并将返回值储存到输出文件中，每一行为一个返回值。
   - 打开输入或输出文件失败时，应当抛出异常。

调用示例

```
    vector<string> strs(12);
    for(size_t i=0;i<12;++i){
        cin>>strs[i];
    }
    for(size_t i=0;i<12;++i){
        try {
            cout << MathUtils::calculator(strs[i])<<endl;
        }
        catch (ArithmeticError& e) {
            cout << e.what();
        }
        catch (OverFlow& e){
            cout << e.what();
        }
        catch (...) {
            cout << "ERROR" << endl;
        }
    }
```

输入测试样例test1，test2（以注释的形式附在MathUtils.cpp中），执行结果分别如下：

```
//test1
2147483644+0i
2147483647+0i
Complex add overflow!
ArithmeticError!
Complex multiply overflow!
-2147483648+1073741824i
Complex multiply overflow!
Complex multiply overflow!
2+2i
0+2i
Complex multiply overflow!
-2147483645+0i
```

```
//test2
-2+5i
Complex minus overflow!
Complex add overflow!
429496729-429496729i
2147483643+2147483646i
Complex multiply overflow!
-2147483645+2147483646i
2147483647+2147483644i
Complex multiply overflow!
Complex multiply overflow!
Complex add overflow!
2147483647+3i
```

注意：上述输入测试样例的形式只是为了你方便测试，实际上在你提交的文件中不应该有任何cin。

```
try {
        MathUtils::processTextFile("input1.txt", "output.txt"); // "input.txt"不
存在
    }
    catch (FileNotFound& e) {
        cout << e.what();
    }
    catch (...) {
        cout << "ERROR" << endl;
    }
    try {
        MathUtils::processTextFile("input2.txt", "output.txt"); // "input2.txt"存
在
        cout << "True" << endl;
    }
    catch (FileNotFound& e) {
        cout << e.what();
    }
    catch (...) {
        cout << "ERROR" << endl;
    }
```

执行结果：

```
Cannot open input1.txt
True
```

Tips:

在处理复数计算表达式的时候有一些函数可以使用：（当然利用for循环遍历处理也是可行的，而且有可能不容易犯错，相较于substr来说）

表 9.14：string 搜索操作

搜索操作返回指定字符出现的下标，如果未找到则返回 npos。

s.find(*args*)	查找 s 中 *args* 第一次出现的位置
s.rfind(*args*)	查找 s 中 *args* 最后一次出现的位置
s.find_first_of(*args*)	在 s 中查找 *args* 中任何一个字符第一次出现的位置。
s.find_last_of(*args*)	在 s 中查找 *args* 中任何一个字符最后一次出现的位置
s.find_first_not_of(*args*)	在 s 中查找第一个不在 *args* 中的字符
s.find_last_not_of(*args*)	在 s 中查找最后一个不在 *args* 中的字符

续表

***args* 必须是以下形式之一**	
c,pos	从 s 中位置 pos 开始查找字符 c。pos 默认为 0
s2,pos	从 s 中位置 pos 开始查找字符串 s2。pos 默认为 0
cp,pos	从 s 中位置 pos 开始查找指针 cp 指向的以空字符结尾的 C 风格字符串。pos 默认为 0
cp,pos,n	从 s 中位置 pos 开始查找指针 cp 指向的数组的前 n 个字符。pos 和 n 无默认值

比如说$(a+bi)+(c+di)$，利用`find_first_of("(")`可以获得右括号第一次出现的位置。

string.substr(pos，len)返回从pos开始长为len的子字符串。

提交说明

1. 提交MyException.h、MathUtils.h、MathUtils.cpp,Complex.h,Complex.cpp五个文件，注意编码方式为 UTF-8，直接打包为 zip 格式压缩包，不要存在多一层的目录。
2. 实现代码请严格按照给定的接口名字，否则不能通过编译。(可以根据实际需要添加其他函数)
3. 提交代码中不要包含main()函数，否则不能通过编译。
4. 严格按照要求的功能实现输出，不要尝试进行其他输入输出活动，否则不能通过测试。