File Operation : Working with external input file and external output file

- **File is an external collection of related data treated as a unit**
  Why do we need files?
  Batching processing vs. interactive processing

- **Read from and Write to external files**
  A stream needs to be created to connect the external files to the program.

  ifstream : input stream connects the input data file to the program
  ofstream : output stream connects the program to the output files

  Input and output streams are defined in <fstream> ➔ #include <fstream>

  - Create output stream – writing to external file
    - If the file does not exist beforehand, it will be created.
    - If the file exists beforehand, all information in the file will be lost after we opened a output stream to it.
    - Make sure the file is properly closed after the information is written to the file.
    - The settings used for formatted output can also be used with the user created output streams

Example 1: Input and output file streams in a program

```
#include <iostream>
#include <fstream>
#include <cassert>
using namespace std;

int main()
{
   float     length;
   float     width;
   float     area;
   ifstream   myIn;
   ofstream  myOut;

   myIn.open("rectangle.data");
   assert(myIn);     // check whether the input file is opened properly

   myOut.open("result");

   myIn >> length >> width;

   area = length * width;

   myOut << "The width of the rectangle is  " << width << endl;
   myOut << "The length of the rectangle is " << length << endl;
   myOut << "The area of the rectangle is    " << area << endl;

   myIn.close();
   myOut.close();

   return 0;
}
```

Example 2: Formatted output used for external file

```
#include <fstream>
#include <cmath>
#include <iomanip>
using namespace std;

int main()
{
    ofstream   outfile;
    int             value=10;

    outfile.open("ex1.result");

    outfile<<fixed << showpoint <<setprecision(2);

    outfile << setw(10) << "Value"  << setw(15)  << "Square"  << setw(15)  << "Squre Root" << endl;

    outfile <<setw(10)<< value<<setw(15)<< pow(double(value), 2.0) << setw(15) << sqrt(double(value))
<< endl;

    outfile.close();

    return 0;
}
```

Example 3: Read numbers from a data file

```
#include <iostream>
#include <fstream>
#include <cassert>
using namespace std;

int main()
{
   float      value;
   int         evenCount=0, oddCount=0;
   ifstream   myIn;

   myIn.open("rectangle.data");
   assert(myIn);     // check whether the input file is opened properly

   cout << "Enter an integer: ";
   while (myIn >> value) {
           if (value%2)
                       oddCount++;
           else
                       evenCount++;

           cout << "Enter an integer: ";
   }

   myIn.close();

   cout << "Total: " << oddCount << " odd numbers << " and " << evenCount << " even numbers." << endl;
   return 0;
}
```