10902 CPP Final Exam

Subject: Triangle Json File

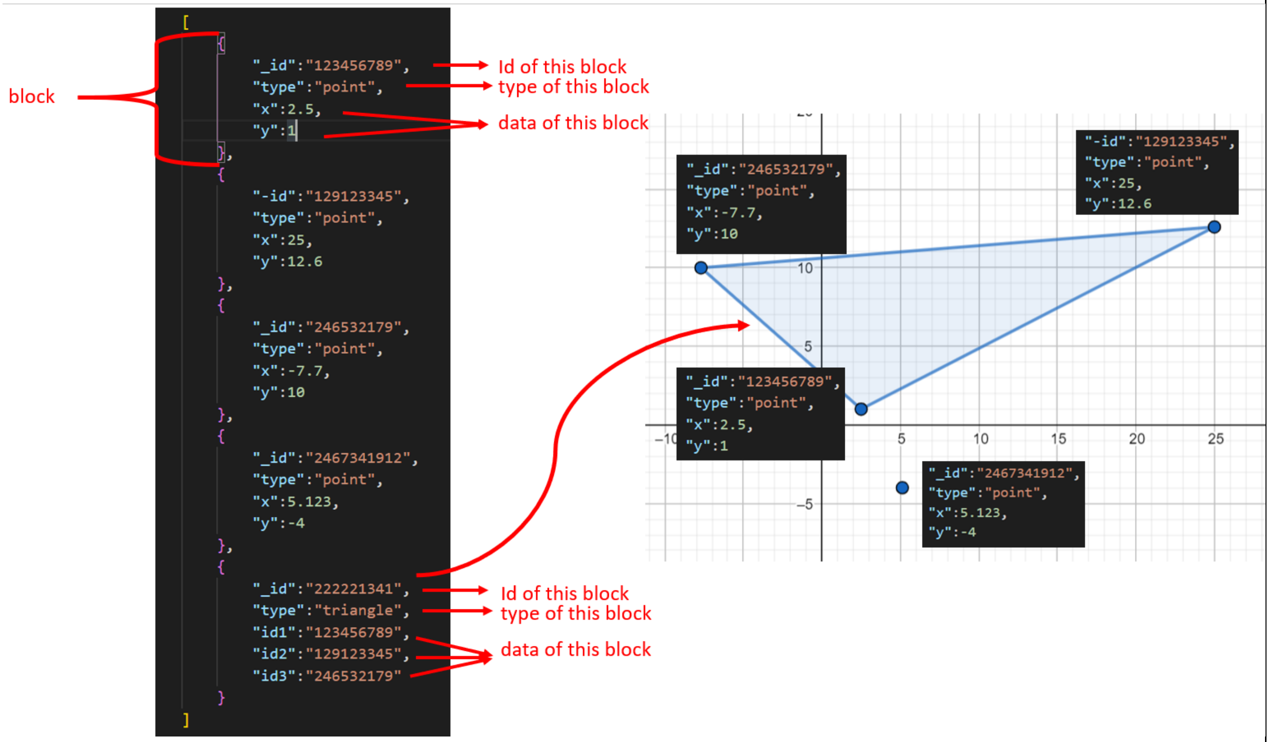
Contributor：Yuan, Wei-Cheng

Main testing concept: File inout

Basics	Functions
■ C++ BASICS	□ SEPARATE COMPILATION AND NAMESPACES
■ FLOW OF CONTROL	■ STREAMS AND FILE I/O
□ FUNCTION BASICS	□ RECURSION
□ PARAMETERS AND OVERLOADING	□ INHERITANCE
□ ARRAYS	□ POLYMORPHISM AND VIRTUAL FUNCTIONS
□ STRUCTURES AND CLASSES	□ TEMPLATES
□ CONSTRUCTORS AND OTHER TOOLS	□ LINKED DATA STRUCTURES
■ OPERATOR OVERLOADING, FRIENDS,AND REFERENCES	□ EXCEPTION HANDLING
■ STRINGS	□ STANDARD TEMPLATE LIBRARY
■ POINTERS AND DYNAMIC ARRAYS	□ PATTERNS AND UML

Description:

CRUD is an acronym for create, read, update, and delete. These are the four basic functions of persistent storage. A pair (name: value) in a JSON file is composed of a name and a value. Each name is followed by a colon, and pairs are separated by a comma. In the sample JSON file, ***test.json,*** containing records of points and triangles in an array, each record is enclosed with curly brackets. Additionally, while points record the raw data of “id”, “type”, and the coordinate of “x”, “y”, triangles record the raw data of “id” and “type” and the index of its three points. In a record, each field is a name-value pair. A name must be a string while a value can be a string, number, boolean, etc. The figure below illustrates the structure of our triangle JSON file.



```
[
    {
        "_id":"123456789",
        "type":"point",
        "x":2.5,
        "y":1
    },
    {
        "-id":"129123345",
        "type":"point",
        "x":25,
        "y":12.6
    },
    {
        "_id":"246532179",
        "type":"point",
        "x":-7.7,
        "y":10
    },
    {
        "_id":"2467341912",
        "type":"point",
        "x":5.123,
        "y":-4
    },
    {
        "_id":"222221341",
        "type":"triangle",
        "id1":"123456789",
        "id2":"129123345",
        "id3":"246532179"
    }
]
```

block
Id of this block
type of this block
data of this block
Id of this block
type of this block
data of this block
"-id":"129123345", "type":"point", "x":25, "y":12.6
"_id":"246532179", "type":"point", "x":-7.7, "y":10
"_id":"123456789", "type":"point", "x":2.5, "y":1
"_id":"2467341912", "type":"point", "x":5.123, "y":-4
10
5
−5
−10
5
10
15
20
25

Given a JSON file containing the record of points and triangles, which has member functions to parse the file, manipulate (RUD) records in the given file, output manipulated records to a JSON file in the same format of input JSON file. Note that the class BasicJSON includes at least the following member functions:

1. **`bool Parse(std::string InputFileName);`** Read a list of records from the file *InputFileName* and parse the contents in the file to construct your data with your selected structure. Also, if the file is parsed successfully, return `true`. Otherwise, return `false`.
2. **`void Write(std::string OutputFileName);`** Output all records to the file *OutputFileName* in JSON format listed above. Do not add tab or space in front of line. see reference in below.
3. **`void Delete(std::string Name);`** Delete a record of the passed `Name`(id). If a point is be deleted, you should remove all triangles which have the deleted point as one of their vertices.
4. **`float TotalArea();`** Return the sum of the area of all triangles in the JSON. If it does not contain any triangle, output "File dont have triangle\n" and return 0. The area of a triangle can be computed using Heron's formula listed at the end of the file.
5. **`overload [] operator`** such that
    1. **JSONobject[*index*]**: an *index* is a non-negative integer for accessing an array element (record).
    2. **JSONobject[*index*][*name*]**: get a value associated with given *index* and *name*.

Please note that:

1. We will provide main.cpp and products.json to test your class. Sample files input-main.cpp, test.json and output.json are shown as an example of testing cases, located under the same directory, e.g.

```
..\
 ├ Q5
   ... ├ CPP 程式設計題-JSON.pdf
       ├ input-main.cpp
       ├ output.json
       └test.json
```

2. No comments are included in the JSON file.
3. The comma at the end of any last pair in an object or array is optional
4. A few redundant spaces and next-line characters are acceptable.
5. All coordinate is float type

Input:

Substitution of your main function.

Output:

Please refer to the sample output.

Sample Input / Output：

Sample Input	Sample Output
According to the given main function and json file	point 5 129123345 37.5 File dont have triangle 0

□ **Easy, Only basic programming syntax and structure are required.**
□**Medium, Multiple programming grammars and structures are required.**
■ **Hard, Need to use multiple program structures or complex data types.**

Expected solving time:

60 minutes

Other notes:

Given main function:

```
int main()
{
    BasicJson json;
    if(json.Parse("Test.json"))
    {
         cout<<json[0]["type"]<<endl;
         cout<<json[0]["x"]<<endl;
         cout<<json[1]["_id"]<<endl;
         json[1]["y"]=20;
         cout<<json.TotalArea()<<endl;
         json.Delete(“129123345”);
         cout<<json.TotalArea()<<endl;
         json.write(“out.json”);
    }
}
```

Given json file (Test.json):

```
[
    {
        "_id":"123456789",
        "type":"point",
        "x":5,
        "y":0
    },
    {
        "-id":"129123345",
        "type":"point",
        "x":0,
        "y":0
    },
    {
        "_id":"246532179",
        "type":"point",
        "x":0,
        "y":5
    },
    {
        "_id":"2467341912",
        "type":"point",
        "x":5.123,
        "y":-4.5
    },
    {
        "_id":"222221341",
        "type":"triangle",
        "id1":"123456789",
        "id2":"129123345",
        "id3":"246532179"
    }
]
```

Output file (output.json):

```
[
{
"_id":"123456789",
"type":"point",
"x":5,
"y":0
},
{
"_id":"246532179",
"type":"point",
"x":0,
"y":5
},
{
"_id":"2467341912",
"type":"point",
"x":5.123,
"y":-4.5
}
]
```

Heron's formula

$$A = \sqrt{s(s-a)(s-b)(s-c)}\text{，其中}s = \frac{a+b+c}{2}$$

a, b, c are triangle edge length