学号：2022141410193　**姓名**：王枭越 **学院**：计算机学院

Assignment 5

正文

本次作业中，保证我的工作文件夹为以下结构

```
|
|--build
|--include--*.h
|--src--*.cpp
|--CMakeLists.txt
|--dataset.csv
|--readme.md
|--table.png
```

给出代码如下

`CMakeLists.txt`

```
cmake_minimum_required(VERSION 3.13)
project(Assignment5)

set(CMAKE_CXX_STANDARD 14)

find_package(GTest REQUIRED)

include_directories(include/)

add_executable(main
        src/main.cpp
        src/assignment5.cpp
        src/author.cpp
        src/publication.cpp
        src/book.cpp
        src/unittest.cpp
)

target_compile_options(main PRIVATE -g)

target_link_libraries(main
        GTest::GTest
        GTest::Main
)
```

其中`target_compile_options(main PRIVATE -g)`是加上调试参数，方便我在`VScode`中调试，所以实际上我的文件夹下还有`.vscode`文件夹设置调试文件。

`*.h`文件

`author.h`

```
#pragma once

#include <vector>
#include <string>

class Book;

class Author{
    public:
        Author(std::string _name);
        void setListOfBooks(std::vector<Book*> all_books);
        std::string getname(){
            return name;
        }
    private:
        std::string name;
        std::vector<Book*> list_of_books;
};
```

`publication.h`

```
#pragma once

#include <vector>
#include <string>

class Book;

class Publisher{
    public:
        Publisher(std::string _name);
        void setListOfBooks(std::vector<Book*> all_books);
        std::string getname(){
            return name;
        }
    private:
        std::string name;
        std::vector<Book*> list_of_books;

};
```

`book.h`

```
#pragma once

#include <vector>
#include <string>
#include <memory>
```

```
class Author;
class Publisher;

class Book{
    public:
        Book(std::string _title,
             std::string _genre,
             double _price,
             Author* author,
             std::shared_ptr<Publisher> publisher);

        std::string gettitle(){
            return title;
        }

        std::string getgenre(){
            return genre;
        }

        double getprice(){
            return price;
        }

        std::string getAuthorName(){
            return author->getname();
        }

        std::string getPublisherName(){
            return publisher->getname();
        }

    private:
        std::string title;
        std::string genre;
        double price;

        Author* author;
        std::shared_ptr<Publisher> publisher;
};
```

`assignment5.h`

```
#pragma once

#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <sstream>
#include <algorithm>
```

```
#include <iomanip>
#include "author.h"
#include "publication.h"
#include "book.h"

typedef std::vector<std::vector<std::string>> Dataframe;
Dataframe read_csv(std::string filename);
std::vector<Book> defineBooks(Dataframe* Table);
void sortBooksByPrice(std::vector<Book> & list_of_books );
void showTable(Dataframe* table,int start, int stop);
```

`*.cpp` 文件

`author.cpp`

```
#include "../include/author.h"
#include "../include/publication.h"
#include "../include/book.h"

Author::Author(std::string _name):name(_name){}
void Author::setListOfBooks(std::vector<Book*> all_books){
    for(auto it:all_books){
        list_of_books.push_back(it);
    }
}
```

`publication.cpp`

```
#include "../include/author.h"
#include "../include/publication.h"
#include "../include/book.h"

Publisher::Publisher(std::string _name):name(_name){};
void Publisher::setListOfBooks(std::vector<Book*> all_books){
    for(auto it:all_books){
        list_of_books.push_back(it);
    }
}
```

`book.cpp`

```
#include "../include/author.h"
#include "../include/publication.h"
#include "../include/book.h"

Book::Book(std::string _title,
           std::string _genre,
           double _price,
           Author* author,
           std::shared_ptr<Publisher> publisher):title(_title),
```

```
                                                    genre(_genre),
                                                    price(_price),
                                                    author(author),
                                                    publisher(publisher)
                                                    {}
```

`assignment5.cpp`

```
#include "../include/assignment5.h"

Dataframe read_csv(std::string filename){
    Dataframe ret;
    std::ifstream in(filename);
    std::string tmp,line;
    std::getline(in,tmp);
    while(std::getline(in,line)){
        //处理引号中逗号
        int pos=line.find('"');
        int pos_=line.find('"',pos+1);
        while(pos!=-1){
            int po=line.find(',',pos+1);
            while(po>pos&&po<pos_){
                line[po]='+';
                po=line.find(',',po+1);
            }
            pos=line.find('"',pos_+1);
            pos_=line.find('"',pos+1);
        }

        std::stringstream ss;
        ss<<line;
        std::string Title,Author,Genre,Price,Publisher;
        std::getline(ss,Title,',');
        if(Title[0]=='"')Title=Title.substr(1,Title.size()-2);
        std::getline(ss,Author,',');
        if(Author[0]=='"')Author=Author.substr(1,Author.size()-2);
        std::getline(ss,Genre,',');
        if(Genre[0]=='"')Genre=Genre.substr(1,Genre.size()-2);
        std::getline(ss,Price,',');
        if(Price[0]=='"')Price=Price.substr(1,Price.size()-2);
        std::getline(ss,Publisher);
        if(Publisher[0]=='"')Publisher=Publisher.substr(1,Publisher.size()-2);

        //还原引号中逗号
        int q=Title.find('+');
        while(q!=-1){
            Title[q]=',';
            q=Title.find('+',q+1);
        }
```

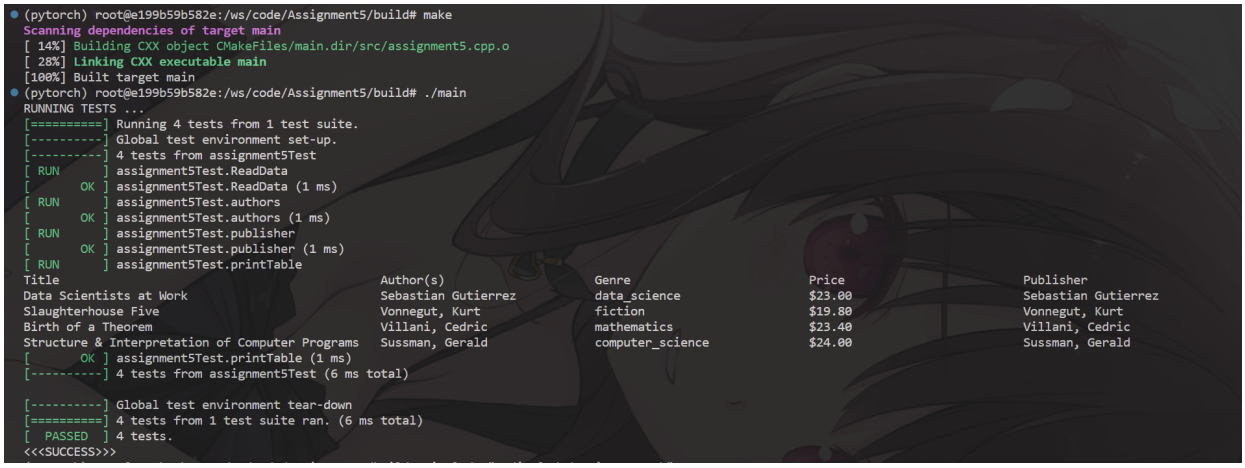
```
        q=Author.find('+');
        while(q!=-1){
            Author[q]=',';
            q=Author.find('+',q+1);
        }

        q=Genre.find('+');
        while(q!=-1){
            Genre[q]=',';
            q=Genre.find('+',q+1);
        }

        q=Price.find('+');
        while(q!=-1){
            Price[q]=',';
            q=Price.find('+',q+1);
        }

        q=Publisher.find('+');
        while(q!=-1){
            Publisher[q]=',';
            q=Publisher.find('+',q+1);
        }

        //存储
        std::vector<std::string> tt{Title,Author,Genre,Price,Publisher};
        ret.push_back(tt);
    }
    in.close();
    return ret;
}

std::vector<Book> defineBooks(Dataframe* Table){
    std::vector<Book> ret;
    for(auto line:*Table){
        std::string title,author,genre,price,publisher;
        title=line[0];
        author=line[1];
        genre=line[2];
        price=line[3];
        publisher=line[4];
        Author* p=new Author(author);
        std::shared_ptr<Publisher> q(new Publisher(publisher));
        Book* t=new Book(title,genre,std::stod(price.substr(1)),p,q);
        std::vector<Book*> tmp{t};
        p->setListOfBooks(tmp);
        q->setListOfBooks(tmp);
        ret.push_back(*t);
    }
    return ret;
}
```

```
bool cmp(Book a,Book b){
    return a.getprice()<b.getprice();
}

void sortBooksByPrice(std::vector<Book> & list_of_books ){
    std::sort(list_of_books.begin(),list_of_books.end(),cmp);
}

void showTable(Dataframe* table,int start, int stop){
    std::vector<Book> t=defineBooks(table);
    std::cout<<std::setw(50)<<std::left<<"Title";
    std::cout<<std::setw(30)<<std::left<<"Author(s)";
    std::cout<<std::setw(30)<<std::left<<"Genre";
    std::cout<<std::setw(30)<<std::left<<"Price";
    std::cout<<std::setw(30)<<std::left<<"Publisher"<<std::endl;
    for(int i=start;i<stop;++i){
        std::cout<<std::setw(50)<<std::left<<t[i].gettitle();
        std::cout<<std::setw(30)<<std::left<<t[i].getAuthorName();
        std::cout<<std::setw(30)<<std::left<<t[i].getgenre();
        std::string tt=std::to_string(t[i].getprice());
        std::cout<<std::setw(30)<<std::left<<"$"+tt.substr(0,tt.find('.')+3);
        std::cout<<std::setw(30)<<std::left<<t[i].getAuthorName()<<std::endl;
    }
}
```

运行结果

```
RUNNING TESTS ...
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from assignment5Test
[ RUN      ] assignment5Test.ReadData
[       OK ] assignment5Test.ReadData (1 ms)
[ RUN      ] assignment5Test.authors
[       OK ] assignment5Test.authors (1 ms)
[ RUN      ] assignment5Test.publisher
[       OK ] assignment5Test.publisher (1 ms)
[ RUN      ] assignment5Test.printTable
Title                                              Author(s)
Genre                          Price                          Publisher

Data Scientists at Work                            Sebastian Gutierrez
data_science                   $23.00                         Sebastian Gutierrez

Slaughterhouse Five                                Vonnegut, Kurt
fiction                        $19.80                         Vonnegut, Kurt

Birth of a Theorem                                 Villani, Cedric
mathematics                    $23.40                         Villani, Cedric
```

```
Structure & Interpretation of Computer Programs   Sussman, Gerald
computer_science              $24.00                        Sussman, Gerald

[       OK ] assignment5Test.printTable (1 ms)
[----------] 4 tests from assignment5Test (6 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (6 ms total)
[  PASSED  ] 4 tests.
<<<SUCCESS>>>
```



各个部分的实现思路是：

1. 交叉引用类： 通过前置声明，解决循环问题，如代码所示。
2. `Dataframe read_csv(std::string filename)` 的实现：通过按 `','` 读入，其中考虑到引号中的逗号，我使用了了替换的方式处理，如代码所示。
3. `std::vector<Book> defineBooks(Dataframe* Table)`：用 `Dataframe read_csv(std::string filename)` 去接收 `dataset.csv`，然后按每部分提出并保存在返回 `vector` 中即可。
4. `void sortBooksByPrice(std::vector<Book> & list_of_books )` 的实现：调用快排即可。
5. `void showTable(Dataframe* table,int start, int stop)` 的实现：按格式打印即可，如代码所示。

至此，所有问题解决。

反馈

在 `readme.md` 中，助教老师有很多语法问题，望改进

例如

> Hi every one 👩‍🎓.\
> In this homework you will be define some classes in C++ and help TA to sort and categorized his bookshelf.\
> He has a csv file that it shows details of his books.\
> You should read a csv file and store it in vector of string vector.\
> **Note**: Suppose this type definition in this homework.

然后还有一个问题，在 `reandme.md` 中有给出

Author Class

This class represents each Author. It has the following method and member variables.

```
class Author{
    public:
        Author(std::string _name);
        void setListOfBooks(std::vector<Book&> all_books);
    private:
        std::string name;
        std::vector<Book&> list_of_books;
};
```

但是

> `std::vector<Book&>` 这句话试图定义一个 `vector`，它包含 `Book` 类型的引用。但是，这是不合法的，因为 `vector` 不能存储引用类型。你可以使用指针或智能指针来代替。

因此我将这里的引用改成了指针来实现。