Student Name: 杨卓
Student ID: 2022141450295

# 高级语言程序设计-II
**Assignment 5**

## 1 Classes

### 1.1 Author Class

```
// author.h
#ifndef AUTHOR
#define AUTHOR

#include <string>
#include <vector>
#include "book.h"

class Book;
// declare class Book

class Author {
public:
    Author(std::string _name);// constructor
    void setListOfBooks(std::vector<Book*> all_books);
    std::string getAuthorName();// to get private variable std::string name
private:
    std::string name;
    std::vector<Book*> list_of_book;
};

#endif
```

在 author.h 中，由于里面有 Book 这个类型的指针变量的 vector，所以需要在一开始声明使用这个类，即 class Book;。除了所需要的函数外，这里还声明了 getAuthorName() 以访问私有成员变量 std::string name。

```
// author.cpp
#include "author.h"
#include <string>
#include <vector>

// constructor
```

```cpp
Author::Author(std::string _name) {
    name = _name;
}

void Author::setListOfBooks(std::vector<Book*> all_books) {
    for (auto book : all_books) {
        if (book->getAuthorName() == this->name)
            list_of_book.push_back(book);
    }
}

std::string Author::getAuthorName() {
    return this->name;
}
```

## 1.2 Publisher Class

```cpp
// publication.h
#ifndef PUBLICATION
#define PUBLICATION

#include <string>
#include <vector>
#include "book.h"

class Book;

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

};

#endif
```

同 author.h，声明的是 Book 类型的指针变量的 vector；设置了 getPublisherName() 函数以访问私有成员变量 std::string name。

```cpp
// publication.cpp
#include "publication.h"
#include <vector>
```

```cpp
#include <string>

Publisher::Publisher(std::string _name) {
    name = _name;
}

void Publisher::setListOfBooks(std::vector<Book*> all_books) {
    for (auto book : all_books) {
        if (book->getPublisherName() == this->name)
            list_of_books.push_back(book);
    }
}

std::string Publisher::getPublisherName() {
    return this->name;
}
```

## 1.3 Book Class

```cpp
// book.h
#ifndef BOOK
#define BOOK

#include <string>
#include <memory>
#include "publication.h"
#include "author.h"

class Publisher;

class Author;

class Book {
    public:
    Book(std::string _title,
        std::string _genre,
        double _price,
        Author* author,
        std::shared_ptr<Publisher> publisher);
    double getPrice();
    std::string getAuthorName();
    std::string getPublisherName();
    private:
    std::string title;
    std::string genre;
```

```
    double price;

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

#endif
```

类似的,使用指针来使用这个上面的类,一个是普通指针 `Author*`;一个智能指针 `std::shared_ptr<Publisher>`,需要先声明两个类: `class Author;` 与 `class Publisher`。

设置了三个函数来获取私有成员变量的值。

```
// book.cpp
#include "book.h"
#include <string>
#include <memory>
#include "publication.h"
#include "author.h"

Book::Book(std::string _title,
        std::string _genre,
        double _price,
        Author* author,
        std::shared_ptr<Publisher> publisher) {
    title = _title;
    genre = _genre;
    price = _price;
    this->author = author;
    this->publisher = publisher;
}

double Book::getPrice() {
    return this->price;
}

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

std::string Book::getPublisherName() {
    return this->publisher->getPublisherName();
}
```

## 2 Main Functions

```h
// assignment5.h
#ifndef ASSGIENMENT
#define ASSGIENMENT

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

typedef std::vector<std::vector<std::string>> Dataframe;

Dataframe read_csv(std::string filename);

double getPrice(std::string pri);

std::vector<Book> defineBooks(Dataframe* Table);

void sortBooksByPrice(std::vector<Book> & list_of_books );

void showTable(Dataframe* table,int start, int stop);

#endif
```

这个头文件中声明了所需要的函数。这些函数在对应的 .cpp 中文件实现。

```cpp
// assignment5.cpp
#include "assignment5.h"
#include "book.h"
#include "publication.h"
#include "author.h"
#include <memory>
#include <vector>
#include <regex>
#include <cstring>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <algorithm>

Dataframe read_csv(std::string filename) {
    std::ifstream fin(filename);
  // title author genre price publisher
    Dataframe table;
    std::string line;
    std::getline(fin, line);
    while (std::getline(fin, line)) {
        std::string cell;
```

```
        int len = line.length();
        int pos = 0;
        std::vector<std::string> info;
        // using regular expression to get books
        std::regex match_mode("(\\\"([^\\\"]*)\\\")|([\\w][&\\w\\s]*[\\w])|([\\w])|(\\$[\\d]*\\.[\\d]*)
            |(,[\\s]*,)");
        auto begin = std::sregex_iterator(line.begin(), line.end(), match_mode);
        auto end = std::sregex_iterator();
        for (auto m = begin; m != end; ++m) {
            std::string str = (*m).str();
            if (str[0] == '\"') str = (*m)[2].str();
            if (str[0] == ',') str = "";
            info.push_back(str);
        }
        table.push_back(info);
    }
    return table;
}

double getPrice(std::string pri) {
    std::stringstream ss; // using stringstream to get double from string
    double result = 0;
    std::regex mode("[\\d]*\\.[\\d]*");
    std::smatch str;
    std::regex_search(pri, str, mode);
    ss = std::stringstream(str.str());
    ss >> result;
    return result;
}

std::vector<Book> defineBooks(Dataframe* Table) {
    std::vector<Book> books;
    for (auto book : *Table) { // initialize a book
        Book *single_book;
        Author *author;
        std::shared_ptr<Publisher> publisher;
        std::string title = book[0];
        std::string gener = book[2];
        double price = getPrice(book[3]);
        author = new Author(book[1]);
        publisher = std::make_shared<Publisher>(book[4]);
        single_book = new Book(title, gener, price, author, publisher);
        books.push_back(*single_book);
    }
    return books;
}
```

```cpp

void sortBooksByPrice(std::vector<Book> &list_of_books) {
    auto cmpF = [](Book A, Book B) { // sort order
        return A.getPrice() > B.getPrice();
    };
    std::sort(list_of_books.begin(), list_of_books.end() ,cmpF);
}

void showTable(Dataframe* table, int start, int stop) {
    size_t maxlen_name = 0;
    size_t maxlen_author = 0;
    size_t maxlen_genre = 0;
    for (int i = start; i <= stop; ++i) {
        maxlen_name = std::max(maxlen_name, table->at(i)[0].length());
        maxlen_author = std::max(maxlen_author, table->at(i)[1].length());
        maxlen_genre = std::max(maxlen_genre, table->at(i)[2].length());
    }
    ++maxlen_name;
    ++maxlen_author;
    ++maxlen_genre; // set margin
    std::cout << std::left;
    std::cout << std::setw(maxlen_name) << "Title";
    std::cout << std::setw(maxlen_author) << "Author(s)";
    std::cout << std::setw(maxlen_genre) << "Genre";
    std::cout << std::setw(8) << "Price";
    std::cout << "Publisher" << std::endl; // print title
    for (int i = start; i <= stop; ++i) { // print infomation about book
        std::cout << std::setw(maxlen_name) << table->at(i)[0];
        std::cout << std::setw(maxlen_author) << table->at(i)[1];
        std::cout << std::setw(maxlen_genre) << table->at(i)[2];
        std::cout << std::setw(8) << table->at(i)[3];
        std::cout << table->at(i)[4] << std::endl;
    }
}
```

值得说的一些点是，使用了正则表达式去匹配 .csv 中的内容；使用了 stringstream 从一个 string 类型的数据中输入 double。

# 3 Test Result

```
root@32514c419ad2:/ws/assignment5/build# ./main
RUNNING TESTS ...
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from assignment5Test
[ RUN      ] assignment5Test.ReadData
[       OK ] assignment5Test.ReadData (219 ms)
[ RUN      ] assignment5Test.authors
[       OK ] assignment5Test.authors (255 ms)
[ RUN      ] assignment5Test.publisher
[       OK ] assignment5Test.publisher (256 ms)
[ RUN      ] assignment5Test.printTable
Title                                       Author(s)           Genre            Price   Publisher
Data Scientists at Work                     Sebastian Gutierrez data_science     $23.00  Apress
Slaughterhouse Five                         Vonnegut, Kurt      fiction          $19.80  Random House
Birth of a Theorem                          Villani, Cedric     mathematics      $23.40  Bodley Head
Structure & Interpretation of Computer Programs Sussman, Gerald computer_science $24.00  MIT Press
Age of Wrath, The                           Eraly, Abraham      history          $23.80  Penguin
[       OK ] assignment5Test.printTable (207 ms)
[----------] 4 tests from assignment5Test (939 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (939 ms total)
[  PASSED  ] 4 tests.
<<<SUCCESS>>>
```

图 1: Caption