|  |  |  |
| --- | --- | --- |
| **American University of Sharjah**  **College of Engineering**  Dept of Computer Science & Engg  P. O. Box 26666  Sharjah, UAE | A picture containing logo  Description automatically generated | **Instructors:** Dr. Aliaa Moualla  **Lab Instructor:** Sameer Alawnah  **Office:** EB1-0012C  **Phone**: 971-6-515-4940  **e-mail**: salawnah@aus.edu  **Semester**: Spring 2024 |

**CMP 220L - Programming II**

**Lab #8 – Classes and Pointers and operators overloading**

**Note: using ChatGPT will be considered a violation of the AUS integrity code.**

**Objectives:**

* Practice building classes.
* Practice pointers.

Using Visual Studio 2022, write the below programs, compile and provide screenshots of output.

Note: you are required to submit copy of the code + screenshots of program run for each exercise.

**Exercise #1**

Write a C++ program that performs the following tasks:

* Create a class called **Book** that has two private attributes: **title** and **author**.

1. Create a default constructor.
2. Create a **parameterize** **constructor** to initialize a Book object with a title and author.
3. Write overload << operator as a member function within the Book class. To use it with global function. This method will display the book's title and author to the output stream.

std::ostream& operator<<(std::ostream& os) const;

1. Define the global overload **<<** operator outside the **Book** class to call << operator member function.

std::ostream& operator<<(std::ostream& os, const Bo ok& book)

1. Write overload >> operator as a member function within the Book class. To use it with global function. This method will read the title and author from the input stream using getline.

std::istream& operator>>(std::istream& is) ;

1. Define the global overload **>>** operator outside the **Book** class to call >> operator member function.

std::istream& operator>>(std::istream& is, Book& book)

Test your class with the following main:

int main()

{

Book b("A Tale of Two Cities","Charles Dickens");

cout<<"The Book is :\n"<<b<<endl;

cout<<"Please enter new Title and Author for the book, each one line : ";

cin>>b;

cout<<"The Book now is:\n"<<b<<endl;

}

Sample output:

The Book is :

Title: A Tale of Two Cities, Author: Charles Dickens

Please enter new Title and Author for the book, each one line : The Good Father

Mario Puzo

The Book now is:

Title: The Good Father, Author: Mario Puzo

#include <iostream>

#include <sstream>

#include <string>

**using** **namespace** std;

**class** Book{

**private**:

string title, author;

**public**:

Book();

Book(**const** string& title, **const** string& author){}

ostream& **operator**<<(ostream& os) **const**

{

os <<"Title: " << title <<" Author: " << author;

**return** os;

}

istream& **operator**>>(istream& is) {

cout << "Title: ";

getline(is, title);

cout << "Author: ";

getline(is, author);

**return** is;

}

};

ostream& **operator**<<(ostream& os, **const** Book& book)

{

**return** book.**operator**<<(os);

}

istream& **operator**>>(istream& is, Book& book)

{

**return** book.**operator**>>(is);

}

**int** main()

{

Book b("A Tale of Two Cities","Charles Dickens");

cout<<"The Book is :\n"<<b<<endl;

cout<<"Please enter new Title and Author for the book, each one line : ";

cin>>b;

cout<<"The Book now is:\n"<<b<<endl;

**return** 0;

}

A black text on a white background

Description automatically generated

**Exercise #2**

* Create **Library** class to represent a library that can hold a collection of books. It has private attributes: **name** (library name), **capacity** (maximum number of books the library can hold), **numBooks** (current number of books in the library, default number is 5), and a dynamic array of Book objects, **books**. Book\* books

1. Create a defalut constructor that intialzie the library name to “None”, capacity to 5, it create a new dynamic array for books.
2. Create a **copy constructor** that copy the library name, capacity, and books. It creates a new dynamic array for books and copies each book from the original library.
3. Overloads the **= operator** to copy the contents of one library into another. It first checks for self-assignment, deallocates the current books array, and then performs a deep copy similar to the copy constructor.
4. Create **addBook** method to add a Book to the library. It checks if the library is not full (based on capacity) and adds the book to the books array.
5. Overload the << operator to print the library's name and the list of books it contains.
6. Destructor:
7. Create **destructor** for the Library class to deallocate the memory of the books array when the library object goes out of scope.

In the main function:

Create a Book object, book1, and read it from the user.

Create a Library object, library1, with the name "My Library" and a capacity of 5.

Add the book1 to library1 using the addBook method.

Create a second Library object, library2 and assign to it the contents of library1. This will demonstrate the use of the copy assignment operator.

Create a third Libraty object, library3 using the default constructor.

Assign the content of library2 to library3. This will demonstarte the use of the = operator

All libraries are displayed using the << operator.

Sample Run

Pleae enter the title and author of the book (each one full line): A Tale of Two Cities

Charles Dickens

Library 1:

Library Name: My library

Books in Library:

Title: A Tale of Two Cities, Author: Charles Dickens

Library 2:

Library Name: My library

Books in Library:

Title: A Tale of Two Cities, Author: Charles Dickens

Library 3:

Library Name: My library

Books in Library:

Title: A Tale of Two Cities, Author: Charles Dickens

#include<iostream>

#include<string>

**using** **namespace** std;

**class** Book {

**public**:

Book()

{

title = "None";

author = "None";

}

Book(string \_title, string \_author)

{

title = \_title;

author = \_author;

}

ostream& **operator**<<(ostream& os) **const**

{

cout << "Title: " << title << ", Author: " << author << endl;

**return** os;

}

istream& **operator**>>(istream& is)

{

getline(is, title);

getline(is, author);

**return** is;

}

**private**:

string title;

string author;

};

ostream& **operator**<<(ostream& os, **const** Book& book)

{

**return** book.**operator**<<(os);

}

istream& **operator**>>(istream& is, Book& book)

{

**return** book.**operator**>>(is);

}

**class** Library {

**public**:

Library()

{

name = "None";

capacity = 5;

numBooks = 0;

books =**new** Book[capacity];

}

Library(string \_name, **int** \_capacity)

{

name = \_name;

capacity = \_capacity;

numBooks = 0;

books = **new** Book[capacity];

}

Library(**const** Library& rhs)

{

capacity = rhs.capacity;

books = **new** Book[capacity];

numBooks = rhs.numBooks;

**for** (**int** i = 0; i < capacity; i++)

{

books[i] = rhs.books[i];

}

}

Library& **operator**=(**const** Library& rhs)

{

**if** (**this** == &rhs)

{

**return** \***this**;

}

**if** (**this**->capacity != rhs.capacity)

{

**delete**[] **this**->books;

**this**->capacity = rhs.capacity;

**this**->books = **new** Book[capacity];

}

**this**->numBooks = rhs.numBooks;

**for** (**int** i = 0; i < capacity; i++)

{

books[i] = rhs.books[i];

}

**return** \***this**;

}

**void** addBook(**const** Book& b)

{

**if** (numBooks < capacity)

{

books[numBooks] = b;

numBooks++;

}

**else**

cout << "Library is full!" << endl;

}

ostream& **operator**<<(ostream& outs) **const**

{

cout << "Library Name: " << name << endl;

cout << "Books in Library: " << endl;

**for** (**int** i = 0; i < numBooks; i++)

{

outs << books[i];

}

**return** outs;

}

~Library()

{

}

**private**:

string name;

**int** capacity;

**int** numBooks;

Book\* books;

};

ostream& **operator**<<(ostream& outs, **const** Library& other)

{

**return** other.**operator**<<(outs);

}

**int** main()

{

Book book1, book2;

cout << "Please Enter the title and the author of the book (each one full line): " << endl;

cin >> book1;

cin >> book2;

Library library1("My library",5);

cout << endl;

library1.addBook(book1);

Library library2 = library1;

Library library3;

library3 = library2;

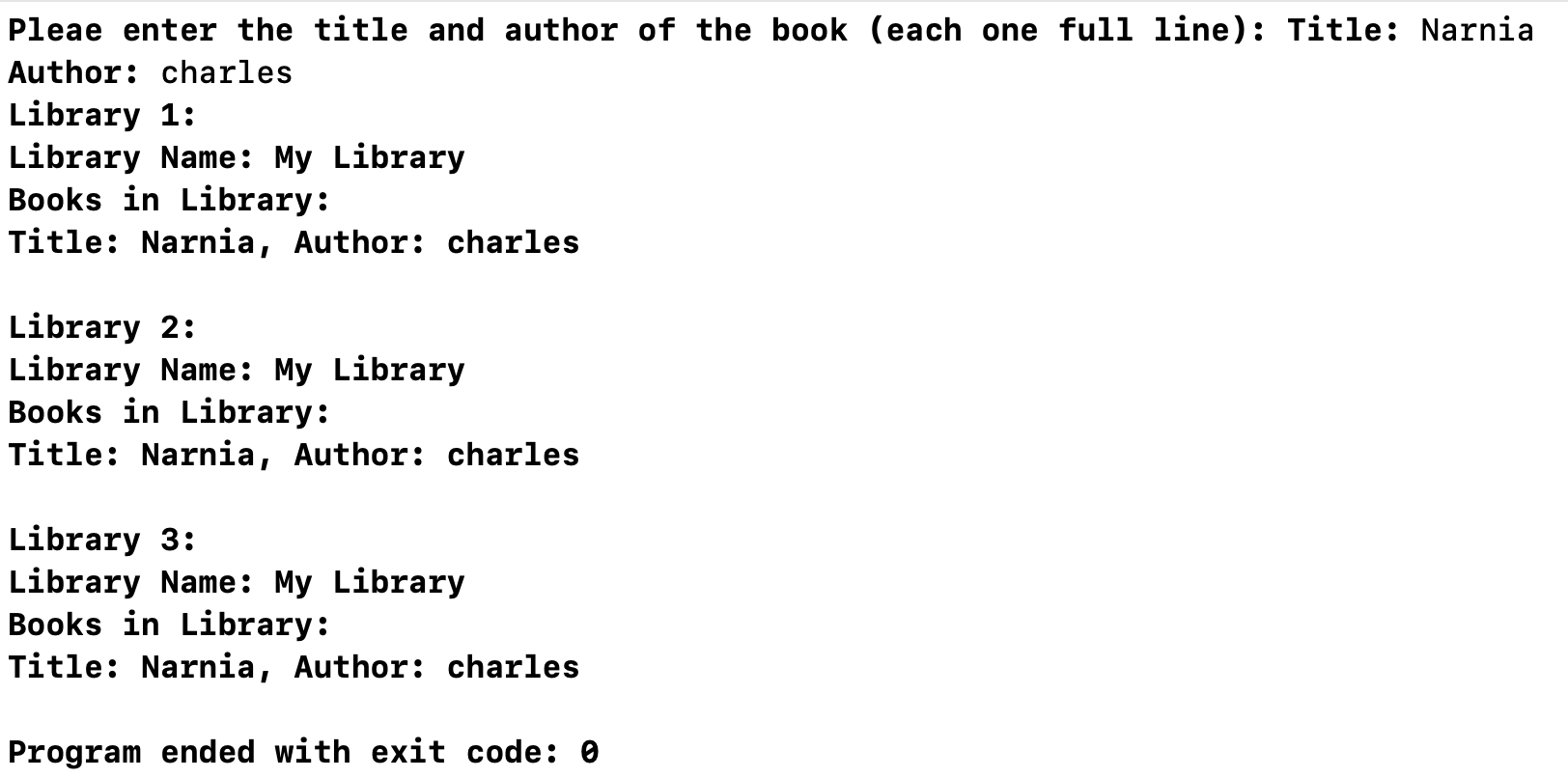
cout << "Library 1: \n"<< library1 << endl;

cout << "Library 2: \n" << library2 << endl;

cout << "Library 3: \n" << library3 << endl;

**return** 0;

}

****