Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Інформатики

Звіт  
з лабораторної роботи №2  
з дисципліни «Об’єктно-орієнтоване програмування»

Варіант 12

Виконав:  
ст. гр. ІТІНФ-21-1  
Кошель В.О.

Перевірили:  
Кириченко І.Ю.

1. **ОСНОВИ РОЗРОБКИ КЛАСІВ НА МОВІ C++**
   1. **Мета роботи:**

Познайомитись з основами розробки класів.

* 1. **Завдання:**

Кожний розроблюваний клас повинен реалізовувати:

* + конструктор за замовченням;
  + деструктор;
  + конструктори с параметрами;
  + конструктор копіювання;
  + перевантажену операцію присвоювання;
  + перевантажену операцію виведення в потік;
  + перевантажену операцію введення з потоку;
  + перевантажену арифметичну операцію;
  + перевантажену логічну операцію;
  + перевантажену операцію приведення до типу (наприклад для отримання середнього значення);
  + необхідні методи.

Розробити клас, що представляє бібліотеку.

* Бібліотека характеризується назвою, адресою, спеціалізацією та списком книг.
* Адреса характеризується містом, вулицею, номером будинку.
* Книга характеризується назвою, роком видання, кількістю у бібліотеці, кількістю виданих книг.

Розробити методи, які дозволяють:

1. дізнатися назву та адресу бібліотеки
2. перевірити наявність книги за назвою
3. дізнатися середнє число виданих книг
4. додати книгу до бібліотеки
5. видалити книгу з бібліотеки
6. видати книгу на руки (збільшити кількість виданих книг)
7. підрахувати процент книжок, які повністю видані
   1. **Хід виконання**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Specialization

{

string city, street;

size\_t number;

public:

Specialization() : city("New York"), street("Example str."), number(1) {}

Specialization(string city, string street, size\_t number)

{

this->city = city;

this->street = street;

this->number = number;

}

Specialization(const Specialization& copy)

{

this->city = copy.city;

this->street = copy.street;

this->number = copy.number;

}

Specialization& operator=(const Specialization& copy)

{

this->city = copy.city;

this->street = copy.street;

this->number = copy.number;

return \*this;

}

bool operator==(const Specialization& spec)

{

Specialization first = \*this;

if (first.city == spec.city

&& first.number == spec.number

&& first.street == spec.street)

return true;

else return false;

}

ostream& print(ostream& out) const

{

out << city << ", " << street << " str., " << number;

return out;

}

operator double() { return (double)number; }

string getCity() const { return city; }

string getStreet() const { return street; }

size\_t getNumber() const { return number; }

void setCity(string city) { this->city = city; }

void setStreet(string street) { this->street = street; }

void setNumber(size\_t number) { this->number = number; }

~Specialization() {}

};

ostream& operator<<(ostream& out, Specialization& spec)

{

out << spec.getCity() << ", " << spec.getStreet() << " str., " << spec.getNumber();

return out;

}

istream& operator>>(istream& in, Specialization spec)

{

string tmpstr;

size\_t tmpint;

cout << "City: ";

in >> tmpstr;

spec.setCity(tmpstr);

cout << "Street: ";

tmpstr = "";

in >> tmpstr;

spec.setStreet(tmpstr);

cout << "Number: ";

tmpstr = "";

in >> tmpint;

spec.setNumber(tmpint);

return in;

}

class Book

{

string name;

size\_t year, amount\_available, amount\_checkedout;

public:

Book() : name("Example Book"), year(2000), amount\_available(1), amount\_checkedout(5) {}

Book(string name, size\_t year, size\_t amount\_available, size\_t amount\_checkedout)

{

this->name = name;

this->year = year;

this->amount\_available = amount\_available;

this->amount\_checkedout = amount\_checkedout;

}

Book(const Book& book)

{

this->name = book.name;

this->year = book.year;

this->amount\_available = book.amount\_available;

this->amount\_checkedout = book.amount\_checkedout;

}

Book operator=(const Book& book)

{

this->name = book.name;

this->year = book.year;

this->amount\_available = book.amount\_available;

this->amount\_checkedout = book.amount\_checkedout;

return \*this;

}

Book operator++()

{

this->amount\_available++;

return \*this;

}

bool operator!=(const Book& book)

{

if (this->amount\_available != book.amount\_available)

return true;

else return false;

}

operator double() { return (double)this->year; }

string getName() const { return name; }

size\_t getYear() const { return year; }

size\_t getAmountAvailable() const { return amount\_available; }

size\_t getAmountCheckedout() const { return amount\_checkedout; }

void setName(string name) { this->name = name; }

void setYear(size\_t year) { this->year = year; }

void setAmountAvailable(size\_t amount) { amount\_available = amount; }

void setAmountCheckedout(size\_t amount) { amount\_checkedout = amount; }

~Book() {}

};

ostream& operator<<(ostream& out, Book& book)

{

out << "Name: " << book.getName() << "\t Year: " << book.getYear() << "\n Available: " << book.getAmountAvailable() << "\t Published: " << book.getAmountCheckedout();

return out;

}

istream& operator>>(istream& in, Book& book)

{

cout << "Add new book" << endl;

string tmpstr = "";

getline(in, tmpstr);

int tmpint;

cout << "Name: ";

getline(in, tmpstr);

book.setName(tmpstr);

cout << "Year: ";

while (true)

{

tmpstr = "";

getline(in, tmpstr);

tmpint = atoi(tmpstr.c\_str());

if (tmpint < 0 || tmpint > 2022)

{

cout << "Try another one..." << endl;

continue;

}

else

{

book.setYear(tmpint);

break;

}

}

cout << "Available: ";

while (true)

{

tmpstr = "";

getline(in, tmpstr);

tmpint = atoi(tmpstr.c\_str());

if (tmpint < 0)

{

cout << "Try another one..." << endl;

continue;

}

else

{

book.setAmountAvailable(tmpint);

break;

}

}

cout << "Published: ";

while (true)

{

tmpstr = "";

getline(in, tmpstr);

tmpint = atoi(tmpstr.c\_str());

if (tmpint > book.getAmountAvailable() || tmpint < 0)

{

cout << "Try another one..." << endl;

continue;

}

else

{

book.setAmountCheckedout(tmpint);

break;

}

}

return in;

}

class Library

{

string name;

public:

Specialization specialization;

vector<Book> books;

Library() : name("Library"), specialization() { books.push\_back({}); }

Library(string name, Specialization spec, vector<Book> books)

{

this->name = name;

specialization = spec;

this->books = books;

}

Library(const Library& lib)

{

name = lib.name;

specialization = lib.specialization;

books = lib.books;

}

Library& operator=(const Library& lib)

{

name = lib.name;

specialization = lib.specialization;

books = lib.books;

return \*this;

}

Library operator++() { books.push\_back(Book()); }

string getName() const { return name; }

void setName(string name) { this->name = name; }

void getNameAndAdress(ostream& out)

{

out << "FIRST TASK:" << endl;

out << "Name: " << name << "\tAdress: " << specialization << endl << endl;

}

void isBookAvailable(ostream& out) const

{

out << "SECOND TASK:" << endl;

string tmp;

cout << "Enter book name: ";

cin >> tmp;

for (size\_t i = 0; i < books.size(); i++)

{

if (tmp == books[i].getName())

out << "Book " << tmp << " is available" << endl << endl;

else

out << "Book " << tmp << " isn't available" << endl << endl;

}

}

void averageCheckedout(ostream& out) const

{

out << "THIRD TASK" << endl;

double average = 0;

for (size\_t i = 0; i < books.size(); i++)

average += books[i].getAmountCheckedout();

average /= books.size();

out << "The average number of checked out books is " << average << endl << endl;

}

void addBook(ostream& out)

{

out << "FOURTH TASK:" << endl;

Book tmp;

cin >> tmp;

books.push\_back(tmp);

out << tmp.getName() << " (" << tmp.getYear() << ") is added" << endl << endl;

}

void removeBook(ostream& out)

{

out << "FIFTH TASK:" << endl;

cout << "Book list:" << endl;

for (size\_t i = 0; i < books.size(); i++)

{

cout << i + 1 << ".\t";

cout << books[i].getName() << "\t";

cout << books[i].getYear() << endl;

}

bool isCorrect = false;

cout << "What book do you want to remove?\nEnter number: ";

string tmp;

size\_t tmpint;

while (!isCorrect)

{

cin >> tmp;

tmpint = atoi(tmp.c\_str());

if (tmpint > books.size() || tmpint < 1)

continue;

else

isCorrect = true, tmp = "";

}

for (size\_t i = 0; i < books.size(); i++)

{

if (tmpint == (i + 1))

{

books.erase(books.begin() + i);

out << "Completed." << endl;

break;

}

}

out << "New book list is:" << endl;

for (size\_t i = 0; i < books.size(); i++)

{

out << i + 1 << ".\t";

out << books[i].getName() << "\t";

out << books[i].getYear() << endl;

}

out << endl;

}

void giveBook(ostream& out)

{

out << "SIXTH TASK:" << endl;

cout << "Book list" << endl;

for (size\_t i = 0; i < books.size(); i++)

{

cout << i + 1 << ".\t";

cout << books[i].getName() << "\t";

cout << books[i].getYear() << endl;

}

bool isCorrect = false;

cout << "What book do you want to check out?\nEnter number: ";

string tmp;

size\_t tmpint;

while (!isCorrect)

{

cin >> tmp;

tmpint = atoi(tmp.c\_str());

if (tmpint > books.size() || tmpint < 1)

continue;

else

isCorrect = true, tmp = "";

}

tmpint--;

if (books[tmpint].getAmountCheckedout() < books[tmpint].getAmountAvailable())

{

books[tmpint].setAmountCheckedout(books[tmpint].getAmountCheckedout() + 1);

out << "Book " << books[tmpint].getName() << " is checked out" << endl << endl;

}

else out << "All books are checked out earlier." << endl << endl;

}

void howManyBooksGiven(ostream& out) const

{

out << "SEVENTH TASK:" << endl;

double percentage = 0;

for (size\_t i = 0; i < books.size(); i++)

if (books[i].getAmountCheckedout() == books[i].getAmountAvailable())

percentage++;

percentage /= (books.size());

percentage \*= 100;

out << "Percentage of fully given book is " << percentage << "%" << endl;

}

~Library() {}

};

ostream& operator<<(ostream& out, Library& lib)

{

out << "LIBRARY:\n";

out << "Name: " << lib.getName() << "\nSpecialization:\n" << lib.specialization << "\nBooks:\n";

for (size\_t i = 0; i < lib.books.size(); i++)

out << lib.books[i] << endl;

return out;

}

int main()

{

Library lib("Name", Specialization("City", "Street", 5), { Book("Name", 2005, 520, 520) });

// ЗАВДАННЯ 1:

// Дізнатися назву та адресу бібліотеки

lib.getNameAndAdress(cout);

// ЗАВДАННЯ 2:

// Перевірити наявність книги за назвою

lib.isBookAvailable(cout);

// ЗАВДАННЯ 3:

// Дізнатися середнє число виданих книг

lib.averageCheckedout(cout);

// ЗАВДАННЯ 4:

// Додати книгу до бібліотеки

lib.addBook(cout);

// ЗАВДАННЯ 5:

// Видалити книгу з бібліотеки

lib.removeBook(cout);

// ЗАВДАННЯ 6:

// Видати книгу на руки (збільшити кількість виданих книг)

lib.giveBook(cout);

// ЗАВДАННЯ 7:

// Підрахувати процент книжок, які повністю видані

lib.howManyBooksGiven(cout);

system("pause");

return 0;

}

* 1. **Результат виконання**

**Изображение выглядит как текст

Автоматически созданное описание**

* 1. **Висновок.** В результаті виконання цієї лабораторної роботи ми ознайомилися з основами розробки класів у мові C++.