Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Інформатики

Звіт  
з лабораторної роботи №5  
з дисципліни «Об’єктно-орієнтоване програмування»

Варіант 12

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

Перевірили:  
Кириченко І.Ю.

1. **АЛГОРИТМИ СТАНДАРТНОЇ БІБЛіОТЕКИ С++**
   1. **Мета роботи:**

Ознайомитись з такими поняттями, як алгоритми стандартної бібліотеки STL. Отримати навички роботи з ними.

* 1. **Завдання:**

Створити контейнер об'єктів класів, реалізованих в лабораторній роботі №2 (студентів, гуртожитків, викладачів і т. д.). Контейнер реалізувати як вектор. Провести певні дії з цим контейнером. Не застосовувати цикли, використовувати тільки відповідні стандартні алгоритми. Перетворити класи для вирішення даного завдання

* Забезпечити для контейнера можливість зчитування з файлу і запису в файл з використанням ітераторів потоків.
* Сортування за замовчуванням: за вулицею і номером будинку, а також сортування за кількістю книг.
* Винести в окремий вектор всі бібліотеки, починаючи з тої, в якій більше 4 книг.
* Зберегти всі номери будинків бібліотек з вектора з кроку 3 в окремому векторі.
* Дізнатися, скільки в векторі з кроку 4 номерів будинків більших від заданого.
* Знайти середню кількість книг у всіх бібліотеках з використанням accumulate()
* Перевірити, чи входить вектор, отриманий на кроці 3, в вихідний вектор як підпослідовність
  1. **Вхідні дані (***input.txt***)**

Library\_1:Kyiv,Khreshchatyk,1;Book1,2004,120,1\_Book2,2021,10,9\_Book3,2020,850,4

Library\_2:Kharkiv,Yarosha,15;First,2010,140,200\_Second,2022,100,0\_Third,2021,450,420\_Fourth,1984,1000,80\_Fifth,2000,1,1

Library\_3:Kharkiv,Nauky,100;First,2010,140,200\_Second,2022,100,0\_Third,2021,450,420\_Fourth,1984,1000,80

* 1. **Хід виконання:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <vector>

#include <iterator>

#include <algorithm>

#include <functional>

#include <numeric>

#include <math.h>

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, const Specialization& spec)

{

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

return out;

}

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

{

string tmp;

// Місто

getline(in, tmp, ',');

spec.setCity(tmp);

// Вулиця

getline(in, tmp, ',');

spec.setStreet(tmp);

// Номер будинку

getline(in, tmp);

spec.setNumber(atoi(tmp.c\_str()));

return in;

}

class Book

{

string name;

int year;

size\_t 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; }

int 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, const Book& book)

{

out << book.getName() << " (" << book.getYear() << "):" << endl;

out << "\tAvailable: " << book.getAmountAvailable() << endl;

out << "\tPublished: " << book.getAmountCheckedout() << endl;

return out;

}

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

{

string tmp;

// Назва книги

getline(in, tmp, ',');

book.setName(tmp);

// Рік видання

getline(in, tmp, ',');

(atoi(tmp.c\_str()) <= 2022)

? book.setYear(atoi(tmp.c\_str()))

: book.setYear(2022);

// Доступно книг всього

getline(in, tmp, ',');

(atoi(tmp.c\_str()) > 0)

? book.setAmountAvailable(atoi(tmp.c\_str()))

: book.setAmountAvailable(0);

// Виписано книг вже

getline(in, tmp, '\_');

(book.getAmountAvailable() > atoi(tmp.c\_str()))

? book.setAmountCheckedout(atoi(tmp.c\_str()))

: book.setAmountCheckedout(book.getAmountAvailable());

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()); }

bool operator<(const Library b) const

{

if (this->specialization.getStreet() == b.specialization.getStreet())

return this->specialization.getNumber() < b.specialization.getNumber();

else

return this->specialization.getStreet() < b.specialization.getStreet();

}

bool operator==(const Library& a)

{

return (name == a.name

&& specialization.getCity() == a.specialization.getCity()

&& specialization.getNumber() == a.specialization.getNumber()

&& specialization.getStreet() == a.specialization.getStreet());

}

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, const Library& lib)

{

out << "=== Library: " << lib.getName() << " ===\n";

out << "Adress:\t" << lib.specialization << endl;

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

out << i + 1 << ".\t" << lib.books[i];

return out;

}

istream& operator>>(istream& in, Library& lib)

{

// БІБЛІОТЕКА:Київ,Хрещатик,1;Книга1,2004,120,1\_Книга2,2021,10,9\_Книга3,2020,850,4

string tmp;

// Назва

getline(in, tmp, ':');

lib.setName(tmp);

// Адреса

getline(in, tmp, ';');

stringstream tmp\_spec(tmp);

tmp\_spec >> lib.specialization;

// Масив книг

getline(in, tmp);

vector<Book> tmp\_vector;

stringstream tmp\_books(tmp);

copy(istream\_iterator<Book>(tmp\_books), istream\_iterator<Book>(), back\_inserter(tmp\_vector));

lib.books = tmp\_vector;

return in;

}

void print\_task(ostream& out, size\_t n)

{

out << ":::::::::::::::::::::::TASK " << n << ":::::::::::::::::::::::" << endl;

}

// Для завдання 2:

bool compare\_by\_books(const Library& a, const Library& b)

{

return a.books.size() < b.books.size();

}

// Для завдання 3:

bool more\_4\_books(const Library& a)

{

return (a.books.size() >= 4);

}

// Для завдання 6:

int average\_books\_num(size\_t sum, const Library& a)

{

return sum += a.books.size();

}

// Для завдання 7:

bool sub\_vector\_search(Library& a, Library& b)

{

return a == b;

}

int main()

{

vector<Library> libs;

/\*

TASK 1:

Забезпечити для контейнера можливість зчитування з файлу і запису в файл з використанням ітераторів потоків.

\*/

// Зчитування

print\_task(cout, 1);

ifstream fin("input.txt");

if (!fin) { cerr << "File not found"; return -1; }

copy(istream\_iterator<Library>(fin), istream\_iterator<Library>(), back\_inserter(libs));

fin.close();

// Запис

ofstream fout("output.txt");

copy(libs.begin(), libs.end(), ostream\_iterator<Library>(fout, "\n"));

copy(libs.begin(), libs.end(), ostream\_iterator<Library>(cout, "\n"));

cout << endl;

/\*

TASK 2:

СОРТУВАННЯ:

1. За вулицею і номером будинку (за замовчуванням);

2. За кількістю книг.

\*/

print\_task(cout, 2);

// За вулицею і номером будинку

cout << "Sorted by adress:\n";

sort(libs.begin(), libs.end());

copy(libs.begin(), libs.end(), ostream\_iterator<Library>(cout, "\n"));

// За кількістю книг

cout << "Sorted by number of books:\n";

sort(libs.begin(), libs.end(), compare\_by\_books);

copy(libs.begin(), libs.end(), ostream\_iterator<Library>(cout, "\n"));

cout << endl;

/\*

TASK 3:

Винести в окремий вектор всі бібліотеки, починаючи з тої, в якій більше 4 книг.

\*/

print\_task(cout, 3);

vector<Library>::iterator it = find\_if(libs.begin(), libs.end(), more\_4\_books);

vector<Library> task\_3;

copy(it, libs.end(), back\_inserter(task\_3));

copy(task\_3.begin(), task\_3.end(), ostream\_iterator<Library>(cout, "\n"));

cout << endl;

/\*

TASK 4:

Зберегти всі номери будинків бібліотек з вектора (3) в окремому векторі.

\*/

print\_task(cout, 4);

vector<size\_t> task\_4;

transform(task\_3.begin(),

task\_3.end(),

back\_inserter(task\_4),

[](const Library& a) { return a.specialization.getNumber(); });

copy(task\_4.begin(), task\_4.end(), ostream\_iterator<size\_t>(cout, "\t"));

cout << endl;

cout << endl;

/\*

TASK 5:

Дізнатися, скільки в векторі з кроку 4 номерів будинків більших від заданого.

\*/

print\_task(cout, 5);

size\_t task\_5 = count\_if(task\_4.begin(), task\_4.end(), bind2nd(greater<size\_t>(), 10));

cout << "Number of buildings that are bigger than 10: " << task\_5 << endl;

cout << endl;

/\*

TASK 6:

Знайти середню кількість книг у всіх бібліотеках з використанням accumulate()

\*/

print\_task(cout, 6);

size\_t task\_6 = accumulate(libs.begin(), libs.end(), 0, average\_books\_num) / libs.size();

cout << "Average number of books for libraries: " << task\_6 << endl;

cout << endl;

/\*

TASK 7:

Перевірити, чи входить вектор, отриманий на кроці 3, в вихідний вектор як підпослідовність.

\*/

print\_task(cout, 7);

vector<Library>::iterator it\_ = search(libs.begin(), libs.end(), task\_3.begin(), task\_3.end(), sub\_vector\_search);

if (it\_ != libs.end())

cout << "Subvector is found!\n";

else

cout << "Subvector isn't found.\n";

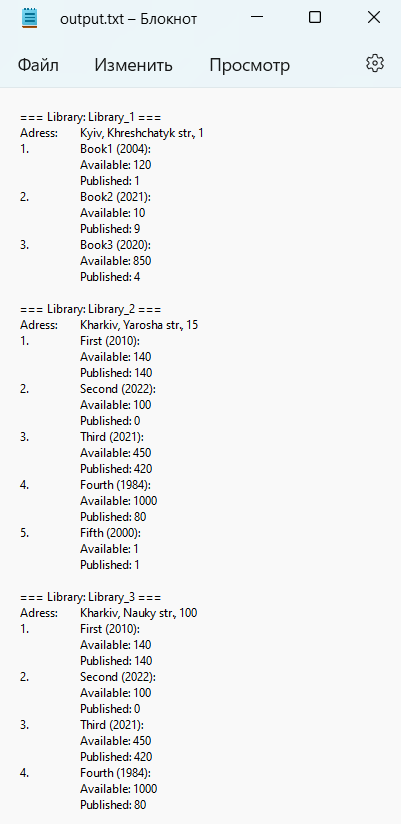
cout << endl;

system("pause");

return 0;

}

* 1. **Результат виконання**
* **Вихідний файл (***output.txt***):**

****

* **Консоль:**

**Изображение выглядит как текст

Автоматически созданное описание**

|  |  |
| --- | --- |
| **Изображение выглядит как текст  Автоматически созданное описание** | **Изображение выглядит как текст  Автоматически созданное описание** |

**Изображение выглядит как текст

Автоматически созданное описание**

* 1. **Висновок.** В результаті виконання цієї лабораторної роботи ми ознайомилися із алгоритмами стандартної бібліотеки та навчилися використовувати отримані знання на практиці.