|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ИУ8 “КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ”

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

Студент Лустин Павел Евгеньевич

*(*Фамилия И.О)

Группа\_\_ИУ8-24\_\_\_\_\_\_\_\_

Студент  **\_\_\_\_\_\_8.05\_\_\_\_\_\_\_\_\_**  Лустин \_\_ \_\_

*дата (*Фамилия И.О)

Преподаватель **\_\_\_\_\_\_8.05\_\_\_\_\_\_\_**  Барыкин Дмитрий

*дата (*Фамилия И.О)*.*

*2024 г.*

Цель работы

Цель работы состоит в изучении использования объектов своих классов в упорядоченных и неупорядоченных контейнерах библиотеки STL (set и map, unordered\_set и unordered\_map)

Условие задачи(вариант 14)

Для класса, разработанного в ЛР4, обеспечить возможность добавления объектов в контейнер set (сортировка как указано в задании на ЛР4) и в контейнер unordered\_set. Исходные данные как в ЛР4 читать из файла, вывести на печать для контроля объекты контейнеров.

Текст программы с комментариями

#include <iostream>

#include <string>

#include <deque>

#include <list>

#include <fstream>

#include <sstream>

#include <algorithm>

#include <set>

#include <unordered\_set>

using namespace std;

class Book //Класс - книга

{

public:

string FIO;

string Title;

string publishing\_house;

int year;

int pages;

Book() //Пустой конструктор

{

FIO = "";

Title = "";

publishing\_house = "";

year = 0;

pages = 0;

}

Book(string FIO, string Title, string publishing\_house, int year, int pages) //Конструктор с начальными значениями

{

this->FIO = FIO;

this->Title = Title;

this->publishing\_house = publishing\_house;

this->year = year;

this->pages = pages;

}

Book(const Book& book) //Конструктор копирования

{

this->FIO = book.FIO;

this->Title = book.Title;

this->publishing\_house = book.publishing\_house;

this->year = book.year;

this->pages = book.pages;

}

Book(Book&& book) //Конструктор перемещения

{

this->FIO = book.FIO;

this->Title = book.Title;

this->publishing\_house = book.publishing\_house;

this->year = book.year;

this->pages = book.pages;

book.FIO = "";

book.Title = "";

book.publishing\_house = "";

book.year = 0;

book.pages = 0;

}

Book& operator = (const Book& book) //Перегрузка оператора присваивания с копированием

{

if (this != &book)

{

this->FIO = book.FIO;

this->Title = book.Title;

this->publishing\_house = book.publishing\_house;

this->year = book.year;

this->pages = book.pages;

}

return \*this;

}

Book& operator = (Book&& book) //Перегрузка оператора присваивания с перемещением

{

if (this != &book)

{

this->FIO = book.FIO;

this->Title = book.Title;

this->publishing\_house = book.publishing\_house;

this->year = book.year;

this->pages = book.pages;

book.FIO = "";

book.Title = "";

book.publishing\_house = "";

book.year = 0;

book.pages = 0;

}

return \*this;

}

friend ostream& operator << (ostream& fout, const Book& book); //Описание дружественной перегруженной операции вставки объекта в поток cout

bool operator < (const Book& book) const //Перегрузка оператора < для сортировки в set [Лабораторная 5]

{

return size(FIO) < size(book.FIO);

}

bool operator ==(const Book& book) const //Перегрузка оператора == для unordered\_set [Лабораторная 5]

{

return size(FIO) == size(book.FIO);

}

};

struct Hasher //Структура с функтором [Лабораторная 5]

{

// Хешеры для отдельных полей

hash<string> shash;

hash<int> ihash;

hash <double> dhash;

size\_t operator()(const Book& obj) const //Функтор для задания хеш-функции для unordered\_set [Лабораторная 5]

{

const size\_t coef = 5171;

return (coef \* coef \* coef \* coef \* shash(obj.FIO) + coef \* coef \* coef \* shash(obj.Title) + coef \* coef \* shash(obj.publishing\_house) + coef \* ihash(obj.year) + dhash(obj.pages));

}

};

ostream& operator << (ostream& fout, const Book& book) //Объявление дружественной перегруженной операции вставки объекта в поток cout

{

fout << "Автор: " << book.FIO << endl;

fout << "Название: " << book.Title << endl;

fout << "Издательство: " << book.publishing\_house << endl;

fout << "Год издания: " << book.year << endl;

fout << "Количество страниц: " << book.pages << endl;

return fout;

}

bool bool\_sort(Book book1, Book book2) //булева функция для правильной сортировки элементов контейнера deque

{

return size(book1.FIO) > size(book2.FIO);

}

int main()

{

setlocale(LC\_ALL, "Rus"); //реализуем возможность печати русских букв

set <Book> books\_set; //Создаём контейнер множество [Лабораторная 5]

unordered\_set <Book, Hasher> books\_unord\_set; //Создаём контейнер неупорядоченное множество [Лабораторная 5]

deque<Book> books\_deque; //Создаём контейнер очередь

list<Book> books\_list; //Создаём контейнер список

string temp, fio, title, publish, pages\_str, year\_str;

int count = -1, flag = 0, pages = 0, year = 0;

deque <string> temp\_deque;

ifstream fin;

fin.open("input.txt"); //Открываем файл для чтения

while (!fin.eof()) //Цикл поочередной записи всех строк из файла в вспомогательную очередь

{

getline(fin, temp);

temp\_deque.push\_back(temp);

temp = "";

++count;

}

fin.close(); //закрываем файл после чтения

for (int i = 0; i < count; ++i) //Основной цикл работы с данными, полученными из файла. На выходе получаем очередь(books\_deque) из объектов класса Book

{

temp = temp\_deque[i];

for (int j = 0; j < size(temp); ++j) //Цикл, с помощью которого мы сплошную строку разбиваем на строки для полей класса Book

{

if (temp[j] == ';')

{

++flag;

continue;

}

if (flag == 0)

{

fio += temp[j];

}

if (flag == 1)

{

title += temp[j];

}

if (flag == 2)

{

publish += temp[j];

}

if (flag == 3)

{

year\_str += temp[j];

}

if (flag == 4)

{

pages\_str += temp[j];

}

}

istringstream(year\_str) >> year; //Преобразуем строку в число

istringstream(pages\_str) >> pages; //Преобразуем строку в число

Book book(fio, title, publish, year, pages); //Создаём объект класса Book

books\_set.insert(book); //Добавляем объект класса Book в множество books\_set [Лабораторная 5]

books\_unord\_set.insert(book); //Добавляем объект класса Book в неупорядоченное множество books\_unord\_set [Лабораторная 5]

books\_deque.push\_back(book); //Добавляем объект класса Book в очередь books\_deque

temp = fio = title = publish = year\_str = pages\_str = "";

flag = 0;

}

copy(books\_deque.begin(), books\_deque.end(), inserter(books\_list, books\_list.begin())); //Вызов функции копирования контейнера типа deque в контейнер типа list

ofstream fout;

fout.open("output.txt"); //Открываем файл для записи

fout << "Исходный контейнер deque:" << endl;

for (int i = 0; i < count; ++i)

{

fout << books\_deque[i] << endl;

}

fout << endl;

sort(books\_deque.begin(), books\_deque.end(), bool\_sort); //Вызов функции сортировки контейнера deque при помощи булевой функции

fout << "Исходный контейнер deque после сортировки:" << endl;

for (int i = 0; i < count; ++i)

{

fout << books\_deque[i] << endl;

}

fout << endl;

fout << "Контейнер, в который копирован исходный контейнер(list):" << endl;

for (auto i = books\_list.begin(); i != books\_list.end(); ++i) //Цикл для вывода контейнера типа list в файл

{

fout << \*i << endl;

}

fout << endl;

fout << "Контейнер set:" << endl;

for(auto i = books\_set.begin(); i != books\_set.end(); ++i) //Цикл для вывода контейнера типа set в файл [Лабораторная 5]

{

fout << \*i << endl;

}

fout << "Контейнер unordered\_set:" << endl;

for (auto i = books\_unord\_set.begin(); i != books\_unord\_set.end(); ++i) //Цикл для вывода контейнера типа unordered\_set в файл [Лабораторная 5]

{

fout << \*i << endl;

}

fout.close(); //Закрываем файл после записи

return 0;

}

Содержимое файла input.txt:

Leo Nikolaevich Tolstoy; War and Peace;Eksmo;2010;736

Fyodor Mikhailovich Dostoevsky; Crime and Punishment;ACT;2021;544

Alexander Sergeevich Pushkin;Eugene Onegin;Myth;2023;288

Содержимое файла output.txt:

Исходный контейнер deque:

Автор: Leo Nikolaevich Tolstoy

Название: War and Peace

Издательство: Eksmo

Год издания: 2010

Количество страниц: 736

Автор: Fyodor Mikhailovich Dostoevsky

Название: Crime and Punishment

Издательство: ACT

Год издания: 2021

Количество страниц: 544

Автор: Alexander Sergeevich Pushkin

Название: Eugene Onegin

Издательство: Myth

Год издания: 2023

Количество страниц: 288

Исходный контейнер deque после сортировки:

Автор: Fyodor Mikhailovich Dostoevsky

Название: Crime and Punishment

Издательство: ACT

Год издания: 2021

Количество страниц: 544

Автор: Alexander Sergeevich Pushkin

Название: Eugene Onegin

Издательство: Myth

Год издания: 2023

Количество страниц: 288

Автор: Leo Nikolaevich Tolstoy

Название: War and Peace

Издательство: Eksmo

Год издания: 2010

Количество страниц: 736

Контейнер, в который копирован исходный контейнер(list):

Автор: Leo Nikolaevich Tolstoy

Название: War and Peace

Издательство: Eksmo

Год издания: 2010

Количество страниц: 736

Автор: Fyodor Mikhailovich Dostoevsky

Название: Crime and Punishment

Издательство: ACT

Год издания: 2021

Количество страниц: 544

Автор: Alexander Sergeevich Pushkin

Название: Eugene Onegin

Издательство: Myth

Год издания: 2023

Количество страниц: 288

Контейнер set:

Автор: Leo Nikolaevich Tolstoy

Название: War and Peace

Издательство: Eksmo

Год издания: 2010

Количество страниц: 736

Автор: Alexander Sergeevich Pushkin

Название: Eugene Onegin

Издательство: Myth

Год издания: 2023

Количество страниц: 288

Автор: Fyodor Mikhailovich Dostoevsky

Название: Crime and Punishment

Издательство: ACT

Год издания: 2021

Количество страниц: 544

Контейнер unordered\_set:

Автор: Leo Nikolaevich Tolstoy

Название: War and Peace

Издательство: Eksmo

Год издания: 2010

Количество страниц: 736

Автор: Fyodor Mikhailovich Dostoevsky

Название: Crime and Punishment

Издательство: ACT

Год издания: 2021

Количество страниц: 544

Автор: Alexander Sergeevich Pushkin

Название: Eugene Onegin

Издательство: Myth

Год издания: 2023

Количество страниц: 288

Вывод:

Я научился использовать объекты своих классов в упорядоченных и неупорядоченных контейнерах библиотеки STL (set и map, unordered\_set и unordered\_map)