|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного автономного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №3**

**«Перегрузка операторов»**

**ДИСЦИПЛИНА: «Высокоуровневое программирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-22Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Кисвянцев Д.М.)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Пчелинцева Н.И.)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2025 | | |

**Цель работы:** приобретение практических навыков и знаний по работе с перегрузкой операторов.

**Задачи**:

1. Изучить понятие оператора;
2. Выяснить виды и способы перегрузки операторов;
3. Изучить методы и случаи применения перегрузок;
4. Научиться соединять пользовательские объекты с потоками ввода / вывода;
5. Познакомиться с понятием функтора.
6. Научиться применять перегрузку операторов на практике;

**Общее задание для всех вариантов**

1. Написать для пользовательских классов дружественные функции перегрузки операторов ввода и вывода.
2. Написать перегрузки операторов сравнения двух объектов пользовательского класса, как методов внутри класса.
3. На данном этапе хранение списков данных в программе (пользователи и другие объекты) может быть реализовано в виде простых массивов в главном файле программы.
4. Создать функции взаимодействия пользователя приложения с данными: добавление (используя перегрузки операторов ввода) и удаление, сортировка (с помощью перегрузок операторов сравнения).
5. Совместить созданный функционал с меню.

**Обновлённый листинг подкласса Astronaut**

wostream& operator<<(wostream& aout, const Astronaut& astronaut) {

aout << L"Космонавт: " << astronaut.getSurname() << L" " << astronaut.getName() << endl;

aout << L"Возраст: " << astronaut.getAge() << endl;

aout << L"Миссия: " << astronaut.getMission() << endl;

return aout;

}

wistream& operator>>(wistream& ain, Astronaut& astronaut){

wstring surname, name, mission;

int age;

wcout << L"Введите фамилию: ";

getline(ain, surname);

wcout << L"Введите имя: ";

getline(ain, name);

wcout << L"Введите возраст: ";

ain >> age;

ain.ignore();

wcout << L"Введите миссию: ";

getline(ain, mission);

astronaut.m\_surname = surname;

astronaut.m\_name = name;

astronaut.m\_age = age;

astronaut.m\_mission = mission;

return ain;

}

bool Astronaut::operator<(const Astronaut& other) const{

return m\_surname < other.m\_surname;

}

bool Astronaut::operator>(const Astronaut& other) const{

return m\_surname > other.m\_surname;

}

**Обновлённый листинг подкласса Engineer**

wostream& operator<<(wostream& eout, const Engineer& engineer) {

eout << L"Инженер: " << engineer.getSurname() << L" " << engineer.getName() << endl;

eout << L"Возраст: " << engineer.getAge() << endl;

eout << L"Специализация: " << engineer.getSpecialisation() << endl;

return eout;

}

wistream& operator>>(wistream& ein, Engineer& engineer){

wstring surname, name, specialisation;

int age;

wcout << L"Введите фамилию: ";

getline(ein, surname);

wcout << L"Введите имя: ";

getline(ein, name);

wcout << L"Введите возраст: ";

ein >> age;

ein.ignore();

wcout << L"Введите специализацию: ";

getline(ein, specialisation);

engineer.m\_surname = surname;

engineer.m\_name = name;

engineer.m\_age = age;

engineer.m\_specialisation = specialisation;

return ein;

}

bool Engineer::operator<(const Engineer& other) const{

return m\_surname < other.m\_surname;

}

bool Engineer::operator>(const Engineer& other) const{

return m\_surname > other.m\_surname;

}

**Обновлённый листинг подкласса Rocket**

wostream& operator<<(wostream& rout, const Rocket& rocket) {

rout << L"Название : " << rocket.getName() << endl;

rout << L"Тип: " << rocket.getType() << endl;

rout << L"Грузоподъёмность: " << rocket.getPayloadCapacity() << endl;

rout << L"Назначение: " << rocket.getPayloadCapacity() << endl;

return rout;

}

wistream& operator>>(wistream& rin, Rocket& rocket){

wstring name, type, purpose;

int payload\_capacity;

wcout << L"Введите название: ";

getline(rin, name);

wcout << L"Введите тип: ";

getline(rin, type);

wcout << L"Введите грузоподъёмность в тоннах: ";

rin >> payload\_capacity;

rin.ignore();

wcout << L"Введите назначение: ";

getline(rin, purpose);

rocket.m\_name = name;

rocket.m\_type = type;

rocket.m\_payload\_capacity = payload\_capacity;

rocket.m\_purpose = purpose;

return rin;

}

bool Rocket::operator<(const Rocket& other) const{

return m\_name < other.m\_name;

}

bool Rocket::operator>(const Rocket& other) const{

return m\_name > other.m\_name;

}

**Обновлённый листинг main.cpp**

int main(){

setlocale(LC\_ALL, "Russian");

vector<Astronaut\*> astronauts;

vector<Engineer\*> engineers;

Authorisation authorisation;

wstring role = authorisation.AuthorisationMenu();

if (!role.empty()) {

MainMenu(role, astronauts, engineers);

}

for (Astronaut\* astronaut : astronauts) {

delete astronaut;

}

for (Engineer\* engineer : engineers){

delete engineer;

}

}

**Листинг Authorisation.cpp**

#include "Authorisation.h"

#include <fstream>

#include <iostream>

#include <vector>

#include <sstream>

using namespace std;

wstring Authorisation::SignIn(){

wstring login, password;

wcout << L"Введите логин: ";

getline(wcin, login);

wcout << L"Введите пароль: ";

getline(wcin, password);

wifstream file("user.txt");

if (!file.is\_open()){

wcout << L"Ошибка открытия файла!" << endl;

return L"";

}

wstring line;

while (getline(file, line)) {

wistringstream wiss(line);

wstring wissLogin, wissPassword, wissRole;

wiss >> wissLogin >> wissPassword >> wissRole;

if (wissLogin == login and wissPassword == password){

if (wissRole == L"admin"){

wcout << L"Вы вошли как администратор!" << endl;

}

else{

wcout << L"Вы вошли как пользователь!" << endl;

}

return wissRole;

}

}

wcout << L"Неверный логин или пароль!" << endl;

return L"";

}

wstring Authorisation::SignUp(){

wstring login, password, role;

wcout << L"Введите логин: ";

getline(wcin, login);

wcout << L"Введите пароль: ";

getline(wcin, password);

wcout << L"Введите роль (admin/user): ";

getline(wcin, role);

wofstream file("user.txt", ios::app);

if (!file.is\_open()){

wcout << L"Ошибка открытия файла!" << endl;

return L"";

}

file << login << " " << password << " " << role << endl;

file.close();

wcout << L"Вы успешно зарегистрировались!" << endl;

return L"";

}

void Authorisation::ExportToFile(vector<Astronaut\*>& astronauts, vector<Engineer\*>& engineers){

wofstream file("user.txt", ios::app);

if (!file.is\_open()){

wcout << L"Ошибка открытия файла!" << endl;

}

for (Astronaut\* emp : astronauts){

file << emp->getLogin() << " " << emp ->getPassword() << " " << "user" << endl;

}

for (Engineer\* sup : engineers){

file << sup->getLogin() << " " << sup->getPassword() << " " << "user" << endl;

}

file.close();

wcout << L"Записи успешно сохранены в файл!" << endl;

}

wstring Authorisation::AuthorisationMenu(){

while (true){

wcout << L"1 - Войти" << endl;

wcout << L"2 - Зарегистрироваться" << endl;

wcout << endl;

int command;

wcout << L"Введите команду >> ";

wcin >> command;

wcin.ignore();

switch(command){

case 1:{

wstring role = Authorisation::SignIn();

if (!role.empty()){

return role;

}

break;

}

break;

case 2:

Authorisation::SignUp();

break;

default:

wcout << L"Неверная команда! Попробуйте еще раз" << endl;

}

}

return L"";

}

**Листинг Utils.cpp**

#include "Utils.h"

#include "../User/User.h"

#include "../Astronaut/Astronaut.h"

#include "../Engineer/Engineer.h"

#include "../User/User.h"

#include "../Authorisation/Authorisation.h"

#include <algorithm>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

void SortAstronauts(vector<Astronaut\*>& astronauts){

int sortcommand;

wcout << L"Сортировать:" << endl;

wcout << L"1 - По возрастанию" << endl;

wcout << L"2 - По убыванию" << endl;

wcout << L"Введите команду >>";

wcin >> sortcommand;

switch(sortcommand){

case 1:

sort(astronauts.begin(), astronauts.end(), [](Astronaut\* a, Astronaut\* b){

return \*a < \*b;

});

break;

case 2:

sort(astronauts.begin(), astronauts.end(), [](Astronaut\* a, Astronaut\* b){

return \*a > \*b;

});

break;

default:

wcout << L"Неверная команда!" << endl;

break;

}

}

void SortEngineers(vector<Engineer\*>& engineers){

int sortcommand;

wcout << L"Сортировать:" << endl;

wcout << L"1 - По возрастанию" << endl;

wcout << L"2 - По убыванию" << endl;

wcout << L"Введите команду >>";

wcin >> sortcommand;

switch(sortcommand){

case 1:

sort(engineers.begin(), engineers.end(), [](Engineer\* a, Engineer\* b){

return \*a < \*b;

});

break;

case 2:

sort(engineers.begin(), engineers.end(), [](Engineer\* a, Engineer\* b){

return \*a > \*b;

});

break;

default:

wcout << L"Неверная команда!" << endl;

break;

}

}

void DeleteAstronaut(vector<Astronaut\*>& astronauts){

int index;

for (int i = 0; i < astronauts.size(); i++){

wcout << i + 1 << L" ";

astronauts[i]->PrintInfo();

}

wcout << L"Введите индекс удаляемой записи: ";

wcin >> index;

delete astronauts[index - 1];

}

void DeleteEngineer(vector<Engineer\*>& engineers){

int index;

for (int i = 0; i < engineers.size(); i++){

wcout << i + 1 << L" ";

engineers[i]->PrintInfo();

}

wcout << L"Введите индекс удаляемой записи: ";

wcin >> index;

delete engineers[index - 1];

}

void FilterAstronaut(vector<Astronaut\*>& astronauts) {

int filterCommand;

wcout << L"Фильтровать космонавтов по:" << endl;

wcout << L"1 - Фамилии" << endl;

wcout << L"2 - Имени" << endl;

wcout << L"3 - Возрасту" << endl;

wcout << L"4 - Миссии" << endl;

wcout << L"Введите команду >>";

wcin >> filterCommand;

wcin.ignore();

switch(filterCommand) {

case 1: {

wstring surname;

wcout << L"Введите фамилию для фильтрации: ";

getline(wcin, surname);

for (Astronaut\* a : astronauts) {

if (a->getSurname().find(surname) != wstring::npos) {

wcout << \*a << endl;

}

}

break;

}

case 2: {

wstring name;

wcout << L"Введите имя для фильтрации: ";

getline(wcin, name);

for (Astronaut\* a : astronauts) {

if (a->getName().find(name) != wstring::npos) {

wcout << \*a << endl;

}

}

break;

}

case 3: {

int minAge, maxAge;

wcout << L"Введите минимальный возраст: ";

wcin >> minAge;

wcout << L"Введите максимальный возраст: ";

wcin >> maxAge;

for (Astronaut\* a : astronauts) {

int age = a->getAge();

if (age >= minAge && age <= maxAge) {

wcout << \*a << endl;

}

}

break;

}

case 4: {

wstring mission;

wcout << L"Введите название миссии: ";

getline(wcin, mission);

for (Astronaut\* a : astronauts) {

if (a->getMission().find(mission) != wstring::npos) {

wcout << \*a << endl;

}

}

break;

}

default:

wcout << L"Неверная команда!" << endl;

break;

}

}

void FilterEngineer(vector<Engineer\*>& engineers) {

int filterCommand;

wcout << L"Фильтровать инженеров по:" << endl;

wcout << L"1 - Фамилии" << endl;

wcout << L"2 - Имени" << endl;

wcout << L"3 - Возрасту" << endl;

wcout << L"4 - Специализации" << endl;

wcout << L"Введите команду >>";

wcin >> filterCommand;

wcin.ignore();

switch(filterCommand) {

case 1: {

wstring surname;

wcout << L"Введите фамилию для фильтрации: ";

getline(wcin, surname);

for (Engineer\* e : engineers) {

if (e->getSurname().find(surname) != wstring::npos) {

wcout << \*e << endl;

}

}

break;

}

case 2: {

wstring name;

wcout << L"Введите имя для фильтрации: ";

getline(wcin, name);

for (Engineer\* e : engineers) {

if (e->getName().find(name) != wstring::npos) {

wcout << \*e << endl;

}

}

break;

}

case 3: {

int minAge, maxAge;

wcout << L"Введите минимальный возраст: ";

wcin >> minAge;

wcout << L"Введите максимальный возраст: ";

wcin >> maxAge;

for (Engineer\* e : engineers) {

int age = e->getAge();

if (age >= minAge && age <= maxAge) {

wcout << \*e << endl;

}

}

break;

}

case 4: {

wstring specialisation;

wcout << L"Введите специализацию: ";

getline(wcin, specialisation);

for (Engineer\* e : engineers) {

if (e->getSpecialisation().find(specialisation) != wstring::npos) {

wcout << \*e << endl;

}

}

break;

}

default:

wcout << L"Неверная команда!" << endl;

break;

}

}

void MainMenu(wstring& role, vector<Astronaut\*>& astronauts, vector<Engineer\*>& engineers) {

Astronaut\* astronaut = nullptr;

Engineer\* engineer = nullptr;

if (role == L"admin") {

while (true) {

wcout << L"1 - Вывести данные о пользователях" << endl;

wcout << L"0 - Выход из программы" << endl;

int command(-1);

wcout << L"Введите команду >> ";

wcin >> command;

switch(command) {

case 1:

for (Astronaut\* astronaut : astronauts) {

astronaut->PrintInfo();

wcout << L"\n" << endl;

}

for (Engineer\* engineer : engineers) {

engineer->PrintInfo();

wcout << L"\n" << endl;

}

break;

case 0:

return;

default:

wcout << L"Неверная команда! Попробуйте еще раз." << endl;

break;

}

}

}

else if (role == L"user") {

while (true) {

wcout << L"Меню:" << endl;

wcout << L"1 - Добавить космонавта" << endl;

wcout << L"2 - Вывести данные о космонавте" << endl;

wcout << L"3 - Добавить инженера" << endl;

wcout << L"4 - Вывести данные об инженере" << endl;

wcout << L"5 - Сортировка" << endl;

wcout << L"6 - Фильтрация" << endl;

wcout << L"7 - Удаление" << endl;

wcout << L"0 - Выход" << endl;

wcout << endl;

int command;

wcout << L"Введите команду >> ";

wcin >> command;

wcin.ignore();

switch(command) {

case 1: {

astronaut = new Astronaut();

wcin >> \*astronaut;

astronauts.push\_back(astronaut);

break;

}

case 2: {

for (Astronaut\* astronaut : astronauts) {

wcout << \*astronaut;

wcout << L"\n" << endl;

}

break;

}

case 3: {

engineer = new Engineer();

wcin >> \*engineer;

engineers.push\_back(engineer);

break;

}

case 4: {

for (Engineer\* engineer : engineers) {

wcout << \*engineer;

wcout << L"\n" << endl;

}

break;

}

case 5: {

int sortcommand(-1);

wcout << L"Кого вы хотите сортировать?" << endl;

wcout << L"1 - Космонавтов" << endl;

wcout << L"2 - Инженеров" << endl;

wcout << L"Введите команду >> ";

wcin >> sortcommand;

wcin.ignore();

switch(sortcommand) {

case 1:

SortAstronauts(astronauts);

break;

case 2:

SortEngineers(engineers);

break;

default:

wcout << L"Неверная команда! Попробуйте еще раз." << endl;

break;

}

break;

}

case 6: {

int filtercommand(-1);

wcout << L"Кого вы хотите фильтровать?" << endl;

wcout << L"1 - Космонавтов" << endl;

wcout << L"2 - Инженеров" << endl;

wcout << L"Введите команду >> ";

wcin >> filtercommand;

wcin.ignore();

switch(filtercommand) {

case 1:

FilterAstronaut(astronauts);

break;

case 2:

FilterEngineer(engineers);

break;

default:

wcout << L"Неверная команда! Попробуйте еще раз." << endl;

break;

}

break;

}

case 7: {

int deletecommand(-1);

wcout << L"Кого вы хотите удалить?" << endl;

wcout << L"1 - Космонавтов" << endl;

wcout << L"2 - Инженеров" << endl;

wcout << L"Введите команду >> ";

wcin >> deletecommand;

wcin.ignore();

switch(deletecommand) {

case 1:

DeleteAstronaut(astronauts);

break;

case 2:

DeleteEngineer(engineers);

break;

default:

wcout << L"Неверная команда! Попробуйте еще раз." << endl;

break;

}

break;

}

case 0: {

Authorisation authorisation;

authorisation.ExportToFile(astronauts, engineers);

return;

}

default: {

wcout << L"Неверная команда! Попробуйте еще раз." << endl;

break;

}

}

}

}

}

**Результат работы меню авторизации**

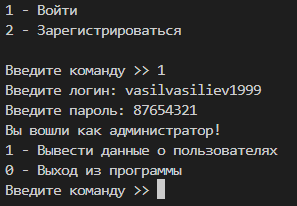
****

Рис. 1.1 Результат работы меню авторизации

**Результат работы меню регистрации**

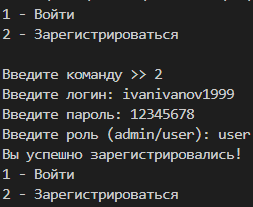


Рис. 1.2 Результат работы меню регистрации

**Выводы:**

В ходе выполнения лабораторной работы были получены практические и теоретические навыки перегрузки операторов