МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

**Лабораторная работа №4**

по дисциплине

«Объектно-ориентированное программирование»

**Выполнил:**

Ермолаев Кирилл Александрович

Студент 2 курса группы \_ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

Ставрополь, 2023 г.

**Тема:** Основы объектно-ориентированного программирования на ЯП С++.

**Цель работы:** Стандартные потоки.

**Ход работы:**

**Вариант 7**

1. Определить класс с именем TRAIN, содержащий следующие поля: название пункта назначения; номер поезда; время отправления. Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа TRAIN.

2. Написать программу, выполняющую следующие действия:

− ввод с клавиатуры данных в массив, состоящий из восьми объектов типа TRAIN; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;

− вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени;

− если таких поездов нет, выдать на дисплей соответствующее сообщение.

// main.cpp

#include "train.h"

#include <algorithm>

int main() {

const int arraySize = 4;

TRAIN trains[arraySize];

// Ввод данных в массив

for (int i = 0; i < arraySize; ++i) {

std::cout << "Enter details for train " << (i + 1) << ":\n";

std::cin >> trains[i];

}

// Сортировка по названиям пунктов назначения

std::sort(trains, trains + arraySize, [](const TRAIN& a, const TRAIN& b) {

return a.getDestination() < b.getDestination();

});

// Вывод информации о поездах, отправляющихся после введенного времени

std::string inputTime;

std::cout << "\nEnter the departure time to filter trains: ";

std::cin >> inputTime;

bool found = false;

for (const auto& train : trains) {

if (train.getDepartureTime() > inputTime) {

std::cout << train;

found = true;

}

}

// Вывод сообщения, если не найдено ни одного поезда

if (!found) {

std::cout << "No trains found after the specified time.\n";

}

return 0;

}

// train.h

#pragma once

#ifndef TRAIN\_H

#define TRAIN\_H

#include <iostream>

#include <string>

class TRAIN {

private:

std::string destination;

int trainNumber;

std::string departureTime;

public:

// Конструкторы

TRAIN();

TRAIN(const std::string& dest, int num, const std::string& time);

// Методы доступа

std::string getDestination() const;

int getTrainNumber() const;

std::string getDepartureTime() const;

// Перегруженные операции извлечения и вставки

friend std::ostream& operator<<(std::ostream& os, const TRAIN& train);

friend std::istream& operator>>(std::istream& is, TRAIN& train);

};

#endif

// train.cpp

#include "train.h"

// Реализация конструкторов

TRAIN::TRAIN() : trainNumber(0) {}

TRAIN::TRAIN(const std::string& dest, int num, const std::string& time)

: destination(dest), trainNumber(num), departureTime(time) {}

// Реализация методов доступа

std::string TRAIN::getDestination() const {

return destination;

}

int TRAIN::getTrainNumber() const {

return trainNumber;

}

std::string TRAIN::getDepartureTime() const {

return departureTime;

}

// Реализация операции вставки

std::ostream& operator<<(std::ostream& os, const TRAIN& train) {

os << "Destination: " << train.destination << "\n";

os << "Train Number: " << train.trainNumber << "\n";

os << "Departure Time: " << train.departureTime << "\n";

return os;

}

// Реализация операции извлечения

std::istream& operator>>(std::istream& is, TRAIN& train) {

std::cout << "Enter destination: ";

is >> train.destination;

std::cout << "Enter train number: ";

is >> train.trainNumber;

std::cout << "Enter departure time: ";

is >> train.departureTime;

return is;

}

// UnitTest4.cpp

#include <algorithm>

#include <vector>

#include "pch.h"

#include "CppUnitTest.h"

#include "../ConsoleApplication1/train.h"

#include "../ConsoleApplication1/train.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TRAINTests

{

TEST\_CLASS(TRAINTests)

{

public:

TEST\_METHOD(TestTrainSorting)

{

// Тест сортировки поездов по пунктам назначения

TRAIN trains[3];

// Заполнение массива поездов

trains[0] = TRAIN("Paris", 123, "08:00");

trains[1] = TRAIN("Berlin", 456, "09:30");

trains[2] = TRAIN("London", 789, "10:45");

// Ожидаемый порядок после сортировки

TRAIN expectedOrder[3] = { TRAIN("Berlin", 456, "09:30"),

TRAIN("London", 789, "10:45"),

TRAIN("Paris", 123, "08:00") };

// Ручная сортировка (пузырьковая сортировка для примера)

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 2 - i; ++j) {

if (trains[j].getDestination() > trains[j + 1].getDestination()) {

std::swap(trains[j], trains[j + 1]);

}

}

}

// Проверка сортировки

for (int i = 0; i < 3; ++i) {

Assert::AreEqual(expectedOrder[i].getDestination(), trains[i].getDestination());

Assert::AreEqual(expectedOrder[i].getTrainNumber(), trains[i].getTrainNumber());

Assert::AreEqual(expectedOrder[i].getDepartureTime(), trains[i].getDepartureTime());

}

}

TEST\_METHOD(TestTrainFiltering)

{

// Тест фильтрации поездов по времени отправления

TRAIN trains[3];

// Заполнение массива поездов

trains[0] = TRAIN("Paris", 123, "08:00");

trains[1] = TRAIN("Berlin", 456, "09:30");

trains[2] = TRAIN("London", 789, "10:45");

// Ожидаемые поезда после фильтрации

TRAIN expectedTrains[2] = { TRAIN("Berlin", 456, "09:30"),

TRAIN("London", 789, "10:45") };

// Фильтрация по времени

std::string filterTime = "09:00";

int filteredCount = 0;

for (int i = 0; i < 3; ++i) {

if (trains[i].getDepartureTime() > filterTime) {

// Печать или другая обработка вместо добавления в вектор

// std::cout << trains[i];

++filteredCount;

}

}

// Проверка фильтрации

Assert::AreEqual(2, filteredCount);

}

TEST\_METHOD(TestTrainConstructor)

{

TRAIN train("TestDestination", 123, "12:34");

Assert::AreEqual("TestDestination", train.getDestination().c\_str());

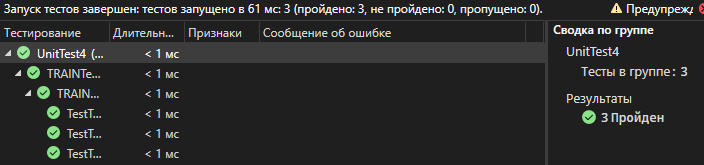
Assert::AreEqual(123, train.getTrainNumber());

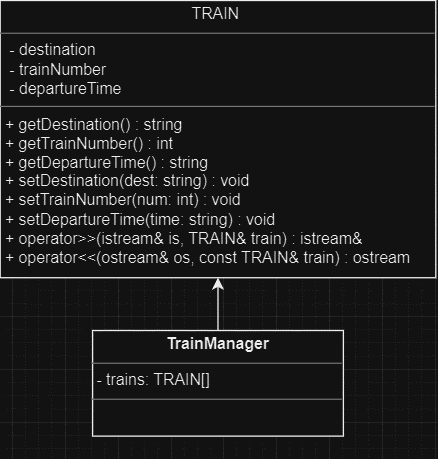
Assert::AreEqual("12:34", train.getDepartureTime().c\_str());

}

};

}





<https://github.com/MoeTomatoki/OOP>

**Вывод:** получил базовые знания по работе со стандартными потоками и научился работать с ними.