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:**

Иванов Даниил Евгеньевич

Студент 2 курса группы \_ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

Ставрополь, 2023 г.

Тема: Стандартные потоки.

Цель работы: изучить стандартные потоки и научится реализовать их.

Выполнение работы:

Вариант -16

1. Определить класс с именем ZNAK, содержащий следующие поля:

Фамилия, имя; знак Зодиака; день рождения (массив из трех чисел).

Определить методы доступа к этим полям и перегруженные операции

извлечения и вставки для объектов типа ZNAK.

2. Написать программу, выполняющую следующие действия:

− ввод с клавиатуры данных в массив, состоящий из восьми объектов типа

ZNAK; записи должны быть упорядочены по датам дней рождения;

− вывод на экран информации о людях, родившихся под знаком, наименование

которого введено с клавиатуры;

− если таких нет, выдать на дисплей соответствующее сообщение.

Листинг:

// main.cpp

#include "ZNAK.h"

#include <algorithm>

#include "UnitTest.h"

//#include <Windows.h>

int main() {

//SetConsoleCP(1251);

//SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

const int size = 8;

ZNAK znaks[size];

// Входные данные

int i = 0;

while (i < size) {

std::cout << "Введите данные о человек " << i + 1 << ":\n";

std::cin >> znaks[i];

// Увеличивайте счетчик только в том случае, если ввод успешен.

if (std::cin) {

++i;

}

else {

// Очистите флаг ошибки и отбросьте неверный ввод.

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

std::cout << "Неверный Ввод. Пожалуйста, попробуйте еще раз.\n";

}

}

// Сортировка по дате рождения

std::sort(znaks, znaks + size, [](const ZNAK& a, const ZNAK& b) {

return std::lexicographical\_compare(a.getBirthdate(), a.getBirthdate() + 3, b.getBirthdate(), b.getBirthdate() + 3);

});

// Вывод отсортированных данных

std::cout << "\nСортированные данные:\n";

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

std::cout << znaks[i];

}

// Поиск по знаку зодиака

std::string searchSign;

std::cout << "\nВведите знак зодиака для поиска: ";

std::cin >> searchSign;

bool found = false;

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

if (znaks[i].getZodiacSign() == searchSign) {

found = true;

std::cout << "\nЛюди, рожденные под " << searchSign << ":\n";

std::cout << znaks[i];

}

}

if (!found) {

std::cout << "Не найдено людей со знаком Зодиака: " << searchSign << "\n";

}

return 0;

testZNAK();

}

// ZNAK.h

#ifndef ZNAK\_H

#define ZNAK\_H

#include <iostream>

#include <string>

class ZNAK {

private:

std::string lastName;

std::string firstName;

std::string zodiacSign;

int birthdate[3];

public:

// Конструктор

ZNAK();

ZNAK(std::string lastName, std::string firstName, std::string zodiacSign, int birthdate[3]);

// Методы доступа

std::string getLastName() const;

std::string getFirstName() const;

std::string getZodiacSign() const;

const int\* getBirthdate() const; // Updated to const

// Перегруженные операторы

friend std::istream& operator>>(std::istream& is, ZNAK& znak);

friend std::ostream& operator<<(std::ostream& os, const ZNAK& znak);

};

#endif // ZNAK\_H

// ZNAK.cpp

#include "ZNAK.h"

// Констурктор

ZNAK::ZNAK() {}

ZNAK::ZNAK(std::string lastName, std::string firstName, std::string zodiacSign, int birthdate[3]) :

lastName(lastName), firstName(firstName), zodiacSign(zodiacSign) {

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

this->birthdate[i] = birthdate[i];

}

}

// Методы доступа

std::string ZNAK::getLastName() const {

return lastName;

}

std::string ZNAK::getFirstName() const {

return firstName;

}

std::string ZNAK::getZodiacSign() const {

return zodiacSign;

}

const int\* ZNAK::getBirthdate() const {

return birthdate;

}

// Перегруженные операторы

std::istream& operator>>(std::istream& is, ZNAK& znak) {

std::cout << "Введите фамилию: ";

is >> znak.lastName;

std::cout << "Введите имя: ";

is >> znak.firstName;

std::cout << "Введите знак зодиака: ";

is >> znak.zodiacSign;

std::cout << "Введите дату рождения (день месяц год): ";

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

is >> znak.birthdate[i];

}

return is;

}

std::ostream& operator<<(std::ostream& os, const ZNAK& znak) {

os << "Имя и Фамилия: " << znak.firstName << " " << znak.lastName << "\n";

os << "Знак зодиака: " << znak.zodiacSign << "\n";

os << "Дата рождения: ";

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

os << znak.birthdate[i] << " ";

}

os << "\n";

return os;

}

// UnitTest.cpp

#include <cassert>

#include <string.h>

#include "ZNAK.h"

#include "UnitTest.h"

using namespace std;

void testZNAK() {

int date[]{ 1, 2, 1990 };

//Тест конструктора

ZNAK znak(string("Иванов"), string("Иван"), string("Овен"), date);

//Тест методов доступа

assert(znak.getLastName() == "Иванов");

assert(znak.getFirstName() == "Иван");

assert(znak.getZodiacSign() == "Овен");

const int\* birthdate = znak.getBirthdate();

assert(birthdate[0] == 1);

assert(birthdate[1] == 2);

assert(birthdate[2] == 1990);

// Если все тесты пройдены успелшно, выоводим сообщение об успешном завершении

cout << "Все тесты пройдены!\n";

}

/\*

int testMain() {

// Вызов тестов

testZNAK();

return 0;

}

\*/

// UnitTest.h

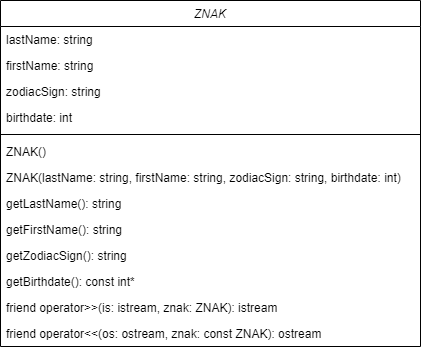
#ifndef UNIT\_TEST\_H

#define UNIT\_TEST\_H

// Объявление функции тестирования

void testZNAK();

#endif // UNIT\_TEST\_H



Вывод: изучил стандартные потоки и научился реализовать их.