Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе**

Дисциплина: Основы алгоритмизации и программирования

Тема: Сохранение данных в файле с использованием потоков

Выполнил работу

студент группы РИС-22-1б

Рыжков Н.С.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2023

**ТЕХНИЧЕСКАЯ СПЕЦИФИКАЦИЯ ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ №10**

Программа предназначена для управления объектами класса Time, которые представляют собой время в минутах и секундах. Пользователь может создавать, сохранять, читать, удалять, уменьшать и добавлять объекты класса Time.

**Функциональные требования**

*Создание объекта*

Функция createObject позволяет пользователю создать новый объект класса Time. При вызове функции пользователь должен ввести значения минут и секунд для нового объекта.

*Сохранение объектов*

Функция saveObjects сохраняет все объекты класса Time в файл times.txt. Если файл уже существует, он будет перезаписан.

*Чтение объектов*

Функция readObjects читает все объекты класса Time из файла times.txt и выводит их на экран.

*Удаление объектов*

Функция deleteObjects позволяет пользователю удалить объект класса Time. При вызове функции пользователь должен ввести значения минут и секунд для объекта, который нужно удалить.

*Уменьшение объектов*

Функция decreaseObjects позволяет пользователю уменьшить значение объекта класса Time на 90 секунд. При вызове функции пользователь должен ввести значения минут и секунд для объекта, который нужно уменьшить.

*Добавление объектов*

Функция addObjects позволяет пользователю добавить несколько новых объектов класса Time после заданного объекта. При вызове функции пользователь должен ввести значения минут и секунд для объекта, после которого нужно добавить новые объекты, а также количество новых объектов.

*Выход из программы*

Пользователь может выйти из программы, выбрав соответствующий пункт в меню.

**Нефункциональные требования**

*Язык программирования*

Программа должна быть написана на языке C++.

*Интерфейс пользователя*

Программа должна иметь текстовый интерфейс пользователя. Все действия должны выполняться через меню.

*Хранение данных*

Все объекты класса Time должны храниться в векторе. При сохранении объектов они должны быть записаны в файл times.txt, а при чтении объектов они должны быть прочитаны из этого файла.

*Обработка ошибок*

Программа должна корректно обрабатывать ошибки, связанные с некорректным вводом пользователем. Если пользователь вводит некорректные данные, программа должна сообщать об этом и предлагать повторить ввод.

**Листинг программы**

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

class Time {

private:

int minutes;

int seconds;

public:

Time() {

minutes = 0;

seconds = 0; }

Time(int m, int s) {

minutes = m;

seconds = s;

if (seconds >= 60) {

minutes += seconds / 60;

seconds %= 60; } }

friend ostream& operator<<(ostream& os, const Time& t) {

os << t.minutes << ":" << t.seconds;

return os; }

friend istream& operator>>(istream& is, Time& t) {

is >> t.minutes >> t.seconds;

if (t.seconds >= 60) {

t.minutes += t.seconds / 60;

t.seconds %= 60; }

return is; }

Time operator+(const int& s) {

Time t(minutes, seconds + s);

return t; }

Time operator-(const int& s) {

Time t(minutes, seconds - s);

if (t.seconds < 0) {

t.minutes -= 1;

t.seconds += 60; }

return t; }

Time operator--() {

if (minutes > 0 || seconds >= 90) {

seconds -= 90;

if (seconds < 0) {

minutes -= 1;

seconds += 60; } }

return \*this; }

bool operator==(const Time& t2) {

return (minutes == t2.minutes && seconds == t2.seconds); }

bool operator!=(const Time& t2) {

return (minutes != t2.minutes || seconds != t2.seconds); } };

void createObject(vector<Time>& times) {

Time t;

cout << "Enter minutes and seconds: ";

cin >> t;

times.push\_back(t);}

void saveObjects(vector<Time>& times) {

ofstream file("times.txt");

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

file << times[i] << endl; }

file.close();}

void readObjects(vector<Time>& times) {

ifstream file("times.txt");

Time t;

while (file >> t) {

times.push\_back(t); }

file.close();}

void deleteObjects(vector<Time>& times) {

Time t;

cout << "Enter minutes and seconds to delete: ";

cin >> t;

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

if (times[i] == t) {

times.erase(times.begin() + i);

i--; } } }

void decreaseObjects(vector<Time>& times) {

Time t;

cout << "Enter minutes and seconds to decrease: ";

cin >> t;

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

if (times[i] == t) {

--times[i];

if (times[i] < Time(0, 0)) {

times[i] = Time(0, 0); } } } }

void addObjects(vector<Time>& times) {

Time t;

cout << "Enter minutes and seconds to add after: ";

cin >> t;

int k;

cout << "Enter number of objects to add: ";

cin >> k;

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

if (times[i] == t) {

for (int j = 0; j < k; j++) {

createObject(times);

}

break;

}

}

}

int main() {

vector<Time> times;

int choice;

do {

cout << "1. Create object\n";

cout << "2. Save objects\n";

cout << "3. Read objects\n";

cout << "4. Delete objects\n";

cout << "5. Decrease objects\n";

cout << "6. Add objects\n";

cout << "7. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

createObject(times);

break;

case 2:

saveObjects(times);

break;

case 3:

readObjects(times);

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

cout << times[i] << endl;

}

break;

case 4:

deleteObjects(times);

break;

case 5:

decreaseObjects(times);

break;

case 6:

addObjects(times);

break;

case 7:

break;

default:

cout << "Invalid choice\n";

break;

}

} while (choice != 7);

return 0;

