Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.10**

Дисциплина: «Информатика»

Тема: Сохранение данных в файле с использованием потоков

Вариант 4

Выполнил:

Студент группы РИС-20-1б

Еске Вячеслав Сергеевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель задачи**

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Разработка программы, в которой данные сохраняются в файле, корректируются и выводятся из файла на печать. Работа с файлом осуществляется с использованием потоковых классов.

Для достижения поставленной цели необходимо решить следующие задачи:

* Проанализировать теоретические сведения о ООП
* Абстрактные типы данных. Контейнеры
* Реализовать алгоритм поставленной задачи
* Учесть все исключительные ситуации на ввод элементов
* Создать отдельные файлы: .cpp для описания методов класса, .cpp для описания главной функции, .h для описания класса

Постановка задачи

Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием.

Реализовать:

 добавление секунд (учесть, что в минуте не может быть более 60 секунд)

 вычитание секунд

Задание:

 Удалить все записи равные заданному значению.

 Уменьшить все записи с заданным значением на 1 минуту 30 секунд. Значение интервала не должно быть меньше 0 минут 0 секунд.

 Добавить K записей после элемента с заданным значением.

Анализ задачи

1. Определить какие операции должны быть выполнены по заданию:

* Создание класса

class Money

* Создание группы сумм, заполнение класса
* Создание методов получения суммы по номеру, печати, конструкторов
* Отчистка памяти

1. Для решения задачи используются переменные:

Несколько переменных целочисленного значения для ввода чисел и реализации меню

int s, in, menu = 4;

Указатель для работы с динамической памятью

1. Ввод данных осуществляется посредством функции

cin >> rub;

Вывод данных осуществляется посредством функции

cout << "Искомый элемент = " << a.give(in);

1. Создание метода класса , который печатает сумму:

void Money::Print()

{

cout << "\nРубли - " << rub << ", копейки - " << cop;

1. Создание методов класса для нахождения размера и доступа по индексу

int give(int index);

int givesize();

Создание контейнера

Money::Money(int r, long int c)

{

rub = r;

cop = c;

double co = c;

sum = r + co / 100;

}

1. Создание инструмента обработки исключительных ситуаций. Для этого была использована логика try:

try

{

cout << "Искомый элемент: " << a.give(in);

}

catch (const int in)

{

cout << "Ошибка ввода";

}

}

Код

#include "Time.h"

#include "file\_work.h"

#include <iostream>

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

Time t,a,b,key;

int k, c;

char file\_name[30];

do {

//Меню

cout << "\n1. Создать файл";//создание файла

cout << "\n2. Печать файла";//печать файла

cout << "\n3. Удаление элементов";

cout << "\n4. Изменение элементов";

cout << "\n5. Добавление элементов";

cout<<"\n0. Выход\n";

cin>>c;

switch (c)

{

case 1:

cout << "Введите имя файла: ";

cin >> file\_name;//задаем имя файла

k = make\_file(file\_name);//вызов функции для записи в файл

if (k < 0)cout << "Не удается создать файл";//вывод сообщения об ошибке

break;

case 2:

cout << "Введите имя файла: ";

cin >> file\_name;//задаем имя файла

k = print\_file(file\_name);//вызов функции для печати файла

if (k == 0)cout << "Пустой файл\n";//если файл пустой

if (k < 0)cout << "Не удается открыть файл\n";//если файл нельзя открыть

break;

case 3:

cout << "Введите имя файла: ";

cin >> file\_name;//задаем имя файла

cout << "Введите левую границу: ";

cin >> a;

do {

cout << "Введите правую границу: ";

cin >> b;

} while (b <= a);

k = del\_file(file\_name, a, b);

if (k == 0)cout << "Пустой файл\n";//если файл пустой

if (k < 0)cout << "Не удается открыть файл\n";//если файл нельзя открыть

break;

case 4:

cout << "Введите имя файла: ";

cin >> file\_name;//задаем имя файла

cout << "Введите ключ для изменения элементов: ";

cin >> key;

k = change\_file(file\_name, key);

if (k == 0)cout << "Нет элементов с таким ключом\n";//если файл пустой

if (k < 0)cout << "Не удается открыть файл\n";//если файл нельзя открыть

break;

case 5:

cout << "Введите имя файла: ";

cin >> file\_name;//задаем имя файла

cout << "Введите количество добавляемых элементов: ";

cin >> k;

k = add\_file(file\_name, k);

if (k == 0)cout << "Пустой файл\n";//если файл пустой

if (k < 0)cout << "Не удается открыть файл\n";//если файл нельзя открыть

break;

}

}

while (c != 0);

}

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Time

{

public:

Time() { min = 0; sec = 0; };

Time(int m, int s) { min = m; sec = s; }

Time(const Time& t) { min = t.min; sec = t.sec; }

~Time() {};

Time& operator=(const Time&);

Time& operator+(int);

Time operator+(const Time&);

friend bool operator==(const Time& t1, const Time& t2);

friend bool operator!=(const Time& t1, const Time& t2);

friend istream& operator>>(istream& in, Time& t);

friend ostream& operator<<(ostream& out, const Time& t);

friend fstream& operator>>(fstream& fin, Time& t);

friend fstream& operator <<(fstream& fout, const Time& t);

bool operator >(const Time&);

bool operator <(const Time&);

bool operator >=(const Time&);

bool operator <=(const Time&);

bool operator ==(const Time&);

bool operator !=(const Time&);

private:

int min;

int sec;

};

#include "Time.h"

//перегрузка операции присваивания

Time& Time::operator=(const Time& t) {

//проверка на самоприсваивание

if (&t == this) return \*this;

min = t.min;

sec = t.sec;

return \*this;

}

//перегрузка операции добавления секунд

Time& Time::operator+(int s) {

int temp = min \* 60 + sec+s;

min = temp / 60;

sec = temp % 60;

return \*this;

}

//перегрузка глобальной функции-операции ввода

istream& operator>>(istream& in, Time& t) {

cout << "Введите минуты - ";

in >> t.min;

do {

cout << "Введите секунды - ";

in >> t.sec;

} while ((t.sec < 0) || (t.sec >= 60));

return in;

}

//перегрузка глобальной функции-операции вывода

ostream& operator<<(ostream& out, const Time& t) {

return (out << t.min << " : " << t.sec);

}

//перегрузка функции сравнения

bool operator==(const Time& t1, const Time& t2) {

if (t1.min == t2.min && t1.sec == t2.sec)

return true;

else return false;

}

//перегрузка функции неравенства

bool operator!=(const Time& t1, const Time& t2) {

if (t1.min != t2.min && t1.sec != t2.sec)

return true;

else return false;

}

//дружественные функции для работы с файловыми потоками

fstream& operator>>(fstream& fin, Time&t) {

string buf;

fin >> t.min;

fin >> buf;

fin >> t.sec;

return fin;

}

fstream& operator<<(fstream& fout, const Time& t) {

fout << t.min << " : " << t.sec;

return fout;

}

bool Time::operator <(const Time& t) {

if (min < t.min)return true;

if (min == t.min && sec < t.sec)

return true;

return false;

}

bool Time::operator >(const Time& t) {

if (min > t.min)return true;

if (min == t.min && sec > t.sec)

return true;

return false;

}

bool Time::operator <=(const Time& t) {

if (min <= t.min)return true;

if (min == t.min && sec <= t.sec)

return true;

return false;

}

bool Time::operator >=(const Time& t) {

if (min >= t.min)return true;

if (min == t.min && sec >= t.sec)

return true;

return false;

}

bool Time::operator ==(const Time& t) {

if (min == t.min && sec == t.sec)

return true;

return false;

}

bool Time::operator !=(const Time& t) {

if (min != t.min || sec != t.sec)

return true;

return false;

}

Time Time::operator+(const Time& t) {

int temp1 = min \* 60 + sec;

int temp2 = t.min \* 60 + t.sec;

Time p;

p.min = (temp1 + temp2) / 60;

p.sec = (temp1 + temp2) % 60;

return p;

}

#pragma once

#include "Time.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int make\_file(const char\* f\_name) {

fstream stream(f\_name, ios::out | ios::trunc);//открыть для записи

if (!stream)return -1;//ошибка открытия файла

int n;

Time t;

cout << "Введите количество атрибутов: ";

cin >> n;

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

{

cin >> t;//ввод атрибутов объекта из стандартного потока

stream << t << "\n";//запись объекта в файловый поток

}

stream.close();//закрыть поток

return n;//вернуть количество записанных объектов

}

int print\_file(const char\* f\_name)

{

fstream stream(f\_name, ios::in);//открыть для чтения

if(!stream)return -1;//ошибка открытия файла

Time t;

int i=0;

while(stream>>t)

{

cout << t << "\n";

i++;

}

stream.close();

return i;

}

int del\_file(const char\* f\_name, Time a, Time b) {

fstream temp("temp", ios::out);//открыть для записи

fstream stream(f\_name, ios::in);//открыть для чтения

if(!stream)return -1;//ошибка открытия файла

int i=0;

Time t;

while (stream >> t)//пока нет конца файла выполняем чтение объекта

{

//если прочитан признак конца файла,то выход из цикла

if (stream.eof())break; i++;

//если номер объекта не равен k, то записываем его во вспомогательый файл

if ((t > b) || (t < a)) temp << t <<"\n";

}

//закрыть файлы

stream.close();

temp.close();

remove(f\_name);//удалить старый файл

rename("temp", f\_name);// переименоватьtemp

return i;//количество прочитанных

}

int change\_file(const char\* f\_name, Time key)

{

fstream temp("temp", ios::out);//открыть для записи

fstream stream(f\_name,ios::in);//открыть для чтения

if(!stream)return -1;//ошибка открытия файла

Time t,a;

a=Time(1, 30);

int i=0,

l=0;

char x;

while(stream>>t)

{

if (stream.eof())break;

i++;

if (t == key)

{

cout << t << " - is changing... Continue[y/n]?\n";

cin >> x;

if (x == 'n' || x == 'N')break;

temp << t + a << "\n";

l++;

}

else temp << t << "\n";

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;//количество измененных элементов

}

int add\_file(const char\* f\_name, int k)

{

fstream temp("temp", ios::out);//открыть для записи

fstream stream(f\_name,ios::in);//открыть для чтения

if(!stream)return -1;//ошибка открытия файла

Time t,addt;

int i=0,

l=0;

while (i < k)

{

cout << "Введите значение нового элемента: ";

cin >> addt;

temp << addt << "\n";//записать в temp новую запись

i++;

l++;

}

while(stream>>t)

{

if (stream.eof())break;

temp << t<<"\n";

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;//количество добавленных

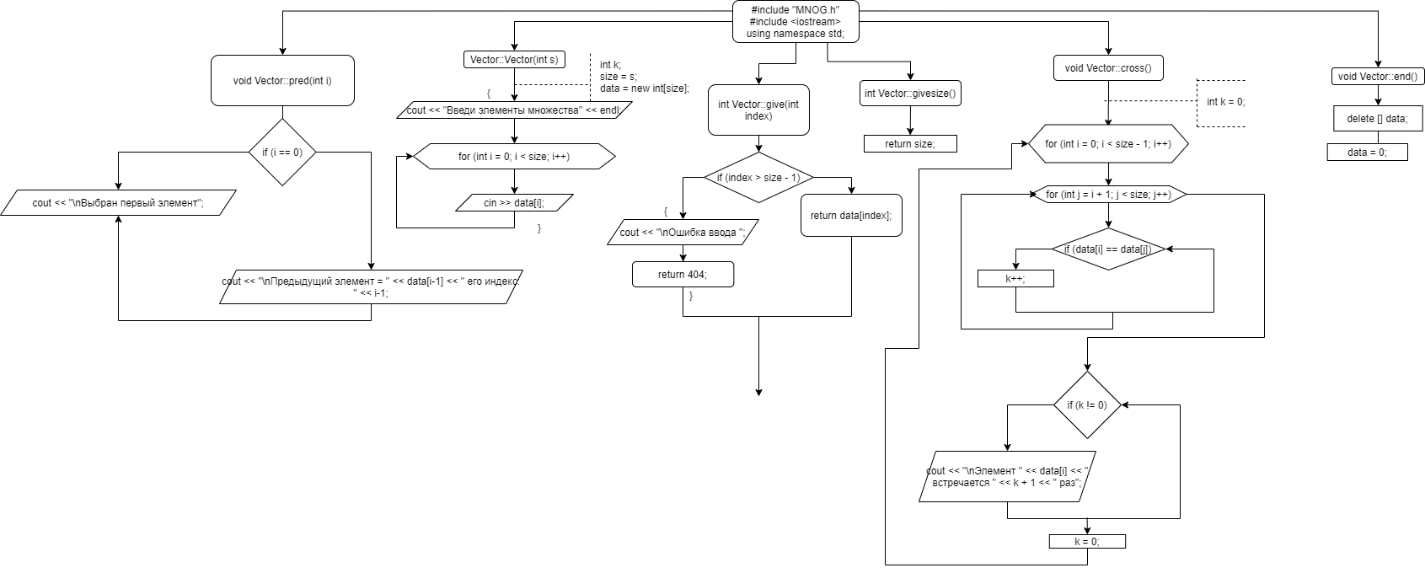
}

Блок-схема:

Main.cpp:



MHETODS.cpp:



MONEY.h:



Работа кода

