Министерство науки и высшего образования РФ

Федеральное бюджетное государственное образовательное учреждение высшего профессионального образования

Пермский национальный исследовательский политехнический университет (ПНИПУ)

Факультет: Электротехнический (ЭТФ)

Направление: 09.03.01 «Информатика и вычислительная техника (Автоматизированные системы обработки информации и управления)»

Кафедра: «Информационные технологии и автоматизированных систем» (ИТАС)

Теория алгоритмов

Лабораторная работа №10

Базы данных

Студент: Балтаев Э. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись)

Группа: АСУ-20-1бзу

Работу проверил:

доцент кафедры ИТАС

Полякова О. А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись:\_\_\_\_\_\_\_\_\_\_\_

Пермь 2021г.

Постановка задачи:

Написать приложение для работы с простой базой данных, хранящей информацию об объекте на внешнем носителе. Приложение должно выполнять следующие функции:

1. Создание базы данных, содержащей записи указанного формата.
2. Просмотр базы данных.
3. Удаление элементов из базы данных (по ключу/ по номеру).
4. Корректировка элементов в базе данных (по ключу / по номеру).
5. Добавление элементов в базу данных (в начало / в конец/ с заданным номером).
6. Выполнение задания, указанного в варианте.

Задание варианта:

1. Типизированный файл содержит данные о ежемесячных доходах подразделений фирмы «Феникс» за пять лет. Программа должна
   1. Добавлять, удалять, корректировать, позволять просматривать записи файла.
   2. Определять:

- для каждого подразделения самые прибыльные годы;

- наиболее длинный период каждого подразделения с доходом ниже среднего по всей фирме

Описание класса

1) Указанная структура

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Finance

{

private:

int id;

string dep;

int year;

int month;

double salary;

bool is\_delete = false;

public:

Finance() { id = 0; dep = ""; year = 0; month = 0; salary = 0; }

Finance(int i, string d, int y, int m, double s);

~Finance() {}

int get\_id() { return id; }

string get\_department() { return dep; }

int get\_year() { return year; }

int get\_month() { return month; }

int get\_salary() { return salary; }

bool get\_isDelete() { return is\_delete; }

void set\_id(int i) { id = i; }

void set\_department(string d) { dep = d; }

void set\_year(int y);

void set\_month(int m);

void set\_salary(double s);

void set\_isDelete(bool id) { is\_delete = id; }

bool operator== (const Finance& fin);

friend ostream& operator<< (ostream& out, const Finance& fin);

friend istream& operator>> (istream& in, Finance& fin);

friend fstream& operator<< (fstream& out, const Finance& fin);

friend fstream& operator>> (fstream& in, Finance& fin);

};

2) Контейнер для хранения структур

#pragma once

#include "Finance.h"

#include <vector>

#include <map>

#include <set>

#include <algorithm>

class Group

{

private:

vector<Finance> fins;

int fin\_AI;

int last\_delete;

public:

Group();

~Group();

// Finance

bool fin\_contains(Finance& fin);

void fin\_select();

void fin\_select\_by\_department();

void fin\_insert();

void fin\_update();

void fin\_delete();

void fin\_cancel\_delete();

void fin\_save();

void fin\_read();

void query1();

void query2();

};

Определение компонентных функций

1) Указанная структура

#include "Finance.h"

Finance::Finance(int i, string d, int y, int m, double s)

{

id = i;

dep = d;

set\_year(y);

set\_month(m);

set\_salary(s);

}

void Finance::set\_year(int y)

{

if (y < 1900 || y > 2021)

throw exception("Не верный год");

year = y;

}

void Finance::set\_month(int m)

{

if(m < 1 || m> 12)

throw exception("Не верный месяц");

month = m;

}

void Finance::set\_salary(double s)

{

if(s < 0)

throw exception("Не верный доход");

salary = s;

}

bool Finance::operator==(const Finance& fin)

{

return dep == fin.dep && year == fin.year && month == fin.month;

}

ostream& operator<<(ostream& out, const Finance& fin)

{

out << fin.id << ": " << fin.dep << " - ";

if (fin.month < 10)

out << "0";

out << fin.month << "." << fin.year << " = " << fin.salary << " у.е.";

return out;

}

istream& operator>>(istream& in, Finance& fin)

{

int temp;

char buf[255];

in.ignore();

cout << "Подраздаление? "; in.getline(buf, 255); fin.dep = string(buf);

cout << "Год? "; in >> temp; fin.set\_year(temp);

cout << "Месяц? "; in >> temp; fin.set\_month(temp);

cout << "Доход? "; in >> temp; fin.set\_salary(temp);

return in;

}

fstream& operator<<(fstream& out, const Finance& fin)

{

out << fin.id << " " << fin.year << " " << fin.month << " " << fin.salary << endl;

out << fin.dep;

return out;

}

fstream& operator>>(fstream& in, Finance& fin)

{

int temp;

char buf[255];

in >> fin.id;

in >> temp; fin.set\_year(temp);

in >> temp; fin.set\_month(temp);

in >> temp; fin.set\_salary(temp);

in.ignore();

in.getline(buf, 255); fin.dep = string(buf);

return in;

}

2) Контейнер для хранения структур

#include "Group.h"

Group::Group() {

fins = vector<Finance>();

fin\_AI = 1;

last\_delete = -1;

}

Group::~Group() {

fins.clear();

}

bool Group::fin\_contains(Finance& fin)

{

// ищем

for (auto it = fins.begin(); it != fins.end(); it++)

if (\*it == fin)

return true;

return false;

}

void Group::fin\_select() {

if (fins.size() == 0)

throw exception("пусто");

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

// если помечено на удаление

if (fins[i].get\_isDelete() == true)

cout << "--] ";

cout << i + 1 << ") " << fins[i] << endl;

}

}

void Group::fin\_select\_by\_department() {

char buf[255];

if (fins.size() == 0)

throw exception("пусто");

cout << "Подразделение? "; cin.ignore(); cin.getline(buf, 255);

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

if (fins[i].get\_department() == string(buf)) {

// если помечено на удаление

if (fins[i].get\_isDelete() == true)

cout << "--] ";

cout << i + 1 << ") " << fins[i] << endl;

}

}

void Group::fin\_insert() {

int x, i = 1;

Finance temp;

auto it = fins.begin();

cout << "\n1. В начало\n2. В конец\n3. С заданным номером\n>"; cin >> x;

switch (x) {

case 1: it = fins.begin(); break; // начало

case 2: it = fins.end(); break; // конец

case 3: // по номеру

cout << "Num? "; cin >> x;

for (it = fins.begin(); it != fins.end(); i++, it++)

if (i == x) {

it = fins.insert(it, temp);

x = 0;

break;

}

// не нашли нужный номер

if (x == 0)

throw exception("неверное значение");

break;

default:

throw exception("неверное значение");

}

cin >> temp;

temp.set\_id(fin\_AI++);

if (fin\_contains(temp) == true)

throw exception("Уже есть значение за данный период");

fins.insert(it, temp);

cout << "Успешно\n";

}

void Group::fin\_update() {

int x, key, i = 1;

auto it = fins.begin();

cout << "\n1. По ключу\n 2. По номеру\n>"; cin >> x;

if (x != 1 && x != 2)

throw exception("Неверный номер");

cout << "key? "; cin >> key;

for (it = fins.begin(); it != fins.end(); i++, it++) {

if (x == 1 && it->get\_id() == key) // ключ

break;

else if (x == 2 && i == key) // номер

break;

}

if(it == fins.end())

throw exception("Нет заданного элемента\n");

cin >> \*it;

cout << "Успешно" << endl;

}

void Group::fin\_delete() {

int x, key, i = 1;

auto it = fins.begin();

cout << "\n1. По ключу\n 2. По номеру\n>"; cin >> x;

if (x != 1 && x != 2)

throw exception("Неверный номер");

cout << "key? "; cin >> key;

// ищем

for (it = fins.begin(); it != fins.end(); i++, it++) {

if (x == 1 && it->get\_id() == key) // ключ

break;

else if (x == 2 && i == key) // номер

break;

}

if (it == fins.end())

throw exception("Нет заданного элемента\n");

it->set\_isDelete(true);

last\_delete = it->get\_id();

cout << "Успешно" << endl;

}

void Group::fin\_cancel\_delete() {

if (last\_delete < 0)

throw exception("Ничего не удалялось");

auto it = fins.begin();

// ищем

for (it = fins.begin(); it != fins.end(); it++)

if (it->get\_id() == last\_delete) {

last\_delete = -1;

break;

}

if (it == fins.end())

throw exception("Ошибка при отмене\n");

it->set\_isDelete(false);

cout << "Успешно" << endl;

}

void Group::fin\_save()

{

string path;

cout << "path? "; cin >> path;

fstream f(path, ios::out);

if (f.is\_open() == false)

throw exception("Файл не был открыт");

f << fin\_AI << endl;

for (auto it = fins.begin(); it != fins.end(); it++)

// помеченные удаляем

if(it->get\_isDelete() == false)

f << \*it << endl;

f.close();

cout << "Успешно" << endl;

}

void Group::fin\_read()

{

string path;

cout << "path? "; cin >> path;

fstream f(path, ios::in);

if (f.is\_open() == false)

throw exception("Файл не был открыт");

fins.clear();

Finance temp;

f >> fin\_AI;

do {

// последнее чтение вызывает исключение в сеттере класса

try { f >> temp; } catch(exception& ex) {}

if (f.eof())

break;

fins.push\_back(temp);

} while (!f.eof());

f.close();

cout << "Успешно" << endl;

}

// самые прибыльные годы

void Group::query1() {

if(fins.size() == 0)

throw exception("пусто");

// собираем статистику

map<string, map<int, vector<double>>> data;

for (auto it = fins.begin(); it != fins.end(); it++) {

// если добавлено подразделение

if (data.find(it->get\_department()) != data.end()) {

if (data[it->get\_department()].find(it->get\_year()) != data[it->get\_department()].end()) {

data[it->get\_department()][it->get\_year()][0] += it->get\_salary();

data[it->get\_department()][it->get\_year()][1] += 1;

}

else {

vector<double> temp;

temp.push\_back(it->get\_salary()); // сумма дохода

temp.push\_back(1); // количество месяцев

data[it->get\_department()].insert(pair<int, vector<double>>(it->get\_year(), temp));

}

}

// если не добавлено

else {

map<int, vector<double>> temp;

data.insert(pair<string, map<int, vector<double>>>(it->get\_department(), temp));

}

}

// анализируем статистику

string dep;

int year = 0;

double salary = 0;

for (auto it = data.begin(); it != data.end(); it++) {

dep = it->first;

cout << dep << endl;

for (auto it2 = it->second.begin(); it2 != it->second.end(); it2++) {

cout << "- " << it2->first << ": " << it2->second[0] << endl;

if (salary < it2->second[0]) {

year = it2->first;

salary = it2->second[0];

}

}

cout << "--- Самый прибыльный год " << year << endl;

}

}

// Наиболее длинные периоды подразделений с доходом меньше среднего

void Group::query2() {

set<string> deps;

double avg = 0;

// считаем сумму и запоминаем уникальные подразделения

for (auto it = fins.begin(); it != fins.end(); it++) {

deps.insert(it->get\_department());

avg += it->get\_salary();

}

avg /= fins.size();

cout << "Средний доход: " << avg << endl;

vector<Finance> temp;

for (auto it = deps.begin(); it != deps.end(); it++) {

// отбираем по данному подразделению

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

if (fins[i].get\_department() == \*it)

temp.push\_back(fins[i]);

// сортируем по датам

sort(temp.begin(), temp.end(), [](Finance& fin1, Finance& fin2) {

return

fin1.get\_year() == fin2.get\_year() && fin1.get\_month() < fin2.get\_month() ||

fin1.get\_year() < fin2.get\_year();

});

int prev = 0, c = 0;

int stY, stM, maxStY, maxStM, maxEndY, maxEndM;

for (auto it2 = temp.begin(); it2 != temp.end(); it2++) {

// если доход меньше среднего

if (it2->get\_salary() < avg) {

// если начало периода

if (c == 0) {

stY = it2->get\_year();

stM = it2->get\_month();

}

c++;

}

else {

// запоминаем максимальный период

if (prev < c) {

prev = c;

maxStY = stY;

maxStM = stM;

maxEndY = it2->get\_year();

maxEndM = it2->get\_month();

}

c = 0;

}

}

cout << \*it << " - наиболее длинный период равен " << prev << " с " << maxStM << "." << maxStY << " по " << maxEndM << "." << maxEndY << endl;

temp.clear();

}

}

Определение главного файла

#include <iostream>

#include <conio.h>

#include "Group.h"

int main()

{

system("chcp 1251");

string x;

Group group;

while (true) {

try {

cout << "Нажмите любую клавишу для продолжения...";

\_getch();

system("cls");

cout << "\nsel. Данные о доходах";

cout << "\nsel1. Данные о доходах подразделения";

cout << "\nins. Добавить";

cout << "\nupd. Изменить";

cout << "\ndel. Удалить";

cout << "\ncan. Удалить последнее изменение";

cout << "\nread. Загрузить из файла";

cout << "\nsave. Сохранить в файл";

cout << "\nqu1. Самые прибыльные годы подразделений";

cout << "\nqu2. Наиболее длинные периоды подразделений с доходом меньше среднего";

cout << "\nesc. Выйти\n>";

cin >> x;

if (x == "sel") group.fin\_select();

else if (x == "sel1") group.fin\_select\_by\_department();

else if (x == "ins") group.fin\_insert();

else if (x == "upd") group.fin\_update();

else if (x == "del") group.fin\_delete();

else if (x == "can") group.fin\_cancel\_delete();

else if (x == "read") group.fin\_read();

else if (x == "save") group.fin\_save();

else if (x == "esc") return 0;

else if (x == "qu1") group.query1();

else if (x == "qu2") group.query2();

else throw exception("неверный номер");

}

catch (exception& ex) {

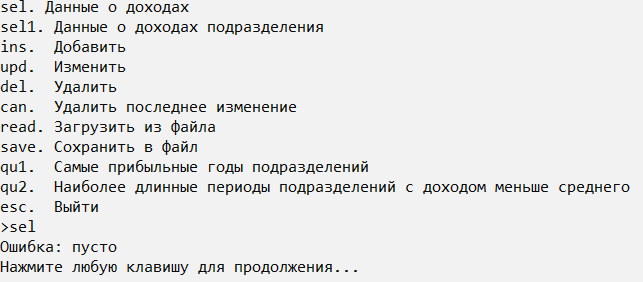
cout << "Ошибка: " << ex.what() << endl;

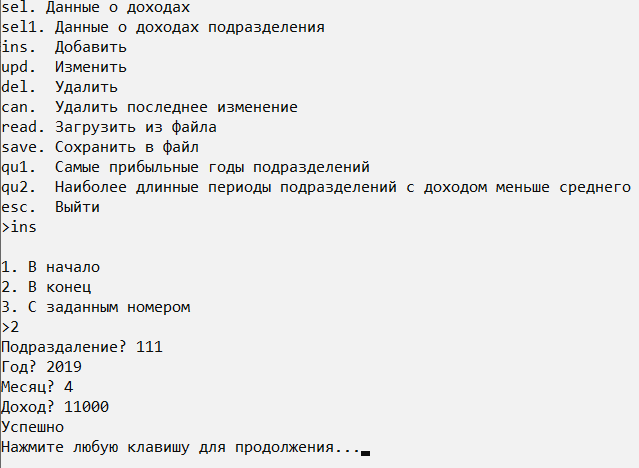
}

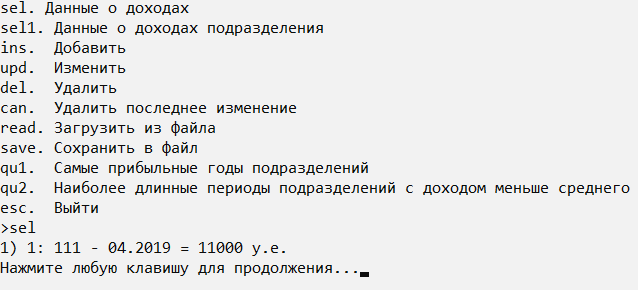
}

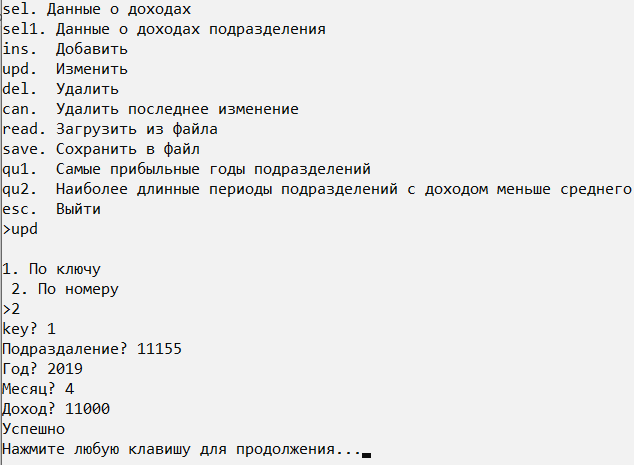
}

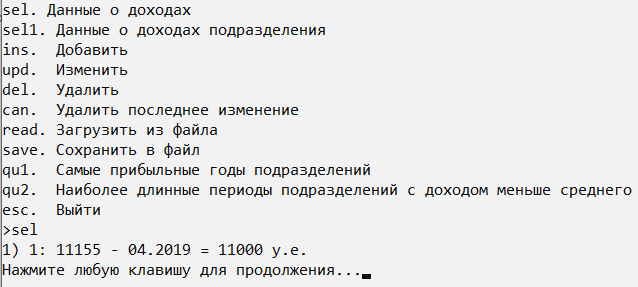
Результаты программы:

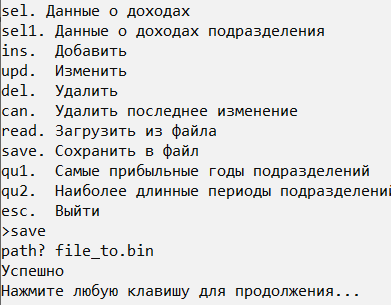
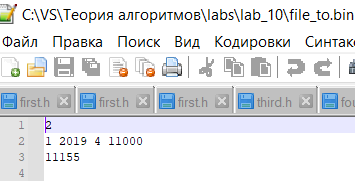


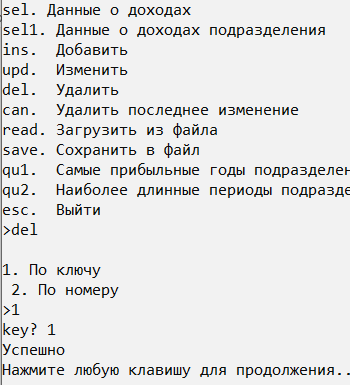
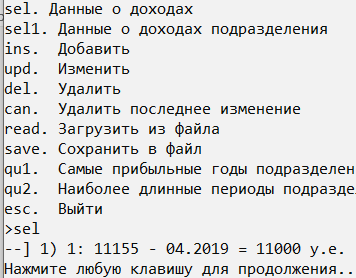


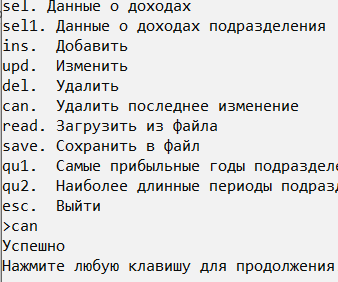
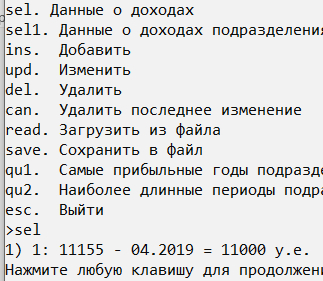


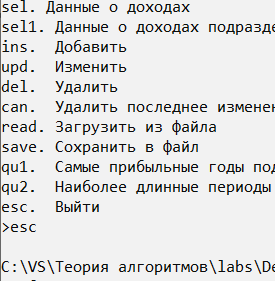
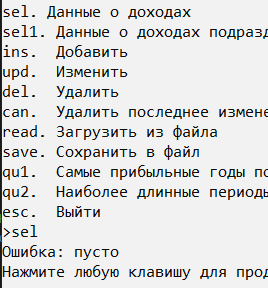


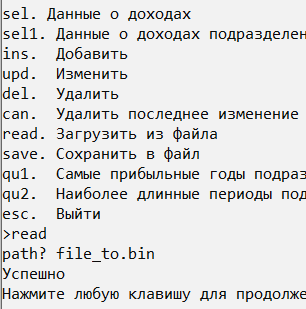
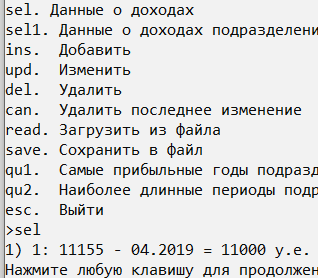


// запросы

