#include "stdafx.h"

#include "AnnualReport.h"

AnnualReport::AnnualReport(TabOfIncome\* pRR, TabOfExpanses\* pER) : ptrPL(pRR), ptrER(pER)

{

}

void AnnualReport::display() //отображение данных на экране

{

cout << " Годовой отчет" << endl;

cout << "--------------------------------------\n";

cout << "--------------------------------------\n";

cout << " Доходы" << endl;

cout << "--------------------------------------\n";

cout << "Выручка:";

income = ptrPL->getSumOfIncomes();

cout << income << endl;

cout << "--------------------------------------\n";

cout << " Расходы" << endl;

cout << "--------------------------------------\n";

cout << " Категория | Стоимость\n";

cout << "-------------------|------------------\n";

expenses = ptrER->getSumOfExpenses();

cout << "--------------------------------------\n";

cout << "Общие расходы:";

cout << expenses << endl;

cout << "--------------------------------------\n";

cout << "--------------------------------------\n";

cout << "Прибыль: " << (income - expenses) << endl;

cout << "--------------------------------------\n";

}

#include "stdafx.h"

#include "Authentication.h"

void Authentication::setUser() //ввести логин и пароль

{

cout << "Введите логин: " << endl;

getaLine(addLogin);

cout << "Введите пароль: " << endl;

getaLine(addPassword);

rules = ptrProfile->access(addLogin, addPassword);

}

int Authentication::getRules()

{

return rules;

}

#include "stdafx.h"

#include "Client.h"

Client::Client(string n, string rn, string s, string d) : name(n), reg\_num(rn), service(s), date(d)

{

}

Client::~Client()

{

}

string Client::getName() //получение имени

{

return name;

}

string Client::getRegNum() //получение вида одежды

{

return reg\_num;

}

string Client::getService() //получение услуги

{

return service;

}

string Client::getDate() //получение даты

{

return date;

}

#include "stdafx.h"

#include "DisplayTabOfClients.h"

void DisplayTabOfClients::setClient() //ввод клиентов

{

cout << "ФИО Клиента: " ;

getaLine(addName);

cout << endl;

cout << "Регистрационый номер автомобиля: ";

getaLine(addRegNum);

cout << endl;

cout << "Оказываемая услуга: ";

getline(cin, addService);

cout << endl;

cout << "Дата обращения (дд.мм.гггг): ";

getaLine(addDate);

cout << endl;

Client\* ptrClient = new Client(addName, addRegNum, addService, addDate);

ptrTabOfClients->insertClient(ptrClient); //добавить клиента в таблицу

}

#include "stdafx.h"

#include "DisplayTabOfExpenses.h"

DisplayTabOfExpenses::DisplayTabOfExpenses(TabOfExpanses\* per) : ptrExpenseList(per)

{

}

void DisplayTabOfExpenses::setExpense() //экран ввода расходов

{

string expenditure, recipient, datePay;

float payment;

cout << "Введите статью: ";

getaLine(expenditure);

cout << "Введите получателя: ";

getaLine(recipient);

cout << "Введите сумму платежа: ";

cin >> payment;

cin.ignore(80, '\n');

cout << "Введите дату платежа: ";

getaLine(datePay);

Expenses\* ptrExpense = new Expenses(expenditure, recipient, payment, datePay);

ptrExpenseList->insertExp(ptrExpense);

}

#include "stdafx.h"

#include "DisplayTabOfIncome.h"

void DisplayTabOfIncome::setIncome()

{

cout << "Введите ФИО клиента: " << endl;

getaLine(addName);

cout << "Введите стоимость услуги: " << endl;

cin >> addPriceService; cin.ignore(80, '\n');

cout << "Введите стоимость материалов: " << endl;

cin >> addPriceMaterials; cin.ignore(80, '\n');

cout << "Введите дату оплаты: " << endl;

getaLine(addDatePrice);

Income\* ptrIncome = new Income(addName, addPriceService, addPriceMaterials, addDatePrice);

ptrTabOfIncome->insertIncome(ptrIncome);

}

#include "stdafx.h"

#include "Income.h"

Income::Income(string n, float pC, float pS, string dP) : name(n), priceService(pC), priceMaterials(pS), datePrice(dP)

{

}

// Методы получения имени, даты, стоимости услуг и материалов

string Income::getName()

{

return name;

}

string Income::getDate()

{

return datePrice;

}

float Income::getPriceMaterials()

{

return priceMaterials;

}

float Income::getPriceService()

{

return priceService;

}

#include "stdafx.h"

#include "Interface.h"

Interface::Interface()

{

ptrTabOfClients = new TabOfClients;

ptrTabOfIncome = new TabOfIncome;

ptrTabOfExpenses = new TabOfExpanses;

}

Interface::~Interface()

{

delete ptrTabOfClients;

delete ptrTabOfIncome;

delete ptrTabOfExpenses;

}

void Interface::interact() //отображение главного меню

{

do

{

ptrAuthentication = new Authentication(ptrUser);

ptrAuthentication->setUser(); //обращение к методу ввода логина и пароля

system("cls");

if (ptrAuthentication->getRules() == 3) cout << "Ошибка авторизацииn"; //неавторизированный пользователь

} while (ptrAuthentication->getRules() == 3);

while (true)

{

cout << "\n";

cout << "| Выберите действие |\n";

cout << "|----------------------------------|\n";

cout << "| Чтобы ввести данные введите i |\n";

if (ptrAuthentication->getRules() == 1) cout << "| Чтобы вывести данные введите d |\n"; //отображать это меню только если это директор

cout << "| Для выхода введите e |\n";

ch = getaChar();

system("cls");

if (ch == 'i') // меню ввода данных

{

cout << "\n";

cout << "| Ввести данные |\n";

cout << "|--------------------------------------|\n";

cout << "| Для ввода нового клиента введите c |\n";

cout << "| Для ввода дохода введите i |\n";

cout << "| Для ввода расхода введите e |\n";

ch = getaChar();

system("cls");

switch (ch)

{

case 'c': ptrDisplayTabOfClients = new DisplayTabOfClients(ptrTabOfClients);

ptrDisplayTabOfClients->setClient();

delete ptrDisplayTabOfClients;

break;

break;

case 'i': ptrDisplayTabOfIncome = new DisplayTabOfIncome(ptrTabOfIncome);

ptrDisplayTabOfIncome->setIncome();

delete ptrDisplayTabOfIncome;

break;

case 'e': ptrDisplayTabOfExpenses = new DisplayTabOfExpenses(ptrTabOfExpenses);

ptrDisplayTabOfExpenses->setExpense();

delete ptrDisplayTabOfExpenses;

break;

default: cout << "Выбран неверный номер\n";

break;

}

}

else if (ch == 'd') //меню вывода данных

{

if (ptrAuthentication->getRules() == 1) // выводить если директор

{

cout << "\n";

cout << "| Вывести данные |\n";

cout << "|-------------------------------|\n";

cout << "| Для вывода таблицы клиентов 1 |\n";

cout << "| Для вывода таблицы доходов 2 |\n";

cout << "| Для вывода таблицы расходов 3 |\n";

cout << "| Для вывода таблицы отчета 4 |\n";

ch = getaChar();

system("cls");

switch (ch)

{

case '1': ptrTabOfClients->display();

break;

case '2':ptrTabOfIncome->display();

break;

case '3': ptrTabOfExpenses->display();

break;

case '4':ptrAnnualReport = new AnnualReport(ptrTabOfIncome, ptrTabOfExpenses);

ptrAnnualReport->display();

delete ptrAnnualReport;

break;

default: cout << "Выбран неверный номер\n";

break;

}

}

}

else if (ch == 'e') //выход из программы

return;

else cout << "Выбран неверный номер\n";

}

delete ptrAuthentication;

}

#include "stdafx.h"

#include "Profile.h"

Profile::Profile(string l, string p) : login(l), password(p)

{

}

Profile::~Profile()

{

}

int Profile::access(string l, string p) //проверка на доступ

{

if ("admin" == l && "admin" == p) return 1;

else if ("user" == l && "user" == p) return 2;

else return 3;

}

#include "stdafx.h"

#include "TabOfClients.h"

TabOfClients::~TabOfClients() //удаление всей памяти о клиенте

{

while (!setPtrsClients.empty())

{

iter = setPtrsClients.begin();

delete \*iter;

setPtrsClients.erase(iter);

}

}

void TabOfClients::insertClient(Client\* ptrT) { //добавление клиента в список

setPtrsClients.push\_back(ptrT);

}

void TabOfClients::display() //отображение таблицы клиентов

{

cout << "\nФИО Клиента |Регистрационный номер |Вид Услуги | Дата обращения \n";

if (setPtrsClients.empty()) //если список пустой

cout << "---Нет клиентов---\n" << endl;

else

{

iter = setPtrsClients.begin();

while (iter != setPtrsClients.end()) //вывести все строки таблицы

{

cout << std::setw(15) << (\*iter)->getName() << " | " << std::setw(15) << (\*iter)->getRegNum() << " | " << std::setw(15) << (\*iter)->getService() << " | " << std::setw(15) << (\*iter)->getDate() << endl;

\*iter++;

}

}

}

#include "stdafx.h"

#include "TabOfExpenses.h"

TabOfExpanses::~TabOfExpanses()

{

while (!vectPtrsExpenses.empty())

{

iter = vectPtrsExpenses.begin();

delete \*iter;

vectPtrsExpenses.erase(iter);

}

}

void TabOfExpanses::insertExp(Expenses\* ptrExp) //добавление в таблицу расходов

{

vectPtrsExpenses.push\_back(ptrExp);

}

void TabOfExpanses::display() //отображение на экране данных о расходах

{

cout << "\nСтаться расходов |Получатель |Сумма платежа |Дата платежа\n";

if (vectPtrsExpenses.size() == 0)

cout << "Расходов не найдено\n" << endl;

else

{

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{

cout << std::setw(15) << (\*iter)->expenditure << " | " << std::setw(15) << (\*iter)->recipient << " | " << std::setw(15) << (\*iter)->payment << " | " << std::setw(15) << (\*iter)->datePay << endl;

iter++;

}

cout << endl;

}

}

float TabOfExpanses::getSumOfExpenses() //общая сумма расходов

{

float totalExpenses = 0;

if (vectPtrsExpenses.size() == 0)

{

cout << "Расходов нет\n";

return 0;

}

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{

cout << std::setw(15) << ((\*iter)->expenditure) << " | " << std::setw(15) << ((\*iter)->payment) << endl;

totalExpenses += (\*iter)->payment;

iter++;

}

return totalExpenses;

}

#include "stdafx.h"

#include "TabOfIncome.h"

using namespace std;

TabOfIncome::~TabOfIncome()

{

while (!setPtrsRR.empty())

{

iter = setPtrsRR.begin();

delete \*iter;

setPtrsRR.erase(iter);

}

}

void TabOfIncome::insertIncome(Income\* ptrRow) {

setPtrsRR.push\_back(ptrRow);

}

void TabOfIncome::display() //отоброжение на экране доходов

{

cout << "\nФИО Клиента |Стоимость работы |Стоимость материалов |Сумма |Дата оплаты\n";

if (setPtrsRR.empty())

cout << "---Нет доходов---\n" << endl;

else

{

iter = setPtrsRR.begin();

while (iter != setPtrsRR.end())

{

cout << std::setw(15) << (\*iter)->getName() << " | " << std::setw(15) << (\*iter)->getPriceService() << " | " << std::setw(15) << (\*iter)->getPriceMaterials() << " | " << std::setw(15) << (\*iter)->getPriceMaterials() + (\*iter)->getPriceService() << " | " << std::setw(15) << (\*iter)->getDate() << endl;

\*iter++;

}

}

}

float TabOfIncome::getSumOfIncomes() //общая сумма доходов

{

float sumProfits = 0.0;

iter = setPtrsRR.begin();

while (iter != setPtrsRR.end())

{

sumProfits += (\*iter)->getPriceMaterials() + (\*iter)->getPriceService();

iter++;

}

return sumProfits;

}

// TeamTask.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "Interface.h"

int \_tmain(int argc, \_TCHAR\* argv[])

{

system("chcp 1251 > nul");

Interface theUserInterface;

theUserInterface.interact();

return 0;

}