Приложение Б

**AnnualReport.cpp**

#include "stdafx.h"

#include "AnnualReport.h"

#include <iostream>

AnnualReport::AnnualReport(IncomeRecord\* pIN, ExpenseRecord\* pER) : ptrIN(pIN), ptrER(pER)

{

}

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

{

cout << " Годовой отчет" << endl;

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

cout << " Доходы" << endl;

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

cout << "Выручка:";

prof = ptrIN->getSumOfIncome();

cout << prof << endl;

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

cout << "Расходы" << endl;

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

cout << " Категория | Стоимость\n";

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

expenses = ptrER->displaySummary();

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

cout << "Общие расходы:";

cout << expenses << endl;

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

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

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

}

**ExpenseInputScreen.cpp**

//// РЕАЛИЗАЦИЯ ЭКРАНА ВВОДА РАССХОДОВ ////////////////

#include "stdafx.h"

#include "ExpenseInputScreen.h"

#include <iostream>

using namespace std;

ExpenseInputScreen::ExpenseInputScreen(ExpenseRecord\* per) : ptrExpenseList(per)

{

}

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

{

string category, org, date;

float amount;

cout << "Введите категорию расходов (Цветы, водитель): ";

cin >> category;

cout << "Введите организацию (ИП Автоперевозки, ОАО Лютики): ";

cin >> org;

cout << "Введите дату платежа (20.12.2020): ";

cin >> date;

cout << "Введите сумму в рублях: ";

cin >> amount;

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

Expense\* ptrExpense = new Expense(category, org, date, amount);

ptrExpenseList->insertExp(ptrExpense); // добавление новой записи в таблицу расходов

}

**ExpenseRecord.cpp**

////// РЕАЛИЗАЦИЯ КЛАССА ТАБЛИЦА РАСХОДОВ/////

#include "stdafx.h"

#include "ExpenseRecord.h"

#include <iostream>

#include <iomanip>

using namespace std;

ExpenseRecord::~ExpenseRecord() // деструктор

{

while (!vectPtrsExpenses.empty()) // очистка вектора

{

iter = vectPtrsExpenses.begin();

delete \*iter;

vectPtrsExpenses.erase(iter);

}

}

void ExpenseRecord::insertExp(Expense\* ptrExp) //добавление в таблицу расходов

{

vectPtrsExpenses.push\_back(ptrExp);

}

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

{

system("cls");

cout << "\nКатегория |Получатель |Дата |Сумма\n";

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

if (vectPtrsExpenses.size() == 0) // если размер вектора 0, то он пустой значит расходов нет

cout << "---Расходов нет---\n" << endl;

else

{

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end()) // пока не конец вектора выводим на экран расходы

{

cout << setw(15) << (\*iter)->category << " | " << std::setw(15) << (\*iter)->org << " | " << std::setw(15) << (\*iter)->datePay << " | " << std::setw(15) << (\*iter)->sum << endl;

iter++;

}

cout << endl;

}

}

float ExpenseRecord::displaySummary() //общая сумма доходов для

{

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)->category) << " | " << std::setw(15) << ((\*iter)->sum) << endl;

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

iter++;

}

return totalExpenses;

}

**Income.cpp**

/////// РЕАЛИЗАЦИЯ КЛАССА ДОХОДЫ /////

#include "stdafx.h"

#include "Income.h"

Income::Income(string i, float t, float s, string p) : id(i), priceTov(t), priceService(s), datePrice(p)

{

}

// получаем ID продавца, название товара, цену и дату продажи

string Income::getID()

{

return id;

}

string Income::getDate()

{

return datePrice;

}

float Income::getPriceTov()

{

return priceTov;

}

float Income::getPriceService()

{

return priceService;

}

**IncomeInputScreen.cpp**

//// РЕАЛИЗАЦИЯ ЭКРАНА ВВОДА ДОХОДОВ ////////////////

#include "stdafx.h"

#include "IncomeInputScreen.h"

#include <iostream>

//Ввод дохода в таблицу

void IncomeInputScreen::setIncome()

{

system("cls");

cout << "Введите ID продавца(000): " << endl;

cin >> ID;

cout << "Введите стоимость товара (1000): " << endl;

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

cout << "Введите стоимость услуги (0): " << endl;

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

cout << "Введите дату продажи (21.12.2020): " << endl;

cin >> DatePrice;

Income\* ptrIncome = new Income(ID, PriceTov, PriceService, DatePrice);

ptrIncomeR->insertIncome(ptrIncome);

}

**IncomeRecord.cpp**

////// РЕАЛИЗАЦИЯ КЛАССА ТАБЛИЦА ДОХОДОВ/////

#include "stdafx.h"

#include "IncomeRecord.h"

#include <iostream>

#include <iomanip>

using namespace std;

IncomeRecord::~IncomeRecord()

{

while (!setPtrsIN.empty())

{

iter = setPtrsIN.begin();

delete \*iter;

setPtrsIN.erase(iter);

}

}

void IncomeRecord::insertIncome(Income\* ptrR) {

setPtrsIN.push\_back(ptrR);

}

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

{

system("cls");

cout << "\nID продавца |Стоимость товара |Стоимость услуги |Общая стоимость |Дата продажи\n";

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

if (setPtrsIN.empty())

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

else

{

iter = setPtrsIN.begin();

while (iter != setPtrsIN.end())

{

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

\*iter++;

}

}

}

float IncomeRecord::getSumOfIncome() //общая сумма доходов для годового отчета

{

float sumIncome = 0.0;

iter = setPtrsIN.begin();

while (iter != setPtrsIN.end())

{

sumIncome += (\*iter)->getPriceTov() + (\*iter)->getPriceService();

iter++;

}

return sumIncome;

}

**Interface.cpp**

#include "stdafx.h"

#include "Interface.h"

#include <iostream>

Interface::Interface() //конструктор

{

ptrProdavciList = new ProdavciList; // возможно исправить лист на рекорд /// книга учета продавцов

ptrIncomeRecord = new IncomeRecord; // записи доходов

ptrExpenseRecord = new ExpenseRecord; // записи расходов

}

Interface::~Interface() // деструктор

{

delete ptrProdavciList;

delete ptrIncomeRecord;

delete ptrExpenseRecord;

}

void Interface::menu() //отображение меню

{

do // если пользователь ошибся вводом

{

ptrTheFirstScreen = new TheFirstScreen(ptrProdavci);

ptrTheFirstScreen->inicial(); //обращение к методу ввода логина и пароля

system("cls");

if (ptrTheFirstScreen->getprava() == 3) cout << "Проверьте идентификатор или пароль\n"; //неавторизированный пользователь

} while (ptrTheFirstScreen->getprava() == 3);

h = -1;

while (true)

{

cout << "\n";

cout << "/МЕНЮ/\n";

cout << "/////////////////////////// \n";

cout << "/1 - Ввести данные / \n";

if (ptrTheFirstScreen->getprava() == 1) cout << "/2 - Вывести данные/ \n"; //отображать это меню только если это директор

cout << "/0 - Выход / \n";

cin >> pr;

if (pr == '1'){

system("cls");

cout << "\n";

cout << "/ МЕНЮ - ВВЕСТИ ДАННЫЕ/ \n";

cout << "///////////////////////\n";

cout << "/1 - Оплата продажи /\n";

cout << "/2 - Провести расходы /\n";

if (ptrTheFirstScreen->getprava() == 1) cout << "/0 - Внести данные об окладе сотрудников /\n";

cin >> h;

switch (h)

{

case '1':

system("cls");

ptrIncomeInputScreen = new IncomeInputScreen(ptrIncomeRecord);

ptrIncomeInputScreen->setIncome();

delete ptrIncomeInputScreen;

break;

case '2': system("cls");

ptrExpenseInputScreen = new ExpenseInputScreen(ptrExpenseRecord);

ptrExpenseInputScreen->setExpense();

delete ptrExpenseInputScreen;

break;

case '0': ptrProdaza = new Prodaza(ptrProdavciList);

ptrProdaza->prodazaProd();

delete ptrProdaza;

break;

default: cout << "Ошибка ввода\n";

break;

}

}

else if (pr == '2') //меню вывода данных

{

if (ptrTheFirstScreen->getprava() == 1) // выводить если директор

{

system("cls");

cout << "\n";

cout << "/ МЕНЮ - ВЫВЕСТИ ДАННЫЕ /\n";

cout << "//////////////////////////////\n";

cout << "/1 - Таблица сотрудников /\n";

cout << "/2 - Таблица доходов /\n";

cout << "/3 - Таблица расходов /\n";

cout << "/4 - Годовой отчет /\n";

cin >> pr;

system("cls");

switch (pr)

{

case '1': system("cls");

ptrProdavciList->display();

break;

case '2':system("cls");

ptrIncomeRecord->display();

break;

case '3': system("cls");

ptrExpenseRecord->display();

break;

case '4':system("cls");

ptrAnnualReport = new AnnualReport(ptrIncomeRecord, ptrExpenseRecord);

ptrAnnualReport->display();

delete ptrAnnualReport;

break;

default: cout << "Ошибка ввода\n";

break;

}

}

}

else if (pr == 3){

ptrTheFirstScreen = new TheFirstScreen(ptrProdavci);

ptrTheFirstScreen->inicial();

}

else if (pr == '0') //выход из программы

return;

else cout << "Ошибка ввода\n";

}

delete ptrProdaza;

}

**KomZad.cpp**

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

//

/////////////////////// ГЛАВНАЯ ФУНКЦИЯ //////////////////////

#include "stdafx.h"

#include <iostream>

#include "Interface.h" // Главный класс подключаем

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

{

setlocale(LC\_ALL, "Russian");

Interface theUserInterface; // создаем объект

theUserInterface.menu();

return 0;

}

**Prodavci.cpp**

///////////////////////////// ФАЙЛ РЕАЛИЗАЦИИИ Prodavci.ch (инициализация пользователя) ////////////////

#include "stdafx.h"

#include "Prodavci.h"

Prodavci::Prodavci(char u, string p) : username(u), password(p)

{

}

Prodavci::~Prodavci() //деструктор

{

}

int Prodavci::avtorizaciya(char u, string p) //получение доступа

{

if ('a' == u && "0000" == p) return 1; // если входит админ то запросить пароль

else if ('p' == u && "" == p) return 2; // если входит продавец, то пароль не требуется

else return 3;

}

**ProdavciList.cpp**

///// РЕАЛИЗАЦИЯ СПИСКА СОТРУДНИКОВ НА СМЕНЕ /////

#include "stdafx.h"

#include "ProdavciList.h"

#include <iostream>

#include <iomanip>

ProdavciList::~ProdavciList() //удаление всей памяти о сотрудниках

{

while (!setPtrsSotr.empty())

{

iter = setPtrsSotr.begin();

delete \*iter;

setPtrsSotr.erase(iter);

}

}

void ProdavciList::insertSotr(Sotrydniki\* ptrP) { //добавление id

setPtrsSotr.push\_back(ptrP);

}

void ProdavciList::display() //отображение таблицы продажи товаров

{

system("cls");

cout << "\nID продавца |Фио продавца |Должность | Дата покупки \n";

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

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

cout << "---Не было продаж---\n" << endl;

else

{

iter = setPtrsSotr.begin();

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

{

cout << setw(15) << (\*iter)->getNumber() << " | " << std::setw(15) << (\*iter)->getFio() << " | " << std::setw(15) << (\*iter)->getKat() << " | " << std::setw(15) << (\*iter)->getDate() << endl;

\*iter++;

}

}

}

**Prodaza.cpp**

#include "stdafx.h"

#include "Prodaza.h"

#include <iostream>

using namespace std;

void Prodaza::prodazaProd()

{

system("cls");

cout << "Введите id сотрудника: " << endl;

cin >> Number;

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

cin >> fio;

cout << "Введите должность: " << endl;

cin >> dol;

cout << "Введите зарплату сотрудника: " << endl;

cin >> zp;

Sotrydniki\* ptrS = new Sotrydniki(Number, fio, dol, zp);

ptrProdavciList->insertSotr(ptrS); //добавить запись в книгу учета продаж

}

**Sotrydniki.cpp**

//// РЕАЛИЗАЦИЯ КЛАСС ПРОДАВЦОВ ////

#include "stdafx.h"

#include "Sotrydniki.h"

Sotrydniki::Sotrydniki(string n, string f, string k, string d) : number(n), fio(f), katprod(k), date(d)

{

}

Sotrydniki::~Sotrydniki()

{

}

string Sotrydniki::getNumber() //получение id продавца

{

return number;

}

string Sotrydniki::getFio() //получение фио продавца

{

return fio;

}

string Sotrydniki::getKat() //получение категории продавца

{

return katprod;

}

string Sotrydniki::getDate() //получение даты смены

{

return date;

}

**TheFirstScreen.cpp**

///////////////// ФАЙЛ РЕАЛИЗАЦИИ TheFirstScreen.h (ЭКРАН АВТОРИЗАЦИИ) ////////////////

#include "stdafx.h"

#include "TheFirstScreen.h"

#include <iostream>

void TheFirstScreen::inicial()

{

system("cls");

cout << "Выберите режим работы администратор/продавец (a/p): " << endl;

cin >> username1;

if (username1 == 'a'){ // только администратору нужен пароль

cout << "Введите пароль: " << endl;

cin >> password1;

}

else password1 = "";

prava = ptrname->avtorizaciya(username1, password1);

}

int TheFirstScreen::getprava()

{

return prava;

}