**ПРИЛОЖЕНИЕ №2**

**14) Написание исходных файлов**

AnnualReport.cpp

#include "stdafx.h"

#include "AnnualReport.h"

AnnualReport::AnnualReport(ProfitList\* pRR, ExpenseList\* 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:";

prof = ptrPL->getSumOfProfits();

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

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

}

Client.cpp

#include "stdafx.h"

#include "Client.h"

Client::Client(string n, string c, string s, string d) : name(n), clother(c), service(s), date(d)

{

}

Client::~Client()

{

}

string Client::getName() //получение имени

{

return name;

}

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

{

return clother;

}

string Client::getService() //получение услуги

{

return service;

}

string Client::getDate() //получение даты

{

return date;

}

ClientInputScreen.cpp

#include "stdafx.h"

#include "ClientInputScreen.h"

void ClientInputScreen::setClient() //ввод клиентов

{

cout << "Введите имя Клиента (Иванов И.И): " << endl;

getaLine(addName);

cout << "Введите вид одежды (Футболка, Куртка): " << endl;

getaLine(addClother);

cout << "Введите вид услуги (в случае отсутствия, оставить поле пустым): " << endl;

getline(cin, addService);

cout << "Введите дату обращения (01.10.2020): " << endl;

getaLine(addDate);

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

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

}

ClientList.cpp

#include "stdafx.h"

#include "ClientList.h"

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

{

while (!setPtrsClients.empty())

{

iter = setPtrsClients.begin();

delete \*iter;

setPtrsClients.erase(iter);

}

}

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

setPtrsClients.push\_back(ptrT);

}

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

{

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

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

\*iter++;

}

}

}

ExpenseInputScreen.cpp

#include "stdafx.h"

#include "ExpenseInputScreen.h"

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

{

}

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

{

string category, payee, date;

float amount;

cout << "Введите категорию расходов (Водитель, Материалы): ";

getaLine(category);

cout << "Введите организацию (ИП Иванов, ОАО Грузоперевозки): ";

getaLine(payee);

cout << "Введите дату платежа (01.10.2020): ";

getaLine(date);

cout << "Введите сумму в рублях(500): ";

cin >> amount;

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

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

ptrExpenseList->insertExp(ptrExpense);

}

ExpenseList.cpp

#include "stdafx.h"

#include "ExpenseList.h"

ExpenseList::~ExpenseList()

{

while (!vectPtrsExpenses.empty())

{

iter = vectPtrsExpenses.begin();

delete \*iter;

vectPtrsExpenses.erase(iter);

}

}

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

{

vectPtrsExpenses.push\_back(ptrExp);

}

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

{

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

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

if (vectPtrsExpenses.size() == 0)

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

else

{

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{

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

iter++;

}

cout << endl;

}

}

float ExpenseList::getSumOfExpenses() //общая сумма доходов

{

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)->amount) << endl;

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

iter++;

}

return totalExpenses;

}

Gets.cpp

#include <iostream>

#include "stdafx.h"

void getaLine(string& inStr) //функция ввода строки

{

char temp[21];

cin.get(temp, 20, '\n');

cin.ignore(20, '\n');

inStr = temp;

}

char getaChar() //функция ввода символа

{

char ch = cin.get();

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

return ch;

}

Profit.cpp

#include "stdafx.h"

#include "Profit.h"

Profit::Profit(string n, float pC, float pS, string dP) : name(n), priceCloth(pC), priceService(pS), datePrice(dP)

{

}

// Методы получения имени, даты, стоимости одежды и услуги

string Profit::getName()

{

return name;

}

string Profit::getDate()

{

return datePrice;

}

float Profit::getPriceCloth()

{

return priceCloth;

}

float Profit::getPriceService()

{

return priceService;

}

ProfitInputScreen.cpp

#include "stdafx.h"

#include "ProfitInputScreen.h"

//Ввод дохода в таблицу

void ProfitInputScreen::setProfit()

{

cout << "Введите имя Клиента (Иванов И.И): " << endl;

getaLine(addName);

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

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

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

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

cout << "Введите дату оплаты (03.10.2020): " << endl;

getaLine(addDatePrice);

Profit\* ptrProfit = new Profit(addName, addPriceCloth, addPriceService, addDatePrice);

ptrProfitList->insertProfit(ptrProfit);

}

ProfitList.cpp

#include "stdafx.h"

#include "ProfitList.h"

using namespace std;

ProfitList::~ProfitList()

{

while (!setPtrsRR.empty())

{

iter = setPtrsRR.begin();

delete \*iter;

setPtrsRR.erase(iter);

}

}

void ProfitList::insertProfit(Profit\* ptrRow) {

setPtrsRR.push\_back(ptrRow);

}

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

{

cout << "\nФИО Клиента |Стоимость стирки |Стоимость услуги |Общая стоимость |Дата оплаты\n";

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

\*iter++;

}

}

}

float ProfitList::getSumOfProfits() //общая сумма доходов

{

float sumProfits = 0.0;

iter = setPtrsRR.begin();

while (iter != setPtrsRR.end())

{

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

iter++;

}

return sumProfits;

}

User.cpp

#include "stdafx.h"

#include "User.h"

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

{

}

User::~User()

{

}

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

{

if ("admin" == l && "1234" == p) return 1;

else if ("user" == l && "4321" == p) return 2;

else return 3;

}

UserInputScreen.cpp

#include "stdafx.h"

#include "UserInputScreen.h"

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

{

cout << "Введите логин: " << endl;

getaLine(addLogin);

cout << "Введите пароль: " << endl;

getaLine(addPassword);

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

}

int UserInputScreen::getRules()

{

return rules;

}

UserInterface.cpp

#include "stdafx.h"

#include "UserInterface.h"

UserInterface::UserInterface()

{

ptrClientList = new ClientList;

ptrProfitList = new ProfitList;

ptrExpenseList = new ExpenseList;

}

UserInterface::~UserInterface()

{

delete ptrClientList;

delete ptrProfitList;

delete ptrExpenseList;

}

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

{

do

{

ptrUserInputScreen = new UserInputScreen(ptrUser);

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

system("cls");

if (ptrUserInputScreen->getRules() == 3) cout << "Пароль или логин неверный, введите еще раз\n"; //неавторизированный пользователь

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

while (true)

{

cout << "\n";

cout << "| Меню |\n";

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

cout << "| 1 | Ввести данные |\n";

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

cout << "| q | Выход |\n";

ch = getaChar();

system("cls");

if (ch == '1') // меню ввода данных

{

cout << "\n";

cout << "| Ввести данные |\n";

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

cout << "| 1 | Новый клиент |\n";

cout << "| 2 | Оплата клиента |\n";

cout << "| 3 | Расходы |\n";

ch = getaChar();

system("cls");

switch (ch)

{

case '1': ptrClientInputScreen = new ClientInputScreen(ptrClientList);

ptrClientInputScreen->setClient();

delete ptrClientInputScreen;

break;

break;

case '2': ptrProfitInputScreen = new ProfitInputScreen(ptrProfitList);

ptrProfitInputScreen->setProfit();

delete ptrProfitInputScreen;

break;

case '3': ptrExpenseInputScreen = new ExpenseInputScreen(ptrExpenseList);

ptrExpenseInputScreen->setExpense();

delete ptrExpenseInputScreen;

break;

default: cout << "Ошибка ввода\n";

break;

}

}

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

{

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

{

cout << "\n";

cout << "| Вывести на экран |\n";

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

cout << "| 1 | Таблица клиентов |\n";

cout << "| 2 | Таблица доходов |\n";

cout << "| 3 | Таблица расходов |\n";

cout << "| 4 | Годовой отчет |\n";

ch = getaChar();

system("cls");

switch (ch)

{

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

break;

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

break;

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

break;

case '4':ptrAnnualReport = new AnnualReport(ptrProfitList, ptrExpenseList);

ptrAnnualReport->display();

delete ptrAnnualReport;

break;

default: cout << "Ошибка ввода\n";

break;

}

}

}

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

return;

else cout << "Ошибка ввода\n";

}

delete ptrClientInputScreen;

}

TeamWork.cpp

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

//

#include "stdafx.h"

#include "UserInterface.h"

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

system("chcp 1251 > nul");

UserInterface theUserInterface;

theUserInterface.interact();

return 0;

}