**Исходный файл AnnualReport.cpp**

#include "stdafx.h"

#include "AnnualReport.h"

AnnualReport::AnnualReport(IncomeList\* pInc, ExpenseList\* pER) : ptrInc(pInc), ptrER(pER)

{

}

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

{

cout << "Годовой отчет|\n\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Доходы |\n\_\_\_\_\_\_\_\_\_\_\_\_\_|\n" << endl;

cout << "Заработок:";

inc = ptrInc->getSumOfIncome();

cout << inc << " Р." << endl;

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "Расходы |" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "Категория, Стоимость|\n";

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n" << endl;

expenses = ptrER->getAllExpenses();

cout << "\n";

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "Общие расходы:";

cout << expenses << " Р." << endl;

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "Итоговая прибыль|" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n" <<endl;

cout << "Прибыль:" << (inc - expenses) << " Р." << endl;

}

**Исходный файл Client.cpp**

#include "stdafx.h"

#include "Client.h"

Client::Client(string n, string f, string s, string d) : name(n), flower(f), service(s), date(d)

{

}

Client::~Client()

{

}

string Client::getName() //получение имени

{

return name;

}

string Client::getFlower() //получение названия цветка

{

return flower;

}

string Client::getService() //получение услуги

{

return service;

}

string Client::getDate() //получение даты

{

return date;

}

**Исходный файл ClientInputScreen.cpp**

#include "stdafx.h"

#include "ClientInputScreen.h"

void ClientInputScreen::setClient() //ввод клиентов

{

cout << "Введите имя Покупателя (Андрианов А.Г.): " << endl;

getaLine(addName);

cout << "Введите название цветка (Роза, Хризантема): " << endl;

getaLine(addFlower);

cout << "Введите вид услуги (Упаковать, Украсить): " << endl;

getline(cin, addService);

cout << "Введите дату обращения (24.11.2020): " << endl;

getaLine(addDate);

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

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

}

**Исходный файл ClientList.cpp**

#include "stdafx.h"

#include "ClientList.h"

ClientList::~ClientList() //деконструктор удаляющий данные о клиенте

{

while (!setPtrsClients.empty()) // удаление всех жильцов

{ // удаление указателей из контейнера

iter = setPtrsClients.begin();

delete \*iter;

setPtrsClients.erase(iter);

}

}

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

setPtrsClients.push\_back(ptrT);

}

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

{

cout << "\nПокупатель, Наименование цветка, Название услуги, Дата:\n";

cout << "\n";

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

cout << "Записей по клиентам не обнаружено\n" << endl;

else

{

iter = setPtrsClients.begin();

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

{

cout << (\*iter)->getName() << ", " << (\*iter)->getFlower() << ", " << (\*iter)->getService() << ", " << (\*iter)->getDate() << endl;

\*iter++;

}

}

}

**Исходный файл ExpenseInputScreen.cpp**

#include "stdafx.h"

#include "ExpenseInputScreen.h"

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

{

}

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

{

string category, payee, dateOfPayment;

float amount;

cout << "Введите категорию расходов (Расчёт по счетам, Закупка товара): ";

getaLine(category);

cout << "Введите поставщика услуг (ТЦ Пионер, ЖКХ): ";

getaLine(payee);

cout << "Введите дату платежа (24.11.2020): ";

getaLine(dateOfPayment);

cout << "Введите сумму в рублях(5555): ";

cin >> amount;

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

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

ptrExpenseList->insertExp(ptrExpense);

}

**Исходный файл 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 список 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 << "Записей по расходам не обнаружено" << endl;

else

{

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{

cout << (\*iter)->category << ", " << (\*iter)->payee << ", " << (\*iter)->dateOfPayment << ", " << (\*iter)->amount << " Р." << endl;

iter++;

}

cout << endl;

}

}

float ExpenseList::getAllExpenses() //подсчёт всех расходов

{

float AllExpenses = 0;

if (vectPtrsExpenses.size() == 0)

{

cout << "Записей по расходам не обнаружено";

return 0;

}

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{

cout << ((\*iter)->category) << ", " << ((\*iter)->amount) << endl;

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

iter++;

}

return AllExpenses;

}

**Исходный файл Getf.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;

}

**Исходный файл Income.cpp**

#include "stdafx.h"

#include "Income.h"

Income::Income(string f, float cOF, float cOS, string dOP) : flower(f), costOfFlower(cOF), costOfService(cOS), dateOfPayment(dOP)

{

}

string Income::getFlower() // Метод получения названия цветка

{

return flower;

}

string Income::getDateOfPayment() // Метод получения даты оплаты

{

return dateOfPayment;

}

float Income::getCostOfFlower() // Метод получения стоимости цветка

{

return costOfFlower;

}

float Income::getCostOfService() // Метод получения стоимости услуги

{

return costOfService;

}

**Исходный файл IncomeInputScreen.cpp**

#include "stdafx.h"

#include "IncomeInputScreen.h"

void IncomeInputScreen::inputIncome() //Добавление записи о доходе в список

{

cout << "Введите название цветка (Роза, Хризантема): " << endl;

getaLine(inputFlower);

cout << "Введите цену цветка: " << endl;

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

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

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

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

getaLine(inputDateOfPayment);

Income\* ptrIncome = new Income(inputFlower, inputCostOfFlower, inputCostOfService, inputDateOfPayment);

ptrIncomeList->insertIncome(ptrIncome);

}

**Исходный файл IncomeList.cpp**

#include "stdafx.h"

#include "IncomeList.h"

using namespace std;

IncomeList::~IncomeList()

{

while (!setPtrsIncome.empty())

{

iter = setPtrsIncome.begin();

delete \*iter;

setPtrsIncome.erase(iter);

}

}

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

setPtrsIncome.push\_back(ptrRow);

}

void IncomeList::display() //вывод заработка на экран

{

cout << "\nНазвание цветка, Стоимость цветка, Стоимость услуги, Общая стоимость, Дата:\n";

cout << "\n ";

if (setPtrsIncome.empty())

cout << "Записей по заработку не обнаружено\n" << endl;

else

{

iter = setPtrsIncome.begin();

while (iter != setPtrsIncome.end())

{

cout << (\*iter)->getFlower() << ", " << (\*iter)->getCostOfFlower() << " Р., " << (\*iter)->getCostOfService() << " Р., " << (\*iter)->getCostOfFlower() + (\*iter)->getCostOfService() << " Р., " << (\*iter)->getDateOfPayment() << endl;

\*iter++;

}

}

}

float IncomeList::getSumOfIncome() //общая сумма заработка

{

float SumOfIncome = 0.0;

iter = setPtrsIncome.begin();

while (iter != setPtrsIncome.end())

{

SumOfIncome += (\*iter)->getCostOfFlower() + (\*iter)->getCostOfService();

iter++;

}

return SumOfIncome;

}

**Исходный файл stdafx.cpp**

// stdafx.cpp : source file that includes just the standard includes

// work.pch will be the pre-compiled header

// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H

// and not in this file

**Исходный файл UserInterface.cpp**

#include "stdafx.h"

#include "UserInterface.h"

using namespace std;

UserInterface::UserInterface()

{

ptrClientList = new ClientList;

ptrIncomeList = new IncomeList;

ptrExpenseList = new ExpenseList;

}

UserInterface::~UserInterface()

{

delete ptrClientList;

delete ptrIncomeList;

delete ptrExpenseList;

}

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

{

while (true)

{

cout << "\n";

cout << " Введите:\n";

cout << " i - для ввода данных\n";

cout << " s - для вывода данных\n";

cout << " o - для выхода из программы\n";

ch = getaChar();

system("cls");

if (ch == 'i') // меню ввода данных

{

cout << "\n";

cout << "| Ввод данных |\n";

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

cout << " 1 - Ввод нового клиента \n";

cout << " 2 - Ввод оплаты \n";

cout << " 3 - Ввод расходов \n";

ch = getaChar();

system("cls");

switch (ch)

{

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

ptrClientInputScreen->setClient();

delete ptrClientInputScreen;

break;

break;

case '2': ptrIncomeInputScreen = new IncomeInputScreen(ptrIncomeList);

ptrIncomeInputScreen->inputIncome();

delete ptrIncomeInputScreen;

break;

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

ptrExpenseInputScreen->inputExpense();

delete ptrExpenseInputScreen;

break;

default: cout << "Ошибка ввода\n";

break;

}

}

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

{

cout << "\n";

cout << "| Вывод данных |\n";

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

cout << " 1 - Вывод списка клиентов \n";

cout << " 2 - Вывод списка доходов \n";

cout << " 3 - Вывод списка расходов \n";

cout << " 4 - Вывод годового отчета \n";

ch = getaChar();

system("cls");

switch (ch)

{

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

break;

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

break;

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

break;

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

ptrAnnualReport->display();

delete ptrAnnualReport;

break;

default: cout << "Ошибка ввода\n";

break;

}

}

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

return;

else cout << "Ошибка! Повторите ввод\n";

}

delete ptrClientInputScreen;

}