**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**"Петрозаводский государственный университет”**

**Физико-технический институт**

**Командное задание**

Дисциплина: «Технология программирования»

**Выполнили работу**:   
студенты 3 курса

физико-технического института,

гр.21316,

Боголюбова Галина Сергеевна,

Назарова Александра Константиновна

Жерновенков Никита Владимирович

**Преподаватель:**

Доцент КИИСФЭ,

Бульба Артём Владимирович

Петрозаводск, 2020

**Цель работы:** в контексте выбранного сюжета выполнить разработку программного обеспечения

**Кратко о программной реализации:** программное обеспечение разработано на языке C++ в IDE Visual Studio 2013 Express. Созданные заголовочники и единицы компиляции: файл LORD.h включает в себя интерфейсные части классов, файл LORDMAIN.cpp включает основную функцию программы, файл LORDMETOD.cpp включает в себя тела методов, которые были объявлены в заголовочных фалах.

**Руководство пользователя** по использованию программы в Приложении 1.

**Листинг программ** находится в Приложении 2 и 3.

**Пошаговое описание процесса разработки:**

**1. Сюжет** рассматриваемого примера – индивидуальный предприниматель Оладьев Олег Олегович обратился с просьбой разработать программу, которая бы упростила его работу на своей пекарне «Булки Олега».   
  
В Петрозаводске Олег только начал свою деятельность и в процессе работы понял, что тяжело составлять все финансовые отчеты вручную для **12** фирм, в которые он поставляет свою выпечку в **12** складов раз в месяц. Заказчику необходимо, чтобы в программе были предусмотрены **4** формы, а именно:

* список организаций, которым поставляется выпечка;
* доход пекарни от продукции;
* расходы пекарни;
* годовой отчет пекарни;

**Список организаций**, которым поставляется выпечка, состоит из номера склада и наименования организации. Таблица состоит из 2 столбцов и 12 строк.

**Доход пекарни** состоит из наименования организации по 1 строке на каждую и 12 столбцов (в соответствии с количеством месяцев). В каждую строку организации в соответствии с месяцем записывается общая сумма за 4 недели доставки продукции.

**Расход пекарни** включает в себя столбцы: дата, получатель, сумма платежа и категория платежа.

**Годовой отчет** включает в себя прибыль за год – разница между доходами и расходами предприятия. Годовой отчет можно вывести в любой момент времени для просмотра примерной прибыли пекарни.

**2. Заказчик** Олег Олегович использовал несколько стандартных журналов для записи   
(см. изображение 1) всех сведений о своих клиентах, доходах и расходах предприятия, таким же образом он составлял годовой отчет.



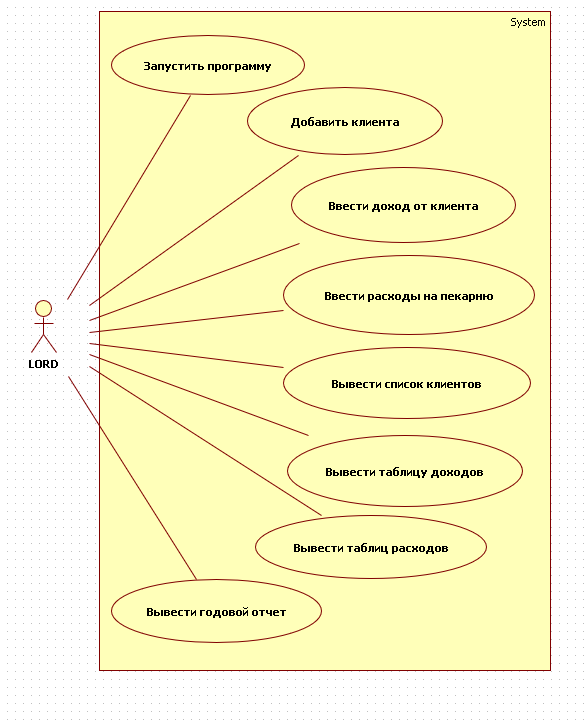
*Фотография 1. Стандартный журнал*

**3. Действующим субъектом** нашей программы является Олег Олегович, т.е. LORD.

**4,5. Варианты использования субъекта и их основные описания:**

* **Запустить программу** - при запуске программы, на экран должно выводиться меню, из которого пользователь может выбрать нужное действие. Это может называться экраном пользовательского интерфейса.
* **Добавить клиента** - на экране должно отобразиться сообщение, в котором программа просит пользователя ввести наименование клиента и номер склада. Эта информация должна заноситься в таблицу.
* **Ввести доход от клиента** - экран ввода дохода от клиента содержит сообщение, из которого пользователь узнает, что ему необходимо ввести наименование клиента (организации), месяц оплаты, а также полученную сумму денег. Программа просматривает список клиентов и по наименованию находит соответствующую запись в таблице доходов. Если этот клиент впервые вносит плату, в этой таблице создается новая строка и указанная сумма заносится в столбец того месяца, за который производится оплата. В противном случае значение вносится в существующую строку.
* **Ввести расходы на пекарню** - экран ввода расхода должен содержать приглашение пользователю на ввод имени получателя (или названия организации), суммы оплаты, дня и месяца, в который производится оплата, бюджетной категории. Затем программа создает новую строку, содержащую эту информацию, и вставляет ее в таблицу расходов.
* **Вывести список клиентов** - программа выводит на экран список клиентов, каждая строка списка состоит из трех полей: наименования организации и номера склада.
* **Вывести таблицу доходов** - каждая строка таблицы, которую выводит программа, состоит из наименования организации и значения ежемесячной оплаты.
* **Вывести таблицу расходов** - каждая строка таблицы, которую выводит программа, состоит из значений месяца, дня, получателя, суммы и бюджетной категории платежа.
* **Вывести годовой отчет** - программа выводит годовой отчет, состоящий из: суммарного дохода за прошедший год; списка всех расходов с указанием бюджетной категории; общая сумма расходов; результирующего годового баланса (доходы/убытки).

**Диаграмма вариантов использования:**

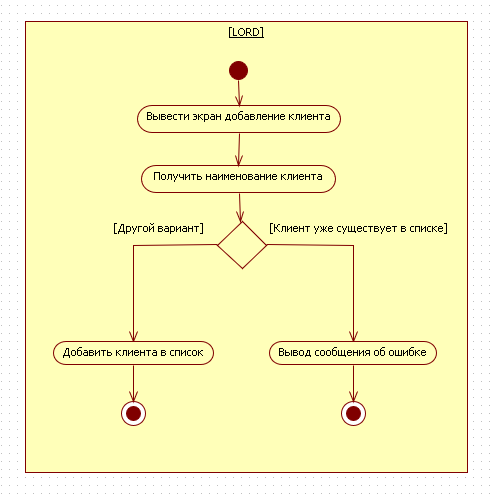
****

*Изображение 1. Диаграмма вариантов использования*

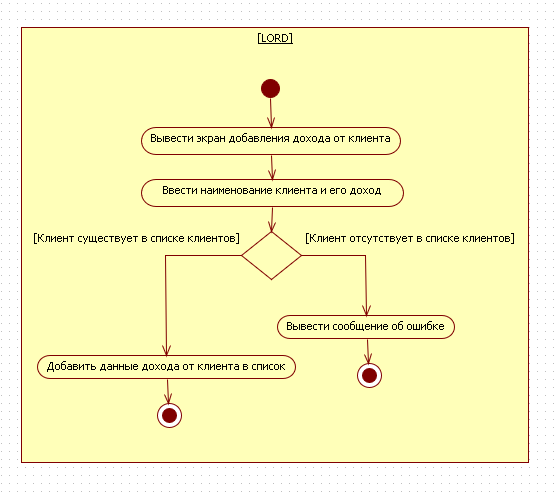
|  |  |  |
| --- | --- | --- |
| **Добавить клиента** | | |
| **№** | **Описание сценария** | **Реакция программы** |
| 1. | На экране отображается экран ввода нового клиента. Введенный клиент уже внесен в список и у него есть свой склад. Пользователю программы выводится сообщение об ошибке. | Выводится ошибка с текстом «Склад уже забронирован» |

|  |  |  |
| --- | --- | --- |
| **Ввести доход от клиента** | | |
| **№** | **Описание сценария** | **Реакция программы** |
| 1. | При вводе данных о доходе от клиента пользователь должен предварительно внести клиента в список и только после этого вводить доход от него. Введенный клиент не существует в списке клиентов. | Выводится ошибка с текстом «Клиент отсутствует в списке клиентов» |

**6. Диаграммы деятельности для 2 вариантов использования:**

****

*Изображение 2. Диаграмма деятельности добавить клиента*

****

*Изображение 3. Диаграмма деятельности ввести доход от клиента*

**7. Список существительных для определения классов:**

* Экран интерфейса пользователя;
* Клиент;
* Экран ввода клиентов;
* Название организации клиента;
* Номер склада;
* Строка клиента;
* Список клиентов;
* Доход;
* Экран ввода дохода от клиента;
* Месяц;
* Общая сумма дохода от клиента за месяц;
* Таблица доходов от клиентов;
* Строка дохода;
* Расход;
* Экран ввода расходов;
* Получатель;
* Размер платежа;
* День;
* Бюджетная категория;
* Строка в таблице расходов;
* Таблица расходов;
* Годовой отчет;
* Суммарный доход от клиентов;
* Суммарные расходы по категориям;

**8. Список существительных для определения классов:**

* Экран пользовательского интерфейса;
* Клиент;
* Экран ввода клиентов;
* Список клиентов;
* Экран ввода дохода от клиентов;
* Таблица доходов;
* Строка таблицы доходов от клиентов;
* Расход;
* Экран ввода расходов;
* Таблица расходов;
* Годовой отчет.

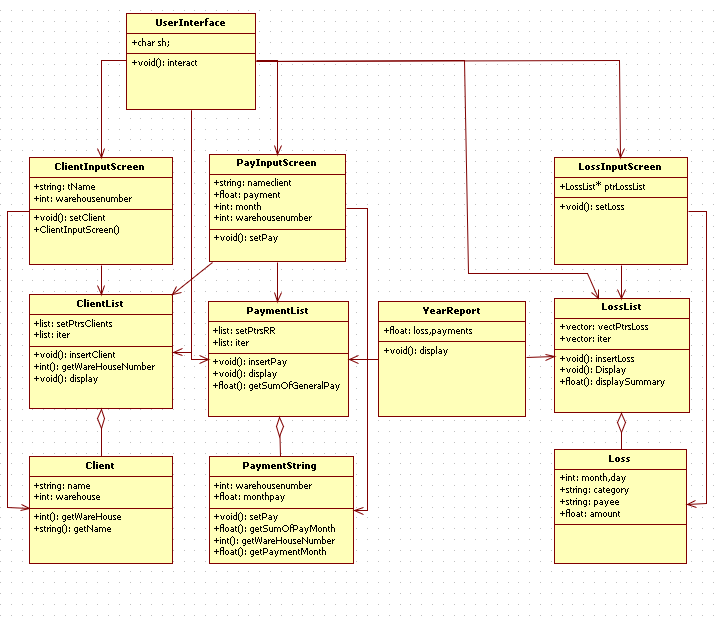
**9. Атрибуты классов:**

* Экран пользовательского интерфейса (выбор функции);
* Экран ввода клиентов (наименование клиента, номер склада);
* Экран ввода дохода от клиентов (наименование клиента, доход, месяц, номер склада);
* Клиент (название организации клиента и номер склада);
* Строка таблицы доходов от клиентов (номер склада, месяц оплаты);
* Таблица доходов (название организации клиента, месяц, день, сумма);
* Таблица расходов (получатель, месяц, день, сумма, бюджетная категория);
* Расход (получатель, месяц, день, сумма, бюджетная категория);
* Годовой отчет (расходы, доход).

**10. Список глаголов для определения методов:**

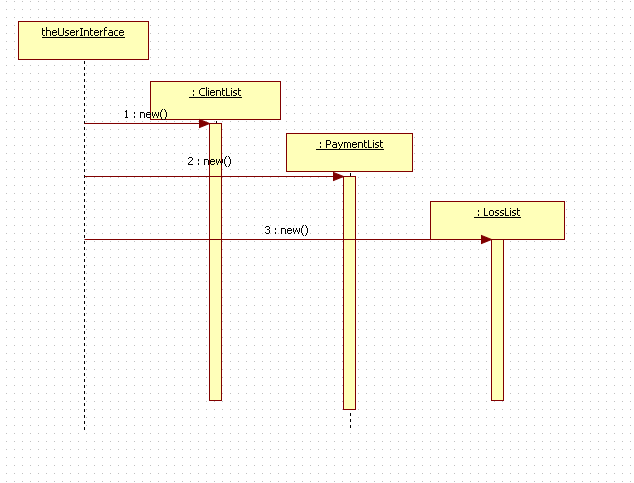
* Записать склад;
* Получить имя;
* Ввести клиента;
* Получить номер склада;
* Вывести на экран;
* Ввести плату;
* Ввести доходы за месяцы;
* Оплатить за месяц;
* Общий доход;
* Ввести потери;
* Вывести потери;
* Вывести годовой отчет;
* Вывести пользовательский интерфейс.

**11.**

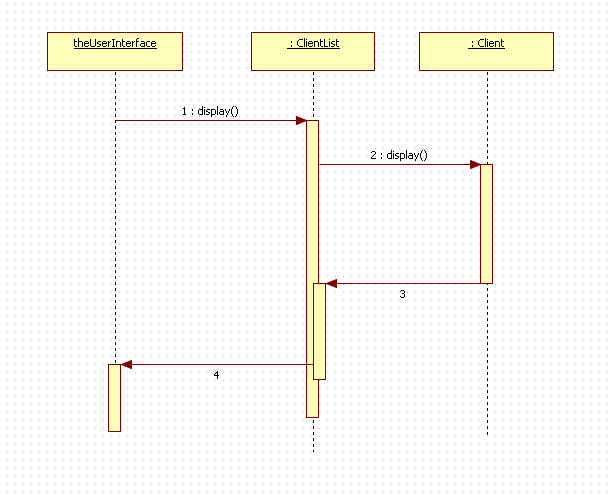


*Изображение 4. Диаграмма классов*

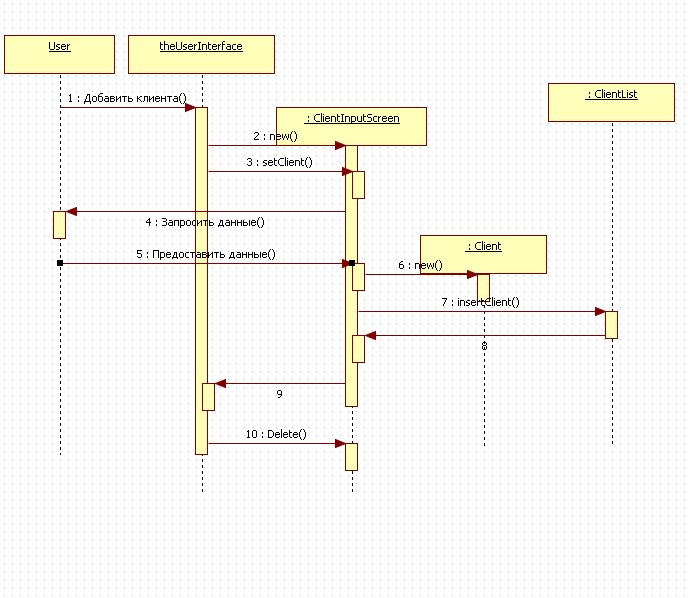
**12.**



*Изображение 5. Диаграмма последовательностей   
для варианта использования «Начать программу»*



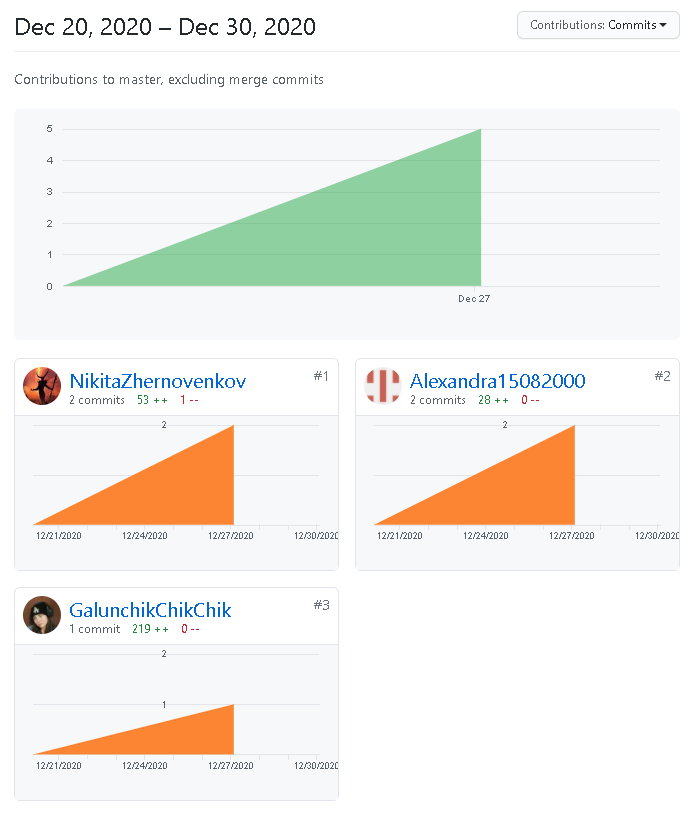
*Изображение 6. Диаграмма последовательностей   
для варианта использования «Вывод списка клиентов»*

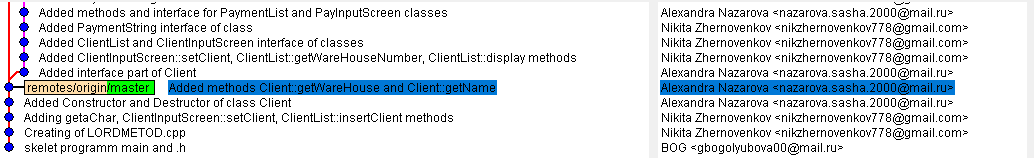


*Изображение 7. Диаграмма последовательностей   
для варианта использования «Добавить нового клиента»*

**История проекта на GitHub:**

[**https://github.com/GalunchikChikChik/CommandProject.git**](https://github.com/GalunchikChikChik/CommandProject.git)







**Заключение:**

Вместе с командой проекта разработали программное обеспечение на языке C++ в IDE Visual Studio 2013 Express. Для этого применяли систему контроля версий GitCMD и использовали удаленный репозиторий на GitHub, но не очень эффективно, что отражается в истории GitHub, большую часть программы создавали лично на локальном репозитории и после пушили на удаленный. Сбоев и зависаний в работе программы нет. Использовали принцип раздельной компиляции. Прописывали отдельно интерфейсную часть и реализацию методов и функций. Предусмотрена очистка памяти после работы программы с удалением всех объектов. Отсутствуют неиспользуемые переменные и нет лишних циклов и массивов для нагрузки программы. Перед началом работы прописали необходимые теоретические данные: субъект программы (актер), его варианты использования, определили список классов и методов; создали диаграммы деятельности, последовательности и классов. Считаем, что цель – разработать ПО в соответствии с сюжетом выполнена и заказчик будет доволен программой.

**Приложение 1.**

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

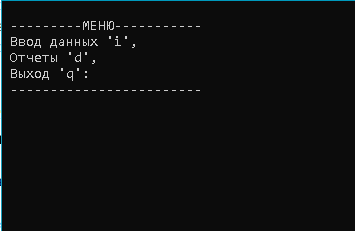
Данное программное обеспечение разработано на языке C++ в IDE Visual Studio 2013 Express и создана для упрощения работы руководителя компании по учету и обработке входящих данных. Разработчики предусмотрели все пожелания заказчика и проанализировали данные, которые будут обрабатываться.

Разработанное ПО предлагает функционал для составления отчетов 4 видов:

* Список клиентов;
* Отчет по доходам;
* Отчет по расходам;
* Годовой отчет.

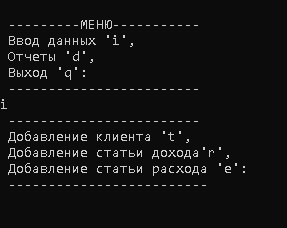
Инструкция по работе с программой:

При запуске ПО вы увидите меню с 3-мя пунктами, предлагающими ввести данные с помощью символа “i”, вывести все данные в отчеты с помощью символа “d”, либо выйти из программы с помощью символа “q”. После написания каждого символа необходимо нажимать клавишу “Enter” для выполнения функции.



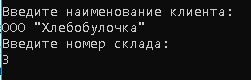
*Изображение 1. Меню программы*

Ввод данных подразделяется на ввод новых клиентов “t”, ввод доходов “r” и расходов “e” с помощью соответствующих клавиш.



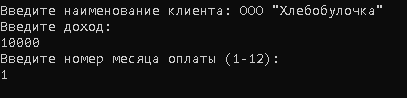
*Изображение 2. Ввод данных*

Для ввода новых клиентов необходимо написать название компании и номер склада, на который поставляется заказ.



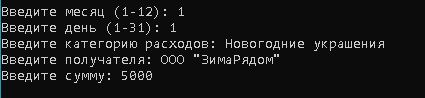
*Изображение 3. Ввод нового клиента*

Для анализа доходов вводится название компании клиента, доход от этой организации и месяц, когда поступил платеж.



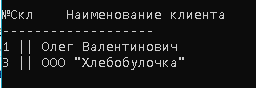
*Изображение 4. Ввод дохода*

Для учета расходов компании заказчика предусматривается введение дня и месяца, категории расхода, получателя текущей выплаты и суммы.



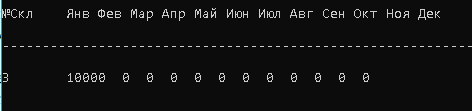
*Изображение 5. Ввод расходов*

В списке клиентов вы можете увидеть номер склада, на который поставляется выпечка и наименование клиента.

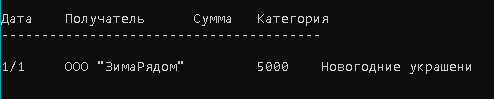


*Изображение 6. Список клиентов*

Доход отображается под каждым месяцем и с определенного склада, куда поставлялась продукция.

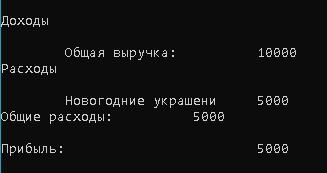


*Изображение 7. Список доходов*



*Изображение 8. Список расходов*

Годовой отчет автоматически рассчитывается на проанализированных данных списка доходов и расходов для анализа общей прибыли компании.



*Изображение 9. Годовой отчет*

**Приложение 2.**

**КОД ЗАГОЛОВОЧНЫХ ФАЙЛОВ**

Файл **LORD.h**

[#include](https://vk.com/im?sel=77215899&st=%23include) <iostream>  
[#include](https://vk.com/im?sel=77215899&st=%23include) <vector>  
[#include](https://vk.com/im?sel=77215899&st=%23include) <list>  
[#include](https://vk.com/im?sel=77215899&st=%23include) <string>  
[#include](https://vk.com/im?sel=77215899&st=%23include) <numeric>  
  
using namespace std;  
  
  
void getaLine(string& inStr);  
  
char getaChar();  
  
//////////////////// класс Client //////////////////  
class Client  
{  
private:  
string name;  
int warehouse;  
  
  
public:  
Client(string n, int warehs);  
~Client();  
int getWareHouse();  
string getName();  
};  
///////////////////класс ClientList ////////////////////////  
  
class ClientList  
{  
private:  
  
list <Client\*> setPtrsClients;  
list <Client\*>::iterator iter;  
public:  
~ClientList();  
void insertClient(Client\*);  
int getWareHouseNumber(string);  
void display();  
};  
////////////////класс ClientInputScreen////////////////////  
  
class ClientInputScreen  
  
{  
private:  
ClientList\* ptrClientList;  
string tName;  
int warehousenumber;  
public:  
ClientInputScreen(ClientList\* ptrTL) : ptrClientList(ptrTL)  
{  
  
}  
void setClient();  
};  
//////////////////// класс PaymentString //////////////////////////  
  
class PaymentString  
{  
private:  
  
int warehousenumber;  
  
float monthpay[12];  
public:  
PaymentString(int);  
void setPay(int, float);  
float getSumOfPayMonth();  
  
int getWareHouseNumber();  
  
float getPaymentMonth(int);  
};  
//////////////////// class PaymentList ///////////////////////  
  
class PaymentList  
{  
private:  
list <PaymentString\*> setPtrsRR;  
list <PaymentString\*>::iterator iter;  
public:  
~PaymentList();  
void insertPay(int, int, float);  
void display();  
float getSumOfGeneralPay();  
};  
//////////////////// class PayInputScreen //////////////////  
  
class PayInputScreen  
{  
private:  
ClientList\* ptrClientList;  
PaymentList\* ptrPaymentList;  
string nameclient;  
float payment;  
int month;  
int warehousenumber;  
  
public:  
PayInputScreen(ClientList\* ptrTL, PaymentList\* ptrRR) : ptrClientList(ptrTL),  
ptrPaymentList(ptrRR)  
{  
  
}  
void setPay();  
};  
///////////////////////класс Loss///////////////////////  
  
class Loss  
{  
public:  
int month, day;  
string category;  
string payee;  
float amount;  
Loss()  
{ }  
Loss(int m, int d, string c, string p, float a) :  
month(m), day(d), category(c), payee(p), amount(a)  
{  
  
}  
};  
///////////////////класс LossList/////////////////////  
  
class LossList  
{  
private:  
vector<Loss\*> vectPtrsExpenses;  
vector<Loss\*>::iterator iter;  
public:  
~LossList();  
void insertLoss(Loss\*);  
void display();  
float displaySummary();  
};  
////////////////класс LossInputScreen///////////////////  
  
class LossInputScreen  
{  
private:  
LossList\* ptrLossList;  
public:  
LossInputScreen(LossList\*);  
void setLoss();  
};  
//////////////////класс YearReport///////////////////////  
  
class YearReport  
{  
private:  
PaymentList\* ptrRR;  
LossList\* ptrER;  
float loss, payments;  
public:  
YearReport(PaymentList\*, LossList\*);  
void display();  
};  
//////////////////Класс UserInterface//////////////////////  
  
class UserInterface  
{  
private:  
ClientList\* ptrClientList;  
ClientInputScreen\* ptrClientInputScreen;  
PaymentList\* ptrPaymentList;  
PayInputScreen\* ptrPayInputScreen;  
LossList\* ptrLossList;  
LossInputScreen\* ptrLossInputScreen;  
YearReport\* ptrYearReport;  
char ch;  
public:  
  
UserInterface();  
~UserInterface();  
void interact();  
};

**Приложение 3.**

**КОД ИСХОДНЫХ ФАЙЛОВ**

Файл **LORDMAIN.cpp**

[#include](https://vk.com/im?sel=77215899&st=%23include) "LORD.h"  
[#include](https://vk.com/im?sel=77215899&st=%23include) "windows.h"  
int main()  
{  
  
SetConsoleCP(1251);  
SetConsoleOutputCP(1251);  
setlocale(LC\_ALL, "rus");  
UserInterface theUserInterface;  
theUserInterface.interact();  
return 0;  
}

Файл **LORDMETOD.cpp**

[#include](https://vk.com/im?sel=77215899&st=%23include) <iostream>  
[#include](https://vk.com/im?sel=77215899&st=%23include) <string>  
[#include](https://vk.com/im?sel=77215899&st=%23include) "LORD.h"  
  
  
void getaLine(string& inStr)  
{  
char temp[21];  
cin.get(temp, 20, '\n');  
cin.ignore(20, '\n');  
inStr = temp;  
}  
  
char getaChar()  
{  
char ch;  
while (true)  
{  
ch = getchar();  
if (ch != '\n')  
{  
break;  
}  
}  
getchar();  
return ch;  
}  
  
  
////////////////методы класса Client///////////////////////  
  
Client::Client(string n, int warehs) : name(n), warehouse(warehs)  
{  
  
}  
  
Client::~Client()  
{  
  
}  
  
int Client::getWareHouse()  
{  
return warehouse;  
}  
  
string Client::getName()  
{  
return name;  
}  
  
///////////метод класса ClientInputScreen//////////////////  
  
void ClientInputScreen::setClient()  
{  
cout « "Введите наименование клиента: " « endl;  
getaLine(tName);  
cout « "Введите номер склада: " « endl;  
cin » warehousenumber;  
cin.ignore(80, '\n');  
Client\* ptrClient = new Client(tName, warehousenumber);  
ptrClientList->insertClient(ptrClient);  
}  
  
////////////////методы класса ClientList///////////////////  
  
ClientList::~ClientList()  
{  
while (!setPtrsClients.empty())  
{  
iter = setPtrsClients.begin();  
delete \*iter;  
setPtrsClients.erase(iter);  
}  
}  
  
void ClientList::insertClient(Client\* ptrT)  
{  
bool exists = false;  
for (iter = setPtrsClients.begin(); iter != setPtrsClients.end(); iter++)  
{  
if ((\*iter)->getWareHouse() == ptrT->getWareHouse())  
{  
exists = true;  
break;  
}  
}  
if (!exists)  
setPtrsClients.push\_back(ptrT);  
else  
{  
  
delete ptrT;  
cout « "\nСклад забронирован ";  
}  
}  
  
  
int ClientList::getWareHouseNumber(string tName)  
{  
int warehousenumber;  
iter = setPtrsClients.begin();  
while (iter != setPtrsClients.end())  
{  
warehousenumber = (\*iter)->getWareHouse();  
if (tName == ((\*iter)->getName()))  
{  
return warehousenumber;  
}  
iter++;  
}  
return -1;  
}  
  
  
void ClientList::display()  
{  
cout « "\n№Скл\tНаименование клиента\n-------------------\n";  
if (setPtrsClients.empty())  
cout « " Клиенты отсутствуют \n" « endl;  
else  
{  
iter = setPtrsClients.begin();  
while (iter != setPtrsClients.end())  
{  
cout « (\*iter)->getWareHouse() « " || " « (\*iter)->getName() « endl;  
\*iter++;  
}  
}  
}  
  
/////////////////методы класса PaymentString/////////////////////  
  
PaymentString::PaymentString(int an) : warehousenumber(an)  
{  
fill(&monthpay[0], &monthpay[12], 0);  
}  
  
void PaymentString::setPay(int m, float am)  
{  
monthpay[m] = am;  
}  
  
int PaymentString::getWareHouseNumber()  
{  
return warehousenumber;  
}  
  
float PaymentString::getPaymentMonth(int month)  
{  
return monthpay[month];  
}  
  
float PaymentString::getSumOfPayMonth()  
{  
return accumulate(&monthpay[0], &monthpay[12], 0);  
}  
  
/////////////////методы класса PaymentList /////////////////  
  
PaymentList::~PaymentList()  
{  
while (!setPtrsRR.empty())  
{  
iter = setPtrsRR.begin();  
delete \*iter;  
setPtrsRR.erase(iter);  
}  
}  
  
void PaymentList::insertPay(int warehousenumber, int month, float amount)  
{  
  
iter = setPtrsRR.begin();  
while (iter != setPtrsRR.end())  
{  
if (warehousenumber == (\*iter)->getWareHouseNumber())  
{  
(\*iter)->setPay(month, amount);  
return;  
}  
else  
iter++;  
}  
PaymentString\* ptrRow = new PaymentString(warehousenumber);  
ptrRow->setPay(month, amount);  
setPtrsRR.push\_back(ptrRow);  
}  
  
void PaymentList::display()  
{  
cout « "\n№Скл\tЯнв Фев Мар Апр Май Июн Июл Авг Сен Окт Ноя Дек\n" « endl  
« "------------------------------------------------------------------\n" « endl;  
if (setPtrsRR.empty())  
cout « " Доходы отсутствуют \n" « endl;  
else  
{  
iter = setPtrsRR.begin();  
while (iter != setPtrsRR.end())  
{  
cout « (\*iter)->getWareHouseNumber() « '\t';  
for (int j = 0; j < 12; j++)  
{  
if (((\*iter)->getPaymentMonth(j)) == 0)  
cout « " 0 ";  
else  
cout « (\*iter)->getPaymentMonth(j) « " ";  
}  
cout « endl;  
iter++;  
}  
cout « endl;  
cout « endl;  
}  
}  
  
float PaymentList::getSumOfGeneralPay()  
{  
float

sumRents = 0.0;  
iter = setPtrsRR.begin();  
  
while (iter != setPtrsRR.end())  
{  
sumRents += (\*iter)->getSumOfPayMonth();  
iter++;  
}  
return sumRents;  
}  
  
/////////////////методы класса PayInputScreen/////////////  
  
void PayInputScreen::setPay()  
{  
cout « "Введите наименование клиента: ";  
getaLine(nameclient);  
  
warehousenumber = ptrClientList->getWareHouseNumber(nameclient);  
if (warehousenumber > 0)  
{  
  
cout « "Введите доход: " « endl;  
cin » payment;  
cin.ignore(80, '\n');  
cout « "Введите номер месяца оплаты (1-12): " « endl;  
cin » month;  
cin.ignore(80, '\n');  
month--;  
  
ptrPaymentList->insertPay(warehousenumber, month, payment);  
}  
else  
{  
cout « "\n —-------------------------";  
cout « "\nКлиент отсутствует в списке\n ";  
cout « " —-------------------------";  
}  
}  
  
//////////////////методы класса LossList//////////////  
  
LossList::~LossList()  
{  
  
while (!vectPtrsExpenses.empty())  
{  
iter = vectPtrsExpenses.begin();  
delete \*iter;  
vectPtrsExpenses.erase(iter);  
}  
}  
  
void LossList::insertLoss(Loss\* ptrExp)  
{  
  
vectPtrsExpenses.push\_back(ptrExp);  
}  
  
void LossList::display()  
{  
cout « "\nДата\tПолучатель\tСумма\tКатегория\n"  
« "----------------------------------------\n" « endl;  
if (vectPtrsExpenses.size() == 0)  
cout « " Расходы отсутствуют \n" « endl;  
else  
{  
iter = vectPtrsExpenses.begin();  
while (iter != vectPtrsExpenses.end())  
{  
cout « (\*iter)->month « '/' « (\*iter)->day « '\t' « (\*iter)->payee « '\t' « '\t';  
cout « (\*iter)->amount « '\t' « (\*iter)->category « endl;  
iter++;  
}  
cout « endl;  
}  
}  
  
  
float LossList::displaySummary()  
{  
float totalLoss = 0;  
if (vectPtrsExpenses.size() == 0)  
{  
cout « "\tВсе категории\t0\n";  
return 0;  
}  
iter = vectPtrsExpenses.begin();  
while (iter != vectPtrsExpenses.end())  
{  
  
cout « '\t' « ((\*iter)->category) « '\t' « ((\*iter)->amount) « endl;  
totalLoss += (\*iter)->amount;  
iter++;  
}  
return totalLoss;  
}  
  
//////////////методы класса LossInputScreen/////////////  
  
  
LossInputScreen::LossInputScreen(LossList\* per) : ptrLossList(per)  
{  
  
}  
  
void LossInputScreen::setLoss()  
{  
int month, day;  
string category, payee;  
float amount;  
cout « "Введите месяц (1-12): ";  
cin » month;  
cin.ignore(80, '\n');  
cout « "Введите день (1-31): ";  
cin » day;  
cin.ignore(80, '\n');  
cout « "Введите категорию расходов: ";  
getaLine(category);  
cout « "Введите получателя: ";  
getaLine(payee);  
cout « "Введите сумму: ";  
cin » amount;  
cin.ignore(80, '\n');  
  
Loss\* ptrLoss = new Loss(month, day, category, payee, amount);  
  
ptrLossList->insertLoss(ptrLoss);  
}  
  
////////////////методы класса YearReport/////////////////  
  
YearReport::YearReport(PaymentList\* pRR, LossList\* pER) : ptrRR(pRR), ptrER(pER)  
{  
  
}  
  
void YearReport::display()  
{  
cout « "Годовой отчет\n--------------\n" « endl;  
cout « "Доходы\n" « endl;  
cout « "\tОбщая выручка:\t\t";  
payments = ptrRR->getSumOfGeneralPay();  
cout « payments « endl;  
cout « "Расходы\n" « endl;  
loss = ptrER->displaySummary();  
cout « "Общие расходы:\t\t";  
cout « loss « endl;  
  
cout « "\nПрибыль:\t\t\t" « (payments - loss) « endl;  
}  
  
////////////////методы класса UserInterface//////////////  
  
UserInterface::UserInterface()  
{  
ptrClientList = new ClientList;  
ptrPaymentList = new PaymentList;  
ptrLossList = new LossList;  
}  
  
UserInterface::~UserInterface()  
{  
delete ptrClientList;  
delete ptrPaymentList;  
delete ptrLossList;  
}  
  
void UserInterface::interact()  
{  
while (true)  
{  
cout « " \n —-------МЕНЮ-----------\n"  
« " Ввод данных 'i', \n"  
« " Отчеты 'd', \n"  
« " Выход 'q': \n"  
« " —----------------------\n";  
ch = getaChar();  
if (ch == 'i')  
{  
cout « " —----------------------\n"  
« " Добавление клиента 't', \n"  
« " Добавление статьи дохода'r', \n"  
« " Добавление статьи расхода 'e': \n"  
« "

-------------------------";  
ch = getaChar();  
switch (ch)  
{  
  
case 't': ptrClientInputScreen =  
new ClientInputScreen(ptrClientList);  
ptrClientInputScreen->setClient();  
delete ptrClientInputScreen;  
break;  
  
case 'r': ptrPayInputScreen =  
new PayInputScreen(ptrClientList, ptrPaymentList);  
ptrPayInputScreen->setPay();  
delete ptrPayInputScreen;  
break;  
case 'e': ptrLossInputScreen =  
new LossInputScreen(ptrLossList);  
ptrLossInputScreen->setLoss();  
delete ptrLossInputScreen;  
break;  
default:  
cout « " \n-----------------ОШИБКА----------------------";  
cout « " Выберите необходимую функцию из меню \n";  
cout « " —-------------------------------------------";  
  
break;  
}  
}  
else if (ch == 'd')  
{  
cout « " \n------------------------\n"  
« " Список клиентов 't' \n"  
« " Список доходов 'r' \n"  
« " Список расходов 'e' \n"  
« " Годовой отчет 'a' \n"  
« " —-----------------";  
ch = getaChar();  
switch (ch)  
{  
case 't': ptrClientList->display();  
break;  
case 'r': ptrPaymentList->display();  
break;  
case 'e': ptrLossList->display();  
break;  
case 'a':  
ptrYearReport = new YearReport(ptrPaymentList,ptrLossList);  
ptrYearReport->display();  
delete ptrYearReport;  
break;  
default:  
  
cout « " \n-----------------ОШИБКА----------------------";  
cout « " Выберите необходимую функцию вывода \n";  
cout « " —-------------------------------------------";  
  
break;  
}  
}  
else if (ch == 'q')  
return;  
else  
cout « " —-----------------";  
}  
}