Министерство науки и высшего образования Российской Федерации   
Федеральное государственное бюджетное образовательное учреждение высшего образования «Петрозаводский государственный университет»  
 Физико-технический институт   
Кафедра информационно-измерительных систем и физической электроники

**Отчет по лабораторной работе**

«КОМАНДНОЕ ЗАДАНИЕ ПО КУРСУ ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ»

Автор работы:

Студент группы 21316

Окоротышев М.В.

Преподаватель:

канд. физ.-мат. наук, доцент

Бульба Артём Владимирович

Петрозаводск 2022

ОГЛАВЛЕНИЕ

[Цель работы 3](#_Toc123144658)

[Кратко о программной реализации: 3](#_Toc123144659)

[**1)** Краткое словесное описание сюжета: 3](#_Toc123144660)

[**2)** Предварительный список существительных: 3](#_Toc123144661)

[**3)** Уточненный список существительных: 3](#_Toc123144662)

[**4)** Список атрибутов: 3](#_Toc123144663)

[**5)** Список сообщений: 4](#_Toc123144664)

[**6)** Листинг заголовочных файлов: 5](#_Toc123144665)

[**7)** Листинг исполнительных файлов: 6](#_Toc123144666)

[Заключение 11](#_Toc123144668)

# Цель работы: Рассмотреть простой пример выполнения заказа на разработку программного обеспечения и, в качестве самостоятельного задания, в контексте ритуального бюро разработать программу, которая бы упростила работу по учету товаров/услуг и ведению сопутствующей финансовой деятельности.

# Кратко о программной реализации:

* Среда разработки: IDE Qt Creator (Qt 5.4.2).
* Язык программирования: С++.
* Созданные в процессе разработки заголовочные файлы и файлы реализации:

"product.h", "product.cpp" – класс Product

"productlist.h", "productlist.cpp" – класс ProductList

"productinputscreen.h", "productinputscreen.cpp" – класс ProductInputScreen

"userinterface.h", "userinterface.cpp" – класс UserInterface и глобальные функции

Пошаговое описание процесса разработки:

1. Краткое словесное описание сюжета:

К нам обратился заказчик – индивидуальный предприниматель Ефремов Н.Б., с недавних пор собственник ритуального бюро. Ритуальное бюро имеет ряд товаров и услуг, учет продаж которых с ростом бизнеса становится все сложнее.

Программа RitualCompany призвана упростить контроль учёта. Она способна осуществлять ввод и вывод данных.

Программа RitualCompany предоставляет возможность добавлять товары и услуги с учетом их артикулов и цены, выводить список товаров/услуг в формате: артикул, название товара/услуги, цена. Также можно вводить кол-во продаж товаров/услуг за месяц для вычисления выручки исходя из установленной на товар/услугу цены и выводить таблицу выручек по месяцам. Предусмотрена возможность фиксировать расходы и выводить таблицу с ними, также их можно использовать их для вычисления прибыли в годовом отчёте.

1. Предварительный список существительных:

1. Экран интерфейса пользователя.

2. Товар.

3. Экран ввода товара.

4. Название товара.

5. Стоимость товара.

6. Список товаров.

1. Уточненный список существительных:

1. Экран пользовательского интерфейса.

2. Товар.

3. Экран ввода товара.

4. Список товаров.

1. Список атрибутов:

**1. Класс UserInterface.**  
a) ptrProductList - указатель на класс список товаров.  
b) ptrProductInputScreen - указатель на класс экран ввода товаров.  
c) ptrRevenueRecord - указатель на класс записей выручки.  
d) ptrSellsInputScreen - указатель на класс экран добавления количества продаж.  
e) ptrExpenseRecord - указатель на класс записей о расходах.  
f) ptrExpenseInputScreen - указатель на класс ввода расходов.  
g) ptrAnnualReport - указатель на класс годового отчета.  
h) ty – переменная для работы оператора множественного выбора switch.  
**2. Класс Product.**  
a) name – название товара/услуги. В последующих пояснениях "товар/услуга" будет сокращено до "товар" для повышения читабельности.  
b) article – артикул товара.  
с) price – цена товара. **3. Класс ProductInputScreen.**  
a) ptrProductList – указатель на список товаров.  
b) pName – название товара.  
c) art – артикул товара.  
d) prc – цена товара. **4. Класс ProductList.**  
a) setPtrsProd – список указателей на класс Product.  
b) iter – итератор для работы со списком setPtrsProd.

1. Список сообщений:

**1. Класс UserInterface.**a) UserInterface() – конструктор.  
b) ~UserInterface() – деструктор.  
c) interact() – функция для работы пользователя с программой.  
**2. Класс Product.**a) Product() – конструктор.  
b) ~Product() – деструктор.  
c) getArticle() - возвращает артикул товара.  
d) getPrice() – возвращает цену товара.  
e) getName() – возвращает название товара. **3. Класс ProductInputScreen.**a) ProductInputScreen() – конструктор.  
b) setProduct() – добавляет данные о товаре и заносит в список товаров.  
**4. Класс ProductList.**  
a) ~ProductList() – деструктор.  
b) insertProduct() – добавить товар в список.  
c) getArt() – возвращает артикул товара по названию.  
d) getPrc() – возвращает цену товара по названию.  
e) display() - вывод списка товаров.

1. Листинг заголовочных файлов:

**Product.h**

#pragma once

#include <string>

using namespace std;

//////////////////// класс Product (товары/услуги) /////////////////

//Он хранит имя товара/услуги, артикул и цену

//В последующих комментариях "товар/услуга" будет сокращено до "товар"

//для повышения читабельности

class Product

{

private:

string name; // название товара

int article; // артикул товара

float price; //цена товара

public:

Product(string n, int art, float pr);

~Product();

int getArticle(); //возвращает артикул товара

float getPrice(); //возращает цену товара

string getName(); //возвращает название товара

};

**ProductList.h**

#pragma once

#include "product.h"

#include <list>

#include <iostream>

#include <iomanip>

///////////////////класс ProductList////////////////////////

//класс ProductList — список всех товаров.

//Он содержит множество указателей на класс Product

//и оперирует ими при выводе

class ProductList

{

private:

// установить указатели на товары

list <Product\*> setPtrsProd; // указатели на класс Product

list <Product\*>::iterator iter; //итератор

public:

~ProductList(); // деструктор (удаление товаров)

void insertProduct(Product\*); // добавить товар в список

int getArt(string); // возвращает артикул товара

float getPrc(string); //возвращает цену товара

void display(); // вывод списка товаров

};

**ProductInputScreen.h**

#pragma once

#include "productlist.h"

////////////////класс ProductInputScreen////////////////////

//класс ProductInputScreen. Это класс, отвечающий за отображение «экрана»,

//куда пользователь может ввести данные о новом товаре:

class ProductInputScreen

{

private:

ProductList\* ptrProductList; //указатель на список товаров

string pName; //название товара

int art; //артикул товара

float prc; //цена

public:

ProductInputScreen(ProductList\* ptrPL) : ptrProductList(ptrPL)

{

/\* тут пусто \*/

}

void setProduct(); // добавить данные о товаре

};

**UserInterface.h**

#pragma once

#include "productlist.h"

#include "productinputscreen.h"

#include "revenuerecord.h"

#include "sellsinputscreen.h"

#include "expenserecord.h"

#include "expenseinputscreen.h"

#include "annualreport.h"

using namespace std;

//////////////////// глобальные методы //////////////////////

void getaLine(string& inStr); // получение строки текста

char getaChar(); // получение символа

//////////////////Класс UserInterface//////////////////////

//Главный класс для запуска приложения:

//этот класс определяет взаимодействие пользователя с программой.

class UserInterface

{

private:

ProductList\* ptrProductList;

ProductInputScreen\* ptrProductInputScreen;

RevenueRecord\* ptrRevenueRecord;

SellsInputScreen\* ptrSellsInputScreen;

ExpenseRecord\* ptrExpenseRecord;

ExpenseInputScreen\* ptrExpenseInputScreen;

AnnualReport\* ptrAnnualReport;

char ty;

public:

UserInterface();

~UserInterface();

void interact();

};

1. Листинг исполнительных файлов:

**Product.cpp**

#include "product.h"

////////////////методы класса Product///////////////////////

//в конструкторе задаём название товара, артикул и цену

Product::Product(string n, int art, float pr) : name(n), article(art), price(pr)

{

/\* тут пусто \*/

}

//--------------------------------------------------------

Product::~Product() // деструктор

{

/\* тут тоже пусто \*/

}

//--------------------------------------------------------

string Product::getName() //геттер возвращает название товара

{

return name;

}

//--------------------------------------------------------

int Product::getArticle() //геттер возвращает артикул товара

{

return article;

}

//--------------------------------------------------------

float Product::getPrice() //геттер возвращает цену товара

{

return price;

}

**ProductList.cpp**

#include "productlist.h"

////////////////методы класса ProductList///////////////////

ProductList::~ProductList() // деструктор

{

while (!setPtrsProd.empty()) // удаление всех товаров, пока список товаров не пуст

{ // удаление указателей из контейнера

iter = setPtrsProd.begin(); // итератор на начало списка

delete\* iter; //освобождение памяти

setPtrsProd.erase(iter); //удаление элемента с текущей позиции итератора erase()

}

}

//--------------------------------------------------------

void ProductList::insertProduct(Product\* ptrP)

{

setPtrsProd.push\_back(ptrP); // вставка нового указателя на класс товара в конец списка

}

//--------------------------------------------------------

int ProductList::getArt(string pName) // получить артикул по названию товара

{

int art;

iter = setPtrsProd.begin(); //итератор на начало списка товаров

while (iter != setPtrsProd.end()) // пока не конец списка

{ // поиск товара в списке (достаем у каждого товара его артикул)

art = (\*iter)->getArticle(); //в переменную заносится артикул по объекту, на котором стоит итератор

if (pName == ((\*iter)->getName())) // сравниваем по названиям и

{

// если получившаяся пара совпадает - значит,

//мы нашли запись об этом товаре в списке, в этом случае

return art; // возвращаем его артикул

}

iter++;

}

return -1; // если нет - возвращаем -1

}

//--------------------------------------------------------

float ProductList::getPrc(string pName) // получить цену товара по его названию

{

float prc;

iter = setPtrsProd.begin(); //итератор на начало списка товаров

while (iter != setPtrsProd.end()) // пока не конец списка

{ // поиск товара в списке (достаем у каждого товара его цену)

prc = (\*iter)->getPrice(); //в переменную заносится значение цены по объекту, на котором стоит итератор

if (pName == ((\*iter)->getName())) // сравниваем по названиям и

{

// если получившаяся пара совпадает - значит,

//мы нашли запись об этом товаре в списке, в этом случае

return prc; // возвращаем его цену

}

iter++;

}

return -1; // если нет - возвращаем -1

}

//--------------------------------------------------------

void ProductList::display() // вывод списка товаров

{

cout << setw(10) <<left <<"\nArticle" << setw(20) <<left << "Product name"<< setw(10)<< left<< "Price\n" ;

if (setPtrsProd.empty()) // если список товаров пуст

cout << "ERROR:Products have not been added yet\n" << endl; // выводим запись, что он пуст)

else

{

iter = setPtrsProd.begin();

while (iter != setPtrsProd.end()) // выводим список всех товаров

{

cout << "\n" << setw(10) << left << (\*iter)->getArticle() << setw(20) << left << (\*iter)->getName() << setw(10) << left << (\*iter)->getPrice()<< endl;

\*iter++;

}

cout << "\n" << endl;

}

}

//--------------------------------------------------------

**ProductInputScreen.cpp**

#include "productinputscreen.h"

#include "userinterface.h"

///////////метод класса ProductInputScreen//////////////////

void ProductInputScreen::setProduct() // добавить данные о товаре

{

cout << "Enter the name of the product/service: " << endl;

getaLine(pName);

cout << "Enter the article number (1234): " << endl;

cin >> art;

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

cout << "Enter the price (499.9): " << endl;

cin >> prc;

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

Product\* ptrProduct = new Product(pName, art, prc); // создать товар

ptrProductList->insertProduct(ptrProduct); // занести в список товаров

}

**UserInterface.cpp**

#include "userinterface.h"

void getaLine(string& inStr) // получение строки текста

{

char temp[22];

cin.get(temp, 21, '\n'); //cin.get не закончит ввод при нажатии пробела при вводе названий товаров,

// состоящих из 2 и более слов

cin.ignore(21, '\n'); //число пропускаемых символов и символ разделения, cin.ignore дает ввести определенное кол-во символов

inStr = temp;

}

char getaChar() // получение символа

{

char ch = cin.get();

cin.ignore(80, '\n'); //число пропускаемых символов и символ разделения

return ch;

}

//--------------------------------------------------------

////////////////методы класса userInterface//////////////

UserInterface::UserInterface()

{

ptrProductList = new ProductList;

ptrRevenueRecord = new RevenueRecord;

ptrExpenseRecord = new ExpenseRecord;

}

//---------------------------------------------------------

UserInterface::~UserInterface()

{

delete ptrProductList;

delete ptrRevenueRecord;

delete ptrExpenseRecord;

}

//---------------------------------------------------------

void UserInterface::interact()

{

while (true)

{

cout << "To enter data, press 'i', \n"

<< "To output data, press 'd', \n"

<< "To exit, press 'q': \n";

ty = getaChar();

if (ty == 'i') // ввод данных

{

cout << "Press 'p' to add the product, \n"

<< "Press 'r' to record the amount of sales, \n"

<< "Press 'e' to record expenses: \n";

ty = getaChar();

switch (ty)

{

//экраны ввода существуют только во время их

//использования

case 'p': ptrProductInputScreen =

new ProductInputScreen(ptrProductList);

ptrProductInputScreen->setProduct();

delete ptrProductInputScreen;

cout << "Product was successfully added\n" << endl;

break;

case 'r': ptrSellsInputScreen =

new SellsInputScreen(ptrProductList, ptrRevenueRecord);

ptrSellsInputScreen->setSells();

delete ptrSellsInputScreen;

cout << "Sales were successfully added\n" << endl;

break;

case 'e': ptrExpenseInputScreen =

new ExpenseInputScreen(ptrExpenseRecord);

ptrExpenseInputScreen->setExpense();

delete ptrExpenseInputScreen;

cout << "Expense was successfully added\n" << endl;

break;

default: cout << "Unknown function\n";

break;

} // конец секции switch

} // конец условия if

else if (ty == 'd') // вывод данных

{

cout << "Press 'p' to output products, \n"

<< "Press 'r' to output revenue, \n"

<< "Press 'e' to output expenses, \n"

<< "Press 'a' to output the annual report: \n";

ty = getaChar();

switch (ty)

{

case 'p': ptrProductList->display();

break;

case 'r': ptrRevenueRecord->display();

break;

case 'e': ptrExpenseRecord->display();

break;

case 'a':

ptrAnnualReport = new AnnualReport(ptrRevenueRecord, ptrExpenseRecord);

ptrAnnualReport->display();

delete ptrAnnualReport;

break;

default: cout << "Unknown output function\n";

break;

} // конец switch

} // конец elseif

else if (ty == 'q')

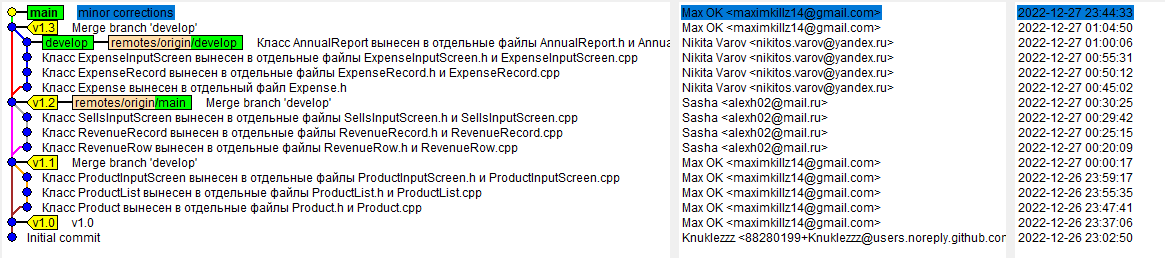
return; // выход

else

cout << "Unknown function. Press only 'i', 'd' or 'q'\n";

} // конец while

} // конец interact()



# Заключение: использовалась среда разработки IDE Qt Creator (Qt 5.4.2), применялась система контроля версий Git. Все прецеденты реализованы, сбои и зависания не наблюдаются, реализована очистка памяти, использованы принципы раздельной компиляции, приложены диаграмма вариантов использования, диаграмма классов, диаграммы действий и последовательностей. Цель достигнута, написанная программа очень упростит работу по учету товаров/услуг и ведению сопутствующей финансовой деятельности.