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»

Преподаватель:

канд. физ.-мат. наук, доцент

Бульба Артём Владимирович

Петрозаводск 2022

СПИСОК ИСПОЛНИТЕЛЕЙ

Варов Н.И., 3 курс, группа 21316.

Окоротышев М.В., 3 курс, группа 21316.

Хаухия А.В., 3 курс, группа 21316.

ОГЛАВЛЕНИЕ

[Цель работы. 4](#_Toc123079294)

[Кратко о программной реализации 4](#_Toc123079295)

[**1)** Краткое словесное описание сюжета 5](#_Toc123079296)

[**2)** Описание имеющихся у заказчика материалов 5](#_Toc123079297)

[**3)** Действующие субъекты 6](#_Toc123079298)

[**4)** Диаграмма вариантов использования 6](#_Toc123079299)

[**5)** Описания вариантов использования 6](#_Toc123079300)

[**6)** Диаграммы действий 7](#_Toc123079301)

[**7)** Предварительный список существительных 8](#_Toc123079302)

[**8)** Уточненный список существительных 8](#_Toc123079303)

[**9)** Список атрибутов 9](#_Toc123079304)

[**10)** Список сообщений 10](#_Toc123079305)

[**11)** Диаграмма классов 11](#_Toc123079306)

[**12)** Диаграммы последовательностей 12](#_Toc123079307)

[**13)** Листинг заголовочных файлов 13](#_Toc123079308)

[**14)** Листинг исполнительных файлов 18](#_Toc123079309)

[**15)** Руководство пользователя 26](#_Toc123079310)

[Заключение 29](#_Toc123079311)

# 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

"revenuerow.h", "revenuerow.cpp" – класс RevenueRow

"revenuerecord.h", "revenuerecord.cpp" – класс RevenueRecord

"sellsinputscreen.h", "sellsinputscreen.cpp" – класс SellsInputScreen

"expense.h" – класс Expense

"expenserecord.h", "expenserecord.cpp" – класс ExpenseRecord

"expenseinputscreen.h", "expenseinputscreen.cpp",

"annualreport.h", "annualreport.cpp" – класс AnnualReport

"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. Описание имеющихся у заказчика материалов:

Таблица 1. Список товаров и услуг.

|  |  |  |
| --- | --- | --- |
| Артикул | Название | Цена |
| 1 | Похороны | 9999.99 |
| 2 | Похороны Luxe | 15999.99 |
| 3 | Цветок | 49.99 |
| 4 | Надгробие | 19999.99 |

Таблица 2. Список продаж.

|  |  |  |
| --- | --- | --- |
| Артикул | Кол-во | Дата |
| 1 | 1 | 04.дек |
| 2 | 1 | 12.дек |
| 3 | 15 | 07.янв |
| 4 | 2 | 08.янв |

Таблица 3. Список затрат.

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Получатель | Сумма | Категория |
| 23.фев | ООО Чисто и быстро | 20000 | Уборка |
| 08.мар | ИП Иванов | 10000 | Ремонт |
| 01.апр | ООО Ритуаль | 178900 | Поставка |
| 09.май | ИП Хаухия | 300 | Организация праздника |

1. Действующие субъекты:

Главное действующее лицо – User (пользователь). Он добавляет товары и услуги, кол-во продаж, фиксирует расходы. Использует функции вывода данных.

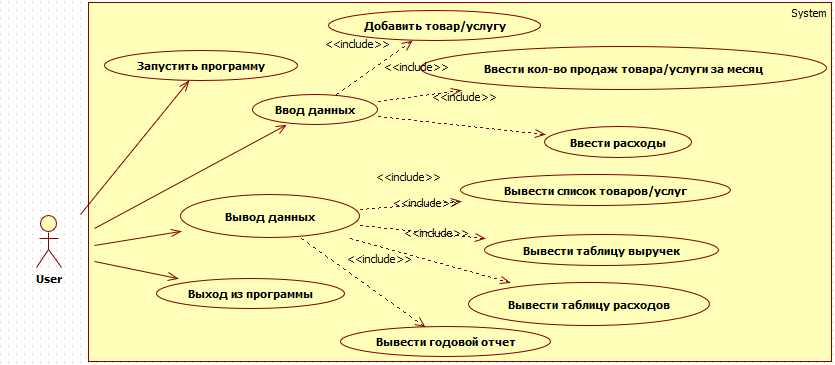
1. Диаграмма вариантов использования:

Рисунок 1 - Диаграмма вариантов использования.

1. Описания вариантов использования:

*-Запустить программу:*При запуске программы на экран выводится меню, из которого пользователь может выбрать нужное действие.

*-Добавить товар/услугу:*На экране отображается сообщение, в котором программа просит пользователя ввести название товара/услуги, его артикул и стоимость. Эта информация заносится в список товаров и услуг.

*-Ввести кол-во продаж товара/услуги за месяц:*Экран ввода кол-ва продаж товара/услуги за месяц содержит сообщение, из которого пользователь узнает, что ему необходимо ввести:1) название товара/услуги;2) месяц за который фиксируется количество продаж;3) количество продаж этого товара/услуги;  
Далее программа вычисляет выручку и если товар заносится впервые, в таблице выручки создается новая строка.

*-Ввести расходы:*Экран ввода расхода содержит приглашение пользователю на ввод категории расходов, размера платежа, дня и месяца, в который производится оплата. Затем программа создает новую строку, содержащую эту информацию, и вставляет ее в таблицу расходов.

-*Вывести список товаров/услуг:*Программа выводит на экран список товаров/услуг, каждая строка списка состоит из трех полей: артикула, названия товара/услуги и цены.

*- Вывести таблицу выручек:*Каждая строка таблицы, которую выводит программа, состоит из артикула товара/услуги и значения выручки с января по декабрь.

*-Вывести таблицу расходов:*Каждая строка таблицы, которую выводит программа, состоит из значений дня и месяца, в который производится оплата, получателя, категории расходов, суммы оплаты.

*-Вывести годовой отчет:*Программа выводит годовой отчет, состоящий из: суммарной выручки за прошедший год;списка всех расходов с указанием категории;общая сумма расходов;результирующего годового баланса (доходы/убытки).

*-Выход из программы:*Происходит закрытие консоли.

1. Диаграммы действий:

Ввести кол-во продаж товара/услуги за месяц:

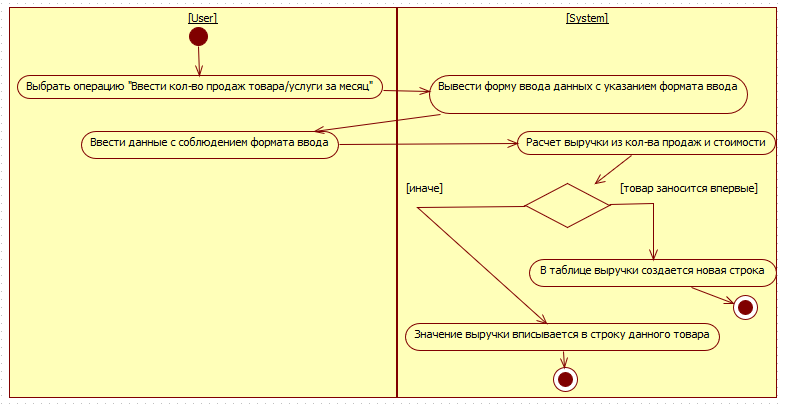


Рисунок 2 - Диаграмма действий.

Добавить товар/услугу:

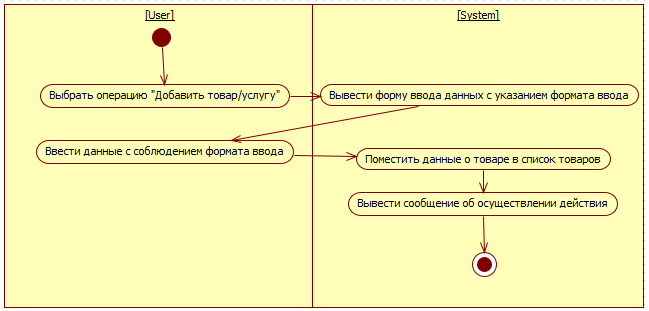


Рисунок 3 - Диаграмма действий.

1. Предварительный список существительных:

1. Экран интерфейса пользователя.

2. Товар.

3. Экран ввода товара.

4. Название товара.

5. Стоимость товара.

6. Список товаров.

7. Количество продаж.

8. Экран ввода количества продаж.

9. Месяц.

10. Выручка.

11. Таблица выручки от продаж.

12. Строка выручки от продаж.

13. Расход.

14. Экран ввода расходов

15. Размер платежа.

16. День и месяц.

17. Получатель

18. Категория расходов.

19. Строка в таблице расходов.

20. Таблица расходов.

21. Годовой отчет.

22. Суммарная арендная плата.

23. Суммарные расходы по категориям.

24. Баланс.

1. Уточненный список существительных:

1. Экран пользовательского интерфейса.

2. Товар.

3. Экран ввода товара.

4. Список товаров.

5. Экран ввода количества продаж.

6. Таблица выручки от продаж.

7. Строка таблицы выручки от продаж.

8. Расход.

9. Экран ввода расходов

10. Таблица расходов.

11. Годовой отчет.

1. Список атрибутов:

**1. 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.  
**2. Класс ProductList.**  
a) setPtrsProd – список указателей на класс Product.  
b) iter – итератор для работы со списком setPtrsProd.  
**3. Класс ProductInputScreen.**  
a) ptrProductList – указатель на список товаров.  
b) pName – название товара.  
c) art – артикул товара.  
d) prc – цена товара.  
**4. Класс RevenueRow.**a) art – артикул товара.   
b) revenue[12] – массив подсчитанной выручки по месяцам.  
c) prc – цена товара.  
**5. Класс RevenueRecord.**  
a) setPtrsRR – список указателей на класс RevenueRow.  
b) iter – итератор для работы со списком setPtrsRR.

**6. Класс SellsInputScreen.**  
a) ptrProductList - указатель на список товаров.  
b) ptrRevenueRecord - указатель на список выручек.  
c) productName - название товара, кол-во продаж которого фиксируем.  
d) amountSold – кол-во продаж.  
e) month – месяц, за который фиксируем количество продаж.  
f) art – артикул товара, кол-во продаж которого фиксируем.  
g) prc - цена товара, кол-во продаж которого фиксируем.  
**7. Класс Expense.**  
a) month – месяц уплаты расходов.  
b) day – день уплаты расходов.  
c) category – категория расходов.  
d) payee – получатель.  
e) amount – сумма, уплаченная за расходы.  
**8. Класс ExpenseRecord.**  
a) vectPtrsExpenses - вектор указателей на расходы.  
b) iter – итератор для работы с вектором vectPtrsExpenses.  
**9. Класс ExpenseInputScreen.**a) ptrExpenseRecord – указатель на вектор расходов.  
**10. Класс AnnualReport.**a) ptrRR – указатель на записи выручек.  
b) ptrER - указатель на записи расходов.  
c) expenses – сумма расходов.  
d) revenues – сумма выручек.  
**11. 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.

1. Список сообщений:

**1. Класс Product.**a) Product() – конструктор.  
b) ~Product() – деструктор.  
c) getArticle() - возвращает артикул товара.  
d) getPrice() – возвращает цену товара.  
e) getName() – возвращает название товара.  
**2. 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.

**3. Класс ProductInputScreen.**a) ProductInputScreen() – конструктор.  
b) setProduct() – добавляет данные о товаре и заносит в список товаров.  
**4. Класс RevenueRow.**a) RevenueRow() – конструктор.  
b) setRevenue() – подсчитывает выручку за месяц и заносит в массив.  
с) getSumOfRow() – подсчитывает сумму выручек с одного товара за год.  
d) getArt() - возвращает артикул товарa.  
e) getRevenueM() – возвращает выручку за выбранный месяц.  
**5. Класс RevenueRecord.**a) ~RevenueRecord() - деструктор  
b) insertRevenue() – добавляет выручку в список.  
c) display() – вывод списка выручек.  
d) getSumOfRevenues() – подсчитывает сумму всех выручек всех товаров.  
**6. Класс SellsInputScreen.**  
a) SellsInputScreen() – конструктор.  
b) setSells() – передача кол-ва продаж для подсчета выручки.  
**7. Класс Expense.**a) Expense() – конструктор.  
**8. Класс ExpenseRecord.**a) ~ExpenseRecord() - деструктор  
b) insertExp() – вставляет в вектор данные о расходах.  
c) display() – вывод таблицы расходов.  
d) displaySummary() – подсчитывает сумму по всем категориям расходов.  
**9. Класс ExpenseInputScreen.**a) ExpenseInputScreen() – конструктор.  
b) setExpense() – создает и добавляет новый расход в вектор расходов.  
**10. Класс AnnualReport.**a) AnnualReport() – конструктор.  
b) display() – вывод годового отчета.  
**11. Класс UserInterface.**a) UserInterface() – конструктор.  
b) ~UserInterface() – деструктор.  
c) interact() – функция для работы пользователя с программой.

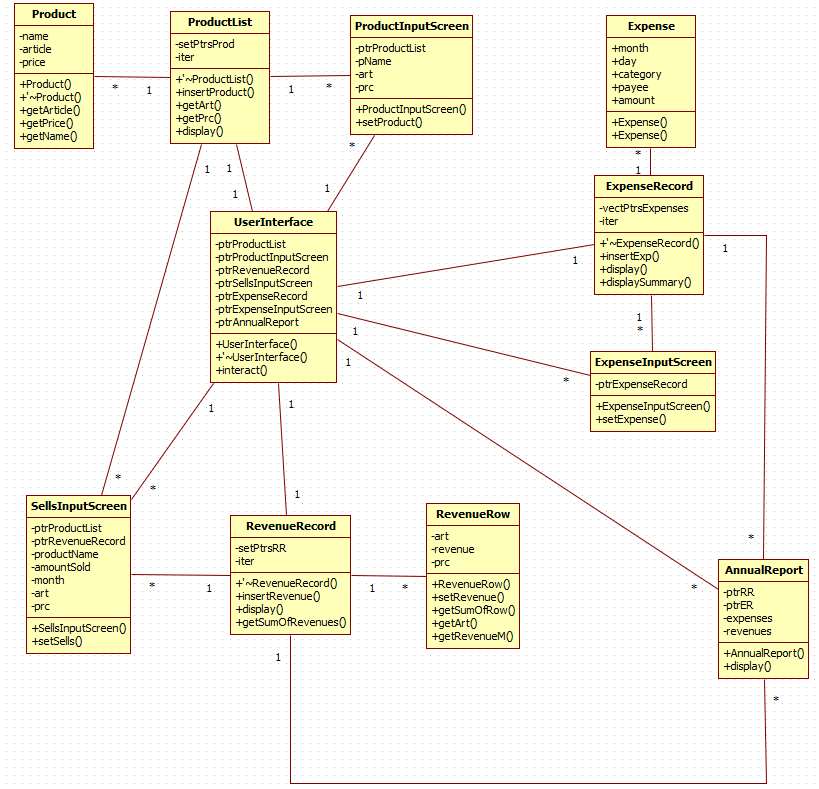
1.  Диаграмма классов:

Рисунок 4 - Диаграмма классов.

1. Диаграммы последовательностей:

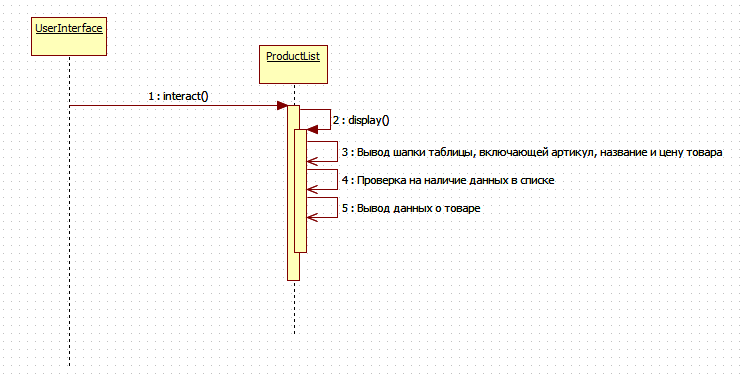
Вывести список товаров/услуг:

Рисунок 5 - Диаграмма последовательностей.

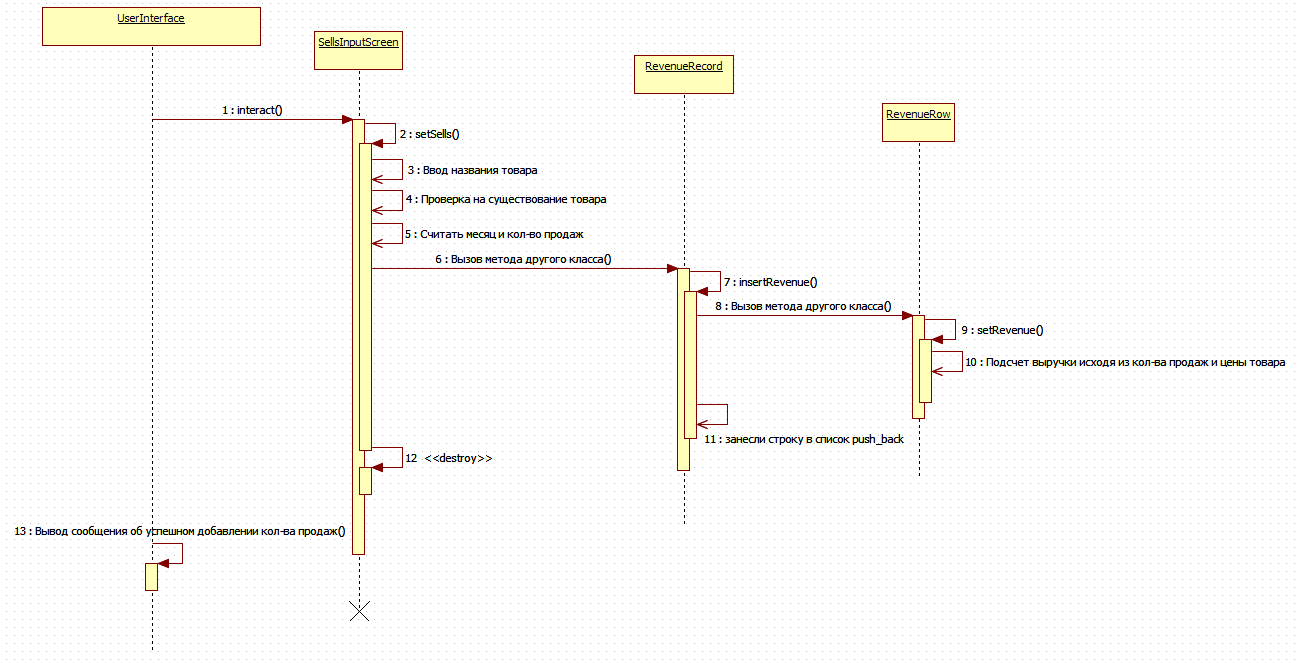
Ввести кол-во продаж товара/услуги за месяц:

Рисунок 6 - Диаграмма последовательностей.

Вывести таблицу выручек:

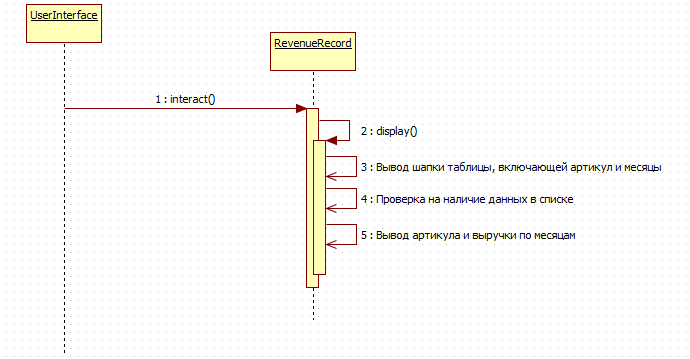


Рисунок 7 - Диаграмма последовательностей.

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

};

**RevenueRow.h**

#pragma once

#include <numeric>

#include <iomanip>

using namespace std;

////////////////////класс RevenueRow//////////////////////////

//класс, хранящий одну табличную строку выручки

// одна строка таблицы прибыли: артикул и 12 выручек по месяцам

class RevenueRow

{

private:

int art; // артикул товара

float revenue[12]; // месяцы

float prc; //стоимость товара

public:

RevenueRow(int); // конструктор с одним параметром

void setRevenue(int, int, float); // подсчитать выручку за месяц

float getSumOfRow(); //подсчитать сумму выручек с одного товара за год

int getArt(); //Запрос артикула товара

float getRevenueM(int); //Запрос выручки за определенный месяц

};

**RevenueRecord.h**

#pragma once

#include <iostream>

#include <list>

#include "revenuerow.h"

//////////////////// класс RevenueRecord ///////////////////////

//класс RevenueRecord. Он хранит непосредственно записи о выручке.

//С ним будет взаимодействовать экран добавления количества продаж.

class RevenueRecord

{

private:

list <RevenueRow\*> setPtrsRR; // список указателей на объекты класса RevenueRow

list <RevenueRow\*>::iterator iter;

public:

~RevenueRecord();

void insertRevenue(int, int, int, float); // добавить выручку

void display(); // отобразить все строки с выручками

float getSumOfRevenues(); // подсчитать сумму всех выручек всех товаров

};

**SellsInputScreen.h**

#pragma once

#include "revenuerecord.h"

#include "productlist.h"

using namespace std;

////////////////////класс SellsInputScreen //////////////////

//Экран для добавления количества продаж

class SellsInputScreen

{

private:

ProductList\* ptrProductList; // список товаров

RevenueRecord\* ptrRevenueRecord; // список записей выручек

string productName; // название товара, кол-во продаж которого фиксируем

int amountSold; // кол-во продаж

int month; // месяц, за который фиксируем количество продаж

int art; // артикул товара, кол-во продаж которого фиксируем

float prc; // цена товара, кол-во продаж которого фиксируем

public:

SellsInputScreen(ProductList\* ptrPL, RevenueRecord\* ptrRR) : ptrProductList(ptrPL), ptrRevenueRecord(ptrRR)

{

/\*тут пусто\*/

}

void setSells(); // добавить количество продаж

};

**Expense.h**

#pragma once

#include <string>

#include <iomanip>

using namespace std;

///////////////////////класс Expense///////////////////////

//Класс затрат

class Expense

{

public:

int month, day; // месяц и день уплаты расходов

string category; // категория расходов (газ, свет, ремонт и тд)

string payee; // кому платим (поставщики газа, света, сервисный центр...)

float amount; // сколько платим

Expense()

{ }

Expense(int m, int d, string c, string p, float a) :

month(m), day(d), category(c), payee(p), amount(a)

{

/\* тут пусто! \*/

}

};

**ExpenseRecord.h**

#pragma once

#include "expense.h"

#include <vector>

#include <iostream>

///////////////////класс ExpenseRecord/////////////////////

//Класс записей о затратах

class ExpenseRecord

{

private:

vector<Expense\*> vectPtrsExpenses; //вектор указателей на расходы

vector<Expense\*>::iterator iter;

public:

~ExpenseRecord();

void insertExp(Expense\*);

void display();

float displaySummary(); // нужно для годового отчета

};

**ExpenseInputScreen.h**

#pragma once

#include "expenserecord.h"

////////////////класс ExpenseInputScreen///////////////////

//Класс для ввода расходов

class ExpenseInputScreen

{

private:

ExpenseRecord\* ptrExpenseRecord; // запись о расходах

public:

ExpenseInputScreen(ExpenseRecord\*);

void setExpense();

};

**AnnualReport.h**

#pragma once

#include "expenserecord.h"

#include "RevenueRecord.h"

#include <iomanip>

//////////////////класс AnnualReport///////////////////////

//Класс годового отчета

class AnnualReport

{

private:

RevenueRecord\* ptrRR; // записи выручек

ExpenseRecord\* ptrER; // записи расходов

float expenses, revenues; // суммы выручек и расходов

public:

AnnualReport(RevenueRecord\*, ExpenseRecord\*);

void display(); // отображение годового отчета

};

**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

}

**RevenueRow.cpp**

#include "revenuerow.h"

/////////////////методы класса RevenueRow/////////////////////

RevenueRow::RevenueRow(int ar) : art(ar) //конструктор

{ //Алгоритм fill() помещает копию значения value (у нас это 0)

//в каждый элемент диапазона, ограниченного парой итераторов [first,last).

//Т.е. в конструкторе просто инициализируем массив значениями 0.

fill(&revenue[0], &revenue[12], 0);

}

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

void RevenueRow::setRevenue(int m, int am, float pr) // сеттер выручка за месяц m, выручка - pr \* am

{

revenue[m] = pr \* am; // привязываем оплату к месяцу

}

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

int RevenueRow::getArt() // геттер запрос артикула

{

return art;

}

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

float RevenueRow::getRevenueM(int month) //Геттер запрос выручки за месяц month

{

return revenue[month];

}

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

float RevenueRow::getSumOfRow() // cумма выручек в строке

{ //По умолчанию алгоритм accumulate() суммирует элементы.

//Нужно указать точку старта, конечную точку и значение от которого начинаем прибавлять.

float a = accumulate(&revenue[0], &revenue[12], revenue[0]);

return a;

}

**RevenueRecord.cpp**

#include "revenuerecord.h"

/////////////////методы класса RevenueRecord//////////////////

RevenueRecord::~RevenueRecord() // деструктор

{ // удалить строки с выручками,

// удалить указатели из множества.

while (!setPtrsRR.empty())

{

iter = setPtrsRR.begin();

delete\* iter;

setPtrsRR.erase(iter);

}

}

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

void RevenueRecord::insertRevenue(int art, int month, int amount, float prc)

{

iter = setPtrsRR.begin(); // Инициализация итератора

while (iter != setPtrsRR.end()) // условие выхода

{ // если текущий объект совпадает с созданным для поиска,

if (art == (\*iter)->getArt())

{ // заносим выручку в список

(\*iter)->setRevenue(month, amount, prc);

return;

}

else

iter++;

} // если не нашли строку - создаем новую

RevenueRow\* ptrRow = new RevenueRow(art);

ptrRow->setRevenue(month, amount, prc); // заносим данные для подсчета выручки

setPtrsRR.push\_back(ptrRow); // заносим строку в список

}

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

void RevenueRecord::display() // отобразить все строки с выручками

{

cout << setw(10) << left << "\nArticle" << setw(8) << left << "Jan" << setw(8) << left << "Feb"

<< setw(8) << left << "Mar" << setw(8) << left << "Apr" << setw(8) << left << "May"

<< setw(8) << left << "June" << setw(8) << left << "July" << setw(8) << left << "Aug"

<< setw(8) << left << "Sep" << setw(8) << left << "Oct" << setw(8) << left << "Nov"

<< setw(8)<< left << "Dec" << "\n" << endl

<< "----------------------------------------------------------------------------------------------------\n" << endl;

if (setPtrsRR.empty())

cout << "\*\*\*No recorded sales\*\*\*\n" << endl;

else

{

iter = setPtrsRR.begin(); //итератор на список с указателями на объекты RevenueRow

while (iter != setPtrsRR.end())

{

cout << setw(9) << left << (\*iter)->getArt(); // вывести артикул товара

for (int j = 0; j < 12; j++) // вывести 12 значений выручки

{

if (((\*iter)->getRevenueM(j)) == 0)

cout << setw(8) << left << "0";

else

cout << setw(7) << left << (\*iter)->getRevenueM(j) << " ";

}

cout << endl;

iter++;

}

cout << endl;

cout << endl;

}

}

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

float RevenueRecord::getSumOfRevenues() // сумма всех выручек всех товаров

{

float sumRevenues = 0.0;

iter = setPtrsRR.begin();

while (iter != setPtrsRR.end())

{

sumRevenues += (\*iter)->getSumOfRow(); //суммирование выручек за все месяцы с разных товаров

iter++;

}

return sumRevenues;

}

**SellsInputScreen.cpp**

#include "sellsinputscreen.h"

#include "userinterface.h"

/////////////////методы класса SellsInputScreen/////////////

void SellsInputScreen::setSells()

{

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

getaLine(productName);

// получить артикул и стоимость по названию товара

art = ptrProductList->getArt(productName);

prc = ptrProductList->getPrc(productName);

if (art > 0) // если товар найден -

{ // получить количество продаж

cout << "Enter the month number (1-12): " << endl;

cin >> month;

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

month--; // внутренняя нумерация 0-11, тк используется массив

cout << "Enter the volume of sales in this month (10): " << endl;

cin >> amountSold; // вводим количество продаж

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

// вставляем данные для подсчета выручки

ptrRevenueRecord->insertRevenue(art, month, amountSold, prc);

}

else

cout << "No such product has been added yet\n" << endl;

}

**ExpenseRecord.cpp**

#include "expenserecord.h"

//////////////////методы класса ExpenseRecord//////////////

ExpenseRecord::~ExpenseRecord() // деструктор

{ // удалить объекты expense

// удалить указатели на вектор

while (!vectPtrsExpenses.empty())

{

iter = vectPtrsExpenses.begin();

delete\* iter;

vectPtrsExpenses.erase(iter);

}

}

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

void ExpenseRecord::insertExp(Expense\* ptrExp)

{

vectPtrsExpenses.push\_back(ptrExp);

}

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

void ExpenseRecord::display() // распечатываем все расходы

{

cout << "\n" << setw(8) << left << "Date" << setw(15) << left << "Recipient" << setw(8) << left << "Sum"<< setw(12) << left << "Category" << endl;

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

if (vectPtrsExpenses.size() == 0) // В контейнере нет расходов

cout << "\*\*\*There are no expenses\*\*\*\n" << endl;

else

{

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{ // распечатываем все расходы

cout << setw(2) << left << (\*iter)->day << '/' << setw(6) << left << (\*iter)->month << setw(15) << left << (\*iter)->payee << setw(8) << left << (\*iter)->amount << setw(12) << left << (\*iter)->category << endl;

iter++;

}

cout << endl;

}

}

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

// используется при составлении годового отчета

float ExpenseRecord::displaySummary()

{

float totalExpenses = 0; // Сумма по всем категориям расходов

if (vectPtrsExpenses.size() == 0)

{

return 0;

}

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{

totalExpenses += (\*iter)->amount; //подсчитываем все расходы

iter++;

}

return totalExpenses;

}

**ExpenseInputScreen.cpp**

#include "expenseinputscreen.h"

#include "userinterface.h"

//////////////методы класса ExpenseInputScreen/////////////

// конструктор

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

{

/\*пусто\*/

}

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

void ExpenseInputScreen::setExpense()

{

int month, day;

string category, payee;

float amount;

cout << "Enter the month (1-12): ";

cin >> month;

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

cout << "Enter the day (1-31): ";

cin >> day;

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

cout << "Enter the expense category (Repairs, Taxes): ";

getaLine(category);

cout << "Enter the recipient (OOO VarovComp): ";

getaLine(payee);

cout << "Enter the sum (39.95): ";

cin >> amount;

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

// создаем новый расход

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

// вставляем расход в список всех расходов

ptrExpenseRecord->insertExp(ptrExpense);

}

**AnnualReport.cpp**

#include "annualreport.h"

////////////////методы класса AnnualReport/////////////////

//Конструктор

AnnualReport::AnnualReport(RevenueRecord\* pRR, ExpenseRecord\* pER) : ptrRR(pRR), ptrER(pER)

{ /\* пусто \*/

}

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

void AnnualReport::display()

{

cout << setw(20) <<"Annual report" << endl;

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

cout << setw(18) << left << "Sales revenue: ";

revenues = ptrRR->getSumOfRevenues();

cout << setw(10) << left << revenues << endl;

//cout << "Expenses\n" << endl;

expenses = ptrER->displaySummary();

cout << setw(19) << left << "\nExpences total: ";

cout << setw(10) << left << expenses << endl;

// вычисляем прибыльность

cout << setw(19) << left << "\nBalance:" << setw(10) << left << (revenues - expenses) << "\n" << endl;

}

**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()

**Main.cpp**

#include "userinterface.h"

int main()

{

UserInterface theUserInterface;

theUserInterface.interact();

return 0;

}

1. Руководство пользователя:

При запуске программы пользователю предоставляется выбор:



-Ввести данные;

-Вывести данные;

-Выйти из программы;

Если выбрано “Ввести данные”, выводится:

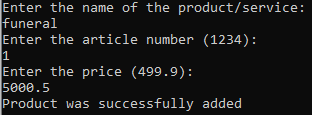


- Добавить товар/услугу;

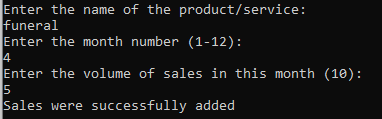
- Записать кол-во продаж;

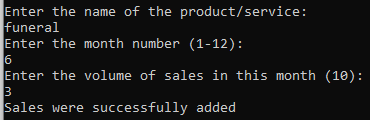
-Записать расходы;

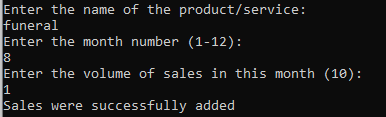
При добавлении товара/услуги, пользователь должен ввести название, артикул и цену:



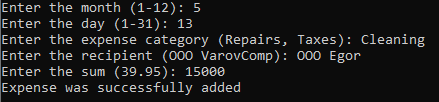
При записи кол-ва продаж, пользователь должен ввести название существующего товара/услуги, номер месяца, за который фиксируем продажи, и кол-во продаж:



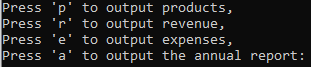




При записи расходов, пользователь должен ввести месяц и день уплаты расходов, категорию, получателя и сумму:



Если выбрано “Вывести данные”, выводится:



-Вывести список товаров/услуг;

-Вывести список выручек;

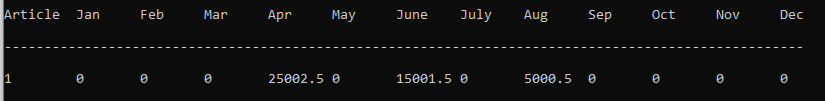
-Вывести список расходов;

-Вывести годовой отчет;

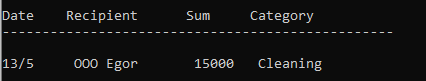
При выводе списка товаров/услуг, будет выведено следующее:



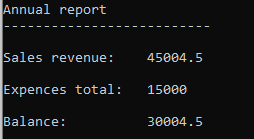
При выводе списка выручек, будет выведено следующее:

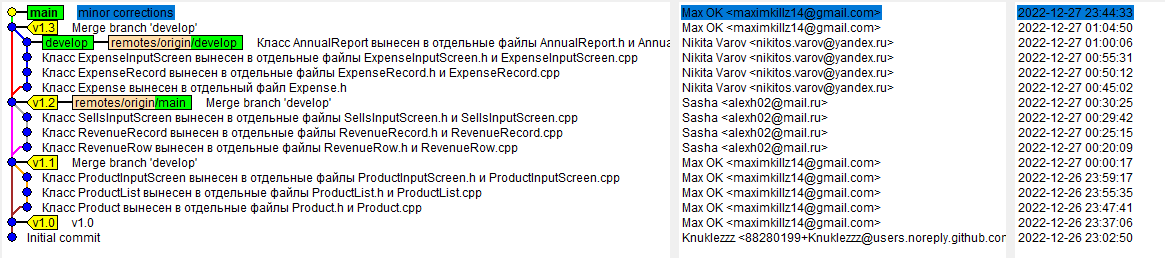


При выводе списка расходов, будет выведено следующее:



При выводе годового отчета, будет выведено следующее:



Если выбрано “Выйти из программы”, консоль закроется.

# 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.