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»

Выполнил:

студент группы 21316

Хаухия Александр

Преподаватель:

канд. физ.-мат. наук, доцент

Бульба Артём Владимирович

Петрозаводск 2022

ОГЛАВЛЕНИЕ

[Цель работы: 3](#_Toc123143359)

[Кратко о программной реализации: 3](#_Toc123143360)

[**1)** Краткое словесное описание сюжета: 3](#_Toc123143361)

[**2)** Предварительный список существительных: 3](#_Toc123143362)

[**3)** Уточненный список существительных: 4](#_Toc123143363)

[**4)** Список атрибутов: 4](#_Toc123143364)

[**5)** Список сообщений: 4](#_Toc123143365)

[**6)** Листинг заголовочных файлов: 5](#_Toc123143366)

[**7)** Листинг исполнительных файлов: 6](#_Toc123143367)

[Заключение: 9](#_Toc123143368)

# 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:

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

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

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

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. Количество продаж.

2. Экран ввода количества продаж.

3. Месяц.

4. Выручка.

5. Таблица выручки от продаж.

6. Строка выручки от продаж.

1. Уточненный список существительных:

1. Экран ввода количества продаж.

2. Таблица выручки от продаж.

3. Строка таблицы выручки от продаж.

1. Список атрибутов:

**1. Класс RevenueRow.**a) art – артикул товара.   
b) revenue[12] – массив подсчитанной выручки по месяцам.  
c) prc – цена товара.  
**2. Класс RevenueRecord.**  
a) setPtrsRR – список указателей на класс RevenueRow.  
b) iter – итератор для работы со списком setPtrsRR.

**3. 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..

1. Список сообщений:

**1. 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.  
**2. Класс RevenueRecord.**a) ~RevenueRecord() - деструктор  
b) insertRevenue() – добавляет выручку в список.  
c) display() – вывод списка выручек.  
d) getSumOfRevenues() – подсчитывает сумму всех выручек всех товаров.  
**3. Класс SellsInputScreen.**  
a) SellsInputScreen() – конструктор.  
b) setSells() – передача кол-ва продаж для подсчета выручки.

1. Листинг заголовочных файлов:

**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

};

1. Листинг исполнительных файлов:

**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

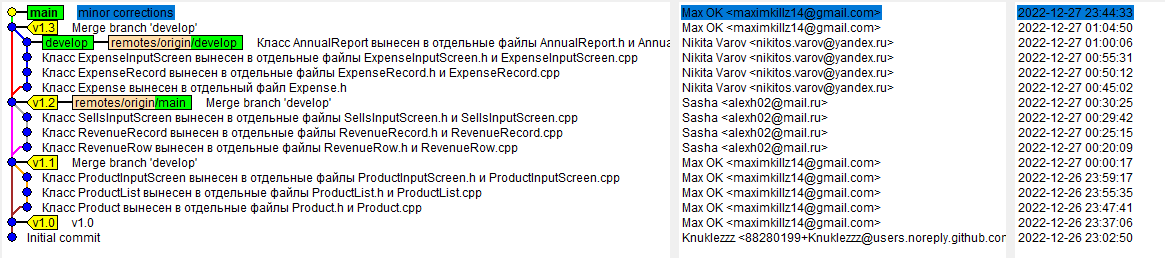
ptrRevenueRecord->insertRevenue(art, month, amountSold, prc);

}

else

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

}



# 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.