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»

Работу выполнил:

Варов Н.И.

Преподаватель:

Бульба А.В.

Петрозаводск 2022

ОГЛАВЛЕНИЕ

[Цель работы. 3](#_Toc123145214)

[Кратко о программной реализации: 3](#_Toc123145215)

[**1)** Краткое словесное описание сюжета: 3](#_Toc123145216)

[**2)** Предварительный список существительных: 3](#_Toc123145217)

[**3)** Уточненный список существительных: 4](#_Toc123145218)

[**4)** Список атрибутов: 4](#_Toc123145219)

[**5)** Список сообщений: 4](#_Toc123145220)

[**6)** Листинг заголовочных файлов: 4](#_Toc123145221)

[**7)** Листинг исполнительных файлов: 6](#_Toc123145222)

[Заключение 8](#_Toc123145223)

# 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:

"expense.h" – класс Expense

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

"expenseinputscreen.h", "expenseinputscreen.cpp" – класс ExpenseInputScreen

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

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. Расход.

2. Экран ввода расходов

3. Размер платежа.

4. День и месяц.

5. Получатель

6. Категория расходов.

7. Строка в таблице расходов.

8. Таблица расходов.

9. Годовой отчет.

10. Суммарная арендная плата.

11. Суммарные расходы по категориям.

12. Баланс.

1. Уточненный список существительных:

1. Расход.

2. Экран ввода расходов

3. Таблица расходов.

4. Годовой отчет.

1. Список атрибутов:

**1. 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.  
**2. Класс ExpenseRecord.**  
a) vectPtrsExpenses - вектор указателей на расходы.  
b) iter – итератор для работы с вектором vectPtrsExpenses.  
**3. Класс ExpenseInputScreen.**a) ptrExpenseRecord – указатель на вектор расходов.  
**4. Класс AnnualReport.**a) ptrRR – указатель на записи выручек.  
b) ptrER - указатель на записи расходов.  
c) expenses – сумма расходов.  
d) revenues – сумма выручек.

1. Список сообщений:

**1. Класс Expense.**a) Expense() – конструктор.  
**2. 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.  
**3. Класс ExpenseInputScreen.**a) ExpenseInputScreen() – конструктор.  
b) setExpense() – создает и добавляет новый расход в вектор расходов.  
**4. Класс AnnualReport.**a) AnnualReport() – конструктор.  
b) display() – вывод годового отчета.

1. Листинг заголовочных файлов:

**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

};

1. Листинг исполнительных файлов:

**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();

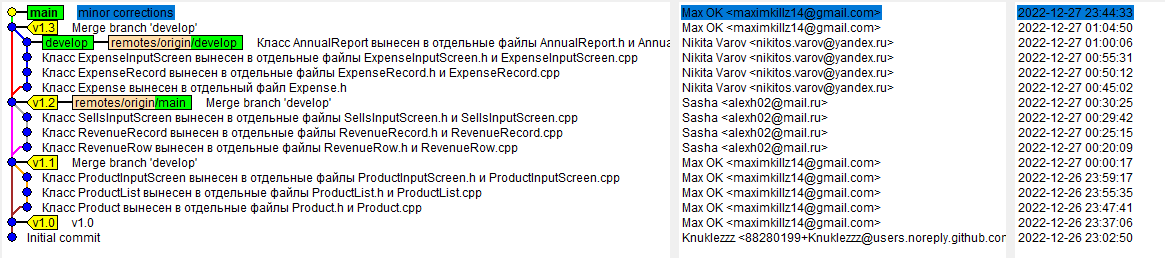
cout << setw(19) << left << "\nExpences total: ";

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

// вычисляем прибыльность

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

}



# 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.