

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

ТРАНСПОРТНАЯ КОМПАНИЯ

отчет по лабораторной работе

Автор работы:
студент группы 21316
_____ Д. А. Сорокин
«26» декабря 2022 г.

Научный руководитель:
канд. физ.-мат. наук, доцент
_____ А. В. Бульба
«26» декабря 2022 г.

Петрозаводск
2022 год

Цель: Рассмотреть простой пример выполнения заказа на разработку программного обеспечения и, в качестве самостоятельного задания, выполнить аналогичный проект (в контексте транспортной компании).

Ход разработки:

Функционал программы позволяет:

- Записывать ежемесячные доходы, которые приносит каждый водитель компании
- Вывести таблицу с доходами водителей по месяцам

Описание имеющихся у заказчика материалов:

Доход от водителей.

Номер рейса	Янв	Фев	Март	Апр	Май	Июнь	Июль	Авг	Сен	Окт	Нояб	Дек
101	0	0	2000	2250	0	0	0	0	0	0	0	0
402	1200	2600	600	2200	2100	0	2420	1630	0	0	0	0
576	0	0	0	0	0	0	0	0	0	0	1200	2150
368	0	0	0	0	0	0	0	0	0	600	1600	1840
204	0	0	0	0	0	0	0	0	0	400	890	1360

Кратко о программной реализации:

Для реализации поставленной задачи была использована среда программирования Qt Creator 5.4.2, в качестве языка программирования был выбран C++. В процессе работы над проектом мною были созданы заголовочные файлы:

1. RevRecord.h
2. FlightRevenue.h
3. RevInputScreen.h

Список атрибутов:

- 1) RevRecord
 - a) setPtrsRR - указателей на список доходов
 - b) iter – итератор для работы со списком доходов
- 2) RevInputScreen
 - a) ptrDriversList - указатель на список водителей
 - b) ptrRevRecord - указатель на список доходов
 - c) revName - имя водителя
 - d) revPaid – оплата

- e) month - месяц
- f) FlightNo – номер рейса
- 3) FlightRevenue
 - a) FlightNo – номер рейса
 - b) months[12] – массив полученной прибыли по месяцам
- 4) UserInterface
 - a) ptrRevRecord - указатель на класс записей дохода
 - b) ptrRevInputScreen - указатель на класс экран добавления дохода

Список сообщений:

- 1) RevRecord
 - a) ~RevRecord() - деструктор
 - b) insertRev() – добавление дохода в список
 - c) display() – вывод списка доходов
 - d) getSumOfRev() – итоговое суммирование доходов по всем водителям
- 2) RevInputScreen
 - a) RevInputScreen() – конструктор
 - b) setRev() – добавляет доход с одного водителя за месяц
- 3) FlightRevenue
 - a) FlightRevenue() – конструктор
 - b) setRev() – устанавливает доход за месяц
 - c) getSumOfRev() – суммирование доходов по каждому водителю
 - d) getFlightNo() – возвращает номер рейса
 - e) getRevNo() – возвращает доход за месяц

Листинг заголовочных файлов:

UserInterface.h

```
//UserInterface.h

#ifndef USERINTERFACE
#define USERINTERFACE

#include <iostream>
#include <list>
#include <vector>
#include <string>
#include <numeric> //для accumulate()
#include "Drivers.h"
#include "DriversList.h"
#include "DriversInputScreen.h"
#include "FlightRevenue.h"
#include "RevRecord.h"
#include "RevInputScreen.h"
#include "Expense.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:
    DriversList* ptrDriversList;
    DriversInputScreen* ptrDriversInputScreen;
    RevRecord* ptrRevRecord;
    RevInputScreen* ptrRevInputScreen;
    ExpenseRecord* ptrExpenseRecord;
    ExpenseInputScreen* ptrExpenseInputScreen;
    AnnualReport* ptrAnnualReport;

    char ch;
public:
    UserInterface();
    ~UserInterface();
    void interact();
};
#endif // USERINTERFACE

```

RevRecord.h

```

//RevRecord.h
#ifndef REVRECORD
#define REVRECORD
#include "tc.h"
////////// класс RevRecord //////////
//класс RevRecord. Он хранит непосредственно записи доходов.
//С ним будет взаимодействовать экран добавления дохода.
class RevRecord
{
private:
    list <FlightRevenue*> setPtrsRR; // указатели на объекты FlightRevenue
    (по одному на водителя)
    list <FlightRevenue*>::iterator iter;
public:
    ~RevRecord();
    void insertRev(int, int, float); // добавить плату
    void display(); // отобразить все строки с платами
    float getSumOfRev(); // подсчитать сумму всех платежей всех водителей

```

```
};
#endif // REVRECORD
```

RevInputScreen.h

```
// RevInputScreen.h
#ifndef REVINPUTSCREEN
#define REVINPUTSCREEN
#include "tc.h"
////////////////////////класс RevInputScreen //////////////////////////
//Экран для добавления платы
class RevInputScreen
{
private:
    DriversList* ptrDriversList; // список водителей
    RevRecord* ptrRevRecord; // список записей об оплате
    string revName; // имя водителя, который приносит прибыль
    float revPaid; // плата
    int month; // за месяц
    int FlightNo; // по рейсам
public:
    RevInputScreen(DriversList* ptrTL, RevRecord* ptrRR) :
ptrDriversList(ptrTL),
    ptrRevRecord(ptrRR)
    {
        /*тут пусто*/
    }
    void setRev(); // добавить доход с одного водителя за месяц
};
#endif // REVINPUTSCREEN
```

FlightRevenue.h

```
//FlightRevenue.h
#ifndef FLIGHTREVENUE_H
#define FLIGHTREVENUE_H
#include "tc.h"
////////////////////////класс FlightRevenue////////////////////////
//класс, хранящий одну табличную строку доходов с рейса
// одна строка таблицы прибыли: рейс и 12 доходов по месяцам
class FlightRevenue
{
private:
    int FlightNo; // рейсы, за которые уплачено
    float months[12]; // месяцы
public:
    FlightRevenue(int); // конструктор с одним параметром
    void setRev(int, float); // добавить плату за месяц
```

```

        //сумма платежей из одной строки (прибыль с одного водителя за все
        месяцы)

        float getSumOfRev();

        int getFlightNo(); //Запрос номера рейса

        float getRevNo(int); //Запрос дохода за месяц int
    };
#endif

```

Листинг исходных файлов:

UserInterface.cpp

```

//UserInterface.cpp

#include <iostream>

#include "UserInterface.h"

void getaLine(string& inStr) // получение строки текста
{
    char temp[21];
    cin.get(temp, 20, '\n');
    cin.ignore(20, '\n'); //число пропускаемых символов и символ разделения
    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()
{

```

```

ptrDriversList = new DriversList;
ptrRevRecord = new RevRecord;
ptrExpenseRecord = new ExpenseRecord;
}

UserInterface::~UserInterface()
{
    delete ptrDriversList;
    delete ptrRevRecord;
    delete ptrExpenseRecord;
}

void UserInterface::interact()
{
    while (true)
    {
        cout << "\n Click to add a driver 'a', \n"    //Нажмите для добавления водителя
            << " Click to record the driver's income 'b', \n" //Нажмите для записи дохода водителя
            << " Click to record expenses 'c', \n"          //Нажмите для записи расходов
            << " Click to withdraw drivers 'd', \n"          //Нажмите для вывода водителей
            << " Click to withdraw revenue 'e', \n"          //Нажмите для вывода доходов
            << " Click to display expenses 'f', \n"          //Нажмите для вывода расходов
            << " Click to display the annual report 'g', \n" //Нажмите для вывода годового отчета
            << " Click to exit the program 'q'\n";    //Нажмите для выхода из программы

        ch = getaChar();

        switch (ch)
        {
            case 'a': ptrDriversInputScreen = new DriversInputScreen(ptrDriversList);

                ptrDriversInputScreen->setDriver();

                delete ptrDriversInputScreen;

```



```

        break;

    case 'b': ptrRevInputScreen = new RevInputScreen(ptrDriversList, ptrRevRecord);
        ptrRevInputScreen->setRev();
        delete ptrRevInputScreen;
        break;

    case 'c': ptrExpenseInputScreen = new ExpenseInputScreen(ptrExpenseRecord);
        ptrExpenseInputScreen->setExpense();
        delete ptrExpenseInputScreen;
        break;

    case 'd': ptrDriversList->display();
        break;

    case 'e': ptrRevRecord->display();
        break;

    case 'f': ptrExpenseRecord->display();
        break;

    case 'g':
        ptrAnnualReport = new AnnualReport(ptrRevRecord, ptrExpenseRecord);
        ptrAnnualReport->display();
        delete ptrAnnualReport;
        break;

    case 'q':
        cout << "The program is completed"; // Программа завершена
        return;
        break;

    default: cout << "Unknown output function\n"; //Неизвестная функция вывода
        break;

    } // конец switch
} // конец while
} // конец interact()

```

RevRecord.cpp

```
//RevRecord.cpp

#include "UserInterface.h"

RevRecord::~RevRecord() // деструктор
{
    // удалить строки с платежами,
    // удалить указатели из множества.
    while (!setPtrsRR.empty())
    {
        iter = setPtrsRR.begin();

        delete* iter;

        setPtrsRR.erase(iter);
    }
}

void RevRecord::insertRev(int FlightNo, int month, float amount)
{
    iter = setPtrsRR.begin(); // Инициализация итератора
    while (iter != setPtrsRR.end()) // условие выхода
    {
        // если текущий объект совпадает с созданным для поиска,
        if (FlightNo == (*iter)->getFlightNo())
        {
            // заносим доход в список
            (*iter)->setRev(month, amount);

            return;
        }
        else
            iter++;
    }
    // если не нашли строку - создаем новую
    FlightRevenue* ptrRow = new FlightRevenue(FlightNo);
    ptrRow->setRev(month, amount); // заносим в нее платеж
```

```

    setPtrsRR.push_back(ptrRow); // заносим строку в вектор
}

void RevRecord::display() // отобразить все строки с доходами
{
    cout << "\nNumber\tJan Feb Mar Apr May June July Aug Sept Oct Nov Dec\n" << endl
        << "-----\n" << endl;
    if (setPtrsRR.empty())
        cout << "***No income***\n" << endl;
    else
    {
        iter = setPtrsRR.begin(); // итератор на список с указателями на объекты FlightRevenue
        while (iter != setPtrsRR.end())
        {
            cout << (*iter)->getFlightNo() << '\t'; // вывести номер рейса
            for (int j = 0; j < 12; j++) // вывести доходы по месяцам
            {
                if (((*iter)->getRevNo(j)) == 0)
                    cout << " 0 ";
                else
                    cout << (*iter)->getRevNo(j) << " ";
            }
            cout << endl;
            iter++;
        }
        cout << endl;
        cout << endl;
    }
}

```

```

float RevRecord::getSumOfRev() // сумма всех платежей
{
    float sumRevs = 0.0;
    iter = setPtrsRR.begin();
    while (iter != setPtrsRR.end())
    { // плюсуем суммы всех доходов водителей за все время
        sumRevs += (*iter)->getSumOfRev();
        iter++;
    }
    return sumRevs;
}

```

RevInputScreen.cpp

```

//RevInputScreen.cpp
#include "UserInterface.h"

void RevInputScreen::setRev()
{
    cout << "Enter the driver's name: ";
    getaLine(revName);
    // получить номер рейса по имени водителя
    FlightNo = ptrDriversList->getFlightNo(revName);
    if (FlightNo > 0) // если имя найдено, и такой водитель существует -
    { // получить сумму платежа
        cout << "Enter the amount of income (800.50): " << endl; // Введите сумму дохода
(800.50)
        cin >> revPaid; // вводим цену
        cin.ignore(80, '\n');
        cout << "Enter the payment month number (1-12): " << endl; // Введите номер месяца
оплаты (1-12)
        cin >> month;
    }
}

```

```

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

month--; // (внутренняя нумерация 0-11)

// вставляем сумму в запись о доходе

ptrRevRecord->insertRev(FlightNo, month, revPaid);

}

else

    cout << "There is no such driver.\n" << endl; //Такого водителя нет

```

FlightRevenue.cpp

```

//FlightRevenue.cpp

#include "UserInterface.h"

FlightRevenue::FlightRevenue(int an) : FlightNo(an) //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(&months[0], &months[12], 0);
}

void FlightRevenue::setRev(int m, float am) // сеттер доход за месяц m, сумма - am
{
    months[m] = am; // привязываем оплату к месяцу
}

int FlightRevenue::getFlightNo() // геттер запрос номера рейса
{
    return FlightNo;
}

float FlightRevenue::getRevNo(int month) //Геттер запрос дохода за месяц month

```

```
{  
    return months[month];  
}
```

```
float FlightRevenue::getSumOfRev() // сумма дохода в строке  
{ //По умолчанию алгоритм accumulate() суммирует элементы.  
    //Нужно указать точку старта, конечную точку и значение от которого начинаем прибавлять.  
    //Чаще всего это ноль, но может быть и результат других вычислений.  
    return accumulate(&months[0], &months[12], 0);  
}
```

Руководство пользователя:

При запуске программы выводится меню:

```
Click to add a driver 'a',  
Click to record the driver's income 'b',  
Click to record expenses 'c',  
Click to withdraw tenants 'd',  
Click to withdraw revenue 'e',  
Click to display expenses 'f',  
Click to display the annual report 'g',  
Click to exit the program 'q'.
```

При выборе «b» Заполнить доходы водителей, пользователю будет предложено ввести имя водителя, сумму, которую он принес компании, и месяц, в котором был доход. После выведется меню.

```
b  
Enter the driver's name: aa  
Enter the amount of income (800.50):  
120  
Enter the payment month number (1-12):  
1
```

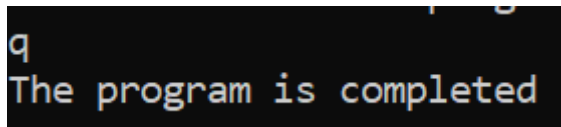
При выборе «е» Вывести доходы, пользователю будет выведена таблица с ежемесячным доходом от водителей, которые учитываются по номеру рейса водителя. После выведется меню.



The screenshot shows a terminal window with a dark background. At the top, the letter 'e' is displayed. Below it is a table with 12 columns representing months from January to December. The first row is the header, and the second row contains numerical values. A dashed line separates the header from the data row.

Number	Jan	Feb	Mar	Apr	May	June	July	Aug	Sept	Oct	Nov	Dec
11	120	0	0	0	0	0	0	0	0	0	0	0

При выборе 'q' Выход из программы, программа завершается, пользователя уведомят об этом.



The screenshot shows a terminal window with a dark background. The letter 'q' is displayed at the top, followed by the text 'The program is completed'.

Заключение:

Использовалась среда разработки Qt Creator 5.4.2, применялась система контроля версий, все прецеденты реализованы, сбои и зависания не наблюдаются, реализована очистка памяти, использованы принципы раздельной компиляции, приложены диаграммы: прецедентов, действий, классов, последовательностей.