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

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

Петрозаводск
2022 год

СПИСОК ИСПОЛНИТЕЛЕЙ:

Клевцов В. Д. группа 21316

Сорокин Д. А. группа 21316

Шевцов М. Н. группа 21316

Содержание

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

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 созданы заголовочные файлы:

1. AnnualReport.h
2. Drivers.h
3. DriversInputScreen.h
4. DriversList.h
5. Expense.h
6. ExpenseInputScreen.h
7. ExpenseRecord.h
8. FlightRevenue.h
9. RevInputScreen.h
10. RevRecord.h
11. UserInterface.h

Ход разработки:

1) Краткое словесное описание сюжета:

Сфера транспортных услуг включает в себя не только большие компании, но и маленькие, которые занимаются перевозками по городу или за его пределами. Такие компании обычно обладают небольшим количеством сотрудников, программа TransportCompany поможет снизить нагрузку с персонала.

Функционал программы позволяет:

- вводить имя водителя, а также номер рейса, который присваивается водителю
- записывать ежемесячные доходы, которые приносит каждый водитель компании
- учитывать расходы, а именно дату, категорию, получателя и сумму

Исходя из данных, которые пользователь занесет в ходе работы, программа сможет вывести:

- список всех водителей компании с их номером рейса, фамилией и именем
- таблицу с доходами водителей по месяцам
- расходы водителей за весь период, с описанием каждого из них
- годовой отчет, который включает в себя все доходы и убытки в течение года. Также, пользователь может вывести отчет не только за год, но и за любой промежуток времени, начиная с начала года

Нельзя не отметить тот факт, что при подготовке годового отчета шанс допустить ошибку снижается, так как все подсчеты ведутся программой.

2) Описание имеющихся у заказчика материалов:

Таблица 1. Список водителей.

Номер рейса	Имя водителя
101	Виктор Хмурый
402	Иван Кузнецов
576	Кирилл Золов
368	Егор Лоскутов
204	Илья Мясников

Таблица 2. 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

Таблица 3. Годовой отчет

Доходы	45000
Расходы	14000
Итого	31000

3) Actors:

В качестве действующего лица будет выступать диспетчер, он будет добавлять водителей, следить за их прибылью в течение месяца и получать годовой отчет.

4) Диаграмма вариантов использования:

5) Описание каждого варианта использования:

Запуск программы:

Когда запускается программа, на экран должно выводиться меню, из которого пользователь.
может выбрать нужное действие. Это может называться экраном пользовательского интерфейса.

Добавить водителя:

На экране должно отобразиться сообщение, в котором программа просит пользователя ввести имя водителя и номер рейса. Эта информация должна заноситься в таблицу.

Ввести данные о рейсе:

Экран ввода данных о рейсе содержит сообщение, из которого пользователь узнает, что ему необходимо ввести имя водителя, месяц рейса, а также полученную сумму денег.
Программа просматривает список водителей и по номеру рейса находит соответствующую запись в таблице доходов от рейса. Если водитель впервые выходит в рейс, в этой таблице создается новая строка и указанная сумма заносится в столбец того месяца, а котором производится рейс. В противном случае значение вносится в существующую строку.

Ввести расходы:

Экран ввода расхода должен содержать приглашение пользователю на ввод имени получателя (или названия организации), суммы оплаты, дня и месяца, в который производится оплата, категории расходов. Затем программа создает новую строку, содержащую эту информацию, и вставляет ее в таблицу расходов.

Вывести список водителей компании:

Программа выводит на экран список водителей, каждая строка списка состоит из двух полей: номер рейса и имя водителя.

Вывести таблицу доходов:

Каждая строка таблицы, которую выводит программа, состоит из номера рейса и значения оплаты рейса за месяц.

Вывести таблицу расходов:

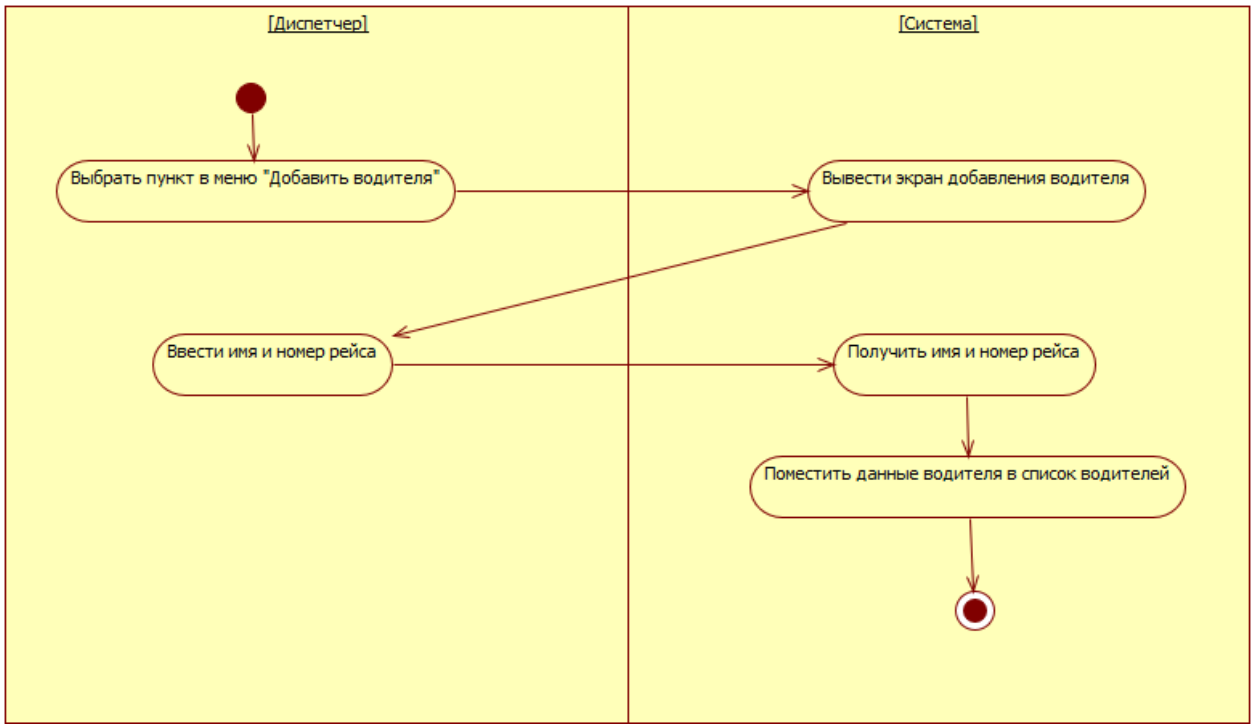
Каждая строка таблицы, которую выводит программа, состоит из номера рейса и значения ежемесячной оплаты рейса.

Вывести годовой отчет:

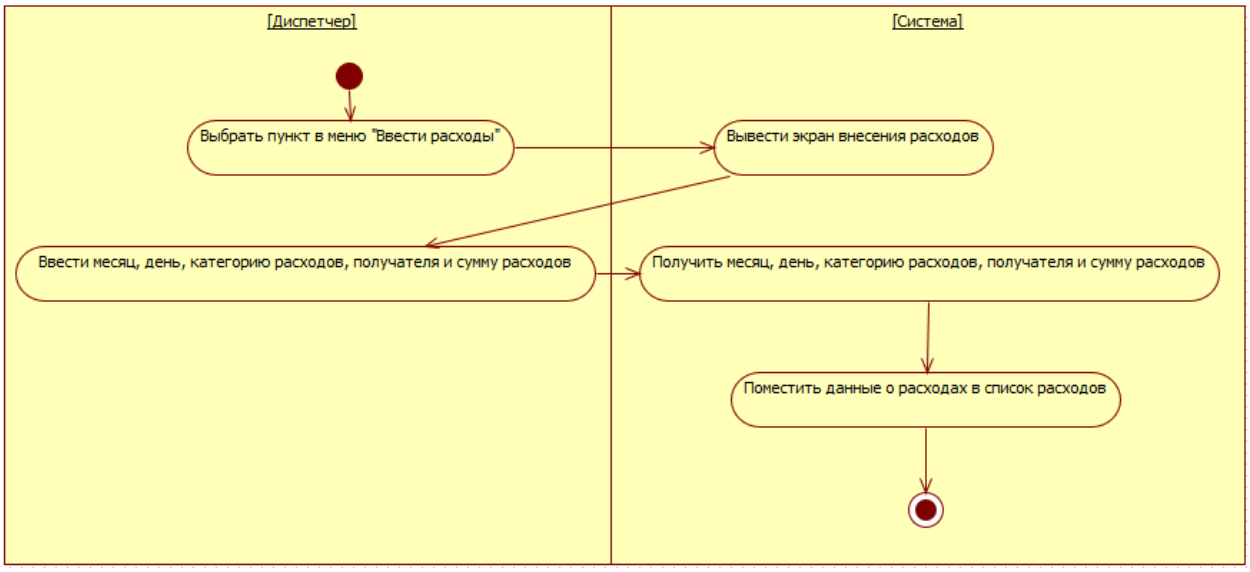
Программа выводит годовой отчет, состоящий из:
суммарного дохода рейсов за прошедший год;
списка всех расходов с указанием категории;
общая сумма расходов;
результирующего годового баланса (доходы/убытки)

6) Диаграммы действий:

Добавить водителя:



Ввести расходы:



7) Предварительный список существительных:
   1. Экран интерфейса пользователя.
   2. Водитель
   3. Экран ввода данных о рейсе
   4. Имя водителя
   5. Номер рейса
   6. Список водителей
   7. Данные о рейсе
   8. Экран ввода данных о рейсе
   9. Месяц
   10. Полученная сумма денег
   11. Таблица доходов

12. Строка оплаты рейса
13. Расход
14. Экран ввода расходов
15. Получатель
16. Сумма оплаты
17. День
18. Категория расходов
19. Строка таблицы расходов
20. Таблица расходов
21. Годовой отчет
22. Суммарные доходы рейсов
23. Суммарные расходы с указанием категории
24. Баланс

8) Уточненный список существительных:
    1. Экран интерфейса пользователя.
    2. Водитель
    3. Экран ввода данных о рейсе
    4. Список водителей
    5. Экран ввода данных о рейсе
    6. Таблица доходов
    7. Строка оплаты рейса
    8. Расход
    9. Экран ввода расходов
    10. Таблица расходов
    11. Годовой отчет
9) Список атрибутов:
    1. UserInterface
        a) ptrDriversList - указатель на класс список водителей
        b) ptrDriversInputScreen - указатель на класс экран ввода водителей
        c) ptrRevRecord - указатель на класс записей дохода
        d) ptrRevInputScreen - указатель на класс экран добавления дохода
        e) ptrExpenseRecord - указатель на класс записей о затратах
        f) ptrExpenseInputScreen - указатель на класс ввода расходов
        g) ptrAnnualReport - указатель на класс годового отчета
        h) ch – переменная для работы switch
    2. RevRecord
        a) setPtrsRR - указателей на список доходов
        b) iter – итератор для работы со списком доходов
    3. 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

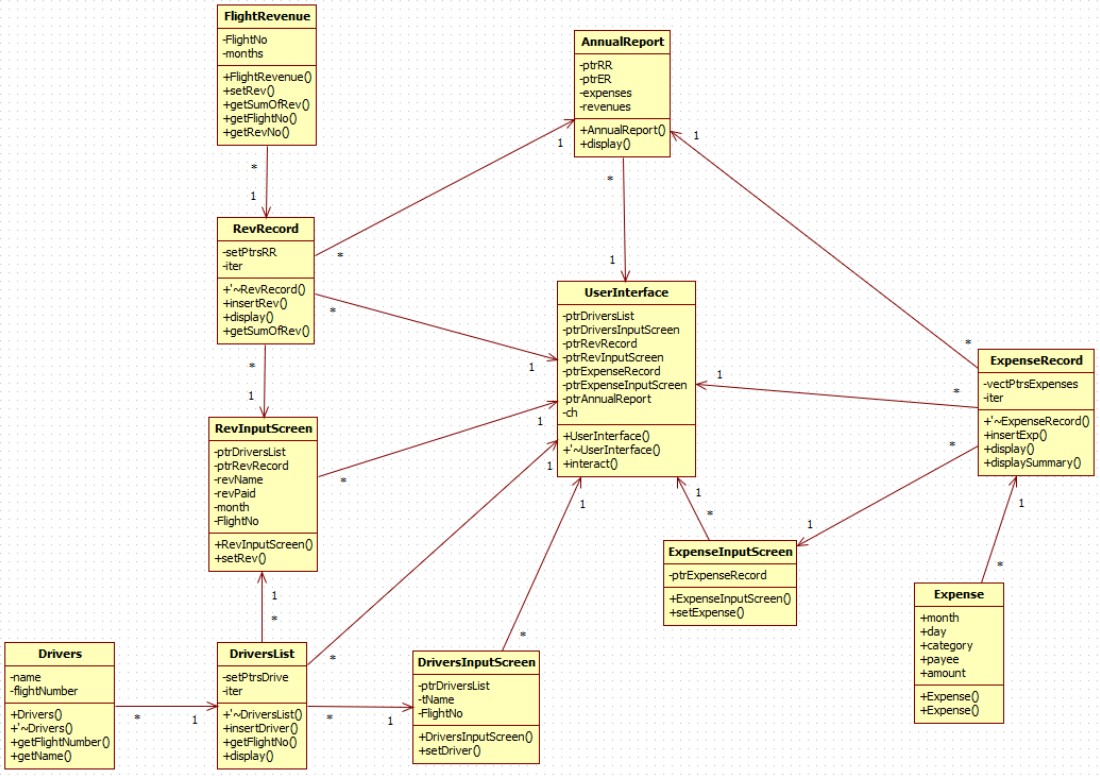
4. FlightRevenue
    a) FlightNo – номер рейса
    b) months[12] – массив полученной прибыли по месяцам
5. ExpenseRecord
    a) vectPtrsExpenses – указатель на вектор расходов
    b) iter – итератор для работы с вектором расходов
6. ExpenseInputScreen
    a) ptrExpenseRecord – указатель на запись о расходах
7. Expense
    a) month - месяц
    b) day - день
    c) category – категория расходов
    d) payee - получатель
    e) amount – сумма, уплаченная за расходы
8. DriversList
    a) setPtrsDrive – указатель на список класса водителей
    b) iter – итератор для работы со списком
9. DriversInputScreen
    a) ptrDriversList – указатель на список водителей
    b) tName – имя водителя
    c) FlightNo – номер рейса
10. Drivers
    a) name – имя водителя
    b) flightNumber – номер рейса
11. AnnualReport
    a) ptrRR – указатель на записи доходов
    b) ptrER - указатель на записи расходов
    c) expenses – сумма расходов
    d) revenues – сумма доходов

10) Список сообщений:
1. UserInterface
    a) UserInterface() – конструктор
    b) ~UserInterface() - деструктор
    c) interact() – функция для взаимодействия пользователя с системой
2. RevRecord
    a) ~RevRecord() - деструктор
    b) insertRev() – добавление дохода в список
    c) display() – вывод списка доходов
    d) getSumOfRev() – итоговое суммирование доходов по всем водителям
3. RevInputScreen
    a) RevInputScreen() – конструктор
    b) setRev() – добавляет доход с одного водителя за месяц
4. 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

5. ExpenseRecord
   a) ~ExpenseRecord() - деструктор
   b) insertExp() – вставляет в конец вектора данные о расходах
   c) display() – вывод таблицы расходов
   d) displaySummary() – сумма по всем категориям расходов
6. ExpenseInputScreen
   a) ExpenseInputScreen() – конструктор
   b) setExpense() – создает и добавляет новый расход в вектор расходов
7. Expense
   a) Expense() - конструктор
   b) Expense(int m, int d, string c, string p, float a) – конструктор с параметром
8. DriversList
   a) ~DriversList() - деструктор
   b) insertDriver() – вставляет нового водителя в конец списка водителей
   c) getFlightNo() – возвращает номер рейса водителя по имени водителя
   d) display() - вывод списка водителей
9. DriversInputScreen
   a) DriversInputScreen() – конструктор
   b) setDriver() – добавляет данные о водителе и заносит в список водителей
10. Drivers
    a) Drivers() – конструктор
    b) ~Drivers() - деструктор
    c) getFlightNumber() – возвращает номер рейса
    d) getName() – возвращает имя водителя
11. AnnualReport
    a) AnnualReport() – конструктор
    b) display() – вывод годового отчета

11) Диаграмма классов:



12) Диаграммы последовательности:

Диаграмма последовательностей “Добавить водителя”:

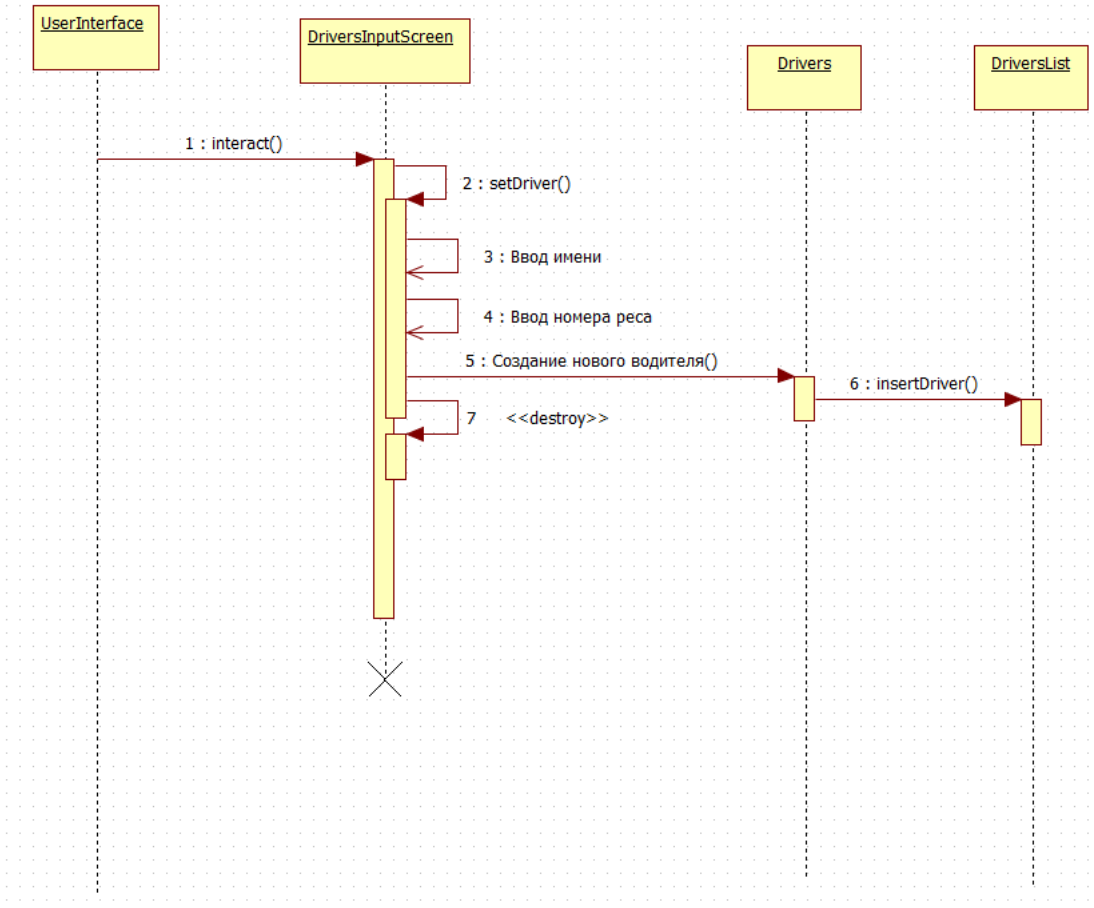


Диаграмма последовательностей “Вывести список водителей компании”:

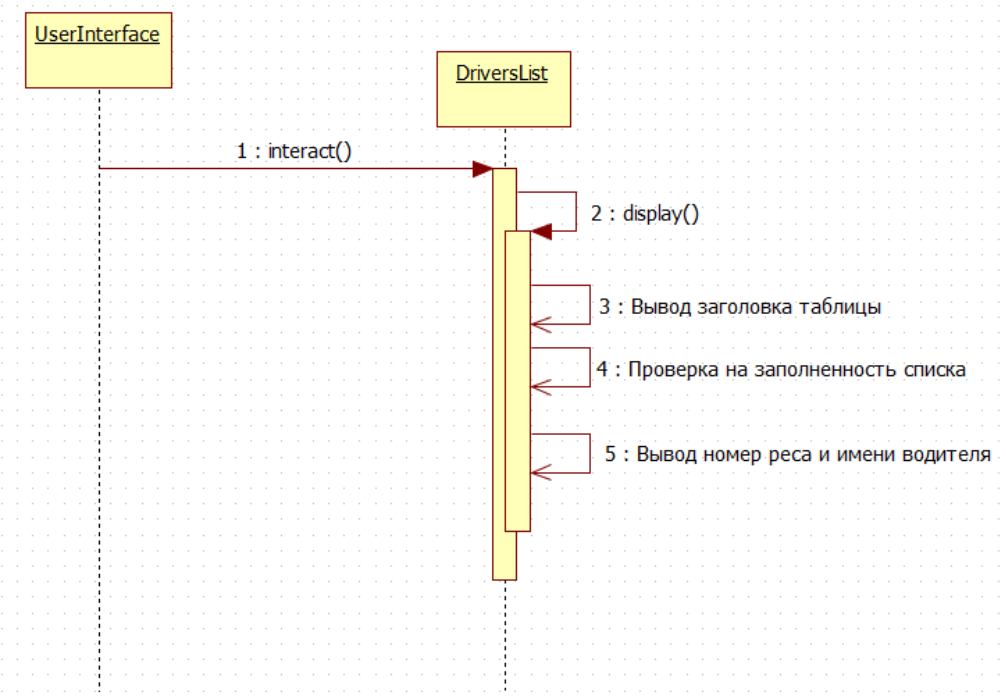
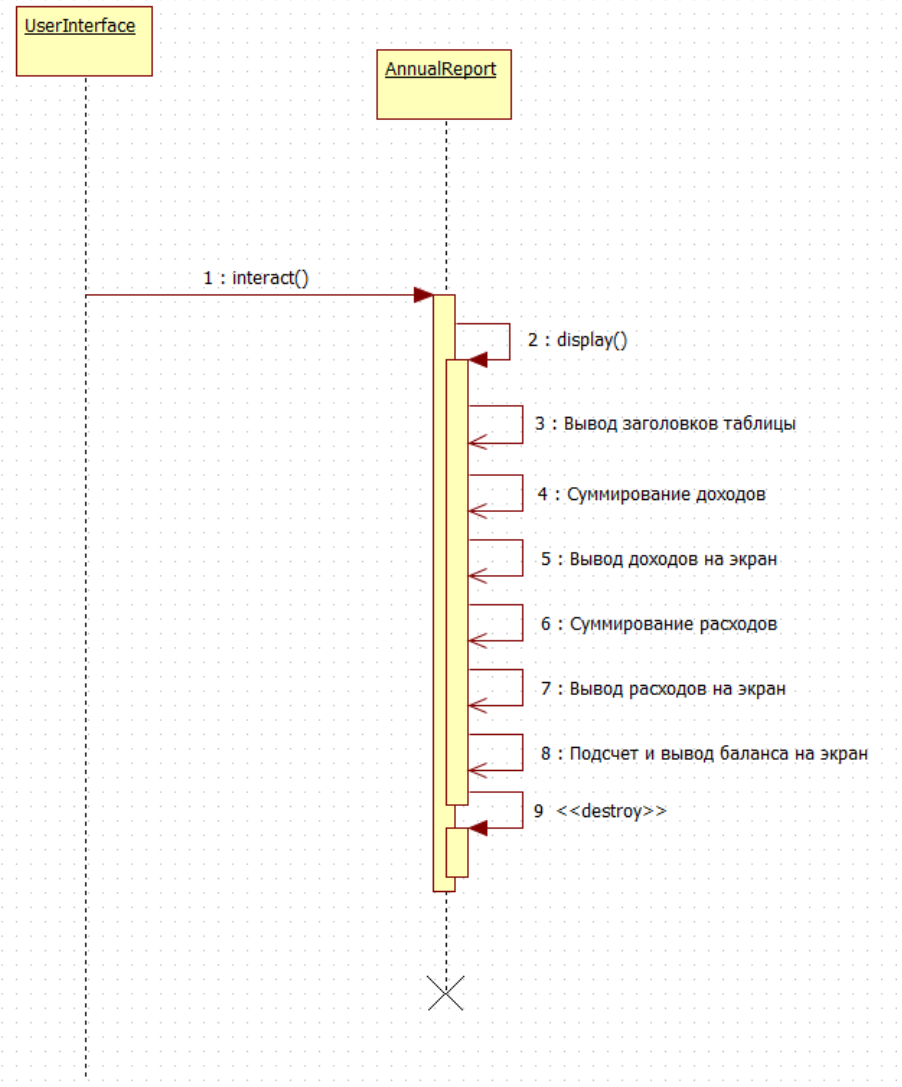


Диаграмма последовательностей “Вывести годовой отчет”:



13) 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 "UserInterface.h"
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 "UserInterface.h"
//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* ptrDL, RevRecord* ptrRR) : ptrDriversList(ptrDL),
      ptrRevRecord(ptrRR)
   {
      /*тут пусто*/
   }
   void setRev(); // добавить доход с одного водителя за месяц
};
#endif // REVINPUTSCREEN
```

FlightRevenue.h

```
//FlightRevenue.h
#ifndef FLIGHTREVENUE
#define FLIGHTREVENUE
#include "UserInterface.h"
//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 пллату 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 // FLIGHTREVENUE
```

ExpenseRecord.h

```
//ExpenseRecord.h
#ifndef EXPENSERECORD
#define EXPENSERECORD
#include "UserInterface.h"
//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
};
#endif // EXPENSERECORD
```

ExpenseInputScreen.h

```
//ExpenseInputScreen.h
#ifndef EXPENSEINPUTSCREEN
#define EXPENSEINPUTSCREEN
#include "UserInterface.h"
//Класс для ввода расходов
class ExpenseInputScreen
{
private:
    ExpenseRecord* ptrExpenseRecord; // запись о расходах
public:
    ExpenseInputScreen(ExpenseRecord*);
    void setExpense();
};
#endif // EXPENSEINPUTSCREEN
```

Expense.h

```
//Expense.h
#ifndef EXPENSE
#define EXPENSE
#include "UserInterface.h"
//Класс затрат
class Expense
{
public:
    int month, day; // месяц и день уплаты расходов
```

```
    string category; // категория расходов (топливо, платные налоги, ремонт/ТО, налоги)
    string payee; // кому платим (АЗС, РОСАВТОДОР, СТО, государство, )
    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! */
    }
};
#endif // EXPENSE
```

DriversList.h

```
//DriversList.h
#ifndef DRIVERSLIST
#define DRIVERSLIST
#include "Drivers.h"
using namespace std;
//класс DriversList — список всех водителей.
//Он содержит множество указателей на класс Drivers
// и оперирует ими при выводе
class DriversList
{
private:
    // установить указатели на водителей
    list <Drivers*> setPtrsDrive; // указатели на класс водителей
    list <Drivers*>::iterator iter; //итератор
public:
    ~DriversList(); // деструктор (удаление водителей)
    void insertDriver(Drivers*); // добавить водителя в список
    int getFlightNo(string); // возвращает номер рейса
    void display(); // вывод списка водителей
};
#endif // DRIVERSLIST
```

DriversInputScreen.h

```
//DriversInputScreen.h
#ifndef DRIVERSINPUTSCREEN
#define DRIVERSINPUTSCREEN
#include "UserInterface.h"
//класс DriversInputScreen. Это класс, отвечающий за отображение «экрана»,
//куда пользователь может ввести данные о новом водителе:
class DriversInputScreen
{
private:
    DriversList* ptrDriversList;
    string tName;
    int FlightNo;
public:
    DriversInputScreen(DriversList* ptrDL) : ptrDriversList(ptrDL)
    {
        /* тут пусто */
    }
```

```
    void setDriver(); // добавить данные о водителе
};
#endif // DRIVERSINPUTSCREEN
```

Drivers.h

```
//Drivers.h
#ifndef DRIVERS
#define DRIVERS
#include "UserInterface.h"
using namespace std;
// класс Drivers (водители) //
//Он хранит имя водителя и номер рейса, в котором он находится.
class Drivers
{
private:
    string name; // имя водителя
    int flightNumber; // номер рейса, в котором он находится
public:
    Drivers(string n, int aNo);
    ~Drivers();
    int getFlightNumber(); //возвращает номер рейса водителя
    string getName(); //возвращает имя водителя
};
#endif // DRIVERS
```

AnnualReport.h

```
//AnnualReport.h
#ifndef ANNUALREPORT
#define ANNUALREPORT
#include "UserInterface.h"
//Класс годового отчета
class AnnualReport
{
private:
    RevRecord* ptrRR; // записи доходов
    ExpenseRecord* ptrER; // записи расходов
    float expenses, revenues; // суммы доходов и расходов
public:
    AnnualReport(RevRecord*, ExpenseRecord*);
    void display(); // отображение годового отчета
};
#endif // ANNUALREPORT
```

14) 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()
```

TransportCompany.cpp

```
//TransportCompany.cpp
#include "UserInterface.h"
int main()
{
    UserInterface theUserInterface;
    theUserInterface.interact();
    return 0;
}
```

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);
}
```

ExpenseRecord.cpp

```
//ExpenseRecord.cpp
#include "UserInterface.h"

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 << "\nDate\tRecipient\tAmount\tCategory\n"  // дата\получатель\сумма\категория
        << "---------------------------------------\n" << endl;
    if (vectPtrsExpenses.size() == 0) // В контейнере нет расходов
        cout << "***There are no expenses***\n" << endl;  // расходов нет
    else
    {
        iter = vectPtrsExpenses.begin();
        while (iter != vectPtrsExpenses.end())
        { // распечатываем все расходы
            cout << (*iter)->month << '/' << (*iter)->day << '\t' << (*iter)->payee << '\t' << '\t';
            cout << (*iter)->amount << '\t' << (*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)
    {
        cout << "\tAll categories\t\t0\n"; // Все категории
```

```
        return 0;
    }
    iter = vectPtrsExpenses.begin();
    while (iter != vectPtrsExpenses.end())
    {
        //выводим на экран категории расходов
        cout << '\t' << ((*iter)->category) << '\t' << ((*iter)->amount) << endl;
        totalExpenses += (*iter)->amount; //подсчитываем все расходы
        iter++;
    }
    return totalExpenses;
}
```

ExpenseInputScreen.cpp

```
//ExpenseInputScreen.cpp
#include "UserInterface.h"

// 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): "; //Введите месяц (1-12)
    cin >> month;
    cin.ignore(80, '\n');
    cout << "Enter the day (1-31): ";   //Введите день (1-31)
    cin >> day;
    cin.ignore(80, '\n');
    cout << "Enter the expense category (fuel, toll roads, repair, taxes): ";  // Введите категорию
расходов (Топливо, Платные дороги, Ремонт, Налоги)
    getaLine(category);
    cout << "Enter the recipient (Gazprom, rosavtodor,service station, Russia): "; // Введите
получателя (Газпром, РОСАВТОДОР, СТО, Россия)
    getaLine(payee);
    cout << "Enter the amount (350.65): "; // Введите сумму
    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);
}
```

DriversList.cpp

```
//DriversList.cpp
#include "UserInterface.h"

DriversList::~DriversList() // деструктор
```

```
{
   while (!setPtrsDrive.empty()) // удаление всех водителей,
   { // удаление указателей из контейнера
      iter = setPtrsDrive.begin();
      delete* iter;
      setPtrsDrive.erase(iter);
   }
}

void DriversList::insertDriver(Drivers* ptrD)
{
   setPtrsDrive.push_back(ptrD); // вставка нового водителя в список
}

int DriversList::getFlightNo(string tName) // получить номер рейса по имени водителя
{
   int FlightNo;
   iter = setPtrsDrive.begin();
   while (iter != setPtrsDrive.end())
   { // поиск водителя в списке (достаем у каждого водителя номер рейса)
      FlightNo = (*iter)->getFlightNumber();
      if (tName == ((*iter)->getName())) // сравниваем по именам и
      {
         // если получившаяся пара совпадает - значит,
         //мы нашли запись об этом водителе в списке, в этом случае
         return FlightNo; // возвращаем номер его рейса
      }
      iter++;
   }
   return -1; // если нет - возвращаем -1
}

void DriversList::display() // вывод списка водителей
{
   cout << "\nNumber#\tName Driver\n--------------------\n";
   if (setPtrsDrive.empty()) // если список водителей пуст
      cout << "***No driver***\n" << endl; // выводим запись, что он пуст)
   else
   {
      iter = setPtrsDrive.begin();
      while (iter != setPtrsDrive.end()) // распечатываем всех водителей
      {
         cout << (*iter)->getFlightNumber() << "   || " << (*iter)->getName() << endl;
         *iter++;
      }
   }
}
```

DriversInputScreen.cpp

```
//DriversInputScreen.cpp
#include "UserInterface.h"

void DriversInputScreen::setDriver() // добавить данные о водителе
```

```
{
   cout << "Enter the driver's name (Alexander Masloy): " << endl; // Введите имя водителя
(Александр Маслов):
   getaLine(tName);
   cout << "Enter the flight number (515): " << endl; // Введите номер рейса (515):
   cin >> FlightNo;
   cin.ignore(80, '\n');
   Drivers* ptrDriver = new Drivers(tName, FlightNo); // создать водителя
   ptrDriversList->insertDriver(ptrDriver); // занести в список водителей
}
```

Drivers.cpp

```
//Drivers.cpp
#include "UserInterface.h"

//в конструкторе задаём имя водителя и номер рейса
Drivers::Drivers(string n, int aNo) : name(n), flightNumber(aNo)
{
   /* тут пусто */
}

Drivers::~Drivers() // деструктор
{
   /* тут тоже пусто */
}

int Drivers::getFlightNumber() //геттер возвращает номер рейса водителя
{
   return flightNumber;
}

string Drivers::getName() //геттер возвращает имя водителя
{
   return name;
}
```

AnnualReport.cpp

```
//AnnualReport.cpp
#include "UserInterface.h"

//Конструктор
AnnualReport::AnnualReport(RevRecord* pRR, ExpenseRecord* pER) : ptrRR(pRR),
ptrER(pER)
{ /* пусто */
}

void AnnualReport::display()
{
   cout << "Annual Report\n----------\n" << endl; //Годовой отчет
   cout << "Income\n" << endl; // Доходы
   cout << "\tPayment for the flight:\t"; // Плата за рейс
   revenues = ptrRR->getSumOfRev();
   cout << revenues << endl;
   cout << "Expenses\n" << endl;  // Расходы
```

```
    expenses = ptrER->displaySummary();
    cout << "Total expenses:\t\t\t";  //Расходы всего
    cout << expenses << endl;
    // вычисляем прибыльность
    cout << "\nBalance:\t\t\t" << (revenues - expenses) << endl; // Баланс
}
```

15) 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 drivers 'd',
Click to withdraw revenue 'e',
Click to display expenses 'f',
Click to display the annual report 'g',
Click to exit the program 'q'.
```

1. При выборе 'a' Добавить водителя, пользователю будет предложено ввести имя водителя и его номер рейса, после чего выведется меню.

```
a
Enter the driver's name (Alexander Masloy):
Alex Berger
Enter the flight number (515):
316
```

2. 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: Alex Berger
Enter the amount of income (800.50):
5000
Enter the payment month number (1-12):
1
```

3. При выборе 'c' Заполнить расходы, пользователю будет предложено ввести месяц, день и категорию расходов, получателя, а также потраченную сумму. После выведется меню.

```
c
Enter the month (1-12): 2
Enter the day (1-31): 22
Enter the expense category (fuel, toll roads, repair, taxes): fuel
Enter the recipient (Gazprom, rosavtodor,service station, Russia): Lukoil
Enter the amount (350.65): 2100
```

4. При выборе 'd' Вывести водителей, пользователю будет выведена таблица: номер рейса || имя водителя. После выведется меню.

```
d

Number# Name Driver
--------------------
316 || Alex Berger
101 || Viktor Maslov
220 || Maxim Nak
```

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

```
Number  Jan Feb Mar Apr May June July Aug Sept Oct Nov Dec

----------------------------------------------------------------

316     5000  0  0  0  0  0  0  0  0  0  0  0
220      0 3500  0 4000  0  0  0  0  0  0  0  0
```

6. При выборе 'f' Вывести расходы, пользователю будет выведена таблица расходов с указанием даты, получателя, суммы и категории расходов. После выведется меню.

```
f

Date    Recipient       Amount  Category
---------------------------------------

2/22    Lukoil          2100    fuel
```

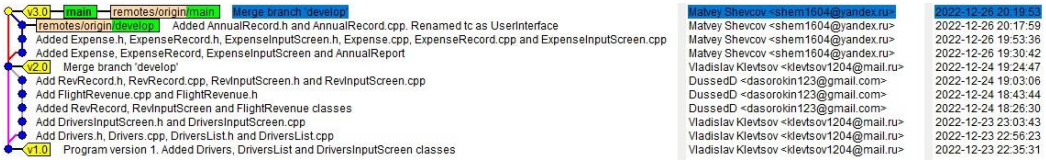
7. При выборе 'g' Вывести годовой отчет, пользователю будет выведена таблица, где будет учтена вся прибыль за прошедший период, указаны расходы, с указанием категории и цены, а также подведен итог, заработала ли компания за этот промежуток времени или же понесла убытки. После выведется меню.

```
g
Annual Report
-------------

Income

        Payment for the flight: 700000
Expenses

        fuel    30000
        toll roads      5000
Total expenses:                 35000

Balance:                        665000
```

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

```
q
The program is completed
```

История проекта на GitHub



Адрес проекта: https://github.com/he4to-hoboe/TransportCompany.git

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.