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

ИСПОЛНИТЕЛЬ:

Шевцов М. Н. группа 21316

Содержание

Цель:

Добавить пользователю возможность учета расходов и упростить составление годового отчета, путём вывода на экран доходов и расходов.

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 созданы заголовочные файлы:

1. AnnualReport.h
2. Expense.h
3. ExpenseInputScreen.h
4. ExpenseRecord.h

Ход разработки:

1) Краткое словесное описание:
В процессе работы компании персоналу требуется вести документацию. Программа TransportCompany поможет пользователю вести учет расходов компании и снять часть задач по составлению годового отчёта.

Функционал программы позволяет:

- Учитывать расходы, а именно дату, категорию, получателя и сумму
- Выводить данные о доходах, расходах и результирующего годового баланса

2) Описание имеющихся у заказчика материалов:

Таблица 1. Расходы

Дата	Получатель	Сумма	Категория
2/22	Лукоил	2100	Топливо
4/22	Механики	1500	Тех обслуживание

Таблица 2. Годовой отчет

Доходы	45000
Расходы	14000
Итого	31000

3) Диаграмма вариантов использования:

4) Описание добавляемых в программу вариантов использования:

Вывести таблицу расходов:

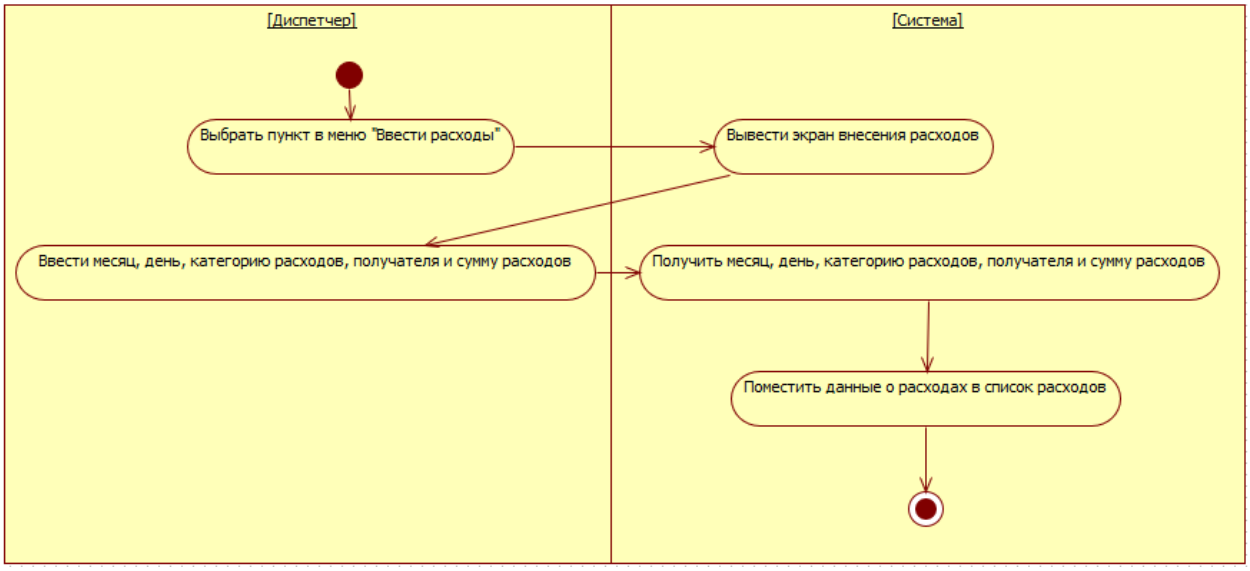
Каждая строка таблицы, которую выводит программа, состоит из номера рейса и значения ежемесячной оплаты рейса.

Вывести годовой отчет:

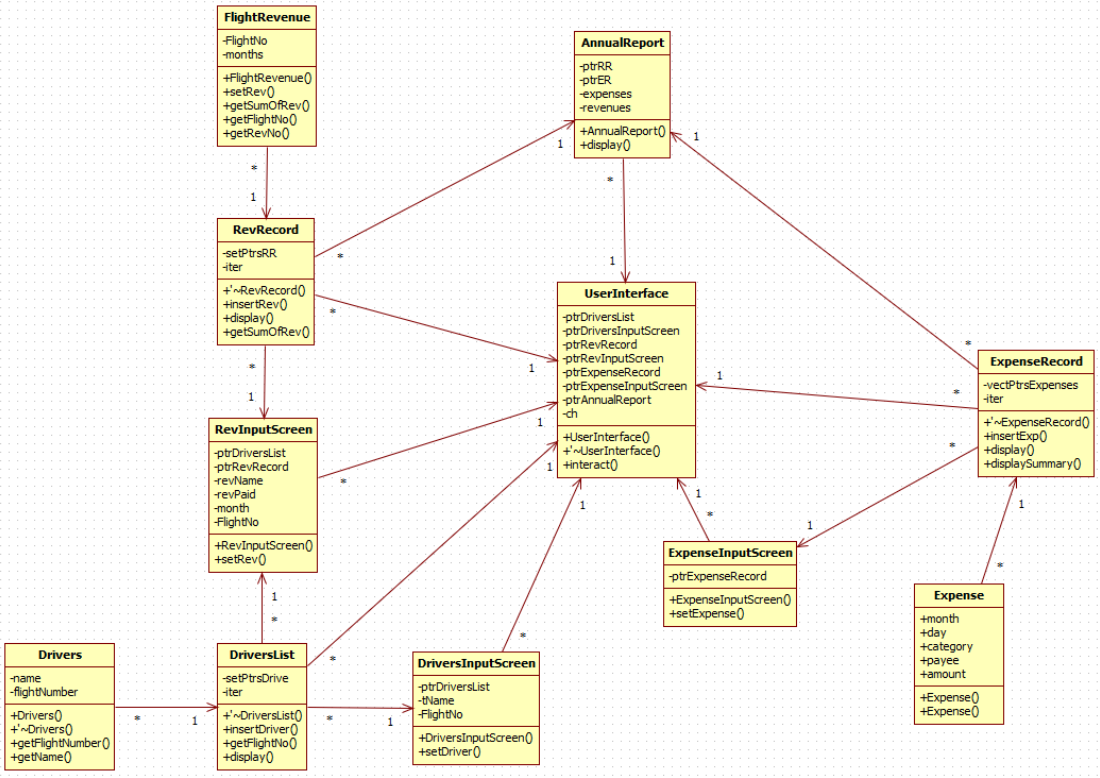
Программа выводит годовой отчет, состоящий из:
суммарного дохода рейсов за прошедший год;
списка всех расходов с указанием категории;
общая сумма расходов;
результирующего годового баланса (доходы/убытки)

5) Диаграммы действий:

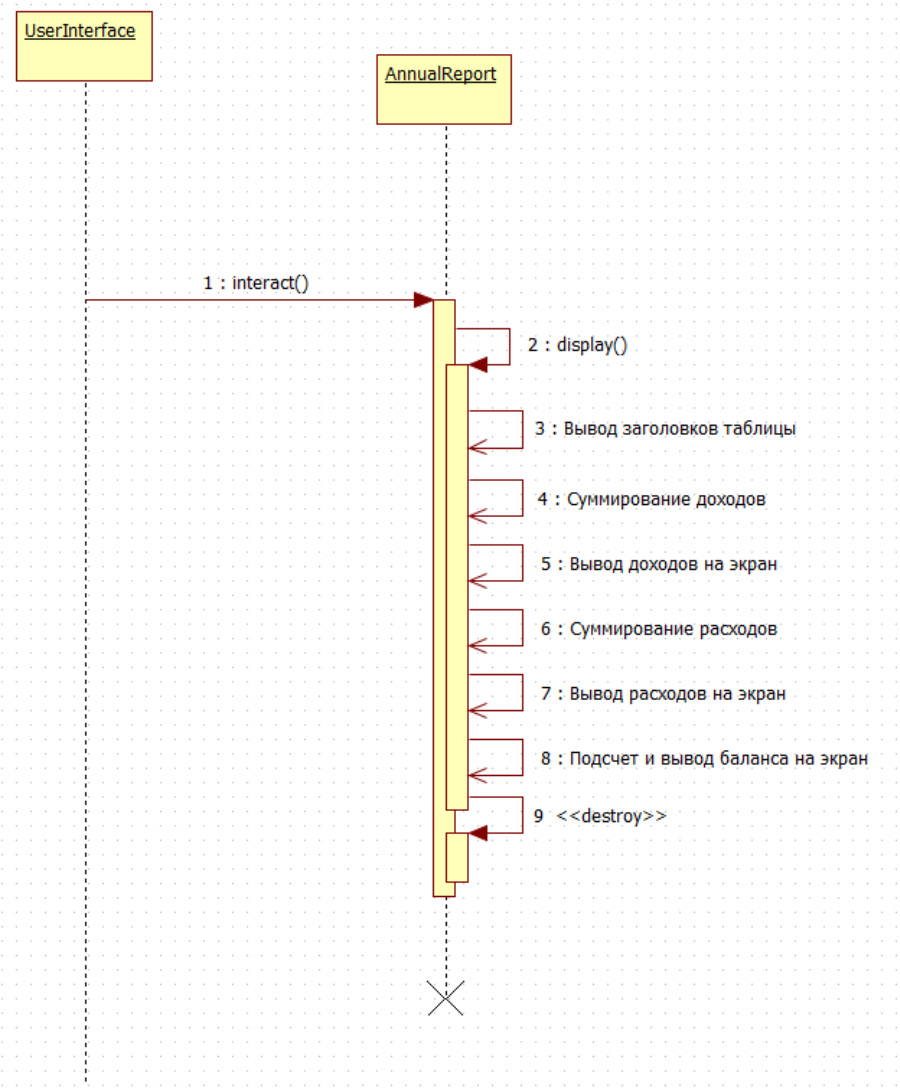
Ввести расходы:



6) Диаграмма классов:



7) Диаграммы последовательности:
Диаграмма последовательностей “Вывести годовой отчет”:



8) 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
```

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
```

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
```

9) 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()
```

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);
}
```

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
}
```

10) 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 '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
```

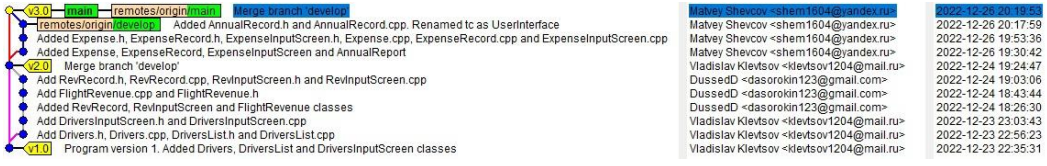
2. При выборе 'f' Вывести расходы, пользователю будет выведена таблица расходов с указанием даты, получателя, суммы и категории расходов. После выведется меню.

```
f

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

2/22    Lukoil          2100    fuel
```

3. 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
```

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.