

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«Петрозаводский государственный университет»

Физико-технический институт

**Кафедра информационно-измерительных систем и физической
электроники**

Отчет по

Командному заданию

по курсу «Технология программирования»

Выполнил:

Студент группы 21316

Федоров Виталий Павлович

Дмитриев Илья Николаевич

Кривецкий Артур Антонович

Проверил:

Бульба Артем Владимирович

Петрозаводск, 2021

Цель работы:

Смоделировать ситуацию выполнения заказа на разработку ПО для облегчения ведения предпринимательской деятельности.

Программная реализация: разработка программы осуществлялась в IDE Visual Studio 2019 на языке C++

В результате работы были созданы следующие заголовочные файлы и единицы компиляции:

AnnualReport.cpp - содержит описание класса AnnualReport

AnnualReport.h - объявление класса AnnualReport (используется для формирования и вывода годового отчета)

ExpenseInputScreen.cpp - содержит описание класса ExpenseInputScreen

ExpenseInputScreen.h - объявление класса ExpenseInputScreen (используется для создания записи расхода)

ExpenseRecord.cpp - содержит описание класса ExpenseRecord

ExpenseRecord.h - объявление класса ExpenseRecord (используется для хранения и вывода записей о расходах)

Glavn.cpp - содержит функцию main()

Global.cpp - содержит описание класса Global

Global.h - объявление класса Global (содержит два глобальных метода, первый используется для получения строки, второй для получения символа)

Order.cpp - содержит описание класса Order

Order.h - объявление класса Order (хранит информацию о заказах)

OrderList.cpp - содержит описание класса OrderList

OrderList.h - объявление класса OrderList (содержит указатели на заказы, выводит список заказов)

OrderInputList.cpp - содержит описание класса OrderInputList

OrderInputList.h - объявление класса OrderInputList (отвечает за экран ввода заказа)

UserInterface.cpp - содержит описание класса UserInterface

UserInterface.h - объявление класса OrderInputList (отвечает за главный экран программы, необходим для взаимодействия пользователя с программой)

Expense.h - объявление класса Expense (содержит поля, необходимые для добавления записи о расходе)

Описание процесса разработки:

1. Легенда:

Некий таинственный гражданин N заходит в офис нашего клуба программистов. Целью его визита к нам является создание программного обеспечения для его транспортной компании "Spreeeed". Гражданин N является Ген. Директором этой компании. В его пожелания входит создание программы, которая будет обрабатывать данные о заказах, прибыли полученной с них.

2. Примерные варианты таблиц:

Номер заказа	Дата	Отправитель	Получатель	Доход

Таб.1 Таблица доходов

Категория расхода	Получатель средств	Сумма расхода
Налог	РФ	500
Ремонт	Сервис	600
Оплата кредита	Банк	800
Аренда офиса	Агентство недвижимости	300

Таб.2 Таблица расходов

Суммы доходов	Сумма расходов	Чистая Прибыль
5156	2200	2956

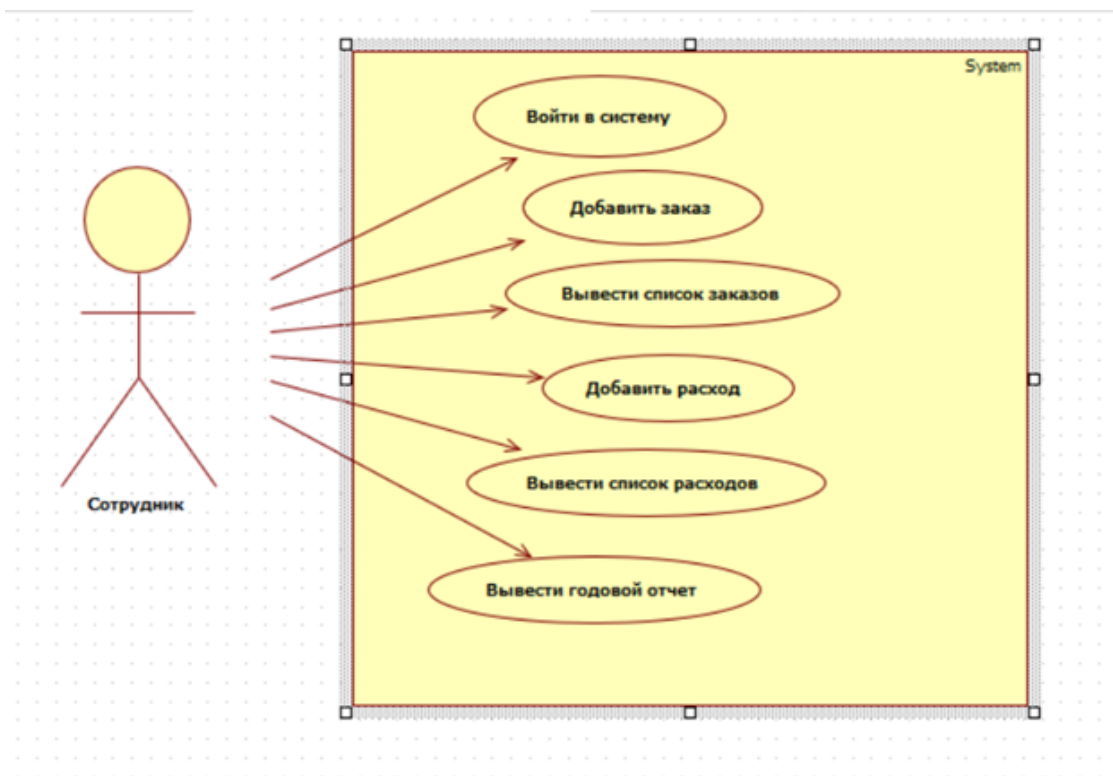
Таб.3 Годовой отчет

3. Действующее лицо - Сотрудник

Имеет возможность:

- Добавлять заказы
- Добавлять расходы
- Выводить годовой отчет

4. Диаграмма вариантов использования:



5. Описание вариантов использования:

Войти в систему - Данный вариант использования выводит на экран меню выбора действий

- 1) Ввести данные
- 2) Вывести данные
- 3) Завершить работу

Добавить заказ - На экране должно отобразиться сообщение, в котором программа просит пользователя ввести номер заказа, дату, данные получателя и отправителя. Эта информация должна заноситься в таблицу.

Вывести список заказов - Программа выводит на экран список заказов, каждая строка списка состоит из пяти полей: номер заказа, дата, отправитель, получатель, доход.

Добавить расход - Экран ввода расхода должен содержать приглашение пользователю на ввод дня и месяца, в который производится оплата, бюджетной категории, и суммы оплаты. Затем программа создает новую строку, содержащую эту информацию, и вставляет ее в таблицу расходов.

Вывести список расходов - Каждая строка таблицы, которую выводит программа, состоит из значений месяца, дня, суммы и бюджетной категории платежа.

Вывести годовой отчет - Программа выводит годовой отчет, состоящий из:

- суммарного дохода от заказов за год;
- списка всех расходов с указанием бюджетной категории;
- общая сумма расходов;

- результирующего годового баланса (доходы/убытки).

6. Диаграмма деятельности варианта использования “Добавить заказ”

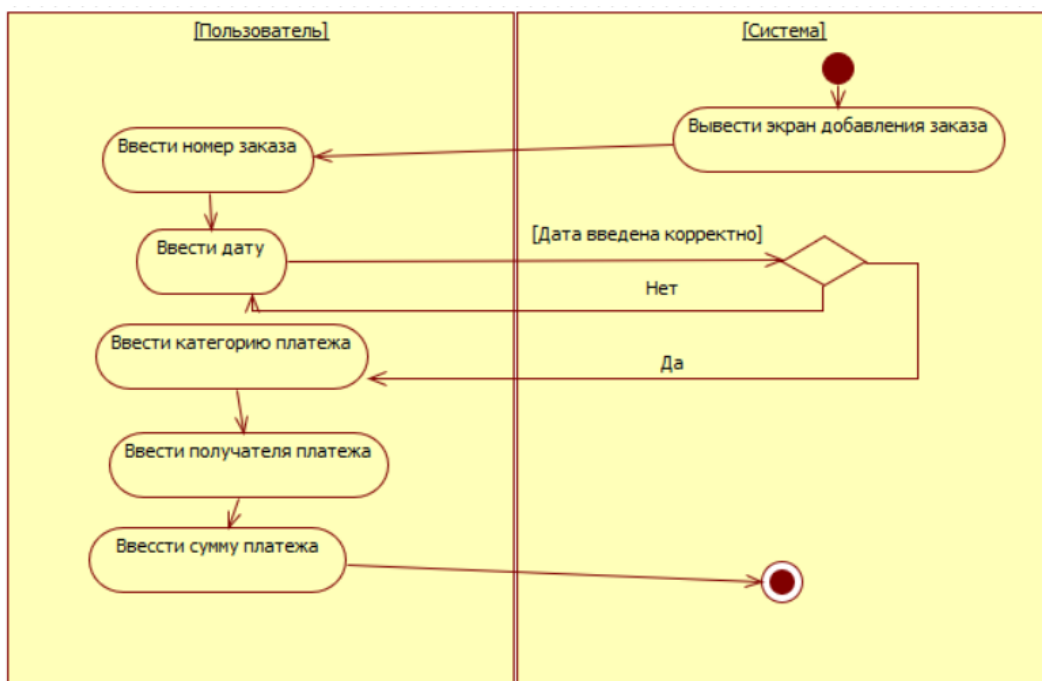
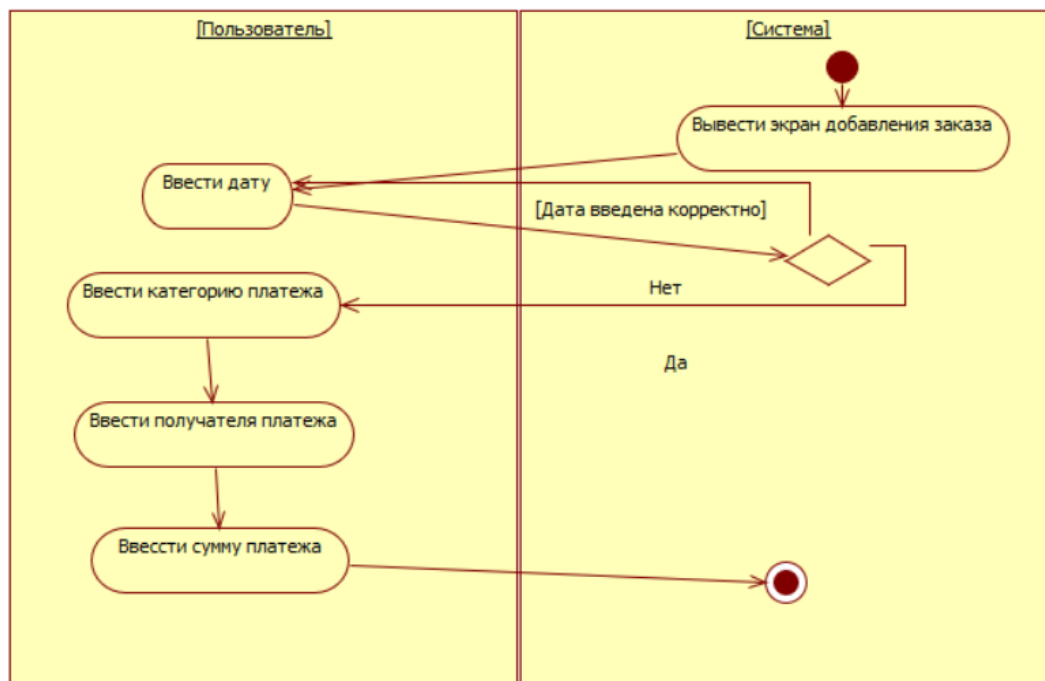


Диаграмма деятельности варианта использования “Добавить расход”



7. Предварительный список существительных:

1. Пользовательский интерфейс
2. Заказ

3. Экран ввода заказа
4. Дата заказа
5. День
6. Месяц
7. Номер заказа
8. Отправитель
9. Получатель
10. Доход
11. Строка списка заказов
12. Список заказов
13. Расход
14. Экран ввода расхода
15. Дата расхода
16. Бюджетная категория
17. Размер платежа
18. Строка расходов
19. Список расходов
20. Годовой отчет
21. Суммарный доход
22. Суммарный расход
23. Результирующий баланс

8. Уточненный список существительных:

1. Пользовательский интерфейс
2. Заказ
3. Экран ввода заказа
4. Список заказов
5. Расход
6. Экран ввода расхода

7. Запись расхода

8. Годовой отчет

9. Список атрибутов классов:

Заказ

- грузоотправитель
- грузополучатель
- номер заказа
- стоимость заказа
- месяц и день заказа

Экран ввода заказа

- грузоотправитель
- грузополучатель
- номер заказа
- стоимость заказа
- месяц и день заказа

Расход

- месяц и день уплаты расходов
- категория расходов
- кому платим
- сколько платим

Годовой отчет

- Записи расходов
- Записи доходов
- Сумма доходов
- Сумма расходов

Экран ввода расхода

- Запись о расходах

Записи расходов

- Указатели на расходы

Список заказов

- Указатели на заказы

Пользовательский интерфейс

- Выбор функции

10. Список сообщений классов:

Класс ввода нового заказа

- 1) Добавить новый заказ

Класс годового отчета

1. Отображение годового отчета

Класс списка всех заказов

- 1) Добавление заказа
- 2) Вывод списка заказов
- 3) Сумма заказа

Класс записи расходов

- 1) Ввод расходов
- 2) Отображение расходов
- 3) Сумма расходов

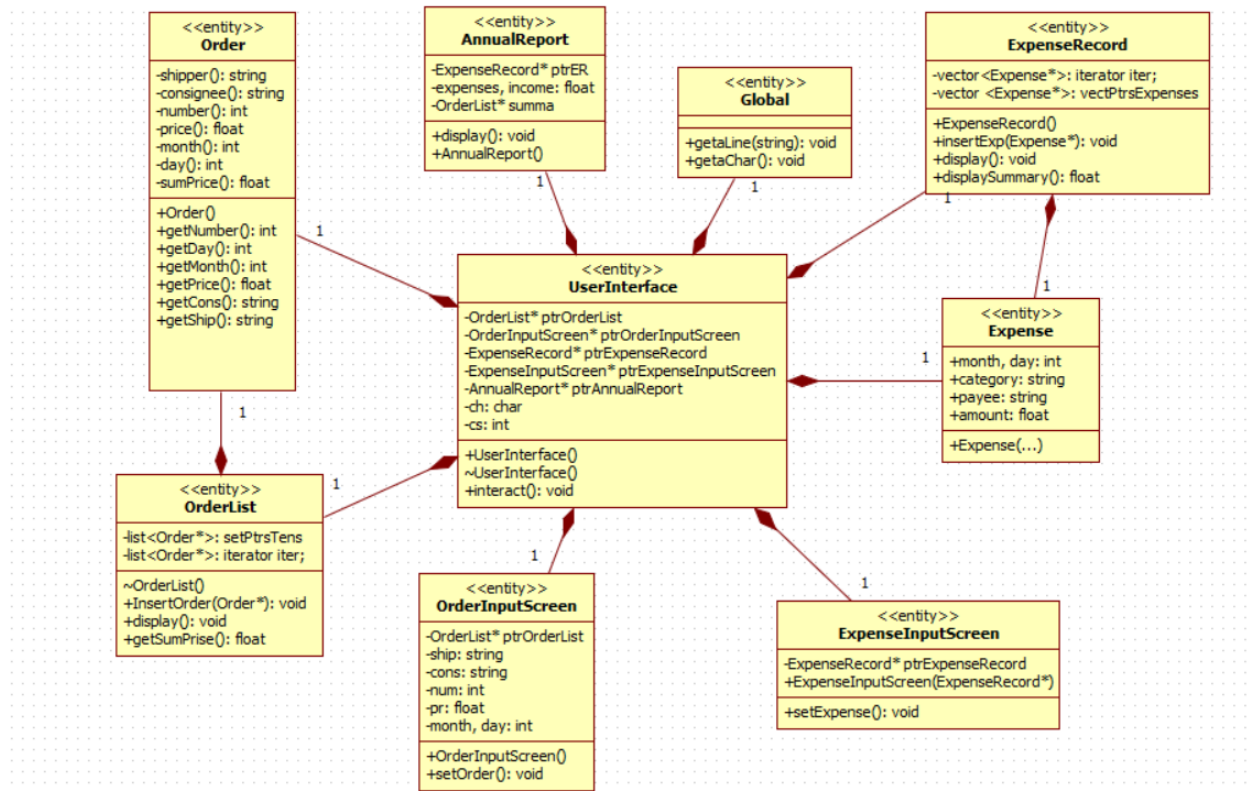
Класс запуска приложения

- 1) Вывод главного меню

Класс хранения списка заказов

- 1) Вывод номера заказа
- 2) Вывод дня заказа
- 3) Вывод месяца заказа
- 4) Вывод стоимости заказа
- 5) Вывод получателя заказа
- 6) Вывод отправителя заказа

11. Диаграмма классов



12. Диаграмма последовательности варианта использования “Добавить заказ”

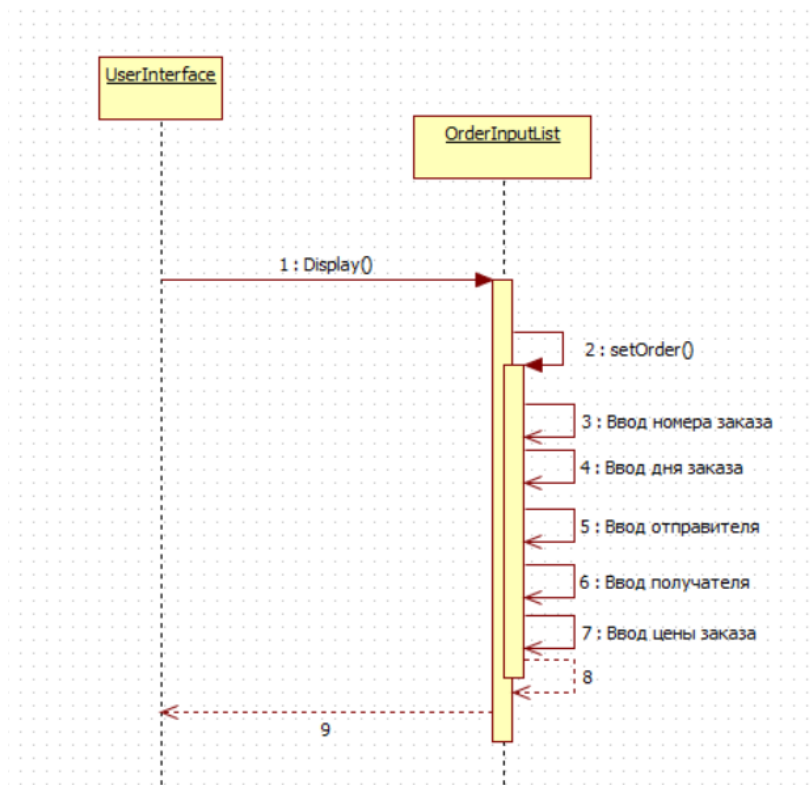
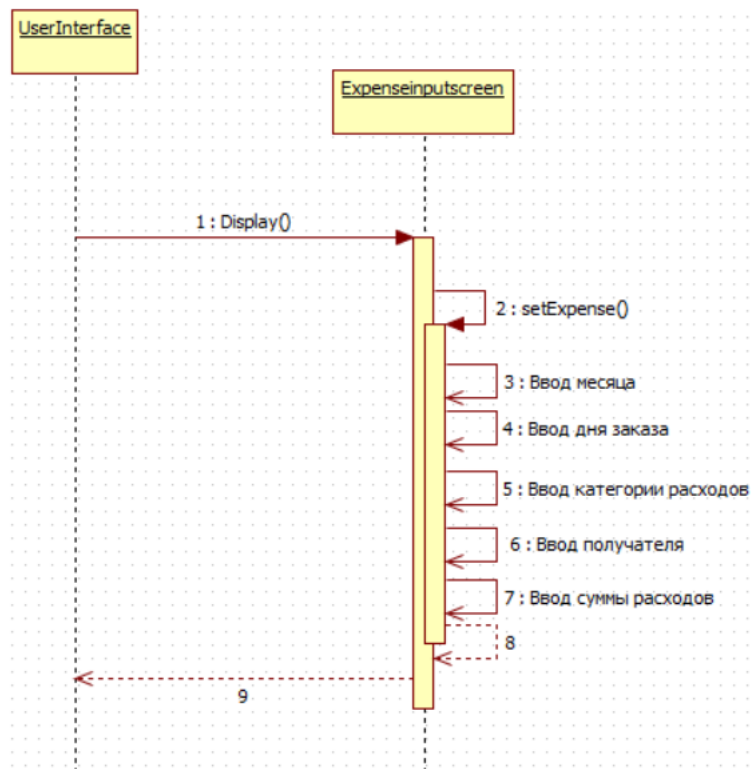


Диаграмма последовательности варианта использования “Добавить расход”



13.Листинги заголовочных файлов

AnnualReport.h

```

#pragma once
#include "ExpenseRecord.h"
#include "OrderList.h"

#include <iostream>
#include <vector>
#include <list>
#include <string>
#include <numeric> //для accumulate()
#include <cctype>
using namespace std;
////////////////////////класс AnnualReport////////////////////////
// ласс годового отчета
class AnnualReport
{
private:
    //RentRecord* ptrRR; // записи доходов
    ExpenseRecord* ptrER; // записи расходов
    float expenses, income; // суммы доходов и расходов
    OrderList* summa;
    //float saa;
public:
    AnnualReport(ExpenseRecord*, OrderList*);
    void display(); // отображение годового отчета
};
  
```

Expence.h

```

#pragma once
  
```

```

#include <iostream>
#include <vector>
#include <list>
#include <string>
#include <numeric> //для accumulate()
#include <cctype>
#include "Global.h"
////////////////////////класс Expense////////////////////////
//Класс затрат
class Expense
{
public:
    int month, day; // месяц и день уплаты расходов
    string category; // категория расходов
    string payee; // кому платим
    float amount; // сколько платим
// Expense(int m, int d, string c, string p, float a) : month(m), day(d), category(c), payee(p), amount(a)
//{ }
    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 <iostream>
#include <vector>
#include <list>
#include <string>
#include <numeric> //для accumulate()
#include <cctype>
using namespace std;
////////////////////////класс ExpenseRecord////////////////////////
// ласс 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////////////////////////
// ласс для ввода расходов
class ExpenseInputScreen
{

```

```
private:
    ExpenseRecord* ptrExpenseRecord; // запись о расходах
public:
    ExpenseInputScreen(ExpenseRecord*);
    void setExpense();
};
```

Global.h

```
#pragma once
#include <string>
#include <Windows.h>
#include <iostream>
#include <vector>
#include <list>
#include <string>
#include <numeric> //для accumulate()

using namespace std;
////////// глобальные методы //////////
void getaLine(string& inStr); // получение строки текста
char getaChar(); // получение символа
```

Order.h

```
#pragma once
#include <iostream>
#include <vector>
#include <list>
#include <string>
#include <numeric> //для accumulate()
#include <cctype>

using namespace std;
////////// класс Order (заказы) //////////
//Хранит информацию о заказах (номер, дата, отправитель, получатель, стоимость)
class Order
{
private:
    string shipper; // грузоотправитель
    string consignee; //грузополучатель
    int number; // номер заказа
    float price; //стоимость заказа
    int month, day; // месяц и день заказа
public:
    Order(int num, int dated, int datem, string ship, string cons, float price);
    ~Order();
    int getNumber(); //возвращает номер заказа
    int getDay(); //возвращает день заказа
    int getMonth(); //возвращает месяц заказа
    float getPrice(); //возвращает стоимость заказа
    string getCons(); //возвращает грузополучателя
    string getShip(); //возвращает грузоотправителя
};
```

OrderInputList.h

```
#pragma once
#include "OrderList.h"
#include "Global.h"
#include <iostream>
#include <vector>
#include <list>
```

```

#include <string>
#include <numeric> //для accumulate()
#include <cctype>
using namespace std;
//////////класс OrderInputList//////////
//класс OrderInputList. Это класс, отвечающий за отображение экрана,
//куда пользователь может ввести данные о новом заказе:
class OrderInputList
{
private:
    OrderList* ptrOrderList;
    string ship; // грузоотправитель
    string cons; // грузополучатель
    int num; // номер заказа
    float pr; //стоимость заказа
    int month, day; // месяц и день заказа
public:
    OrderInputList(OrderList* ptrTL)/*, string sh, string cons, int num, int mont, int day, float pr) */:
ptrOrderList(ptrTL)//, ship(sh), cons(cons), num(num), month(month), pr(pr)*/
    {
        /* тут пусто */
    }
    void setOrder(); // добавить данные о заказе
};

```

OrderList.h

```

#pragma once
#include <iostream>
#include <vector>
#include <list>
#include <string>
#include <numeric> //для accumulate()
#include <cctype>
using namespace std;
#include "Order.h"

//////////класс OrderList//////////
//класс OrderList — список всех заказов.
//Он содержит множество указателей на класс Order
//и оперирует ими при выводе
class OrderList
{
private:
    // установить указатели на заказ
    list <Order*> setPtrsOrd; // указатели на класс жильцов
    list <Order*>::iterator iter; //итератор

public:
    ~OrderList(); // деструктор (удаление списка заказов)
    void insertOrder(Order*); // добавить заказ в список
    void display(); // вывод списка заказов
    float getSumPrice();

};

```

UserInterface.h

```

#include "Global.h"
#include "OrderList.h"

```

```

#include "OrderInputList.h"
#include "ExpenseRecord.h"
#include "ExpenseInputScreen.h"
#include "AnnualReport.h"
#include <vector>
#include <list>
#include <string>
#pragma once
#include <numeric> //для accumulate()
#include <cctype>
using namespace std;

////////// ласс UserInterface//////////
//главный класс для запуска приложения:
//этот класс определяет взаимодействие юзера с программой.
class UserInterface
{
private:
    OrderList* ptrOrderList;
    OrderInputList* ptrOrderInputScreen;
    ExpenseRecord* ptrExpenseRecord;
    ExpenseInputScreen* ptrExpenseInputScreen;
    AnnualReport* ptrAnnualReport;
    char ch;
    int cs;
public:

    UserInterface();
    ~UserInterface();
    void interact();
}; // конец класса userInterfac

```

14. Листинги исполнительных файлов

AnnualReport.cpp

```

#include "AnnualReport.h"

//////////методы класса AnnualReport//////////
//Конструктор
AnnualReport::AnnualReport(ExpenseRecord* pER, OrderList* sP) : ptrER(pER), summa(sP)
{ /* пусто */
}
//-----
void AnnualReport::display()
{
    cout << "Годовой отчет\n-----\n" << endl;
    cout << "Доходы\n" << endl;
    cout << "\tДоход со всех заказов:\t\t";
    income = summa->getSumPrice();
    cout << income << endl;

    cout << "Расходы\n" << endl;
    expenses = ptrER->displaySummary();
    cout << "Сумма расходов:\t\t";
    cout << expenses << endl;
    // вычисляем прибыльность
    cout << "\nПрибыль:\t\t" << (income - expenses) << endl;
}

```

```
//-----
```

ExpenseInputScreen.cpp

```
#include "ExpenseInputScreen.h"
//////////методы класса ExpenseInputScreen//////////

// конструктор
ExpenseInputScreen::ExpenseInputScreen(ExpenseRecord* per) : ptrExpenseRecord(per)
{
    /*пусто*/
}
//-----
void ExpenseInputScreen::setExpense()
{
    int month, day;
    string category, payee;
    float amount;
    cout << "-ведите месяц (1-12): ";
    do
        cin >> month;
    while (month > 12);
    cin.ignore(80, '\n');
    cout << "-ведите день (1-31): ";
    do
        cin >> day;
    while (day > 31);
    cin.ignore(80, '\n');
    cout << "-ведите категорию расходов (ремонт, налоги): ";
    getaLine(category);
    cout << "-ведите получателя: ";
    getaLine(payee);
    cout << "-ведите сумму (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);
}
//-----
```

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 << "\nДата\tПолучатель\tСумма\tКатегория\n"
        << "-----\n" << endl;
    if (vectPtrsExpenses.size() == 0) // В контейнере нет расходов
        cout << "***Расходов нет***\n" << endl;
    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
// используется при составлении годового отчета
float ExpenseRecord::displaySummary()
{
    float totalExpenses = 0; // Сумма по всем категориям расходов
    if (vectPtrsExpenses.size() == 0)
    {
        cout << "\tВсе категории\t0\n";
        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;
}
//-----

```

Glavn.cpp

```

#include "Global.h"
#include "AnnualReport.h"
#include "Expense.h"
#include "ExpenseInputScreen.h"
#include "ExpenseRecord.h"
#include "Order.h"
#include "OrderInputList.h"
#include "OrderList.h"
#include "UserInterface.h"
int main()

```



```

{

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    UserInterface theUserInterface;
    theUserInterface.interact();
    return 0;
}

```

Global.cpp

```

#include "Global.h"
#include <string>
#include <iostream>
#include <vector>
#include <list>
#include <string>
#include <numeric> //для accumulate()

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;
}
//-----

```

Order.cpp

```

#include "Order.h"

//////////методы класса Order//////////
//в конструкторе задаём номер заказа, дату, отправителя, получателя и стоимость
Order::Order(int num, int dated, int datem, string ship, string cons, float price) : number(num), day(dated),
month(datem), shipper(ship), consignee(cons), price(price)
{
    /* тут пусто */
}
//-----
Order::~Order() // деструктор
{
    /* тут тоже пусто */
}
//-----
int Order::getNumber() //геттер возвращает номер заказа
{
    return number;
}
//-----
int Order::getDay() //геттер возвращает день

```

```

{
    return day;
}
//-----
int Order::getMonth() //геттер возвращает месяц
{
    return month;
}
//-----
float Order::getPrice() //геттер возвращает стоимость заказа
{
    return price;
}
//-----
string Order::getShip() //геттер возвращает грузоотправителя
{
    return shipper;
}
//-----
string Order::getCons() //геттер возвращает грузоприемателя
{
    return consignee;
}
//-----

```

OrderInputList.cpp

```

#include "OrderInputList.h"
//////////метод класса OrderInputList//////////
void OrderInputList::setOrder() // добавить данные о заказе
{
    cout << "-ведите номер заказа: " << endl;
    cin >> num;
    cin.ignore(80, '\n');
    cout << "-ведите день " << endl;
    do {
        cin >> day;
        cin.ignore(80, '\n');
    } while (day > 31);
    cout << "-ведите месяц: " << endl;
    do {
        cin >> month;
        cin.ignore(80, '\n');
    } while (month > 12);

    cout << "-ведите имя отправителя: " << endl;
    getaLine(ship);
    cout << "-ведите имя получателя: " << endl;
    getaLine(cons);
    cout << "-ведите цену: " << endl;
    cin >> pr;
    cin.ignore(80, '\n');
    Order* ptrOrder = new Order(num, day, month, ship, cons, pr); // создать заказ
    ptrOrderList->insertOrder(ptrOrder); // занести в список заказов
}

```

OrderList.cpp

```
#include "OrderList.h"
//////////методы класса OrderList//////////
OrderList::~OrderList() // деструктор
{
    while (!setPtrsOrd.empty()) // удаление всех заказов,
    { // удаление указателей из контейнера
        iter = setPtrsOrd.begin();
        delete* iter;

        setPtrsOrd.erase(iter);
    }
}
//-----
void OrderList::insertOrder(Order* ptrT)
{
    setPtrsOrd.push_back(ptrT); // вставка нового заказа в список
}

//-----
float OrderList::getSumPrice() // получить сумму стоимости всех заказов
{
    float sum = 0;
    iter = setPtrsOrd.begin();
    while (iter != setPtrsOrd.end())
    { // Считает сумму всех доходов от заказов
        sum += (*iter)->getPrice();

        iter++;
    }
    return sum;
}
//-----
void OrderList::display() // вывод списка заказов
{
    cout <<
    "\nНомер#\tДата\tОтправитель\tПолучатель\tДоход\n-----\n";
    if (setPtrsOrd.empty()) // если список заказов пуст
        cout << "***Нет заказов***\n" << endl; // выводим запись, что он пуст
    else
    {
        iter = setPtrsOrd.begin();
        while (iter != setPtrsOrd.end()) // распечатываем все заказы
        {
            cout << (*iter)->getNumber() << " || " << (*iter)->getMonth() << "/" << (*iter)->getDay()
            << " || " << (*iter)->getShip() << " || " << (*iter)->getCons() << " || " << (*iter)->getPrice() << endl;
            *iter++;
        }
    }
}
//-----
```

UserInterface.cpp

```
#include "UserInterface.h"

//////////методы класса userInterface//////////
```



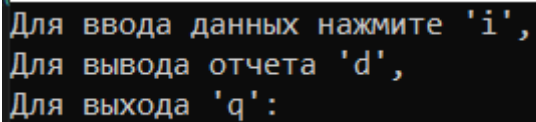
```

        ptrAnnualReport = new AnnualReport(ptrExpenseRecord, ptrOrderList);
        ptrAnnualReport->display();
        delete ptrAnnualReport;
        break;
    default: cout << "Неизвестная функция вывода\n";
        break;
    } // конец switch
} // конец elseif
else if (ch == 'q')
    return; // выход
else
    cout << "Неизвестная функция. Нажимайте только 'i', 'd' или 'q'\n";
} // конец while
} // конец interact()

```

15. Руководство пользователя

При запуске программы выводится главное меню, где пользователю предлагается выбрать действие.

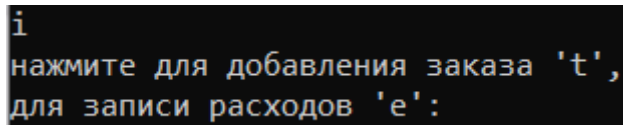


```

Для ввода данных нажмите 'i',
Для вывода отчета 'd',
Для выхода 'q':

```

- Если ввести “i”, пользователю будет предоставлен выбор, что вводить (заказ или расход)

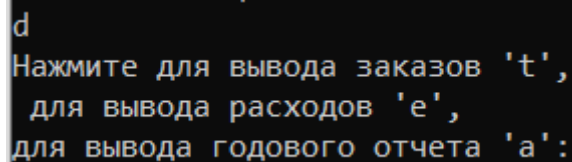


```

i
нажмите для добавления заказа 't',
для записи расходов 'e':

```

- Если ввести “d”, пользователю будет предоставлен выбор, что выводить (список заказов, список расходов или годовой отчет)



```

d
Нажмите для вывода заказов 't',
для вывода расходов 'e',
для вывода годового отчета 'a':

```

- Если ввести “q”, работа программы будет завершена

Порядок добавления заказа:

```

известной функции вывода
Для ввода данных нажмите 'i',
Для вывода отчета 'd',
Для выхода 'q':
i
нажмите для добавления заказа 't',
для записи расходов 'e':
t
Введите номер заказа:
111
Введите день
12
Введите месяц:
12
Введите имя отправителя:
Иванов И.И.
Введите имя получателя:
Сидоров С.С.
Введите цену:
5000

```

Порядок вывода списка заказов:

```

Для ввода данных нажмите 'i',
Для вывода отчета 'd',
Для выхода 'q':
d
Нажмите для вывода заказов 't',
для вывода расходов 'e',
для вывода годового отчета 'a':
t

Номер#   Дата      Отправитель   Получатель   Доход
-----
111 || 12/12 || Иванов И.И. || Сидоров С.С. || 5000

```

Порядок добавления расхода:

```

Для ввода данных нажмите 'i',
Для вывода отчета 'd',
Для выхода 'q':
i
нажмите для добавления заказа 't',
для записи расходов 'e':
e
Введите месяц (1-12): 12
Введите день (1-31): 12
Введите категорию расходов (ремонт, налоги): Топливо
Введите получателя: АО "Роснефть"
Введите сумму (39.95): 2400

```

Порядок вывода списка расходов:

```

Для ввода данных нажмите 'i',
Для вывода отчета 'd',
Для выхода 'q':
d
Нажмите для вывода заказов 't',
для вывода расходов 'e',
для вывода годового отчета 'a':
e

Дата      Получатель      Сумма      Категория
-----
12/12     АО "Роснефть"          2400      Топливо

```

Порядок вывода годового отчета:

```

Для ввода данных нажмите 'i',
Для вывода отчета 'd',
Для выхода 'q':
d
Нажмите для вывода заказов 't',
для вывода расходов 'e',
для вывода годового отчета 'a':
a
Годовой отчет
-----

Доходы

        Доход со всех заказов:          5000
Расходы

        Топливо 2400
Сумма расходов:          2400

Прибыль:          2600

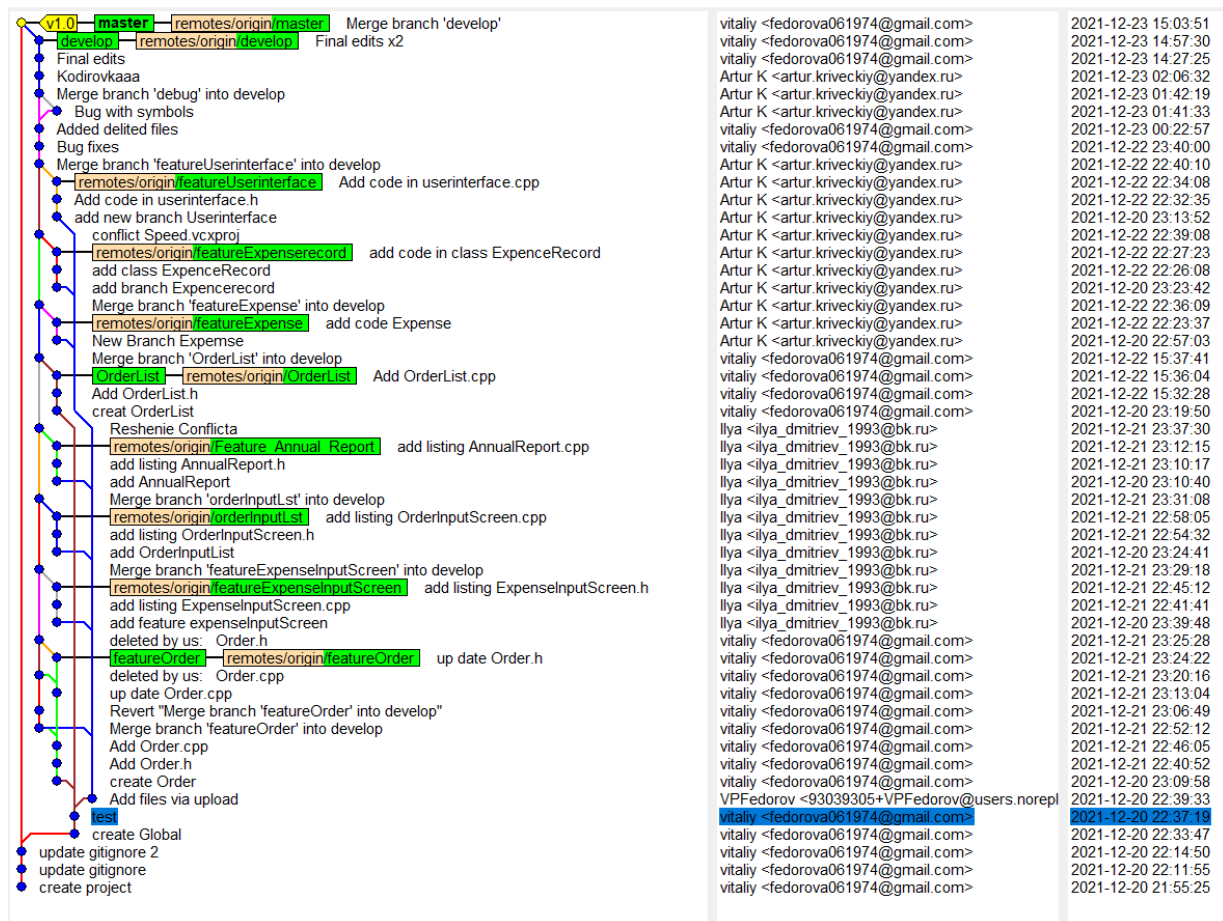
```

История проекта

Удаленный репозиторий проекта:

<https://github.com/VPFedorov/speed.git>

Графическая интерпретация истории проекта:



Описание коммитов:

- 1) Merge branch 'develop' // Выполнили слияние ветки develop с веткой master, поставил тег
- 2) Final edits x2 // Поправил пропущенный файл после сбоя кодировки
- 3) Fina edits // Поправил русский текст после сбоя кодировки
- 4) Kodirovkaaa // Изменили кодировку
- 5) Merge branch 'debug' into develop // Выполнили слияние ветки debug с веткой develop
- 6) Bug with symbols // Произошел баг с отрисовкой символов
- 7) Added delited files // Добавили удаленные файлы
- 8) Bug fixes // Исправил ошибки связанные с неправильным названием
- 9) Merge branch 'featureUserInterface' into develop // Выполнили слияние ветки featureUserInterface с веткой develop
- 10) Add code in UserInterface.cpp // Добавили код в файл UserInterface.cpp
- 11) Add code in UserInterface.h // Добавили код в файл UserInterface.h
- 12) Add new branch UserInterface // Создали ветку UserInterface и добавили файлы

- 13) conflict Speed.vcxproj // Решили конфликт в файле Speed.vcxproj
- 14) Add code in class ExpenseRecord // Добавлен код в файл класса ExpenseRecord
- 15) add class ExpenceRecord // Добавлен класс ExpenseRecord
- 16) add branch Expencerecord // Добавлена ветка ExpenseRecord
- 17) Merge branch 'featureExpense' into develop // Выполнили слияние ветки featureExpense с веткой develop
- 18) add code Expense // Добавлен код в файл класса Expense
- 19) New Branch Expemse // Добавлена ветка Expemse
- 20) Merge branch 'OrderList' into develop // Выполнили слияние ветки OrderList с веткой develop
- 21) Add OrderList.cpp // Добавлен файл OrderList.cpp
- 22) Add OrderList.h // Добавлен файл OrderList.h
- 23) creat OrderList // Создан класс OrderList
- 24) Reshenie Conflicta // решили конфликт
- 25) add listing AnnualReport.cpp // Добавлен код в файл AnnualReport.cpp
- 26) add listing AnnualReport.h // Добавлен код в файл AnnualReport.h
- 27) add AnnualReport // Добавлен Класс AnnualReport
- 28) Merge branch 'orderinputLst' into develop // Выполнили слияние ветки orderinputLst с веткой develop
- 29) add listing OrdernputScreen.cpp // Добавлен код в файл OrdernputScreen.cpp
- 30) add listing OrderinputScreen.h // Добавлен код в файл OrdernputScreen.h
- 31) add OrderinputList // Добавлен Класс OrderinputList
- 32) Merge branch 'featureExpenseInputScreen' into develop // Выполнили слияние ветки featureExpenseInputScreen с веткой develop
- 33) Add listing ExpenseInputScreen.h // Добавлен код в файл ExpenseInputScreen.h
- 34) add listing ExpenseInputScreen.cpp // Добавлен код в файл ExpenseInputScreen.cpp
- 35) add feature expenselInputScreen // Добавлена ветка feature expenselInputScreen
- 36) deleted by us: Order.h // Удален файл Order.h
- 37) up date Order.h // Обновлен файл Order.h

- 38) deleted by us: Order.cpp // Удален файл Order.cpp
- 39) up date Order.cpp // Обновлен файл Order.cpp
- 40) Revert "Merge branch featureOrder' into develop" // Отмена слияния веток
- 41) Merge branch featureOrder' into develop // Слита ветка featureOrder с Develop
- 42) Add Order.cpp // Написан код в файле order.cpp
- 43) Add Order.h // Написан код в файле order.h
- 44) create Order // создана ветка Order, добавлены заголовочный и исполняемый файлы Order.h и Order.cpp
- 45) Add files via upload // Добавлена ветка Develop
- 46) Test // Убрал из отслеживания файл Speed.vcxproj
- 47) Create Global // Создал заголовочный и исполняемый файлы global
- 48) Update gitignore 2 // Обновили gitignor 2 раз, исправили ошибки
- 49) Update gitignore // Обновили gitignore
- 50) Create project // Создали основу проекта, добавлен gitignore

Заключение

Разработка проекта проводилась в среде программирования Visual Studio 2019 на языке программирования C++ с использованием системы контроля версий Git и удаленного репозитория на GitHub. Все запланированные прецеденты были реализованы, сбоев и зависаний в работе программы не наблюдается. В проекте используются принципы раздельной компиляции, очистка памяти реализована, неиспользуемые переменные отсутствуют. Конструкции, без которых можно обойтись, не используются. К отчету прикладываются следующие диаграммы: диаграмма прецедентов, диаграммы вариантов использования, диаграмма классов и диаграммы последовательности.

Цель данной работы была достигнута.

