



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Новосибирский государственный университет экономики и управления «НИНХ»  
(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра базовой подготовки

## **КУРСОВАЯ РАБОТА**

Дисциплина: Языки программирования

Ф.И.О студента: Алёхин Никита Павлович

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Специализация: Программная инженерия

Номер группы: ФИ202

Номер зачетной книжки: 220107

Номер варианта курсовой работы: 7

Проверил: Ковригин Алексей Викторович, канд. педагогических наук,  
преподаватель

Новосибирск 2024

## Оглавление

1. Создание классов модели.....	3
2. Создание класса представления. ....	6
3. Создание класса управления. ....	11
4. Вся программа.....	11

## 1. Создание классов модели

Класс `orders` представляет собой данные и бизнес-логику приложения. Класс содержит методы такие как метод добавление данных в файл, также для получения данных, их обработки и передачи их представлению с помощью контроллера для отображения пользователю.

Метод добавление данных в файл:

```
void append_orders_to_file()
{
    ofstream out;
    out.Open("orders.txt", ios::app);
    if (out.is_open())
    {
        for (int i=0;i<num;i++)
        {
            entry object = orders[i];
            out << object.getSender().getName() << " — " <<
object.getSender().getSurname() << " — " << object.getSender().getNumber() << " — ";
            out << object.getSender_adres() << " — " <<
object.getRecipient().getName() << " — " << object.getRecipient().getSurname() << " — " <<
object.getRecipient().getNumber() << " — ";
            out << object.getRecipient_adres() << " — " << object.getTransport() << " — " <<
object.getWeight() << " — " << object.getVolume() << " — " <<
object.getDate() << ":";
        }
    }
    out.close();
}
```

Метод получение данных из файла:

```
void load_orders()
{
```

```

ifstream in;
in.open("orders.txt");
if (in.is_open())
{
    string str;
    while (getline(in, str))
    {
        string s_name = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string s_surname = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string s_number = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string s_adres = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string r_name = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string r_surname = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string r_number = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string r_adres = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string transport = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        int weight = stoi(str.substr(0, str.find(" — ")));
        str = str.substr(str.find(" — ") + 3);
        int volume = stoi(str.substr(0, str.find(" — ")));
        str = str.substr(str.find(" — ") + 3);
    }
}

```

```

        string date = str.substr(0, str.find(":"));

        entry* str = new
entry(s_name,s_surname,s_number,s_adres,r_name,r_surname,r_number,r_adres,transp
ort,weight,volume,date);

        entry* newzapros = new entry[num + 1];
        for(int i=0;i<num;i++)
        {
            newzapros[i] = orders[i];
        }
        newzapros[num] = *str;
        num++;
        orders = newzapros;
    }
}
in.close();
}

```

(human) — это пользовательский класс, имеющий имя, фамилию, номер телефона. Эти поля должны иметь символьный тип данных. Наиболее подходящий для нашей программы — тип данных string. Все эти свойства не являются общедоступными и определены частным модификатором доступа (private).

private:

```
string name, surname, number;
```

Класс должен содержать функции возвращающие имя, фамилию и его номер телефона, а также создающие и/или сохраняющие образ класса в виде файла с расширением “.h”. Так как данные функции должны быть вызваны внешними функциями они определены общим модификатором доступа (public):

public:

```

human()
{
    name = "Иван";
    surname = "Иванов";
    number = "8-880-555-35-55";
}

```

```

human(string name, string surname, string number)
{
    this->name = name;
    this->surname = surname;
    this->number = number;
}
string print()
{
    return surname + " " + name + " тел. " + number;
}
string getName(){return name;}
string getSurname(){return surname;}
string getNumber(){return number;}

```

## 2. Создание класса представления.

Класс представления Entry содержит метод создания заказов, где пользователь вводит данные. Также содержит метод print который выводит всю информацию о заказе.

```

void create_order()
{
    cout << "Вид доставки: " << endl;
    view = 0;
    for (int i=0;i<3;i++)
    {
        cout << "[" << i + 1 << "]" << vid_order[i] << endl;
    }
    while(view < 1 || view > 3)
    {
        cin>>view;
    }
    if (view == 1)// Доставка по городу
    {

```

```

int type1 = 0;
for (int i=0;i<2;i++)
{
    cout << "[" << i + 1 << "]" " << gorod_type_transport[i] << endl;
}
while(type1 < 1 || type1 > 2)
{
    cin>>type1;
}
transport = gorod_type_transport[type1 - 1];
string city,street, house;
cout << "Укажите адрес отправителя: " << endl;
cout << "Введите город: ";
cin >> city;
cout << "Введите улицу: ";
cin >> street;
cout << "Введите номер дома: ";
cin >> house;
send_adres = city + ", " + street + " " + house;
cout << "Укажите адрес получателя: " << endl;
cout << "Введите улицу: ";
cin >> street;
cout << "Введите номер дома: ";
cin >> house;
recip_adres = street + " " + house;
}
else if (view == 2) // Доставка по России
{
    int type2 = 0;

```

```

for (int i=0;i<2;i++)
{
    cout << "[" << i + 1 << "]" " << rus_type_transport[i] << endl;
}
while(type2 < 1 || type2 > 2)
{
    cin>>type2;
}
transport = rus_type_transport[type2 - 1];
string city;
cout << "Укажите адрес отправителя: " << endl;
cout << "Введите город: ";
cin >> city;
send_adres = city;
cout << "Укажите адрес получателя: " << endl;
cout << "Введите город: ";
cin >> city;
recip_adres = city;
}
else if (view == 3)// Международная доставка
{
    int type3 = 0;
    for (int i=0;i<3;i++)
    {
        cout << "[" << i + 1 << "]" " << mejnarod_type_transport[i] << endl;
    }
    while(type3 < 1 || type3 > 3)
    {
        cin>>type3;
    }
}

```



```

    }
    transport = mejnarod_type_transport[type3 - 1];
    string country, city;
    cout << "Укажите адрес отправителя: " << endl;
    cout << "Введите страну: ";
    cin >> country;
    cout << "Введите город: ";
    cin >> city;
    send_adres = country + ", " + city;
    cout << "Укажите адрес получателя: " << endl;
    cout << "Введите страну: ";
    cin >> country;
    cout << "Введите город: ";
    cin >> city;
    recip_adres = country + ", " + city;
}
cout << "Укажите дату перевозки шаблон(**.**.****)" << endl;
string data;
cin >> data;
while(!proverka_dat(data))
{
    cin >> data;
}
date = data;
cout << "Укажите вес (кг.): " << endl;
int ves = 0;
while(ves < 1)
{
    cin >> ves;

```

```

    }
    weight = ves;
    int vol = 0;
    cout << "Укажите объём (м^3): " << endl;
    while(vol < 1)
    {
        cin >> vol;
    }
    volume = vol;
    cout << "Укажите ФИ отправителя и его номер тел.: " << endl;
    string name, surname, number;
    cout << "Введите фамилию: ";
    cin >> surname;
    cout << "Введите имя: ";
    cin >> name;
    cout << "Введите номер телефона: ";
    cin >> number;
    sender = *(new human(name, surname, number));

    cout << "Укажите ФИ получателя и его номер тел.: " << endl;
    cout << "Введите фамилию: ";
    cin >> surname;
    cout << "Введите имя: ";
    cin >> name;
    cout << "Введите номер телефона: ";
    cin >> number;
    recipient = *(new human(name, surname, number));
}

```

Метод print

```

void print()
{
    cout << "отправитель: "<<sender.print() << " Адрес: " << send_adres
<< "\nПолучатель: "<< recipient.print() << " Адрес: " << recip_adres << endl;

    cout << "Способ доставки: " << transport << " Вес: " << weight << "
кг. Объём: " << volume << " (м^3) ?? Дата: " << date << "\n" << endl;
}

```

### 3. Создание класса управления.

Класс управления Controller — управляющий класс, который будет отслеживать введенные пользователем данные и соответственно изменять модель. Класс содержит в себе функцию начала работы программы, которую она будет рекурсивно вызывать до получения команды остановки программы.

### 4. Вся программа

Цельная программа будет выглядеть следующим образом:

```

#include <iostream>
#include "controller.h"
#include <stdlib.h>
#include <time.h>
#include <windows.h>
using namespace std;

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    srand(time(NULL));
    controller run;
    run.execute();
}

#include <iostream>
#include "orders.h"
using namespace std;
class controller {
private:

```

```

Orders n;
public:
controller(){ }
void execute() {
    int runner = 1;
    while (runner) {
        menu();
        int var = 0;
        while (var < 1 || var > 7) {
            cout << "Введите вариант: ";
            cin >> var;
        }

        if (var == 1) {
            n.append_order();
            cout<<"Заказ успешно отправлен!"<<endl;
        }
        if (var == 2) {
            cout<<"Выберите номер заказа, который вы хотите изменить"<<endl;
            int nom;
            while(nom < 1 || nom > n.getNum() + 1)
            {
                n.print();
                cout << "[" << n.getNum() + 1 << "]" " << "Вернуться назад." << endl;
                cin >> nom;
            }
            if ((n.getNum() + 1) != nom)
            {
                n.change_order(nom);
            }
            else{continue;}
        }

        if (var == 3) {
            n.print();
            cout<<"Выберите номер заказа, который вы хотите отменить"<<endl;
            int num; cin >> num;
            n.pop_order(num);
        }
        if (var == 4) {
            n.print();
        }
        if (var == 5) {
            n.append_orders_to_file();
        }
    }
}

```

```

        cout << "Заказы успешно добавлены в базу!" << endl;
    }
    if (var == 6) {
        n.load_orders();
        cout<<endl<<"Заказы загружены"<<endl;
    }
    if (var == 7) {
        runner = 0;
    }

}

}

void menu() {
    cout << "Команды: " << endl;
    cout << "[1] Добавить заказ" << endl;
    cout << "[2] Изменить заказ" << endl;
    cout << "[3] Отменить заказ" << endl;
    cout << "[4] Показать все заказы" << endl;
    cout << "[5] Выгрузить заказы в базу" << endl;
    cout << "[6] Загрузить заказы из базы" << endl;
    cout << "[7] Выход из программы" << endl;
}
};

```

```

#include <iostream>
#include <cstring>
#include <fstream>
#include <vector>
#include "entry.h"
using namespace std;

```

```

class Orders
{
private:
    entry *orders;
    int num;
public:
    Orders()
    {
        num = 1;
        orders = new entry[num];
        entry neworder;
    }

```

```

    for(int i=0;i<num;i++)
    {
        neworder.create_order();
        orders[i] = neworder;
    }
}
void append_order()
{
    entry zapros;
    zapros.create_order();
    entry* newzapros = new entry[num + 1];
    for(int i=0;i<num;i++)
    {
        newzapros[i] = orders[i];
    }
    newzapros[num] = zapros;
    orders = newzapros;
    num++;
}
void change_order(int num)
{
    orders[num - 1].change();
    cout << "Заказ успешно изменен." << endl;
}
void pop_order(int n)
{
    entry * newzapros = new entry[num - 1];
    for (int i=0,k=0;i < num;i++,k++)
    {
        if (i==(n-1))i++;
        if (i<num)newzapros[k] = orders[i];
    }
    n--;
    orders = newzapros;
}
void print()
{
    for(int i=0;i<num;i++)
    {
        entry all_entry = orders[i];
        cout << "[" << i + 1 << "]" ";
        all_entry.print();
    }
}

```

```

int getNum(){return num;}
void append_orders_to_file()
{
    ofstream out;
    out.open("orders.txt", ios::app);
    if (out.is_open())
    {
        for (int i=0;i<num;i++)
        {
            entry object = orders[i];
            out << object.getSender().getName() << " — " <<
object.getSender().getSurname() << " — " << object.getSender().getNumber() << " —
";
            out << object.getSender_adres() << " — " <<
object.getRecipient().getName() << " — " << object.getRecipient().getSurname()<<" —
"<<object.getRecipient().getNumber()<< " — ";
            out << object.getRecipient_adres() << " — " << object.getTransport() << "
— " << object.getWeight() << " — " << object.getVolume() << " — " <<
object.getDate() <<":";
        }
    }
    out.close();
}
void load_orders()
{
    ifstream in;
    in.open("orders.txt");
    if (in.is_open())
    {
        string str;
        while (getline(in, str))
        {
            string s_name = str.substr(0, str.find(" — "));
            str = str.substr(str.find(" — ") + 3);
            string s_surname = str.substr(0, str.find(" — "));
            str = str.substr(str.find(" — ") + 3);
            string s_number = str.substr(0, str.find(" — "));
            str = str.substr(str.find(" — ") + 3);
            string s_adres = str.substr(0, str.find(" — "));
            str = str.substr(str.find(" — ") + 3);
            string r_name = str.substr(0, str.find(" — "));
            str = str.substr(str.find(" — ") + 3);
            string r_surname = str.substr(0, str.find(" — "));
            str = str.substr(str.find(" — ") + 3);

```

```

        string r_number = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string r_adres = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        string transport = str.substr(0, str.find(" — "));
        str = str.substr(str.find(" — ") + 3);
        int weight = stoi(str.substr(0, str.find(" — ")));
        str = str.substr(str.find(" — ") + 3);
        int volume = stoi(str.substr(0, str.find(" — ")));
        str = str.substr(str.find(" — ") + 3);
        string date = str.substr(0, str.find(":"));
        entry* str = new
entry(s_name,s_surname,s_number,s_adres,r_name,r_surname,r_number,r_adres,transpo
rt,weight,volume,date);
        entry* newzapros = new entry[num + 1];
        for(int i=0;i<num;i++)
        {
            newzapros[i] = orders[i];
        }
        newzapros[num] = *str;
        num++;
        orders = newzapros;
    }
}
in.close();
}

};

#include <iostream>
#include "human"
#include <cstring>
#include <stdlib.h>
#include <time.h>
using namespace std;

class entry
{
private:
    human sender;
    human recipient;
    string date;
    string vid_order[3] = {"Доставка по городу","Доставка по
России","Международная доставка"};

```



```

        string gorod_type_transport[2] = {"Малогабаритный
транспорт","Крупногабаритный транспорт"};
        string rus_type_transport[2] = {"Крупногабаритный
транспорт","Авиаперевозка"};
        string mejnarod_type_transport[3] = {"Автотранспортная
перевозка","Авиаперевозка","Морская перевозка"};
        string transport;
        string send_adres, recip_adres;
        int weight, volume, view;
public:
    entry()
    {
        send_adres = "Новосибирск","Ленина","64";
        recip_adres = "Танковая","20";
        date = "01.01.2000";
        transport = "Малогабаритный";
        weight = 20;
        volume = 50;
    }
    entry(string s_name, string s_surname, string s_number,string send_adres,string
r_name, string r_surname, string r_number, string recip_adres,string transport, int
weight, int volume,string date)
    {
        this->sender = *(new human(s_name, s_surname, s_number));
        this->recipient = *(new human(r_name, r_surname, r_number));
        this->send_adres = send_adres;
        this->recip_adres = recip_adres;
        this->date = date;
        this->weight = weight;
        this->volume = volume;
        this->transport = transport;
    }
    void create_order()
    {
        cout << "Вид доставки: " << endl;
        view = 0;
        for (int i=0;i<3;i++)
        {
            cout << "[" << i + 1 << "]" " << vid_order[i] << endl;
        }
        while(view < 1 || view > 3)
        {
            cin>>view;
        }
    }

```

```

if (view == 1) // Доставка по городу
{
    int type1 = 0;
    for (int i=0;i<2;i++)
    {
        cout << "[" << i + 1 << "]" " << gorod_type_transport[i] << endl;
    }
    while(type1 < 1 || type1 > 2)
    {
        cin>>type1;
    }
    transport = gorod_type_transport[type1 - 1];
    string city,street, house;
    cout << "Укажите адрес отправителя: " << endl;
    cout << "Введите город: ";
    cin >> city;
    cout << "Введите улицу: ";
    cin >> street;
    cout << "Введите номер дома: ";
    cin >> house;
    send_adres = city + ", " + street + " " + house;
    cout << "Укажите адрес получателя: " << endl;
    cout << "Введите улицу: ";
    cin >> street;
    cout << "Введите номер дома: ";
    cin >> house;
    recip_adres = street + " " + house;
}
else if (view == 2) // Доставка по России
{
    int type2 = 0;
    for (int i=0;i<2;i++)
    {
        cout << "[" << i + 1 << "]" " << rus_type_transport[i] << endl;
    }
    while(type2 < 1 || type2 > 2)
    {
        cin>>type2;
    }
    transport = rus_type_transport[type2 - 1];
    string city;
    cout << "Укажите адрес отправителя: " << endl;
    cout << "Введите город: ";
    cin >> city;
}

```

```

    send_adres = city;
    cout << "Укажите адрес получателя: " << endl;
    cout << "Введите город: ";
    cin >> city;
    recip_adres = city;
}
else if (view == 3)// Международная доставка
{
    int type3 = 0;
    for (int i=0;i<3;i++)
    {
        cout << "[" << i + 1 << "]" " << mejnarod_type_transport[i] << endl;
    }
    while(type3 < 1 || type3 > 3)
    {
        cin>>type3;
    }
    transport = mejnarod_type_transport[type3 - 1];
    string country, city;
    cout << "Укажите адрес отправителя: " << endl;
    cout << "Введите страну: ";
    cin >> country;
    cout << "Введите город: ";
    cin >> city;
    send_adres = country + ", " + city;
    cout << "Укажите адрес получателя: " << endl;
    cout << "Введите страну: ";
    cin >> country;
    cout << "Введите город: ";
    cin >> city;
    recip_adres = country + ", " + city;
}
cout << "Укажите дату перевозки шаблон(**.**.****)" << endl;
string data;
cin >> data;
while(!proverka_dat(data))
{
    cin >> data;
}
date = data;
cout << "Укажите вес (кг.): " << endl;
int ves = 0;
while(ves < 1)
{

```

```

        cin >> ves;
    }
    weight = ves;
    int vol = 0;
    cout << "Укажите объём (м^3): " << endl;
    while(vol < 1)
    {
        cin >> vol;
    }
    volume = vol;
    cout << "Укажите ФИ отправителя и его номер тел.: " << endl;
    string name, surname, number;
    cout << "Введите фамилию: ";
    cin >> surname;
    cout << "Введите имя: ";
    cin >> name;
    cout << "Введите номер телефона: ";
    cin >> number;
    sender = *(new human(name, surname, number));

    cout << "Укажите ФИ получателя и его номер тел.: " << endl;
    cout << "Введите фамилию: ";
    cin >> surname;
    cout << "Введите имя: ";
    cin >> name;
    cout << "Введите номер телефона: ";
    cin >> number;
    recipient = *(new human(name, surname, number));
}
void print()
{
    cout << "отправитель: " << sender.print() << " Адрес: " << send_adres <<
"\nПолучатель: " << recipient.print() << " Адрес: " << recip_adres << endl;
    cout << "Способ доставки: " << transport << " Вес: " << weight << " кг.
Объём: " << volume << " (м^3) ?? Дата: " << date << "\n" << endl;
}
void change()
{
    cout << "Выберите пункт заказа, который хотите изменить: " << endl;
    int p = 0;
    cout << "[1] Фи отправителя и его номер тел." << endl;
    cout << "[2] Адрес отправителя: " << endl;
    cout << "[3] Способ доставки: " << endl;
    cout << "[4] Вес: " << endl;

```

```

cout << "[5] Объём: " << endl;
cout << "[6] Дату:" << endl;
while (p < 1 || p > 6)
{
    cin >> p;
}
if (p == 1)
{
    string change_name, change_surname, change_number;
    cout << "Введите фамилию: "; cin >> change_surname;
    cout << "Введите имя: "; cin >> change_name;
    cout << "Введите номер: "; cin >> change_number;
    recipient = *(new human(change_name, change_surname, change_number));
}
else if (p == 2)
{
    if (getView() == 1)
    {
        string city, street, house;
        cout << "Укажите новый адрес отправителя: " << endl;
        cout << "Введите улицу: "; cin >> street;
        cout << "Введите номер дома: "; cin >> house;
        recip_adres = street + " " + house;
    }
    else if (getView() == 2)
    {
        string city;
        cout << "Укажите новый адрес отправителя: " << endl;
        cout << "Введите город: "; cin >> city;
        recip_adres = city;
    }
    else if (getView() == 3)
    {
        string country, city;
        cout << "Укажите новый адрес отправителя: " << endl;
        cout << "Введите страну: "; cin >> country;
        cout << "Введите город: "; cin >> city;
        recip_adres = country + ", " + city;
    }
}
else if (p == 3)
{
    if (getView() == 1)
    {

```

```

int type1 = 0;
for (int i=0;i<2;i++)
{
    cout << "[" << i + 1 << "]" " << gorod_type_transport[i] << endl;
}
while(type1 < 1 || type1 > 2)
{
    cin>>type1;
    if (type1 < 1 || type1 > 2)
    {
        cout << "Возникла ошибка, повторите снова! " << endl;
    }
}
transport = gorod_type_transport[type1 - 1];
}
else if (getView() == 2)
{
    int type2 = 0;
    for (int i=0;i<2;i++)
    {
        cout << "[" << i + 1 << "]" " << rus_type_transport[i] << endl;
    }
    while(type2 < 1 || type2 > 2)
    {
        cin>>type2;
        if (type2 < 1 || type2 > 2)
        {
            cout << "Возникла ошибка, повторите снова! " << endl;
        }
    }
    transport = rus_type_transport[type2 - 1];
}
else if (getView() == 3)
{
    int type3 = 0;
    for (int i=0;i<3;i++)
    {
        cout << "[" << i + 1 << "]" " << mejnarod_type_transport[i] << endl;
    }
    while(type3 < 1 || type3 > 3)
    {
        cin>>type3;
        if (type3 < 1 || type3 > 3)
        {

```

```

        cout << "Возникла ошибка, повторите снова! " << endl;
    }
}
transport = mejnarod_type_transport[type3 - 1];
}
}
else if (p == 4)
{
    int ves = 0;
    cout << "Укажите новый вес (кг.): ";
    while (ves < 1)
    {
        cin >> ves;
        if (ves < 1)
        {
            cout << "Возникла ошибка, повторите снова! " << endl;
        }
    }
    weight = ves;
}
else if (p == 5)
{
    int vol = 0;
    cout << "Укажите новый объём (м^3): ";
    while (vol < 1)
    {
        cin >> vol;
        if (vol < 1)
        {
            cout << "Возникла ошибка, повторите снова! " << endl;
        }
    }
    volume = vol;
}
else if (p == 6)
{
    string data;
    cout << "Укажите новую дату: ";
    cin >> data;
    while(!proverka_dat(data))
    {
        cin >> data;
        if(!proverka_dat(data))

```

```

        {
            cout << "Некорректная дата!";
        }
    }
    date = data;
}
}
bool proverka_year(int y)
{
    if (y % 400 == 0) return true;
    else {
        if (y % 100 == 0) return false;
        else {
            if (y % 4 == 0) return true;
            else return false;
        }
    }
}
bool proverka_dat(string date)
{
    int d, m, y = 0;
    int t = date.find(".");
    string day = date.substr(0, t);
    string month = date.substr(t + 1, 2);
    string year = date.substr(t * 2 + 2, 4);
    d = stoi(day);
    m = stoi(month);
    y = stoi(year);
    int * arr30;
    int * arr31;
    arr30 = new int[4]{4,6,9,11};
    arr31 = new int[7]{1,3,5,7,8,10,12};
    for (int k=0;k<4;k++)
    {
        if (m == arr30[k] && 0 < d && d < 31) return true;
    }
    for (int i;i<7;i++){
        if (m == arr31[i] && 0 < d && d < 32) return true;
    }
    if (proverka_year(y) && m == 2 && 0 < d && d < 30) return true;
    else {
        if (!(proverka_year(y)) && m == 2 && 0 < d && d < 29) return true;
        else {
            return false;
        }
    }
}

```



```

        }
    }
}
human getSender(){return sender;}
human getRecipient(){return recipient;}
string getSender_adres(){return send_adres;}
string getRecipient_adres(){return recip_adres;}
string getTransport(){return transport;}
string getDate(){return date;}
int getWeight(){return weight;}
int getVolume(){return volume;}
int getView(){return view;}

};

#include <iostream>
#include <cstring>

using namespace std;

class human
{
public:
    human()
    {
        name = "Иван";
        surname = "ИВАНОВ";
        number = "8-880-555-35-55";
    }
    human(string name, string surname, string number)
    {
        this->name = name;
        this->surname = surname;
        this->number = number;
    }
    string print()
    {
        return surname + " " + name + " тел. " + number;
    }
    string getName(){return name;}
    string getSurname(){return surname;}
    string getNumber(){return number;}

private:

```

```
    string name, surname, number;  
};
```

**Техническое задание**  
**Project Record** версия 1.0

## Оглавление

1. Термины и определения.....	3
1.1. Общие термины .....	3
1.2. Бизнес термины .....	3
1.3. Технические термины.....	3
1.4. Другие термины.....	3
2. Общие положения.....	3
2.1. Назначение документа.....	3
2.2. Цели создания Системы .....	4
2.3. Основные функциональные возможности Системы .....	4
2.4. Использование Технического Задания .....	4
3. Функциональные требования .....	4
3.1. Диаграммы Вариантов Использования.....	4
3.2. Описание Вариантов Использования .....	5
3.3. Дополнительные функциональные требования .....	6
4. Требования к Сущностям приложения .....	6
4.1. Класс Project.....	6
4.2. Класс Project_List.....	7
5. Модель данных.....	7
6. Требования к приемке-сдаче проекта.....	7
7. История изменения документа .....	8

## **1. Термины и определения**

### **1.1. Общие термины**

Система – программа «Заказы компании перевозок», требования к которому указаны в данном документе.

Компания – владелец и оператор программы «Заказы компании перевозок». TBD – To Be Defined – секция в ТЗ, которая должна быть определена позже.

FAQ – Frequently Asked Questions. Часто задаваемые вопросы.

Вариант Ипользования (ВИ) – описание поведения системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды.

Диаграмма Вариантов Ипользования (ДВИ) – диаграмма, отражающая отношения между авторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

### **1.2. Бизнес термины**

ДС – денежные средства.

### **1.3. Технические термины**

ОС – операционная система.

ИС – информационная система.

БД – база данных, место хранения информации ИС.

Проект — файл с расширением “pj”.

## **2. Общие положения**

### **2.1. Назначение документа**

В настоящем документе приводится полный набор требований к Системе, необходимых для реализации.

Подпись Заказчика и Исполнителя на настоящем документе подтверждает их согласие с нижеследующими фактами и условиями:

2.1.1. При реализации необходимо выполнить работы в объёме, указанном в настоящем Техническом Задании.

2.1.2. Все неоднозначности, выявленные в настоящем Техническом задании после его подписания, подлежат двухстороннему согласованию между Сторонами.

## 2.2. Цели создания Системы

2.2.1. С точки зрения создателей Системы:

2.2.1.1. Построить продукт для создания заказов транспортной компании.

## 2.3. Основные функциональные возможности Системы

2.3.1. Создание и изменение созданного списка заказов.

2.3.2. Чтение и вывод списка заказов.

2.3.3. Создание и сохранение заказов — файлов расширения «.txt».

2.3.4. Удаление проектов.

## 2.4. Использование Технического Задания

2.4.1. Отношения между Исполнителем и Заказчиком в отношении информации, содержащейся в настоящем Техническом Задании, регулируются договором о конфиденциальности, подписанным Исполнителем и Заказчиком [2024]г.

## 3. Функциональные требования

### 3.1. Диаграммы Вариантов Использования

На Диаграммах представлены основные Варианты Использования Системы, детальное описание которых можно найти в п. 3.2 «Описание Вариантов Использования».



Диаграмма 1. Действующие лица.

## **3.2. Описание Вариантов Ипользования**

### **3.2.1.Вариант использования «Создать заказ»**

#### **3.2.1.1. Описание**

**3.2.1.1.1.** Пользователь имеет возможность выбрать один из видов доставки.(по городу, по России, Международная)

#### **3.2.1.2. Основной поток действий для Пользователя**

**3.2.1.2.1.** Система выводит в консоли список видов доставки.

### **3.2.2.Вариант использования «Создать заказ»**

#### **3.2.2.1.Описание**

**3.2.2.1.1.** Выбрав один из видов доставки, Пользователь создаёт заказ.

#### **3.2.2.2. Предусловия**

**3.2.2.2.1.** Пользователь выбирает вид доставки.

#### **3.2.2.3.Основной поток действий для Пользователя**

**3.2.2.3.1.** Пользователь выбирает способ доставки

**3.2.2.3.2.** Пользователь в зависимости от способа доставки заполняет данные заказа.

### **3.2.3. Вариант использования «Добавить заказ»**

#### **3.2.3.1.Описание**

Пользователь имеет возможность добавить ещё несколько заказов.

#### **3.2.3.2. Предусловия**

**3.2.3.2.1.** Пользователь даёт команду добавить заказ.

#### **3.2.3.3. Основной поток действий для Пользователя**

**3.2.3.3.1.** Пользователь выбирает вид доставки.

**3.2.3.3.2.** Пользователь выбирает способ доставки.

**3.2.3.3.3.** Пользователь заполняет данные заказа.

#### **3.2.3.4. Альтернативный поток действий для Пользователя**



3.2.3.4.1. Система возвращается в меню

### **3.2.4 Вариант использования «Изменить заказ»**

#### **3.2.4.2. Описание**

Пользователь имеет возможность изменить заказ.

#### **3.2.4.3. Предусловия**

3.2.4.3.1. Пользователь даёт команду изменить заказ.

#### **3.2.4.4. Основной поток действий для Пользователя**

3.2.4.4.1. Пользователь выбирает заказ, который он хочет изменить.

3.2.4.4.2. Пользователь выбирает пункт заказа, который он хочет изменить.

#### **3.2.4.5. Альтернативный поток действий для Пользователя**

3.2.4.5.1. Система возвращается в меню.

### **3.2.5 Вариант использования «Отменить заказ».**

#### **3.2.5.2. Описание**

Пользователь имеет возможность отменить заказ.

#### **3.2.5.3. Предусловия**

3.2.5.3.1. Пользователь даёт команду отменить заказ.

#### **3.2.5.4. Основной поток действий для Пользователя**

3.2.5.4.1. Пользователь выбирает заказ, который хочет изменить.

#### **3.2.5.5. Альтернативный поток действий для Пользователя**

3.2.5.5.1. Система возвращается в меню.

### **3.2.6 Вариант использования «Показать все заказы».**

#### **3.2.6.2. Описание**

Пользователь имеет возможность посмотреть все заказы.

#### **3.2.6.3. Предусловия**

3.2.6.3.1. Пользователь даёт команду показать все заказы.

#### **3.2.6.4. Альтернативный поток действий для Пользователя**

3.2.6.4.1. Система возвращается в меню.

### **3.2.7 Вариант использования «Загрузить заказы в файл»**

#### **3.2.7.2. Описание**

Пользователь имеет возможность загрузить заказы в файл.

#### **3.2.7.3. Предусловия**

3.2.7.3.1. Пользователь даёт команду загрузить заказы в файл.

#### **3.2.7.4. Альтернативный поток действий для Пользователя.**

3.2.7.4.1. Система возвращается в меню.

### **3.2.8 Вариант использования «Выгрузить заказы из файла».**

#### **3.2.8.2. Описание**

Пользователь имеет возможность выгрузить заказы из файла.

### **3.2.8.3. Предусловия**

3.2.8.3.1. Пользователь даёт команду выгрузить заказы из файла

### **3.2.8.4. Альтернативный поток действий для Пользователя.**

3.2.8.4.1. Система возвращается в меню.

## **3.3. Дополнительные функциональные требования**

### **3.3.1. Сохранение данных о заказах в текстовом документе**

Приложение должно сохранять данные о заказах в текстовом документе.

Приложение должно сохранять данные в автоматическом режиме при выходе из программы.

### **3.3.2. Загрузка данных о заказах из текстового документа**

Приложение должно загружать данные о заказах из текстового документа.

Приложение должно загружать данные в автоматическом режиме при запуске программы.

## **4. Требования к Сущностям приложения**

### **4.1.Класс Orders**

№	Название	Тип	Описание
1	orders	entry	Список заказов
2	num	Целое число	Количество заказов

### **4.2.Класс Human**

№	Название	Тип	Описание
1	name	Строка	Имя отправителя, получателя
2	surname	Строка	Фамилия отправителя, получателя
3	number	Строка	Номер тел. Отправителя, получателя

## **5. Требования к приемке-сдаче проекта**

5.1 Исполнитель должен предоставить следующий комплект поставки при сдаче проекта:

- Техническое задание
- Исходный код Системы
- Исполняемые модули Системы
- Тестовые сценарии
- Пользовательскую документацию

5.2 Приемочные испытания должны проводиться по каждому этапу отдельно на сервере Заказчика в сроки, оговоренные договором.

5.3 Ход проведения приемочных испытаний Заказчик и Исполнитель документируют в Протоколе проведения испытаний.

5.4 На основании Протокола проведения испытаний Исполнитель совместно с Заказчиком подписывает Акт приемки-сдачи программы