**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций им Э.Т. Кренкеля**

**Отчёт о выполнении лабораторной работы №13**

**Тема:** **Реализация принципов наследования. Создание иерархии  
классов**

**по дисциплине "Системное программирование"**

**Вариант№ 9**

Принял:

преподаватель Кривоносова Н.В

Выполнил: студенты группы ЗФ-053/054

Фомина Е.А.

Ткачук А.В

Шилаковски М.Э

Семененков К.И

Санкт-Петербург

2022 год

**Цель работы**

1) Создание консольного приложения, состоящего из нескольких файлов в системе

программирования Visual Studio.

2) Создание иерархии классов с использованием простого наследования.3) Изучение принципа подстановки.

**Ссылка на GitHub:**

https://github.com/SPbCTTSFS/SPbCT\_TA\_SM\_FE\_SK

1. **Ход выполнения работы**
2. Постановка задачи.

Задача

Базовый класс:

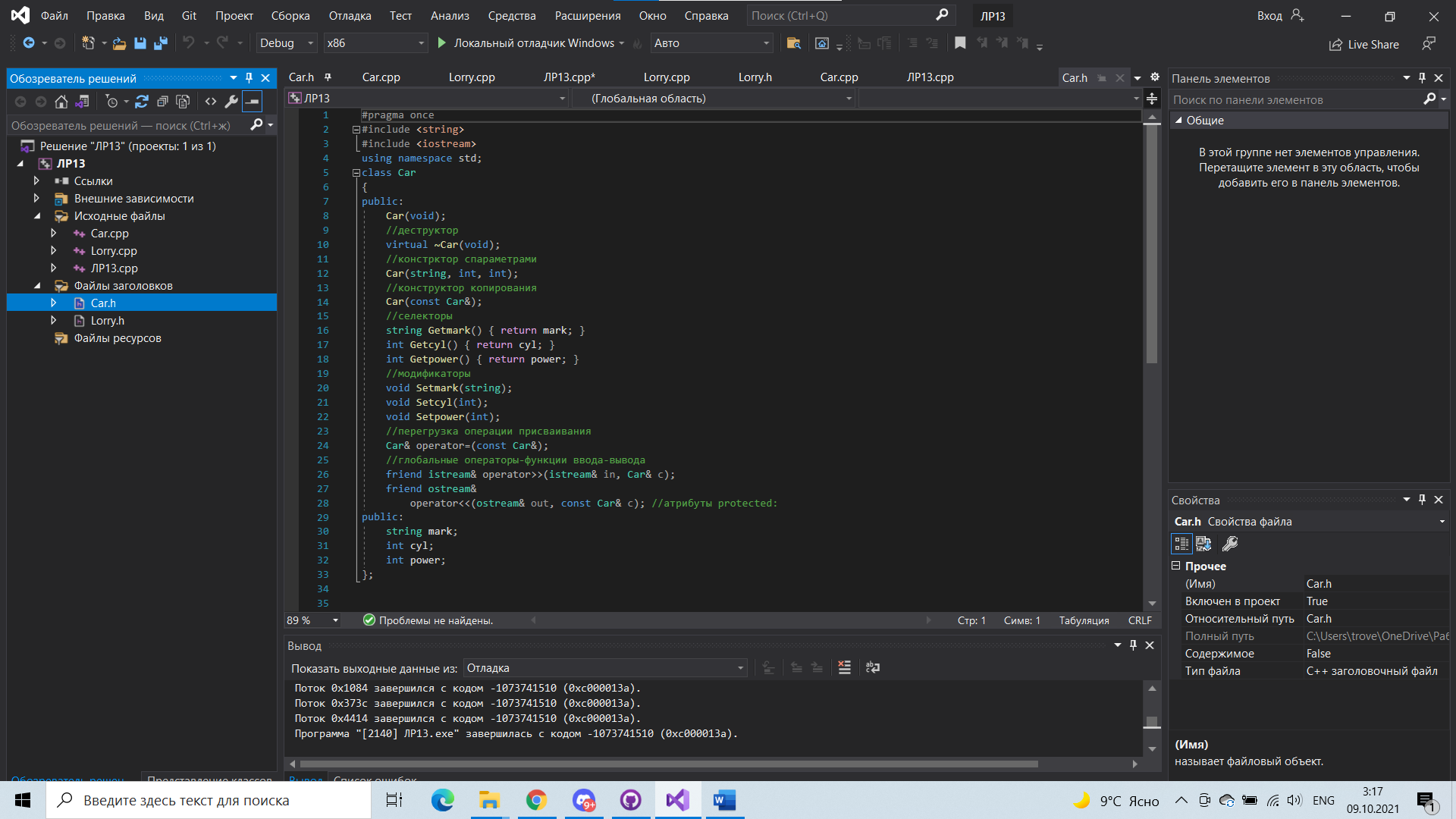
МАШИНА

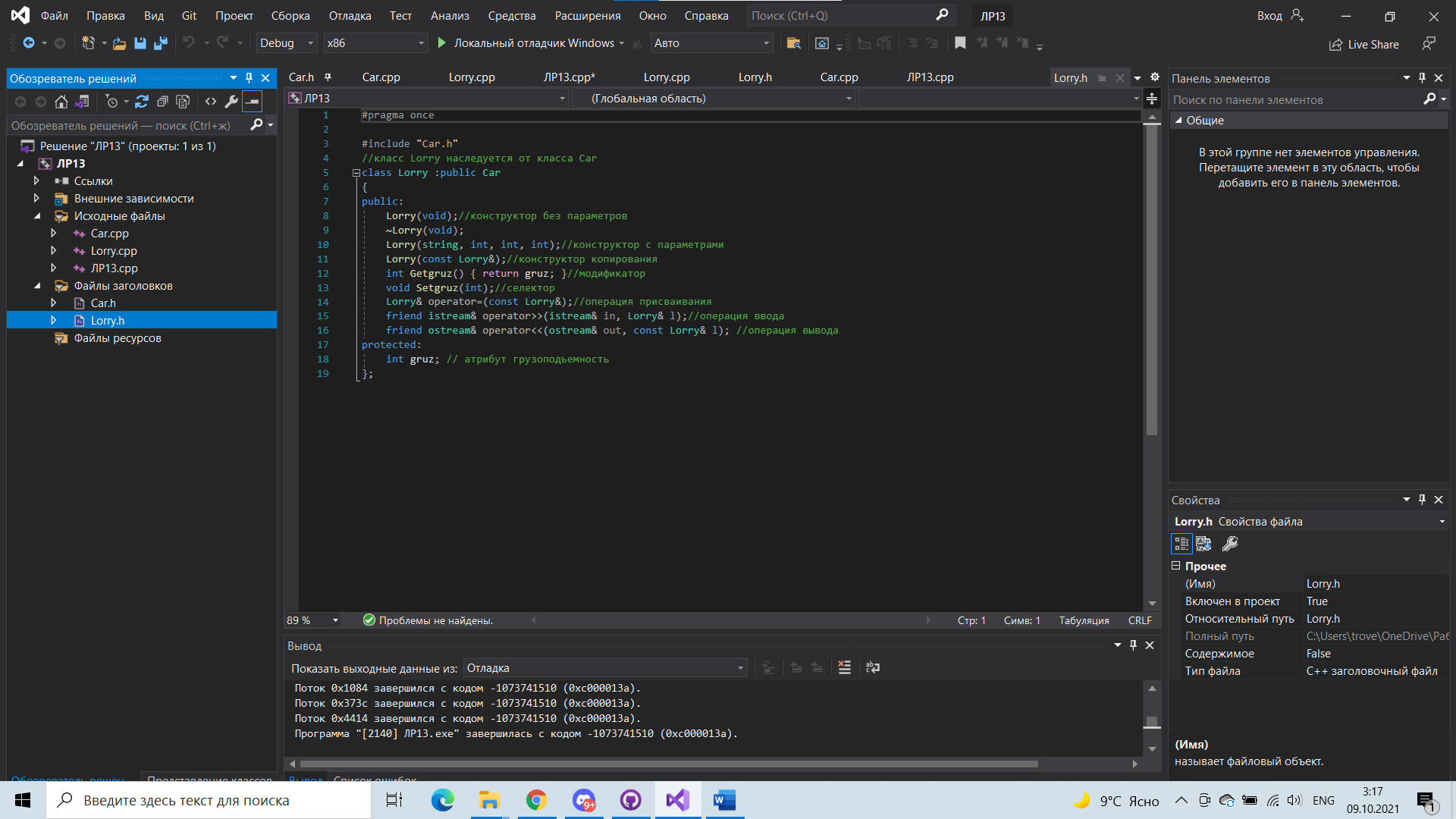
торговая\_марка - string число\_цилиндров - int мощность - int

Создать производный класс ГРУЗОВИК, добавив в него характеристику

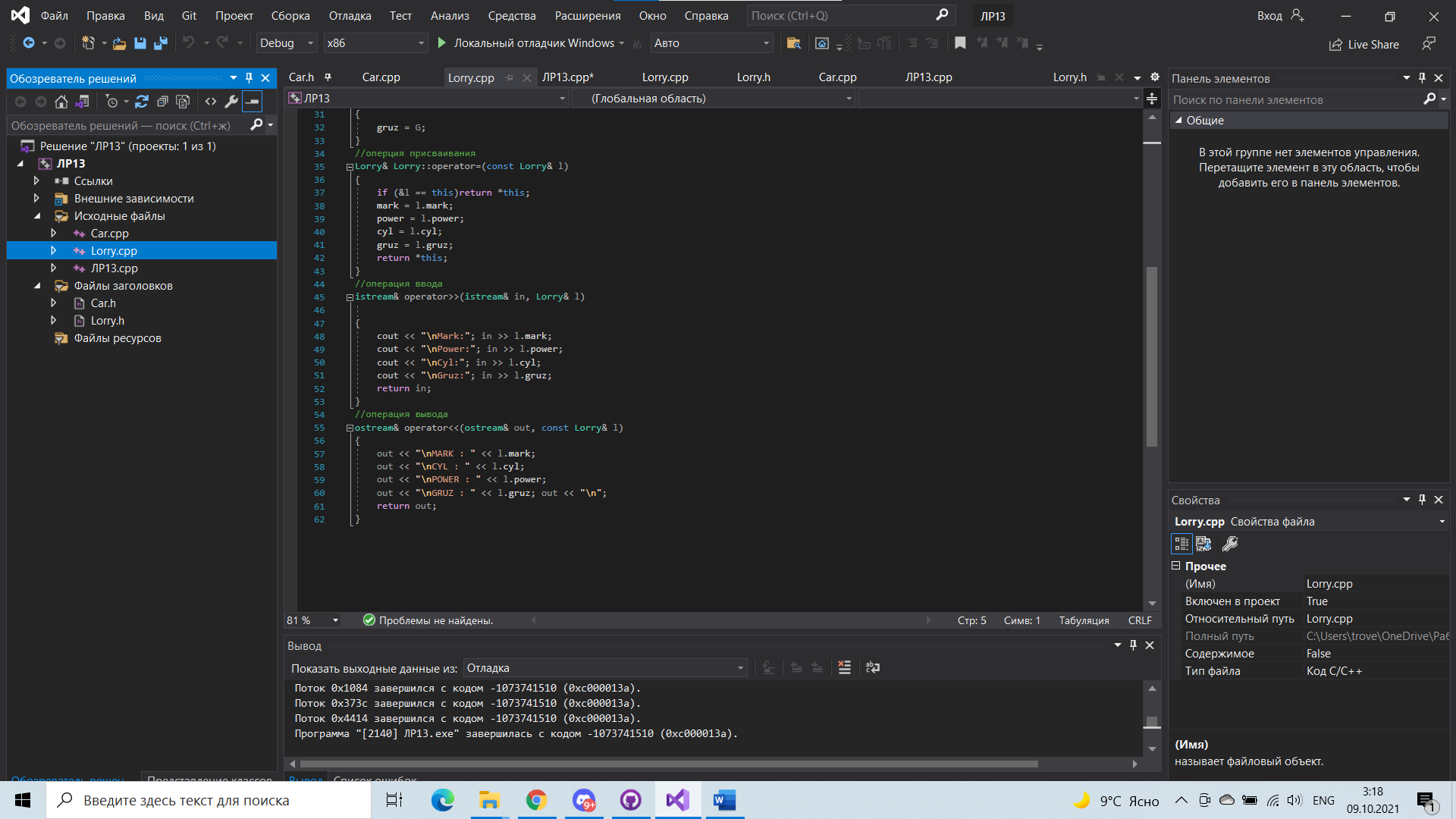
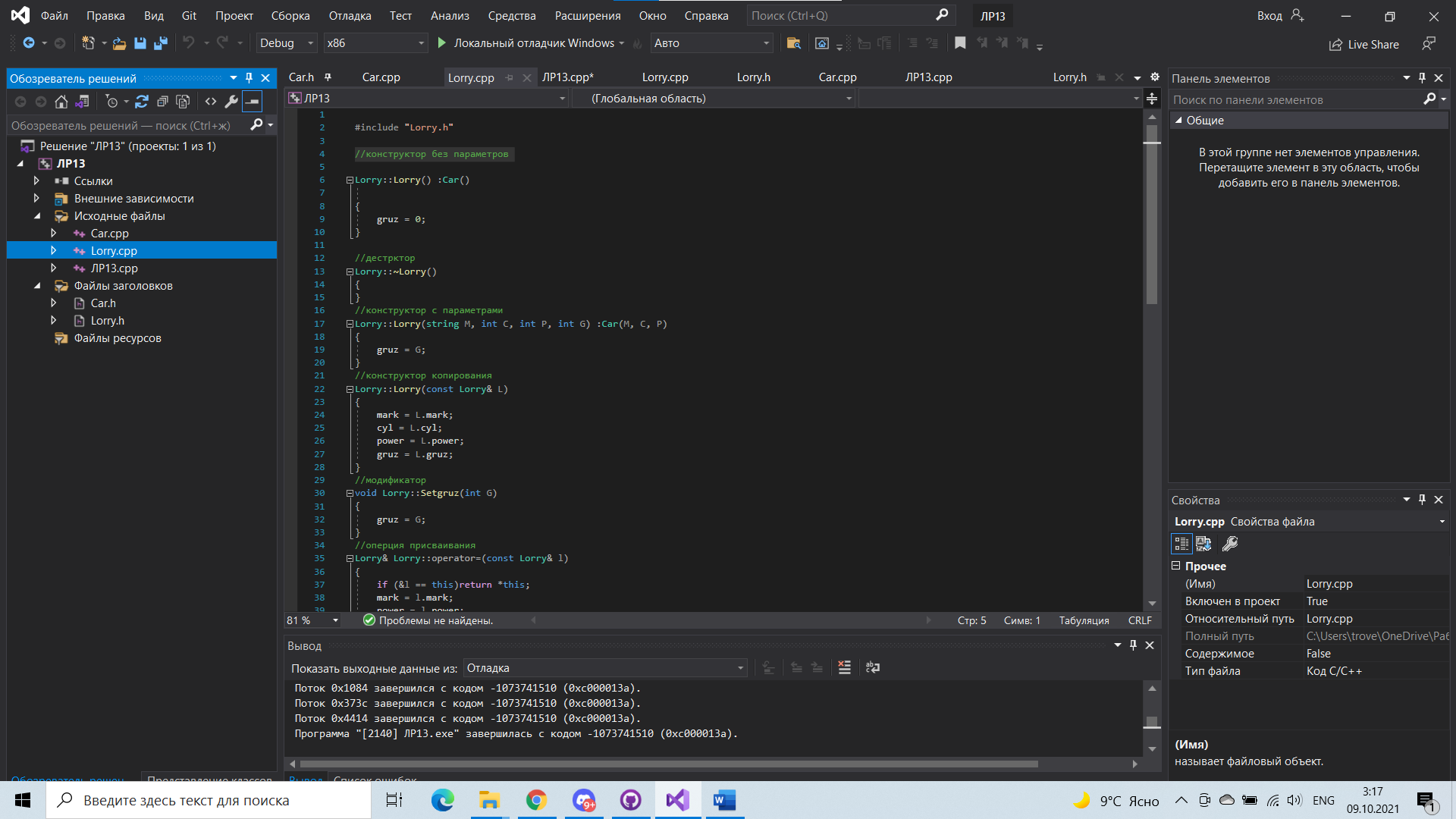
грузоподъемности кузова типа int.

1. Описание класса.

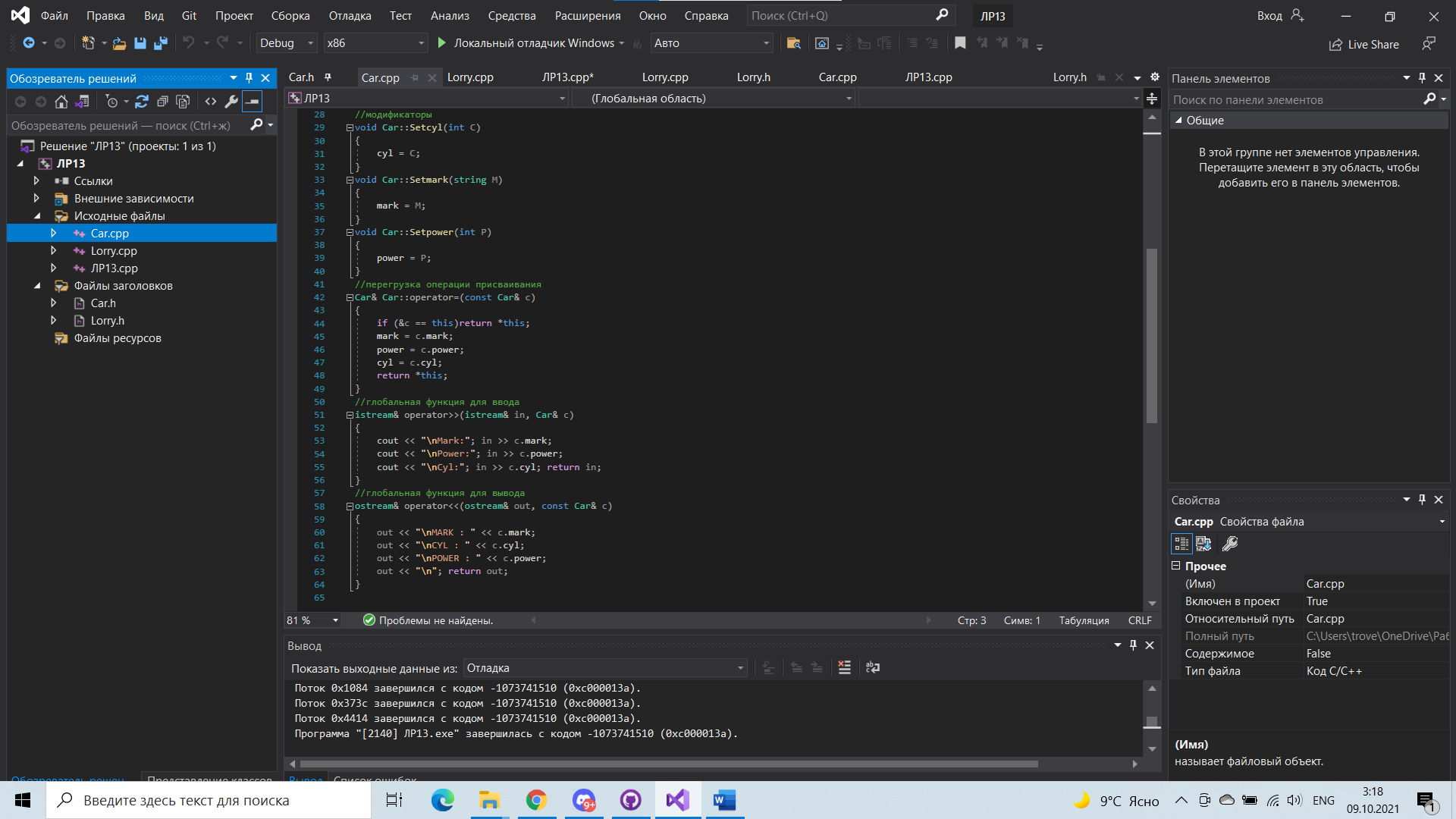
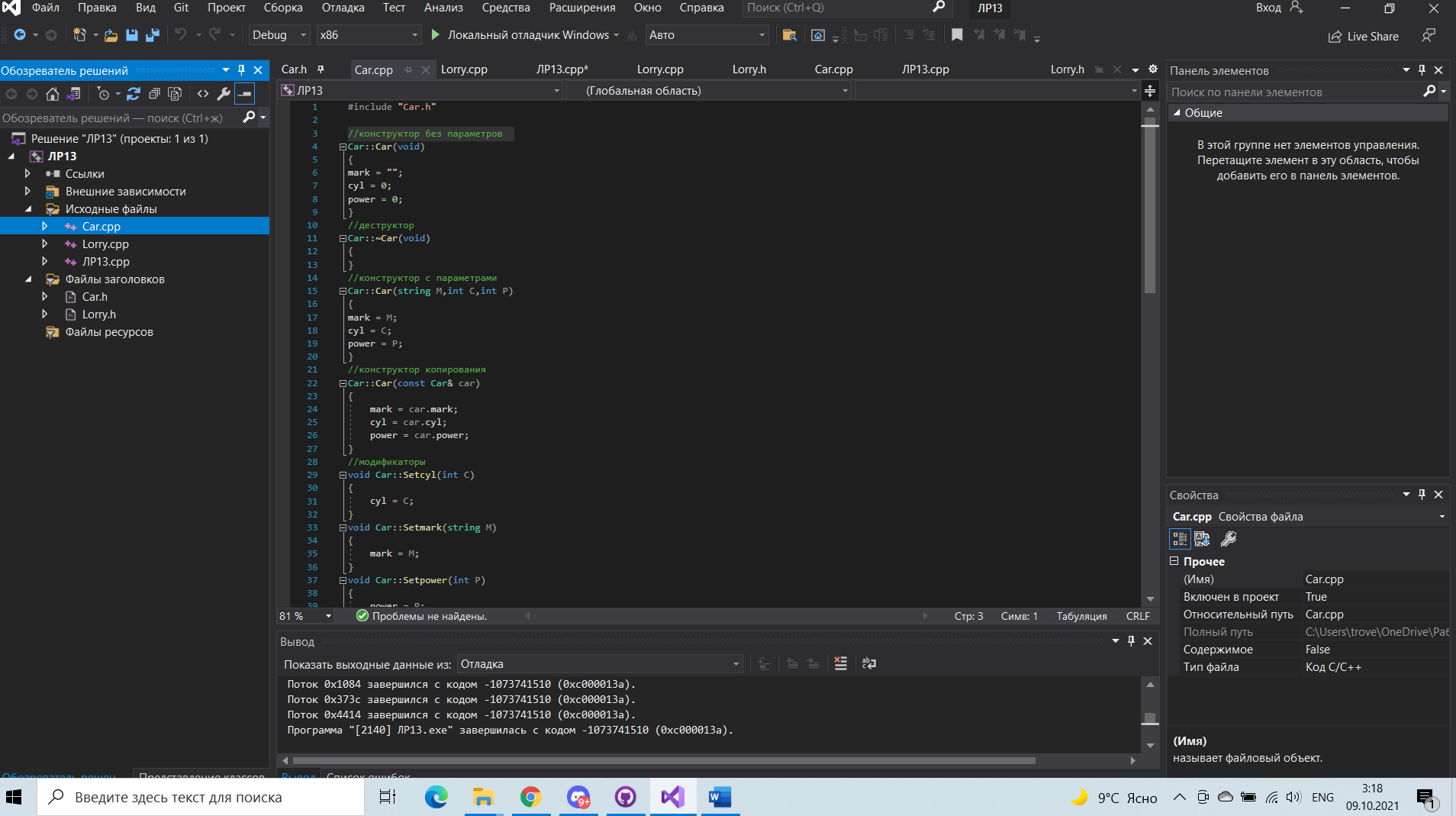




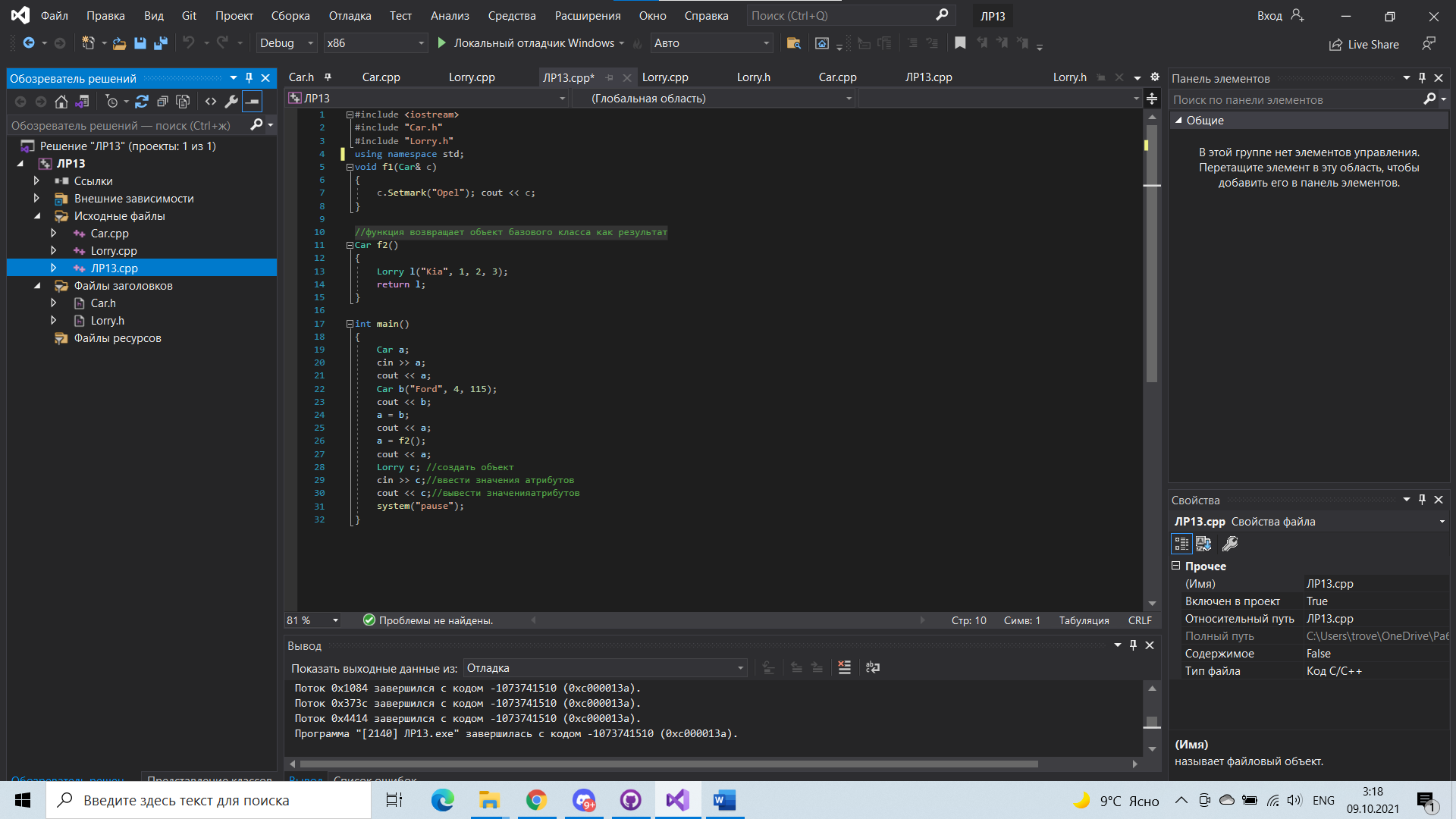
1. Определение компонентных функций.



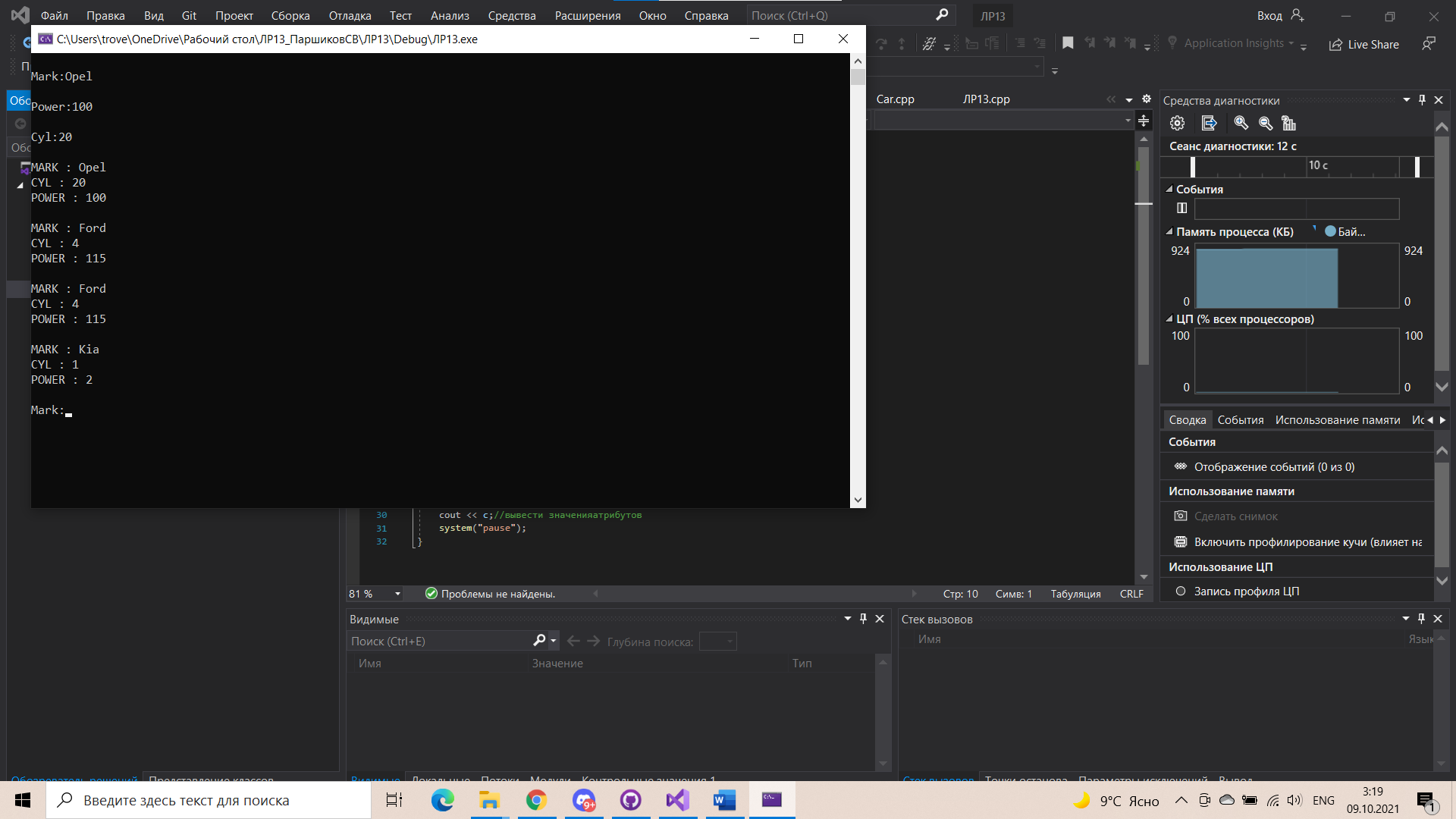
1. Определение глобальных функций.



1. Функция main().



1. Объяснение результатов работы программы.



1. **Контрольные вопросы**
2. Для чего используется механизм наследования?

Наследование позволяет реализовать типизированное повторное использование объектов, создание иерархии «тип-подтип» с сохранением подтипом всех свойств своего предка. Можно рассматривать наследование для всех трех составляющих объекта.

1. Каким образом наследуются компоненты класса, описанные со спецификатором public?

Открытое наследование является одним из наиболее используемых типов наследования. Очень редко вы увидите или будете использовать другие типы, поэтому основной упор следует сделать на понимание именно этого типа наследования. К счастью, открытое наследование является самым легким и простым из всех типов. Когда вы открыто наследуете родительский класс, то унаследованные public-члены остаются public, унаследованные protected-члены остаются protected, а унаследованные private-члены остаются недоступными для дочернего класса. Ничего не меняется.

1. Каким образом наследуются компоненты класса, описанные со

спецификатором private?

При закрытом наследовании все члены родительского класса наследуются как закрытые. Это означает, что private-члены остаются недоступными, а protected- и public-члены становятся private в дочернем классе.

1. Каким образом наследуются компоненты класса, описанные со спецификатором protected?

Этот тип наследования почти никогда не используется, за исключением особых случаев. С защищенным наследованием, public- и protected-члены становятся protected, а private-члены остаются недоступными.

1. Каким образом описывается производный класс?

Производный

класс наследует описание базового класса; затем он может быть изменен добавлением новых

членов, изменением существующих функций-членов и изменением прав доступа. С помощью

наследования может быть создана иерархия классов, которые совместно используют код и

интерфейсы.

1. Наследуются ли конструкторы?

нет

1. Наследуются ли деструкторы?

нет

1. В каком порядке конструируются объекты производных классов?

Объекты класса конструируются снизу вверх: сначала базовый, потом компонентыобъекты (если они имеются), а потом сам производный класс. Таким образом, объект

производного класса содержит в качестве подобъекта объект базового класса.

1. В каком порядке уничтожаются объекты производных классов?

Уничтожаются объекты в обратном порядке: сначала производный, потом его

компоненты-объекты, а потом базовый объект.

1. Что представляют собой виртуальные функции и механизм позднего связывания?

При применении раннего связывания, мы как бы говорим компилятору: "Я точно знаю, чего я хочу. Поэтому жестко(статически) связывай все вызовы функций". При применении механизма позднего связывания мы как бы говорим компилятору: "Я пока не знаю чего я хочу. Когда придет время, я сообщу что и как я хочу".

1. Могут ли быть виртуальными конструкторы? Деструкторы?

Конструктора нет

Деструктор есть

1. Наследуется ли спецификатор virtual?

Да

1. В чем заключается принцип подстановки?

Простыми словами принцип звучит так: Наследующий класс должен дополнять, а не замещать поведение базового класса.

1. Имеется иерархия

классов: class Student

{

int age;

public:

string name;

};

class Employee : public Student

{

protected:

string post;

};

class Teacher : public Employee

{

protected: int stage;

};

Teacher x;

Какие компонентные данные будет иметь объект х?

stage

1. Для классов Student, Employee и Teacher написать конструкторы без параметров.

Student(){};

Employee(){};

Teacher(){};

1. Для классов Student, Employee и Teacher написать конструкторы с параметрами.

Student(int i){};

Employee(int y){};

Teacher(int u){};

1. Для классов Student, Employee и Teacher написать конструкторы копирования.

Student(const a & b);

Employee(const a & b);

Teacher(const a & b);

20. Для классов Student, Employee и Teacher определить операцию присваивания

Student& operator= (const a &b);

Employee& operator= (const a &b);

Teacher& operator= (const a &b);