

车载软件开发基础

课后实践2



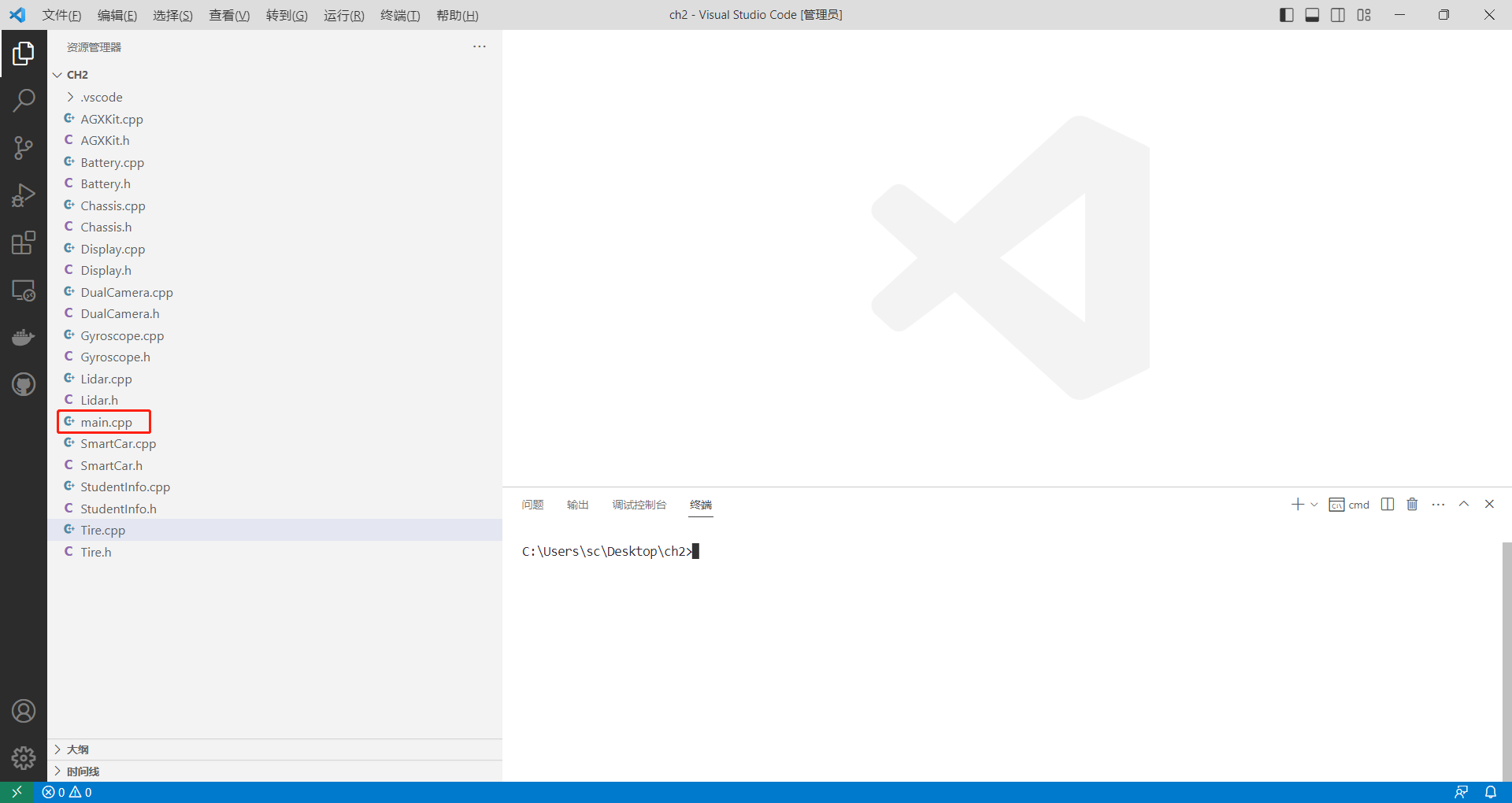
**2024至2025学年第 1 学期**

|  |  |
| --- | --- |
| 学号 | 姓名 |
| E2021077 | 沈冲 |
| 任课教师 | 刘骥 |
| 成 绩 |  |

|  |  |
| --- | --- |
| 任务书 | |
| 任务内容 | 请针对课后实践1的内容，将各模块用class改写。各模块的属性保持不变，增加相关操作行为。补充信息如下：   1. 智能小车   增加操作：   1. setID 实现编号录入 2. print 实现智能小车信息的屏幕输出 3. save 实现智能小车信息的本地文件保存 4. 底盘   增加操作：   1. 各属性的set方法 2. print 实现底盘信息的屏幕输出 3. save 实现底盘信息的本地文件保存 4. 轮胎   增加操作：   1. 各属性的set方法 2. print 实现轮胎信息的屏幕输出 3. save 实现轮胎信息的本地文件保存 4. AGX套件   增加操作：   1. 各属性的set方法 2. print 实现AGX套件的屏幕输出 3. save 实现AGX套件的本地文件保存 4. 双目摄像头   增加操作：   1. 各属性的set方法 2. print 实现双目摄像头信息的屏幕输出 3. save 实现双目摄像头信息的本地文件保存 4. 9轴陀螺仪   增加操作：   1. 各属性的set方法 2. print 实现9轴陀螺仪信息的屏幕输出 3. save 实现9轴陀螺仪信息的本地文件保存 4. 多线激光雷达   增加操作：   1. 各属性的set方法 2. print 实现多线激光雷达信息的屏幕输出 3. save 实现多线激光雷达信息的本地文件保存 4. 液晶显示屏   增加操作：   1. 各属性的set方法 2. print 实现液晶显示屏信息的屏幕输出 3. save 实现液晶显示屏信息的本地文件保存 4. 电池模块   增加操作：   1. 各属性的set方法 2. print 实现电池模块信息的屏幕输出 3. save 实现电池模块信息的本地文件保存 4. 学生信息包括：   增加操作：   1. 各属性的set方法 2. print 实现学生信息的屏幕输出 3. save 实现学生信息的本地文件保存   要求：   1. 上述方法的参数自定义。构造函数自定义 2. Main函数中实现小车信息录入、分配、保存和屏幕显示。 3. 小车各模块的保存和显示分别通过相应对象的操作完成 4. 用程序打开保存的文件，然后将相关信息显示在屏幕上（含分配的学生信息）。一次显示1辆小车信息，按n键显示下一辆小车信息，按p显示上一辆小车信息。请注意：显示第1台显示按p无效，显示第10台小车，按n无效。 |
| 程序规范 | （1）所有程序代码采用C++编写，使用git进行源代码管理；  （2）类名、变量名、函数名应符合C++的命名规范，并在代码中前后保持一致；  （3）涉及面向对象的程序，例如自定义的类，应符合面向对象的设计原则；  （4）正确使用头文件和源文件，自定义的头文件应符合头文件的编写原则，例如用条件宏定义确保头文件不被多次引用、不在头文件中进行类和函数的实现（模板除外）； |
| 报告要求 | （1）报告至少应该包括程序设计、程序效果展示、总结分析3个部分；  （2）程序设计描述组成程序的模块、类、函数以及他们之间的相互关系，若有算法，可以描述算法流程；  （3）程序效果展示除了程序运行效果截图之外，应该有必要的文字说明；  （4）总结分析可以分析实现的效果与理想情况的差异，分析导致这些差异的原因，切忌不要写成心得体会；  （5）报告应该格式规范、排版整洁、少语病和错误。 |
| 作业提交 | （1）含有git仓库（有.git目录）的完整源代码；  （2）任务报告。 |
| 评分标准 | 按照五级制打分，分为优秀、良好、中等、及格、不及格，各评分项占总成绩的比例为：  （1）任务完成情况占评分的60%；  （2）报告占评分的40%。  评分老师根据各部分的完成情况，直接给出总成绩。 |

1. **程序设计**

本程序在实践1程序的基础上，将智能小车、底盘、轮胎、AGX套件、双目摄像头、9轴陀螺仪、多线激光雷达、液晶显示屏、电池模块、学生信息10个部分分模块编写，一共编写了20个独立的每个模块的头文件和cpp文件，然后在main函数中实现小车信息录入、分配、保存和屏幕显示等功能。



1. 类设计

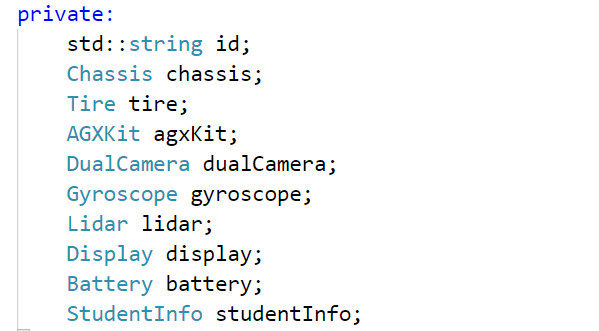
**SmartCar 类**

SmartCar 类是程序的核心，代表一辆智能小车。它包含了小车的基本信息和对各个组件(包含学生信息)的引用。

* 成员变量：

id：小车的唯一标识符。

chassis、tire、agxKit、dualCamera、gyroscope、lidar、display、battery、studentInfo，分别代表小车的不同组件信息。



* 成员函数：

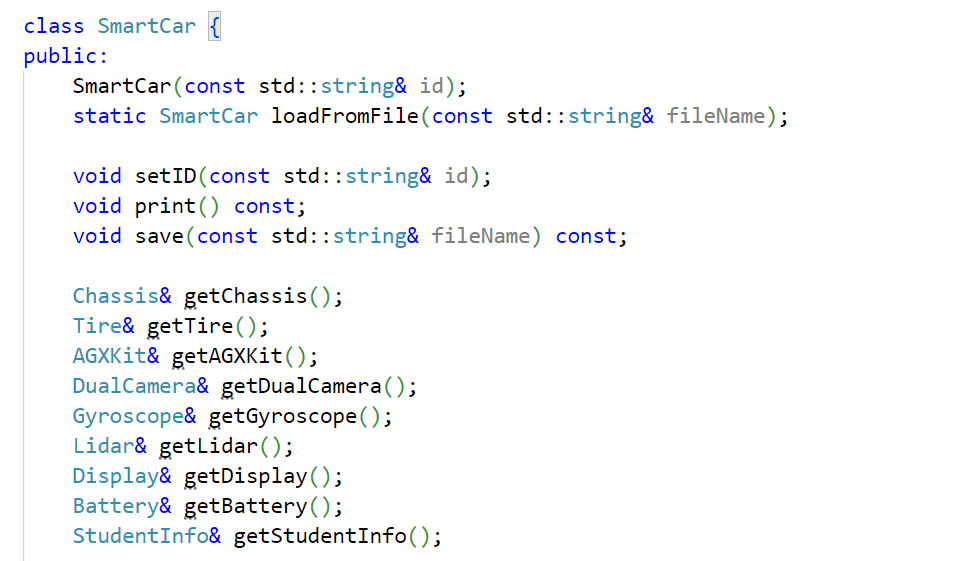
setID()：设置小车的ID。

print()：遍历所有属性并打印它们的信息。

save()：将小车的信息保存到指定的文件中。

loadFromFile()：从文件中加载小车的信息。

getXXX()：访问器函数，用于访问小车的各个属性。



**组件类**

每个组件（如 Chassis、Tire 等）都是一个独立的类，代表小车的一个部分。

* 成员变量：

各个组件特有的属性，如chassisId、tireModel等。

* 成员函数：

print()：打印组件的详细信息。

save()：将组件的信息保存到输出文件流中。

setXXX()：设置器函数，用于设置组件的属性。

1. 函数详细描述

**main() 函数**

main() 函数是程序的入口点，负责处理用户输入和程序流程。

* 流程：

初始化小车数量。

循环录入每辆小车的信息，包括学生信息。

将每辆小车的信息保存到文件中。

提供用户选项，查看或退出程序。

如果用户选择查看，加载并显示每辆小车的信息。

**SmartCar::loadFromFile() 函数**

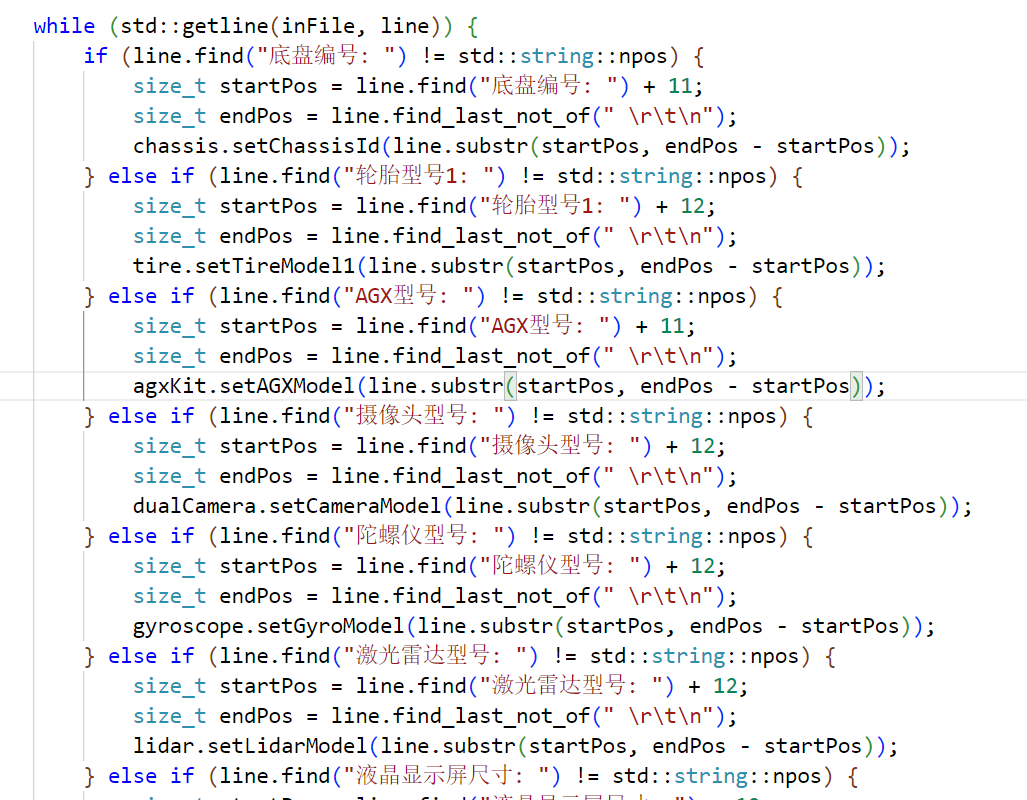
该函数用于从文件中加载小车的信息。

* 流程：

打开指定的文件。

逐行读取文件内容。

解析每行内容，根据关键字将信息分配给相应的组件。



构造并返回 SmartCar 对象。

**SmartCar::save() 函数**

该函数用于将小车的信息保存到文件中。

* 流程：

打开指定的文件。

遍历所有组件，将它们的信息写入文件。

1. 算法流程

**信息录入流程**

初始化小车数量。

对于每辆小车，提示用户输入学生信息。

创建 SmartCar 对象并设置学生信息。

调用 save() 函数将小车信息保存到文件。

**信息显示流程**

提醒用户确认是否查看信息。

如果用户同意，遍历所有小车文件。

对于每个文件，使用 loadFromFile() 函数加载小车信息。

调用 print() 函数显示小车信息。

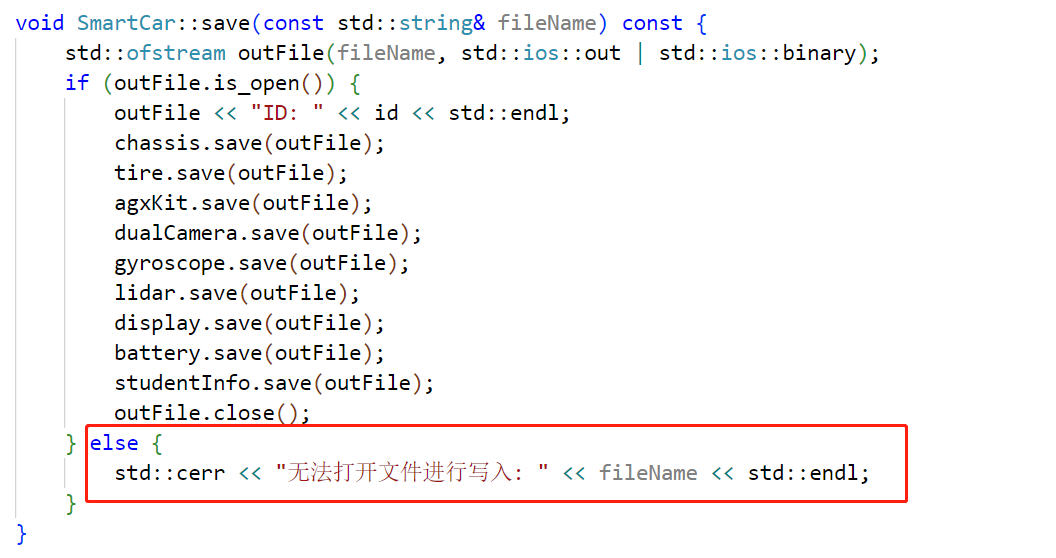
**文件操作流程**

使用 std::ifstream 和 std::ofstream 进行文件读写操作。

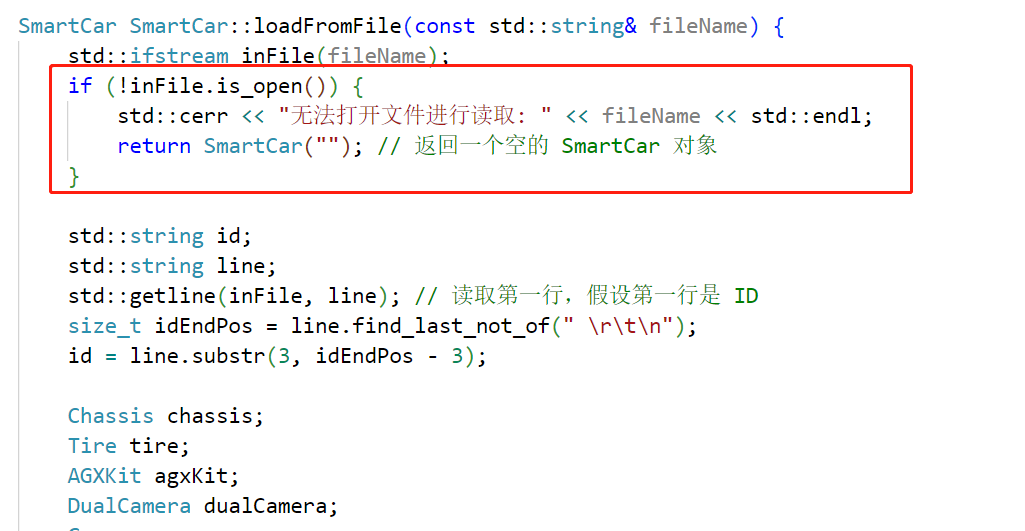
在读写操作中，使用 std::getline() 逐行处理文件内容。

1. 异常处理

程序中应包含对异常的检测和处理，例如文件读写错误、区域设置错误等。



该段代码是对保存信息到文件时的异常处理。

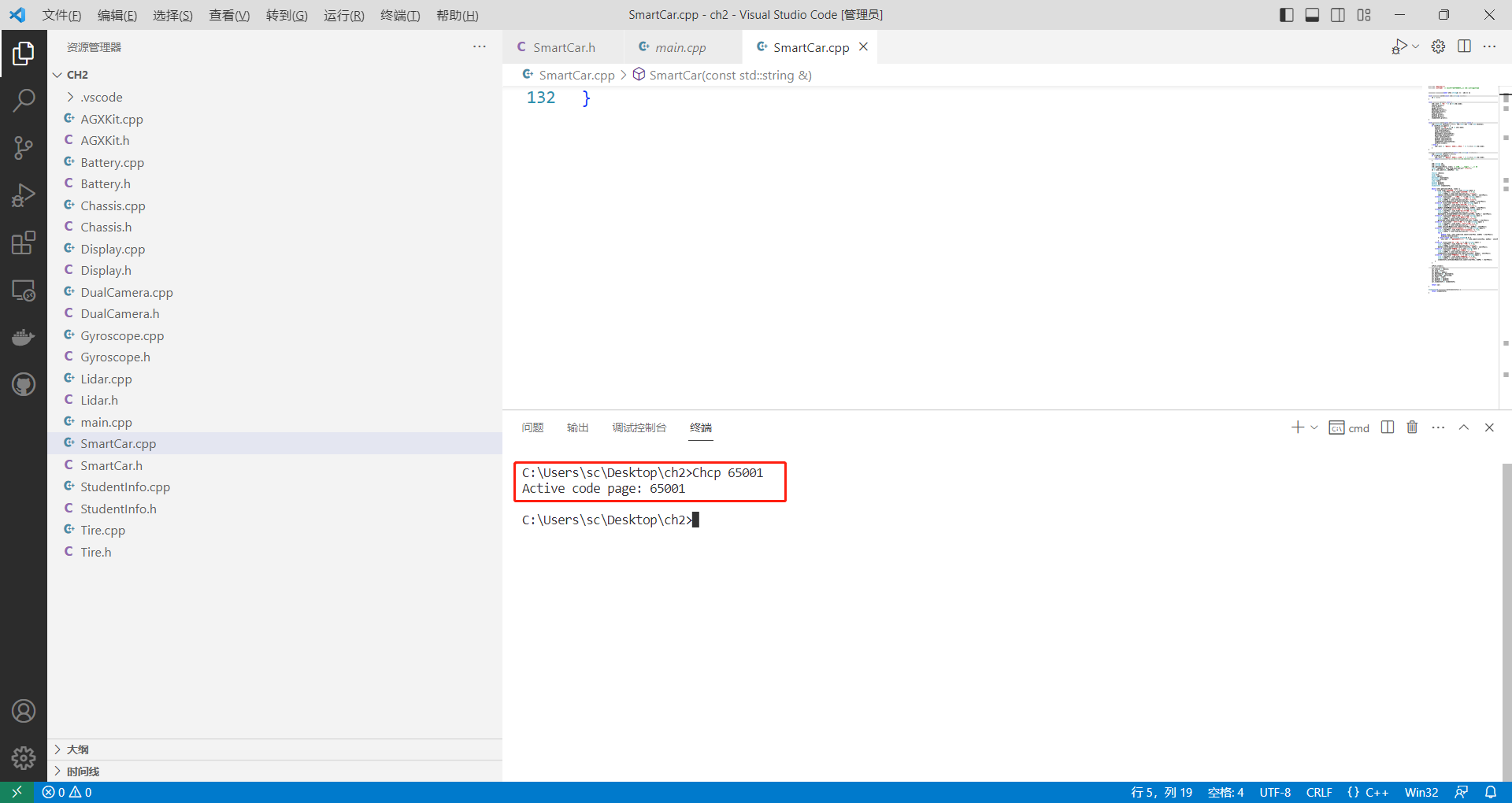


该段代码是对从文件中加载小车信息时的异常处理。

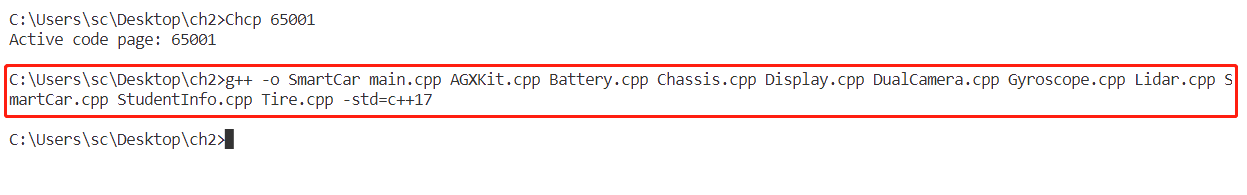
程序通过 SmartCar 类和一系列组件类来管理小车信息，提供了信息录入、保存、加载和显示的功能。通过 main() 函数实现程序主要流程和用户交互。

1. **程序效果展示**

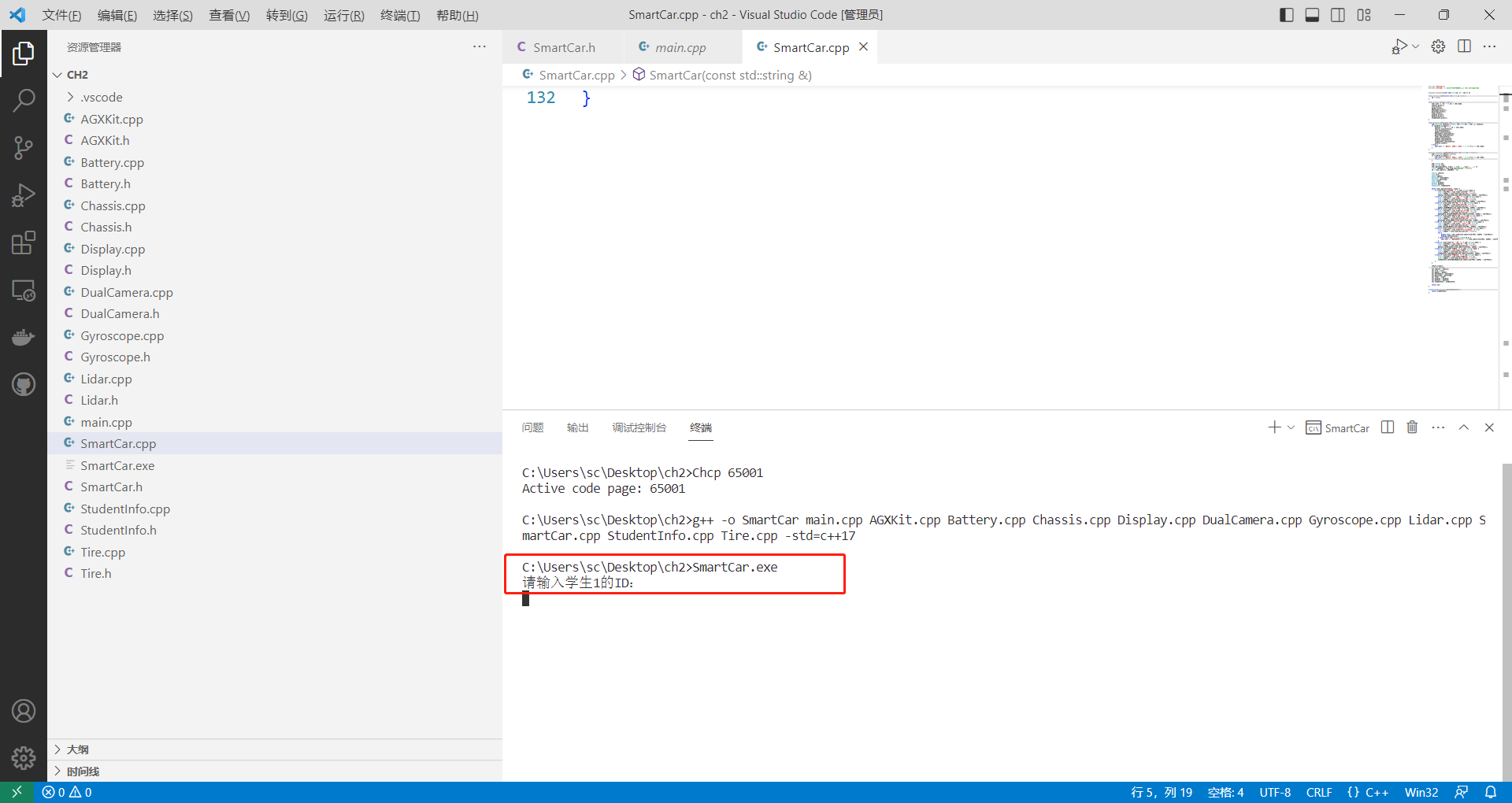
首先，在终端输入Chcp 65001，解决终端显示中文乱码问题。



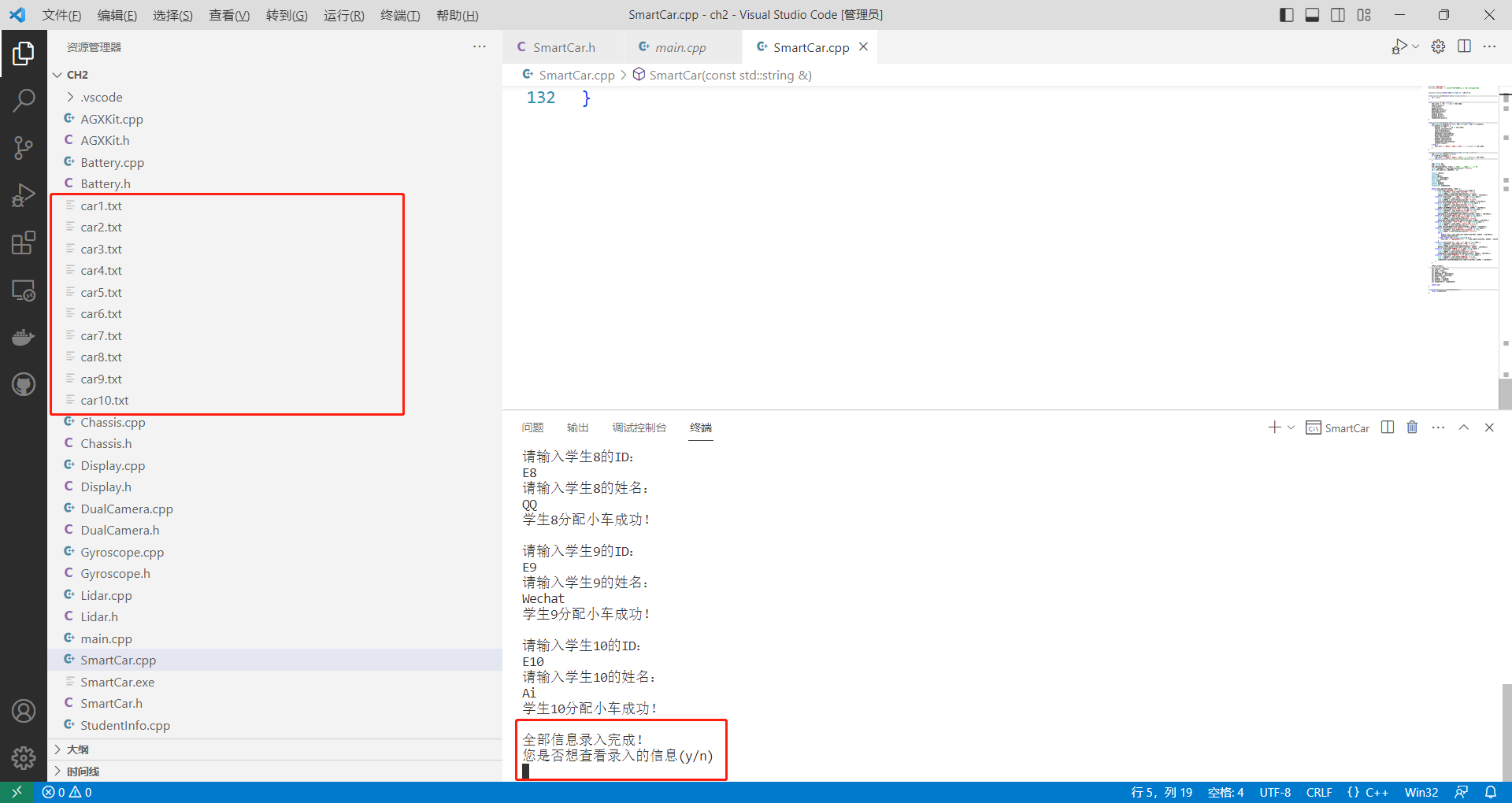
然后终端输入g++ -o SmartCar main.cpp AGXKit.cpp Battery.cpp Chassis.cpp Display.cpp DualCamera.cpp Gyroscope.cpp Lidar.cpp SmartCar.cpp StudentInfo.cpp Tire.cpp -std=c++17，将程序编译成SmartCar.exe可执行文件。



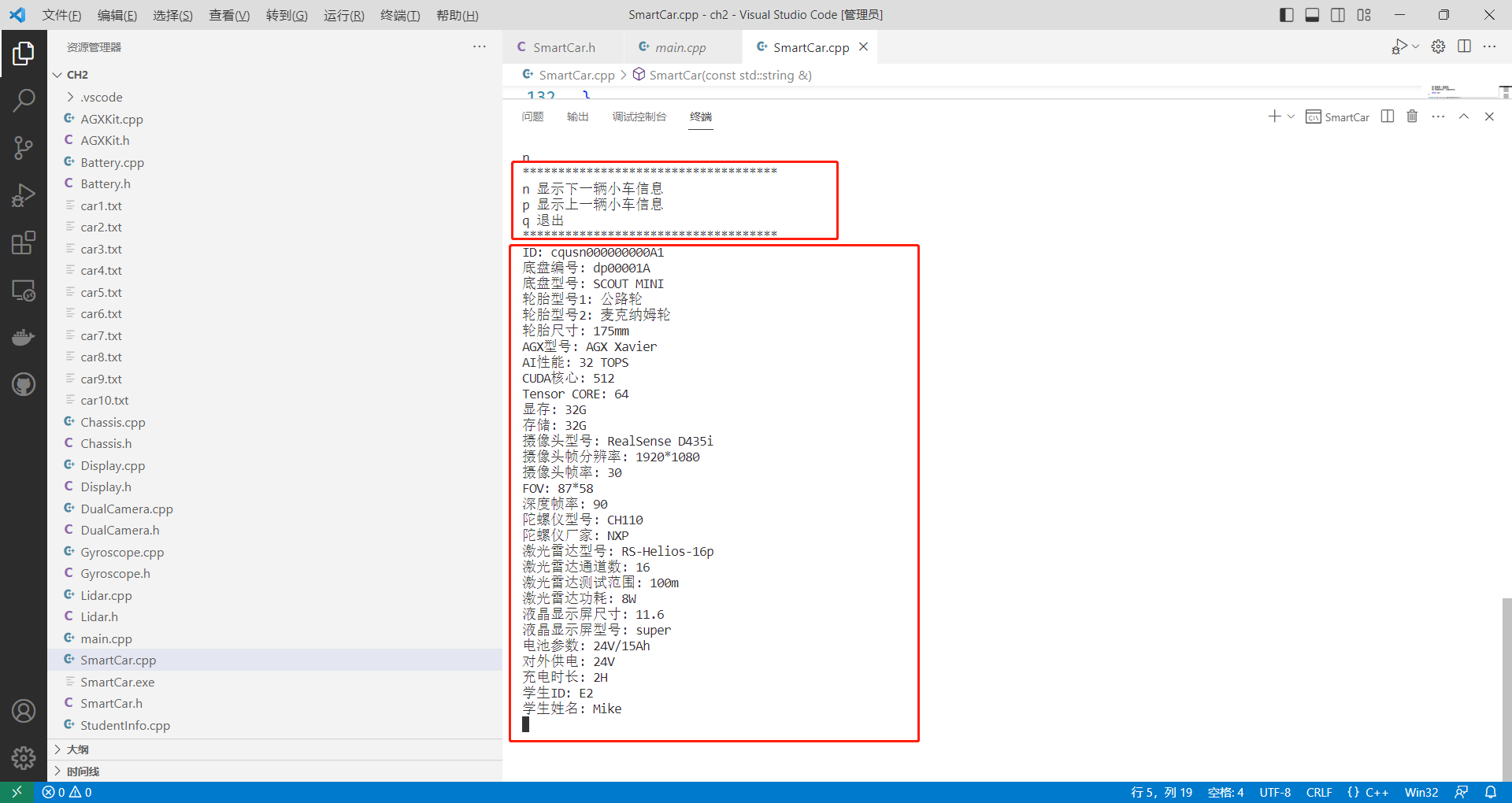
运行SmartCar.exe文件。这时程序检测到当前文件夹下没有录入小车信息，因此提醒用户输入与小车相关的学生信息，进行信息录入。我们依次完成10台小车信息的录入。



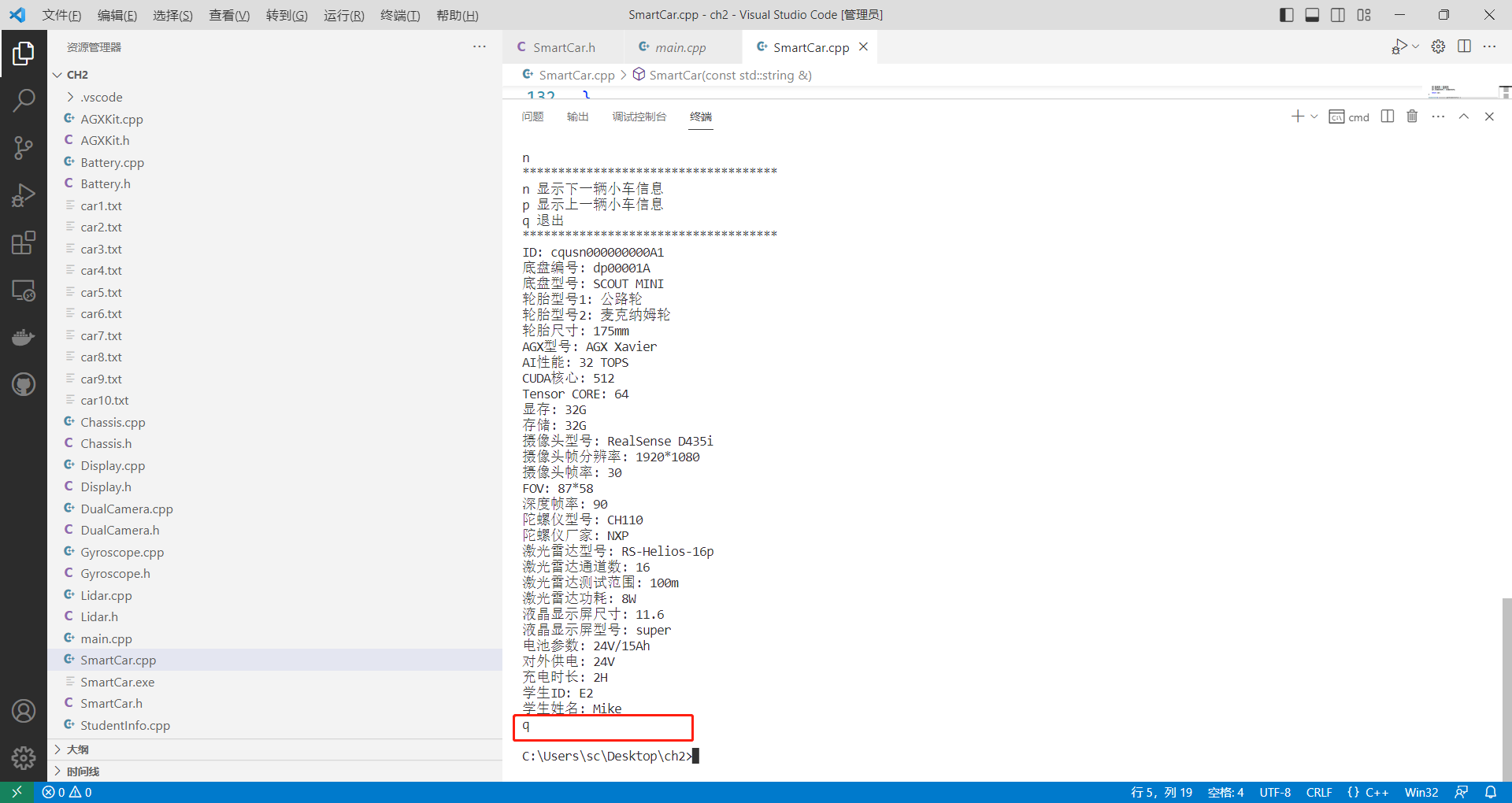
10台小车信息录入完成后，可以看到程序自动生成了包含10台小车信息的txt文件，同时程序提示“全部信息录入完成！您是否想查看录入的信息（y/n）”。我们输入y。



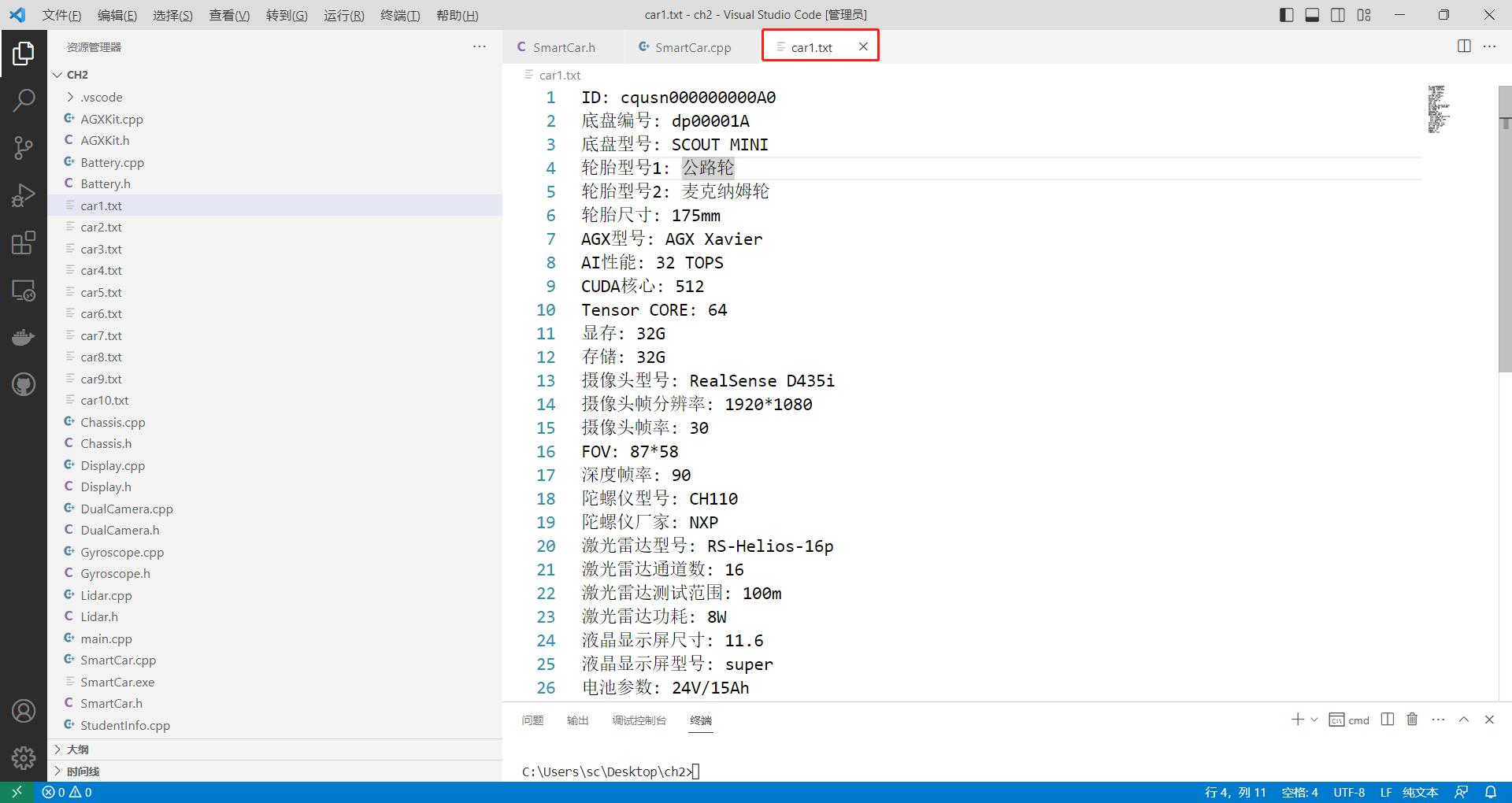
程序提示用户按键操作，并在终端完整显示了小车的信息。



我们可以通过按n键查看下一台小车的信息，按p键查看上一台小车的信息。如果当前显示的是第一台小车信息，那么按p无效；如果当前显示的是最后一台小车的信息，那么按n无效。我们按q键退出程序。



以下是txt文件中存储的小车信息。



当我们再次运行SmartCar.exe文件时，这时程序检测到当前文件夹下有10台小车完整的信息，因此不会再提醒用户录入信息，而是直接进入信息读取功能。

同样地，也可以通过按n，p，q实现相同的效果。

通过以上的测试，程序可以正确地实现报告要求的功能。包括信息的录入、终端信息的提示、用户的交互、信息的读取、信息的显示、信息的刷新显示、异常的处理等等。

1. **总结和分析**

本程序实现了一个智能小车信息管理系统，它能够创建、存储、加载和显示智能小车及其组件的信息。系统通过 SmartCar 类和一系列组件类（如 Chassis、Tire 等）来封装数据和行为。主要功能包括：

* 信息录入：用户可以输入智能小车的详细信息，包括组件信息和学生信息。
* 信息保存：将智能小车的信息保存到文件中。
* 信息加载：从文件中加载智能小车的信息。
* 信息显示：在终端显示智能小车的详细信息。

当然，程序在以下的方面还有改进的空间。

异常处理：在当前实现中，异常处理主要针对文件操作和数字格式转换。然而，对于用户输入错误或不一致的数据，系统没有进行充分的验证和处理。比如假设学号必须是以E开头的8个字母+数字，程序中可以增加相应的校验代码。

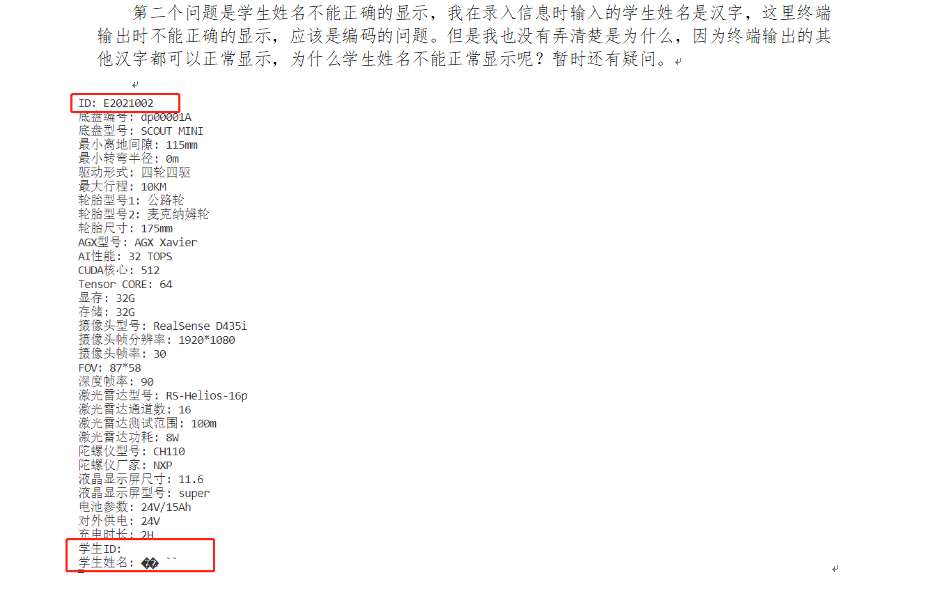
数据验证：系统假设所有输入都是有效的，没有实现对输入数据的验证逻辑，这可能导致数据不一致或程序运行错误。

用户交互：用户界面是基于控制台的，交互性有限。理想情况下，应该有更友好的用户界面，如图形界面，以提高用户体验。

扩展性：虽然系统通过组件类实现了一定的模块化，但添加新的组件或功能可能需要修改多个文件，这在大型项目中可能不够灵活。

性能考虑：当前实现未对性能进行优化，例如，每次保存或加载操作都会完整地读写文件，这在处理大量数据时效率不高。

此外还有一点问题是在实践1中就已经存在的，以下是实践1报告的截图。



这个问题暂时不知道原因，没有得到解决，在本代码中，我是通过输入英文的学生姓名来避免这个问题的。当然，在本次代码中，值得高兴的是，学生ID和小车的ID可以正确地显示了。