

车载软件开发基础

课后实践1



**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 | 某校购买了多台智能小车，并将其分配给同学们实验使用。请使用C++语言编写一个程序，实现对这批智能小车的信息录入及分配，并以文件方式保存相关资料。  具体资料如下：   1. 智能小车信息包括： 2. 编号：自定义，cqusn打头的16位数字+字母 3. 底盘    1. 编号：dp打头的8位数字+字母    2. 型号：SCOUT MINI    3. 轴距：451mm    4. 轮距：490mm    5. 最小离地间隙：115mm    6. 最小转弯半径：0m    7. 驱动形式：四轮四驱    8. 最大行程：10KM    9. 轮胎（4个）       1. 型号：公路轮、麦克纳姆轮       2. 尺寸：175mm 4. AGX套件（1个）    1. 型号：AGX Xavier    2. AI：32 TOPS    3. CUDA核心：512    4. Tensor CORE：64    5. 显存：32G    6. 存储：32G 5. 双目摄像头（1个）    1. 型号：RealSense D435i    2. 摄像头：D430    3. RGB帧分辨率：1920\*1080    4. RGB帧率：30    5. FOV：87\*58    6. 深度帧率：90 6. 多线激光雷达（1个）    1. 型号：RS-Helios-16p    2. 通道数：16    3. 测试范围：100m    4. 功耗：8W 7. 9轴陀螺仪（1个）    1. 型号：CH110    2. 厂家：NXP 8. 液晶显示屏（1个）    1. 尺寸：11.6    2. 型号：super 9. 电池模块（1个）    1. 参数：24V/15Ah    2. 对外供电：24V    3. 充电时长：2H 10. 学生信息包括： 11. 学号 12. 姓名   要求：   1. 共采购了10台，请先分别完成10台小车的信息录入，并完成编号。 2. 根据编号，将每台小车分配给每名同学（含学号和姓名） 3. 将相关信息以文件方式保存到本地硬盘中 4. 用程序打开3中保存的文件，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；  （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）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**

首先，我对程序的功能设计如下。

程序主要包含两个功能，一个是信息的录入（智能小车的编号、学生学号、学生姓名需要在主程序中进行录入，其中智能小车的编号在主程序中写死了，默认每个小车的编号前15位都一样，最后一位为从0到9的数字，学生学号和学生姓名需要用户输入，其他小车信息都采用题目中的给定值）和将录入的信息写到txt文件中，每一个文件存储一个智能小车的信息，一共有10台智能小车，所以信息录入完成后应该有10个txt文件；另一个是信息的读取，从txt文件中读取智能小车的信息。

在main.cpp中，逻辑是这样的，首先判断是否有10个txt文件，也就是判断是否10台智能小车的信息都已经录入完毕，如果没有，那么程序从录入信息开始执行，在用户录入信息时，终端会有智能提示，等到10台小车的信息都录入完毕，终端提示“全部信息录入完成！”，并询问“您是否想查看录入的信息(y/n)”，用户可自行选择是否查看。在下一次执行程序时，由于之前信息已经录好，所以就会跳过录入信息的程序，直接进入信息读取部分，用户可以根据智能提示按n读取下一个小车的信息，按p读取上一个小车的信息，按q退出。

以下是main.cpp文件的内容。

1. #include <iostream>
2. #include <vector>
3. #include <filesystem>
4. #include "SmartCar.h"
5. using namespace std;
6. int main()
7. {
8. const int numCars = 10;
9. vector<SmartCar> cars;
10. string choice;
11. int txtCount = 0;
12. string current\_path = "./";
13. for (const auto &entry : std::filesystem::directory\_iterator(current\_path))
14. {
15. if (entry.is\_regular\_file() && entry.path().extension() == ".txt")
16. {
17. txtCount++;
18. }
19. }
20. if (txtCount != numCars)
21. {
22. for (int i = 0; i < numCars; ++i)
23. {
24. string id = "cqusn000000000A" + to\_string(i);
25. SmartCar car(id);
26. string stuId;
27. string stuName;
28. cout << "请输入学生" + to\_string(i + 1) + "的ID：" << endl;
29. cin >> stuId;
30. cout << "请输入学生" + to\_string(i + 1) + "的姓名：" << endl;
31. cin >> stuName;
32. car.setStudentId(stuId);
33. car.setStudentName(stuName);
34. cout << "学生" + to\_string(i + 1) + "分配小车成功！" << endl;
35. cout << "" << endl;
36. string fileName = "car" + to\_string(i + 1) + ".txt";
37. car.saveToFile(fileName);
38. cars.push\_back(car);
39. }
40. cout << "全部信息录入完成！" << endl;
41. cout << "您是否想查看录入的信息(y/n)" << endl;
42. cin >> choice;
43. }
44. if (choice == "y" || txtCount == 10)
45. {
46. for (int i = 0; i < numCars; ++i)
47. {
48. string fileName = "car" + to\_string(i + 1) + ".txt";
49. SmartCar car = SmartCar::loadFromFile(fileName);
50. cars.push\_back(car);
51. }
52. int currentIndex = 0;
53. while (true)
54. {
55. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
56. cout << "n 显示下一辆小车信息" << endl;
57. cout << "p 显示上一辆小车信息" << endl;
58. cout << "q 退出" << endl;
59. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
60. cars[currentIndex].display();
61. char ch;
62. cin >> ch;
63. if (ch == 'n')
64. {
65. if (currentIndex < numCars - 1)
66. {
67. ++currentIndex;
68. }
69. }
70. else if (ch == 'p')
71. {
72. if (currentIndex > 0)
73. {
74. --currentIndex;
75. }
76. }
77. else if (ch == 'q')
78. {
79. break;
80. }
81. }
82. }
83. return 0;
84. }

main.cpp的主要功能是管理一个智能小车（SmartCar）的列表，包括信息的录入、保存、读取和显示。下面是程序的详细功能解释：

1. 初始化变量：

numCars：定义智能小车的数量，这里设置为10。

cars：一个 SmartCar 类型的 vector，用于存储小车对象。

choice：用于存储用户的选择（y/n）。

txtCount：用于计数当前目录下 .txt 文件的数量。

current\_path：设置当前目录路径为 ./，即程序所在的当前目录。

1. 检查文件数量：

使用 std::filesystem::directory\_iterator 遍历当前目录下的所有文件。

如果文件是常规文件并且扩展名为 .txt，则 txtCount 计数加一。

1. 信息录入：

如果 txtCount 不等于 numCars（即目录下没有足够的 .txt 文件），则执行信息录入流程。

循环创建 SmartCar 对象，每次循环代表一辆小车。

通过控制台输入，为每辆小车录入学生ID和学生姓名。

将录入的信息保存到对应的 .txt 文件中，并更新 SmartCar 对象后加入到 cars 列表中。

所有信息录入完成后，提示用户信息录入完成，并询问是否查看录入的信息。

1. 信息显示：

如果用户选择查看信息（输入 y），或者目录下已经有10个 .txt 文件（即信息已经被录入过），则执行信息显示流程。

循环读取每个 .txt 文件，创建 SmartCar 对象，并将对象加入到 cars 列表中。

使用一个循环，允许用户通过输入 n（显示下一辆车信息）、p（显示上一辆车信息）或 q（退出程序）来控制程序的行为。

每次循环显示当前索引对应小车的信息。

1. 退出程序：

当用户输入 q 时，程序退出信息显示循环，结束程序。

整个程序的流程是先检查是否需要录入信息，如果需要则进行录入，如果已经存在信息则直接读取并显示。用户可以通过简单的控制台指令来查看小车的信息。

以下是SmartCar.cpp文件的内容。该文件中包含程序的核心功能实现。

1. #include "SmartCar.h"
2. #include <iostream>
3. #include <fstream>
4. const std::string SmartCar::defaultChassisId = "dp00001A";
5. const std::string SmartCar::defaultChassisModel = "SCOUT MINI";
6. const int SmartCar::defaultWheelBase = 451;
7. const int SmartCar::defaultWheelTrack = 490;
8. const int SmartCar::defaultGroundClearance = 115;
9. const int SmartCar::defaultTurningRadius = 0;
10. const std::string SmartCar::defaultDriveType = "四轮四驱";
11. const int SmartCar::defaultMaxRange = 10;
12. const std::string SmartCar::defaultTireModel1 = "公路轮";
13. const std::string SmartCar::defaultTireModel2 = "麦克纳姆轮";
14. const int SmartCar::defaultTireSize = 175;
15. const std::string SmartCar::defaultAgxModel = "AGX Xavier";
16. const int SmartCar::defaultAi = 32;
17. const int SmartCar::defaultCudaCores = 512;
18. const int SmartCar::defaultTensorCores = 64;
19. const int SmartCar::defaultMemorySize = 32;
20. const int SmartCar::defaultStorageSize = 32;
21. const std::string SmartCar::defaultCameraModel = "RealSense D435i";
22. const std::string SmartCar::defaultCameraFrameResolution = "1920\*1080";
23. const int SmartCar::defaultCameraFrameRate = 30;
24. const std::string SmartCar::defaultCameraFov = "87\*58";
25. const int SmartCar::defaultDepthFrameRate = 90;
26. const std::string SmartCar::defaultLidarModel = "RS-Helios-16p";
27. const int SmartCar::defaultLidarChannels = 16;
28. const int SmartCar::defaultLidarRange = 100;
29. const int SmartCar::defaultLidarPower = 8;
30. const std::string SmartCar::defaultGyroModel = "CH110";
31. const std::string SmartCar::defaultGyroManufacturer = "NXP";
32. const double SmartCar::defaultDisplaySize = 11.6;
33. const std::string SmartCar::defaultDisplayModel = "super";
34. const std::string SmartCar::defaultBatteryParameter = "24V/15Ah";
35. const int SmartCar::defaultBatterySupply = 24;
36. const int SmartCar::defaultChargeTime = 2;
37. SmartCar::SmartCar(const std::string& id) : id(id) {}
38. void SmartCar::setStudentId(const std::string& studentId) {
39. this->studentId = studentId;
40. }
41. void SmartCar::setStudentName(const std::string& studentName) {
42. this->studentName = studentName;
43. }
44. void SmartCar::saveToFile(const std::string& fileName) const {
45. std::ofstream outFile(fileName, std::ios::out | std::ios::binary);
46. if (outFile.is\_open()) {
47. outFile << "ID: " << id << std::endl;
48. outFile << "底盘编号: " << defaultChassisId << std::endl;
49. outFile << "底盘型号: " << defaultChassisModel << std::endl;
50. outFile << "轴距: " << defaultWheelBase << "mm" << std::endl;
51. outFile << "轮距: " << defaultWheelTrack << "mm" << std::endl;
52. outFile << "最小离地间隙: " << defaultGroundClearance << "mm" << std::endl;
53. outFile << "最小转弯半径: " << defaultTurningRadius << "m" << std::endl;
54. outFile << "驱动形式: " << defaultDriveType << std::endl;
55. outFile << "最大行程: " << defaultMaxRange << "KM" << std::endl;
56. outFile << "轮胎型号1: " << defaultTireModel1 << std::endl;
57. outFile << "轮胎型号2: " << defaultTireModel2 << std::endl;
58. outFile << "轮胎尺寸: " << defaultTireSize << "mm" << std::endl;
59. outFile << "AGX型号: " << defaultAgxModel << std::endl;
60. outFile << "AI性能: " << defaultAi << " TOPS" << std::endl;
61. outFile << "CUDA核心: " << defaultCudaCores << std::endl;
62. outFile << "Tensor CORE: " << defaultTensorCores << std::endl;
63. outFile << "显存: " << defaultMemorySize << "G" << std::endl;
64. outFile << "存储: " << defaultStorageSize << "G" << std::endl;
65. outFile << "摄像头型号: " << defaultCameraModel << std::endl;
66. outFile << "摄像头帧分辨率: " << defaultCameraFrameResolution << std::endl;
67. outFile << "摄像头帧率: " << defaultCameraFrameRate << std::endl;
68. outFile << "FOV: " << defaultCameraFov << std::endl;
69. outFile << "深度帧率: " << defaultDepthFrameRate << std::endl;
70. outFile << "激光雷达型号: " << defaultLidarModel << std::endl;
71. outFile << "激光雷达通道数: " << defaultLidarChannels << std::endl;
72. outFile << "激光雷达测试范围: " << defaultLidarRange << "m" << std::endl;
73. outFile << "激光雷达功耗: " << defaultLidarPower << "W" << std::endl;
74. outFile << "陀螺仪型号: " << defaultGyroModel << std::endl;
75. outFile << "陀螺仪厂家: " << defaultGyroManufacturer << std::endl;
76. outFile << "液晶显示屏尺寸: " << defaultDisplaySize << std::endl;
77. outFile << "液晶显示屏型号: " << defaultDisplayModel << std::endl;
78. outFile << "电池参数: " << defaultBatteryParameter << std::endl;
79. outFile << "对外供电: " << defaultBatterySupply << "V" << std::endl;
80. outFile << "充电时长: " << defaultChargeTime << "H" << std::endl;
81. outFile << "学生ID: " << studentId << std::endl;
82. outFile << "学生姓名: " << studentName << std::endl;
83. outFile.close();
84. } else {
85. std::cerr << "无法打开文件进行写入: " << fileName << std::endl;
86. }
87. }
88. SmartCar SmartCar::loadFromFile(const std::string& fileName) {
89. std::ifstream inFile(fileName, std::ios::in | std::ios::binary);
90. if (inFile.is\_open()) {
91. std::string id;
92. std::string studentId;
93. std::string studentName;
94. std::string line;
95. while (std::getline(inFile, line)) {
96. if (line.find("ID: ") != std::string::npos) {
97. id = line.substr(line.find("ID: ") + 4);
98. } else if (line.find("学生ID: ") != std::string::npos) {
99. studentId = line.substr(line.find("学生ID: ") + 8);
100. } else if (line.find("学生姓名: ") != std::string::npos) {
101. studentName = line.substr(line.find("学生姓名: ") + 10);
102. }
103. }
104. inFile.close();
105. SmartCar car(id);
106. car.setStudentId(studentId);
107. car.setStudentName(studentName);
108. return car;
109. } else {
110. std::cerr << "无法打开文件进行读取: " << fileName << std::endl;
111. return SmartCar("");
112. }
113. }
114. void SmartCar::display() const {
115. std::cout << "ID: " << id << std::endl;
116. std::cout << "底盘编号: " << defaultChassisId << std::endl;
117. std::cout << "底盘型号: " << defaultChassisModel << std::endl;
118. std::cout << "轴距: " << defaultWheelBase << "mm" << std::endl;
119. std::cout << "轮距: " << defaultWheelTrack << "mm" << std::endl;
120. std::cout << "最小离地间隙: " << defaultGroundClearance << "mm" << std::endl;
121. std::cout << "最小转弯半径: " << defaultTurningRadius << "m" << std::endl;
122. std::cout << "驱动形式: " << defaultDriveType << std::endl;
123. std::cout << "最大行程: " << defaultMaxRange << "KM" << std::endl;
124. std::cout << "轮胎型号1: " << defaultTireModel1 << std::endl;
125. std::cout << "轮胎型号2: " << defaultTireModel2 << std::endl;
126. std::cout << "轮胎尺寸: " << defaultTireSize << "mm" << std::endl;
127. std::cout << "AGX型号: " << defaultAgxModel << std::endl;
128. std::cout << "AI性能: " << defaultAi << " TOPS" << std::endl;
129. std::cout << "CUDA核心: " << defaultCudaCores << std::endl;
130. std::cout << "Tensor CORE: " << defaultTensorCores << std::endl;
131. std::cout << "显存: " << defaultMemorySize << "G" << std::endl;
132. std::cout << "存储: " << defaultStorageSize << "G" << std::endl;
133. std::cout << "摄像头型号: " << defaultCameraModel << std::endl;
134. std::cout << "摄像头帧分辨率: " << defaultCameraFrameResolution << std::endl;
135. std::cout << "摄像头帧率: " << defaultCameraFrameRate << std::endl;
136. std::cout << "FOV: " << defaultCameraFov << std::endl;
137. std::cout << "深度帧率: " << defaultDepthFrameRate << std::endl;
138. std::cout << "激光雷达型号: " << defaultLidarModel << std::endl;
139. std::cout << "激光雷达通道数: " << defaultLidarChannels << std::endl;
140. std::cout << "激光雷达测试范围: " << defaultLidarRange << "m" << std::endl;
141. std::cout << "激光雷达功耗: " << defaultLidarPower << "W" << std::endl;
142. std::cout << "陀螺仪型号: " << defaultGyroModel << std::endl;
143. std::cout << "陀螺仪厂家: " << defaultGyroManufacturer << std::endl;
144. std::cout << "液晶显示屏尺寸: " << defaultDisplaySize << std::endl;
145. std::cout << "液晶显示屏型号: " << defaultDisplayModel << std::endl;
146. std::cout << "电池参数: " << defaultBatteryParameter << std::endl;
147. std::cout << "对外供电: " << defaultBatterySupply << "V" << std::endl;
148. std::cout << "充电时长: " << defaultChargeTime << "H" << std::endl;
149. std::cout << "学生ID: " << studentId << std::endl;
150. std::cout << "学生姓名: " << studentName << std::endl;
151. }

SmartCar.cpp是 SmartCar 类的实现，它定义了智能小车的各种属性和行为。以下是程序的主要功能：

1. 定义常量属性：

定义了一系列与智能小车相关的默认属性，如底盘编号、型号、尺寸、性能参数等。这些属性被设置为类的静态常量成员变量。

1. 构造函数：

SmartCar::SmartCar(const std::string& id)：构造函数，用于创建 SmartCar 对象时初始化小车的 ID。

1. 设置器方法：

SmartCar::setStudentId(const std::string& studentId)：设置小车对应学生的 ID。

SmartCar::setStudentName(const std::string& studentName)：设置小车对应学生的姓名。

1. 保存到文件：

SmartCar::saveToFile(const std::string& fileName) const：将小车的所有属性写入到指定的文件中。这个方法使用输出文件流 std::ofstream 并以二进制模式打开文件。如果文件打开成功，它会逐行写入小车的各个属性值。

1. 从文件加载：

SmartCar::loadFromFile(const std::string& fileName)：从指定的文件中读取小车的属性。这个方法使用输入文件流 std::ifstream 并以二进制模式打开文件。如果文件打开成功，它会逐行读取文件内容，并根据行内容提取小车的 ID、学生 ID 和学生姓名。

1. 显示信息：

SmartCar::display() const：在控制台上打印出小车的所有属性信息。这个方法逐个打印小车的各个属性值，包括小车自身的属性和默认的配置参数。

整体来看，这个程序提供了创建智能小车对象、设置其属性、将属性保存到文件、从文件中加载属性以及显示所有属性的功能。这样的设计使程序更容易恢复智能小车的状态，同时也便于用户查看智能小车的详细信息。

以下是SmartCar.h文件的内容。

1. #ifndef SMARTCAR\_H
2. #define SMARTCAR\_H
3. #include <string>
4. #include <vector>
5. class SmartCar {
6. public:
7. SmartCar(const std::string& id);
8. void setStudentId(const std::string& studentId);
9. void setStudentName(const std::string& studentName);
10. void saveToFile(const std::string& fileName) const;
11. static SmartCar loadFromFile(const std::string& fileName);
12. void display() const;
13. private:
14. std::string id;
15. std::string studentId;
16. std::string studentName;
17. static const std::string defaultChassisId;
18. static const std::string defaultChassisModel;
19. static const int defaultWheelBase;
20. static const int defaultWheelTrack;
21. static const int defaultGroundClearance;
22. static const int defaultTurningRadius;
23. static const std::string defaultDriveType;
24. static const int defaultMaxRange;
25. static const std::string defaultTireModel1;
26. static const std::string defaultTireModel2;
27. static const int defaultTireSize;
28. static const std::string defaultAgxModel;
29. static const int defaultAi;
30. static const int defaultCudaCores;
31. static const int defaultTensorCores;
32. static const int defaultMemorySize;
33. static const int defaultStorageSize;
34. static const std::string defaultCameraModel;
35. static const std::string defaultCameraFrameResolution;
36. static const int defaultCameraFrameRate;
37. static const std::string defaultCameraFov;
38. static const int defaultDepthFrameRate;
39. static const std::string defaultLidarModel;
40. static const int defaultLidarChannels;
41. static const int defaultLidarRange;
42. static const int defaultLidarPower;
43. static const std::string defaultGyroModel;
44. static const std::string defaultGyroManufacturer;
45. static const double defaultDisplaySize;
46. static const std::string defaultDisplayModel;
47. static const std::string defaultBatteryParameter;
48. static const int defaultBatterySupply;
49. static const int defaultChargeTime;
50. };
51. #endif *// SMARTCAR\_H*

SmartCar.h是 SmartCar 类的头文件，它定义了智能小车（SmartCar）的类结构和成员函数原型。以下是程序的主要功能和组成部分：

1. 类定义：

SmartCar：一个代表智能小车的类。

1. 构造函数：

SmartCar(const std::string& id)：构造函数，用于创建 SmartCar 对象时初始化小车的 ID。

1. 设置器方法：

void setStudentId(const std::string& studentId)：用于设置小车对应学生的 ID。

void setStudentName(const std::string& studentName)：用于设置小车对应学生的姓名。

1. 文件操作方法：

void saveToFile(const std::string& fileName) const：用于将小车对象的所有属性保存到指定的文件中。

static SmartCar loadFromFile(const std::string& fileName)：用于从指定的文件中加载小车对象的属性。

1. 显示方法：

void display() const：用于在控制台上显示小车对象的所有属性。

1. 私有成员变量：

std::string id：小车的 ID。

std::string studentId：小车对应学生的 ID。

std::string studentName：小车对应学生的姓名。

1. 静态常量成员变量：

定义了一系列与智能小车相关的默认属性，如底盘编号、型号、尺寸、性能参数等。这些属性被声明为静态常量，意味着它们的值在所有 SmartCar 对象中是共享的，并且它们的值在初始化后不会改变。

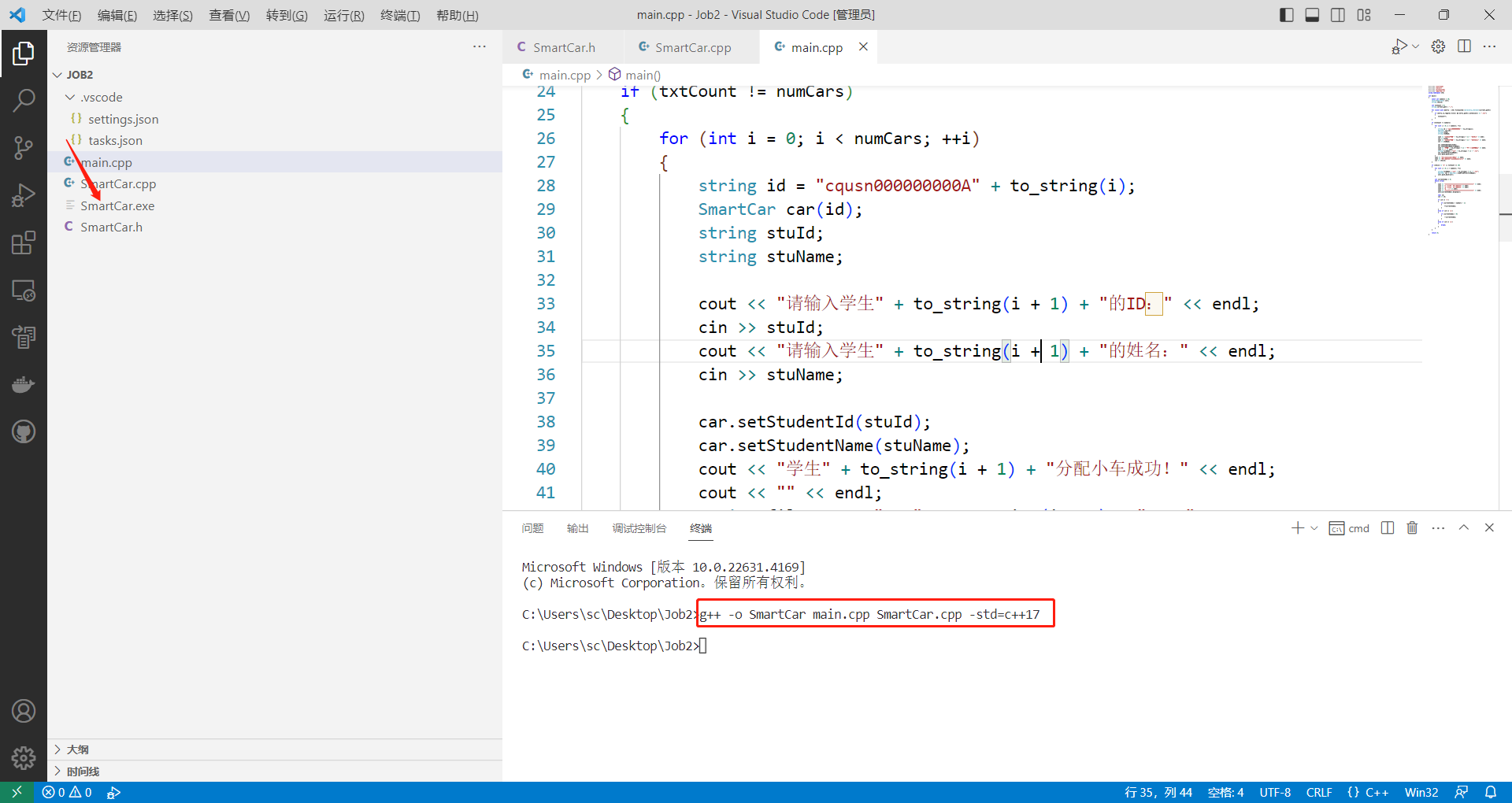
1. 头文件保护：

#ifndef SMARTCAR\_H、#define SMARTCAR\_H、#endif：这些预处理器指令用于防止头文件被多次包含，这是一种常见的头文件保护机制。

总结来说，这个头文件定义了 SmartCar 类的结构，包括它的属性、方法和静态常量。这个类可以用来创建智能小车的对象，设置和保存它们的属性，以及在需要时从文件中加载和显示这些属性。

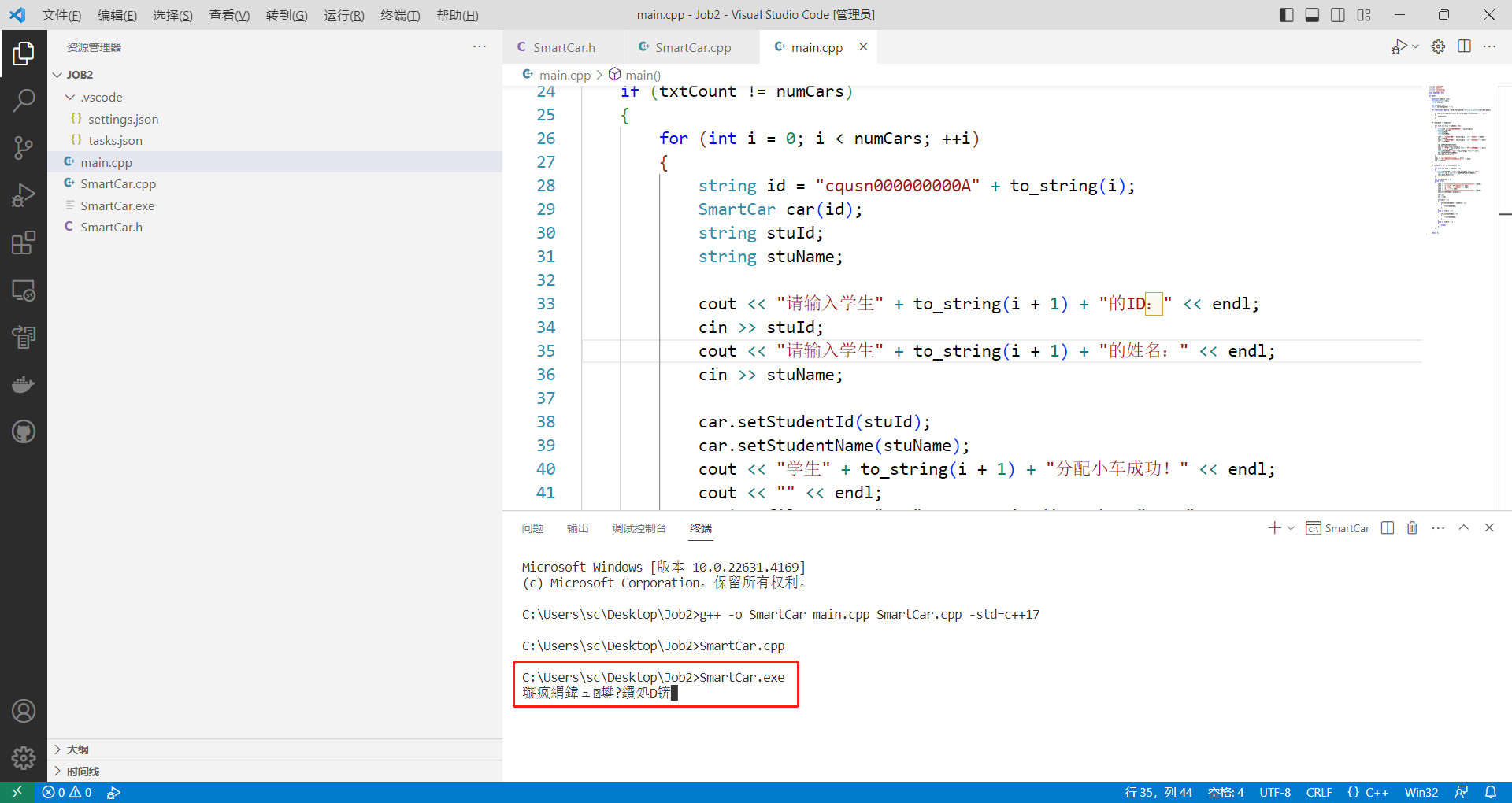
1. **程序效果展示**

首先编译程序。执行g++ -o SmartCar main.cpp SmartCar.cpp -std=c++17进行编译。该命令的意思是：告诉 g++ 编译器使用 C++17 标准来编译 main.cpp 和 SmartCar.cpp 这两个源代码文件，并把编译后的可执行文件命名为 SmartCar。

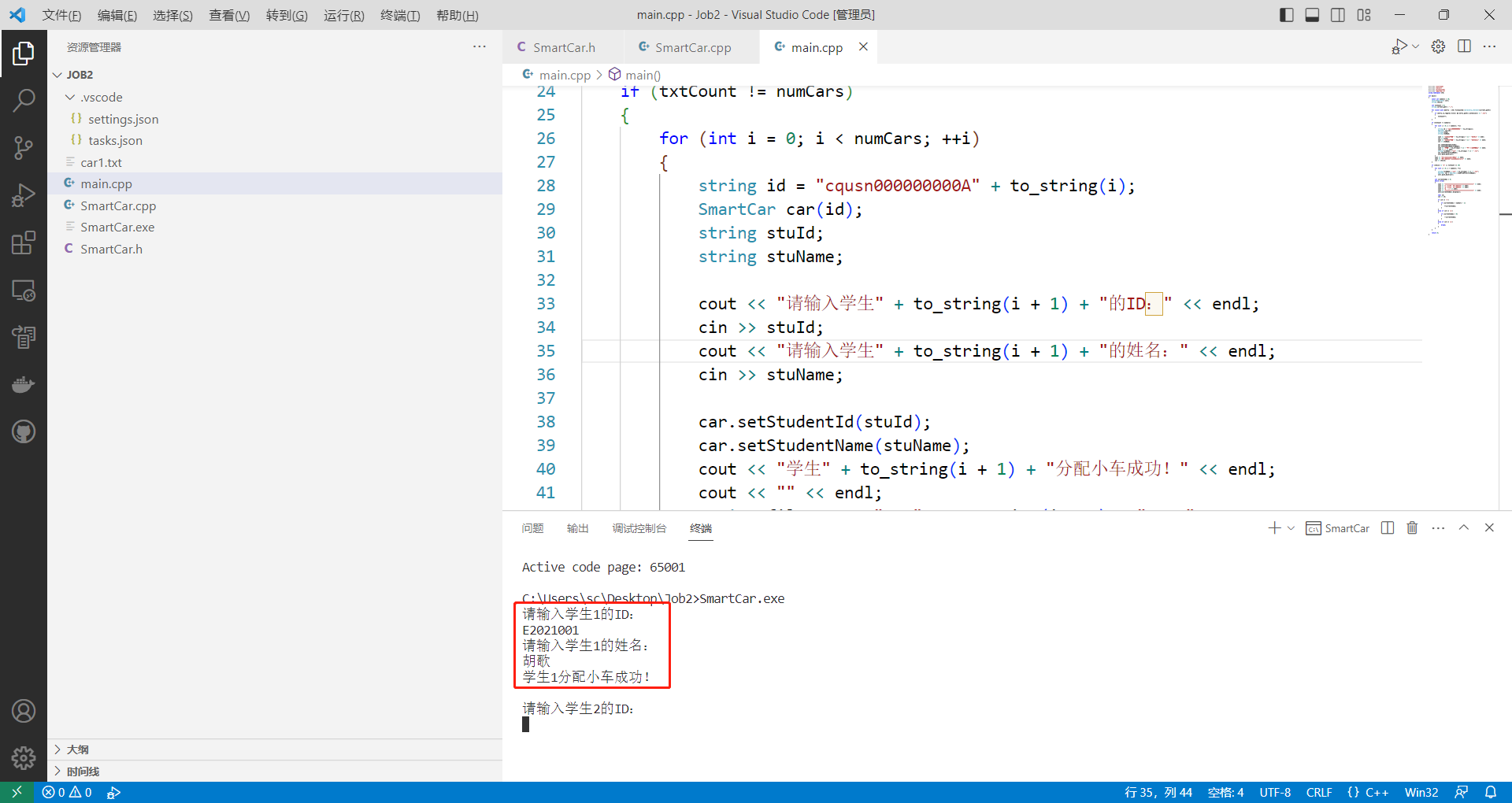


执行该命令编译成功后，就会生成SmartCar.exe可执行文件了。

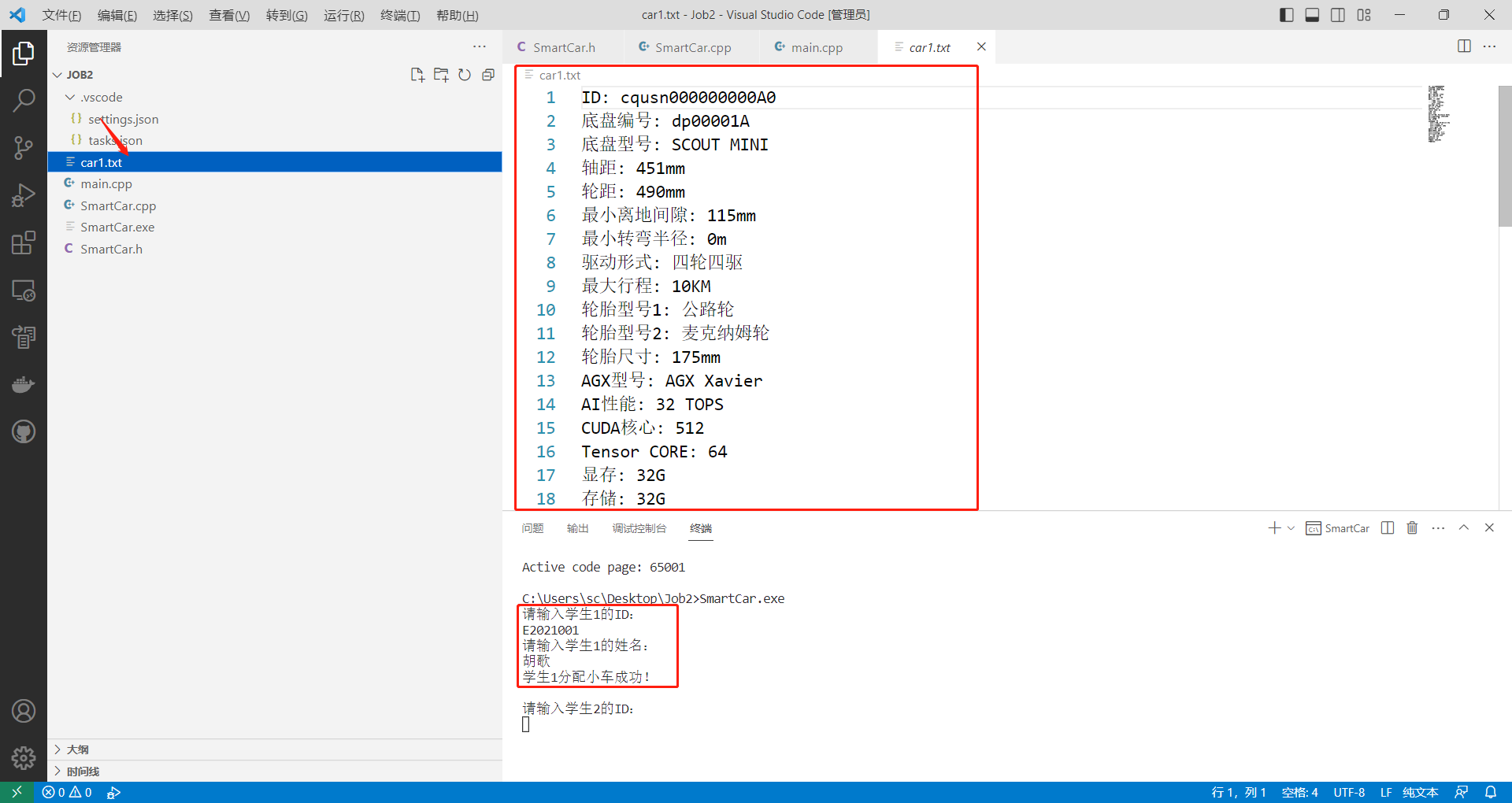
然后在终端输入SmartCar.exe运行程序。



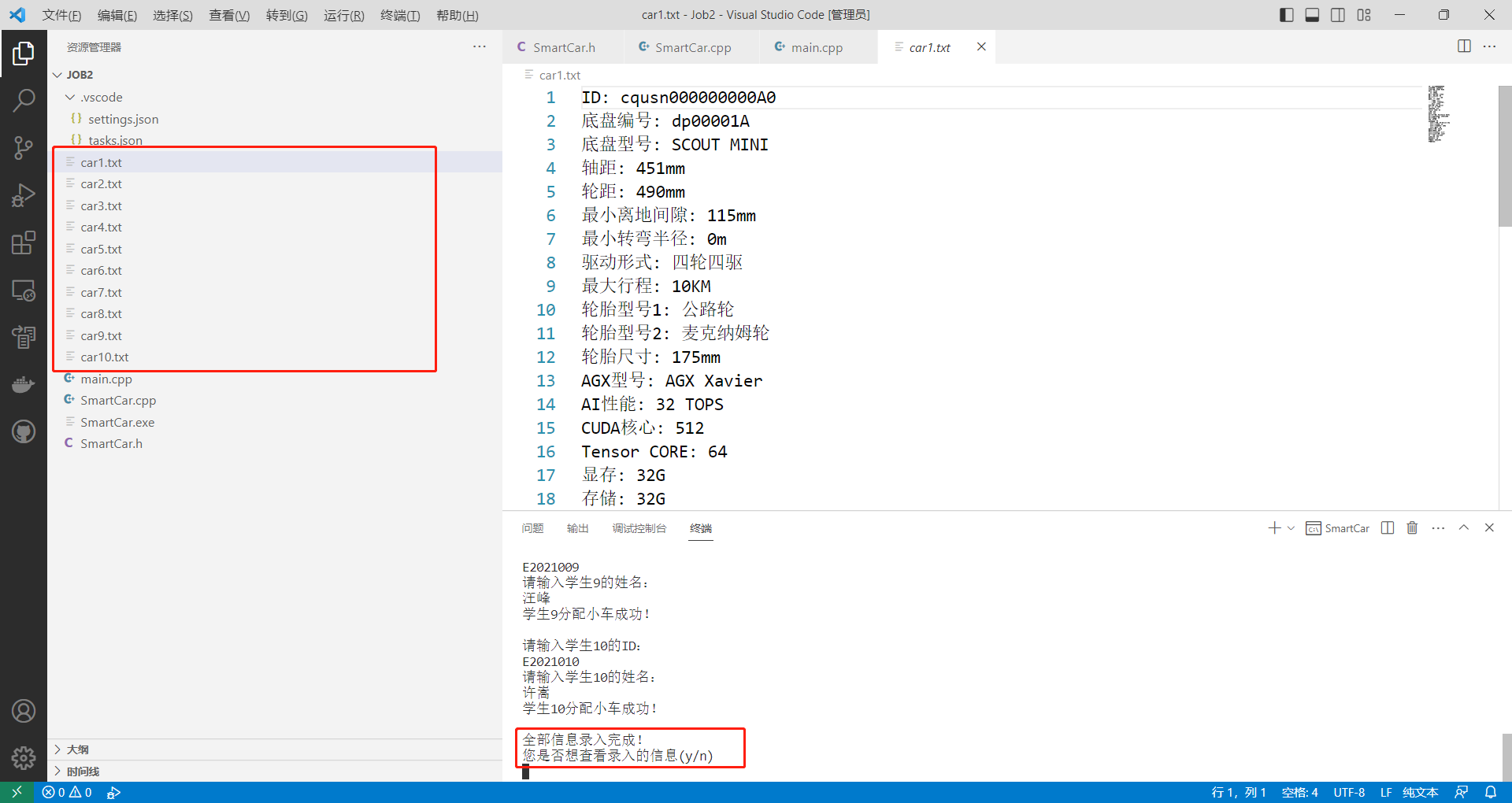
如图，运行完成后会发现终端输出的中文会出现乱码现象，我们先按ctrl+c终止掉程序，然后在终端输入Chcp 65001，就可以解决乱码问题了。再次运行SmartCar.exe。



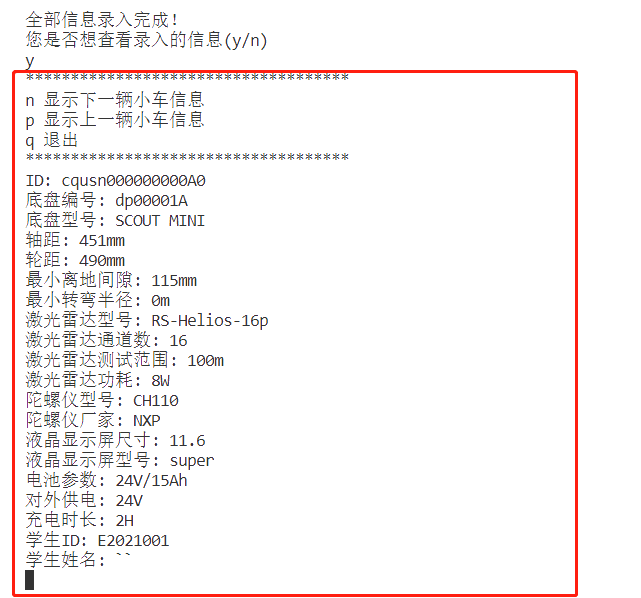
终端会提示用户输入学生的学号ID和姓名，按照要求输入后回车，信息录入完成后按回车，终端会提示“学生1分配小车成功！”，此时，文件夹下会新生成一个car1.txt文件，该文本文件中包含小车的信息，以及小车对应的学生信息。



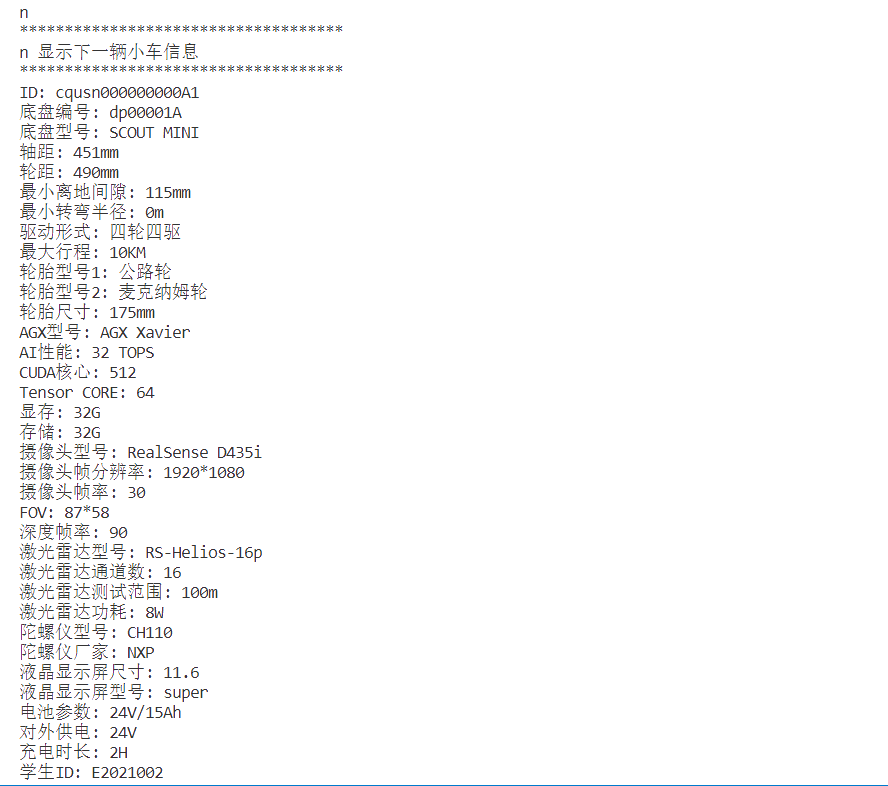
用户根据提示依次完成10台小车信息的录入，录入完成后，会生成10个包含小车信息的txt文件，每个文件存储一台小车的信息。并且终端会提示用户“您是否想查看录入的信息（y/n）”，按y代表查看，n代表退出。我们按y查看输出结果。



此时终端输出第一台小车的信息。并有按键提示，按n可以显示下一辆小车信息，按p显示上一辆小车信息，按q退出。我们输入n并回车。

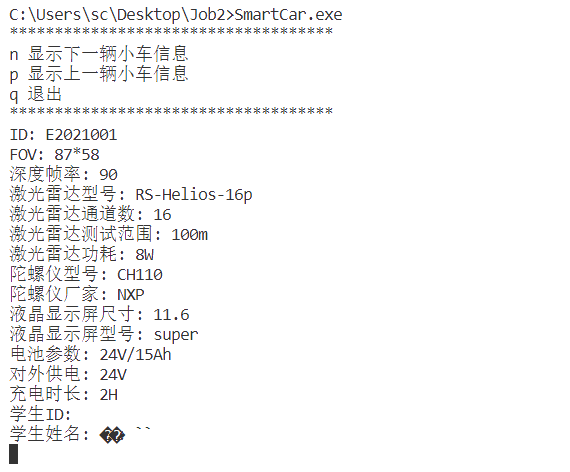


终端显示第二辆小车的信息。

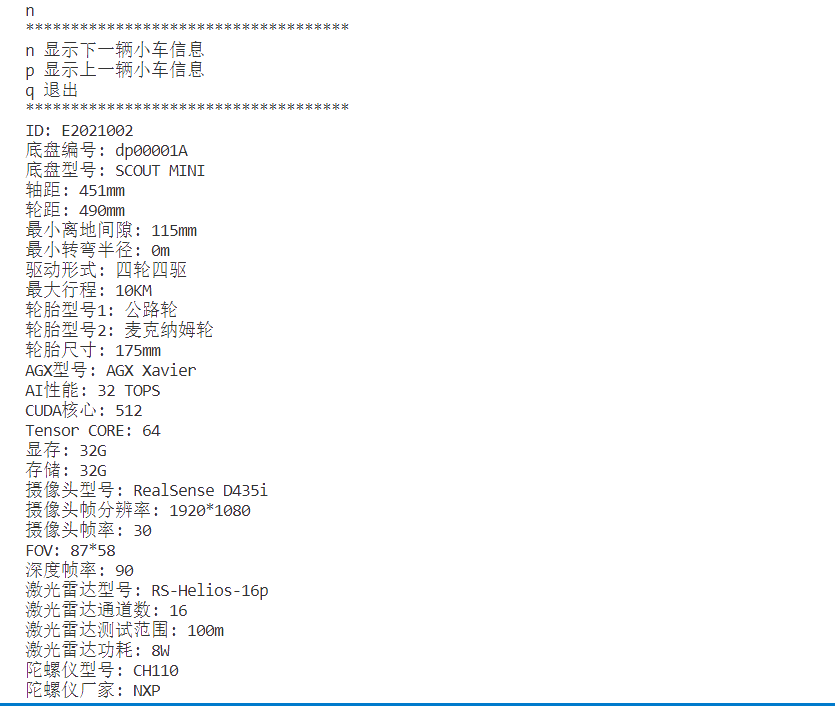


好的，经过测试，该部分程序的功能是正确的，如果想测试是否按p能显示上一辆小车的信息，可以输入p然后回车进行测试，这里就不再测试了，我们输入q回车退出程序。

接下来我们再次执行SmartCar.exe，由于我们之前已经录入好了小车的信息，这次程序就不会执行信息录入的部分了，而直接执行信息获取的部分。



执行后终端首先显示出第一台小车的信息，和之前一样，我们可以按n显示下一辆小车信息，按p显示上一辆小车信息，按q退出程序。



值得注意的是，当显示的是第一辆小车时，按p无效，当显示的是最后一辆小车时，按n无效。在程序中是这样实现的。currentIndex < numCars - 1和currentIndex > 0实现了该功能。

1. if (ch == 'n')
2. {
3. if (currentIndex < numCars - 1)
4. {
5. ++currentIndex;
6. }
7. }
8. else if (ch == 'p')
9. {
10. if (currentIndex > 0)
11. {
12. --currentIndex;
13. }
14. }
15. **总结和分析**

当然，程序的实际运行结果并不完全正确。有两个地方不正确：

第一个问题是学生ID和小车编号ID输出错误，在小车信息读取时，程序把学生ID当成了小车编号进行了输出，而学生ID输出的是一个空字符串。原因出在下面这段代码中，程序将学生ID和小车编号弄混淆了。

1. while (std::getline(inFile, line)) {
2. if (line.find("ID: ") != std::string::npos) {
3. id = line.substr(line.find("ID: ") + 4);
4. } else if (line.find("学生ID: ") != std::string::npos) {
5. studentId = line.substr(line.find("学生ID: ") + 8);
6. } else if (line.find("学生姓名: ") != std::string::npos) {
7. studentName = line.substr(line.find("学生姓名: ") + 10);
8. }
9. }

当然，我知道问题出现在这段代码中，但是我并没有想清楚为什么程序会把学生ID当成小车编号进行输出，在txt文件中，小车编号不是出现在学生ID前面吗？不应该先搜索到小车编号吗？所以这个地方的错误暂时没有得到解决。

第二个问题是学生姓名不能正确的显示，我在录入信息时输入的学生姓名是汉字，这里终端输出时不能正确的显示，应该是编码的问题。但是我也没有弄清楚是为什么，因为终端输出的其他汉字都可以正常显示，为什么学生姓名不能正常显示呢？暂时还有疑问。

