

车载软件设计基础

课后实践2



**2022至2023学年第 1 学期**

|  |  |
| --- | --- |
| 学号 | 姓名 |
| E2020196 | 侯加喜 |
| 任课教师 | 陈恒鑫 |
| 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**

**1.1目录结构**

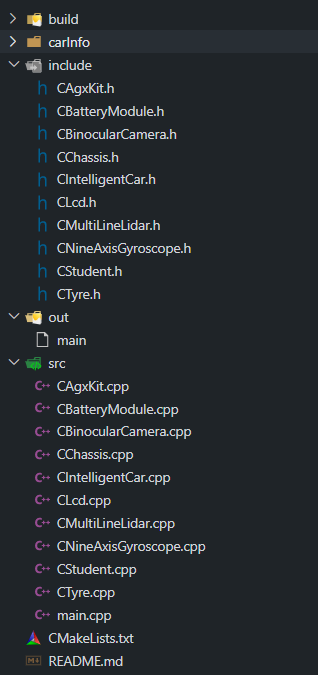
****

图 1.1-1 目录结构

**1.2数据结构**

根据题意可知，智能小车由底盘、Agx套件、轮胎、摄像头等部件组成，并且每位学生分配一辆智能小车。故首先设计了10个class用于表示这些组件及学生，数值通过程序输入。具体数据结构如下：

小车整体数据结构CIntelligentCar:



图1.2-1 CIntelligentCar

底盘数据结构CChassis:



图 1.2-2 CChassis

其中轮胎的数据结构CTyre为:

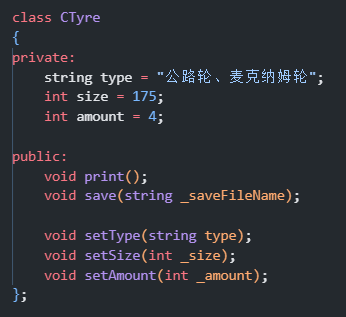


图1.2-3 CTyre

Agx套件数据结构:

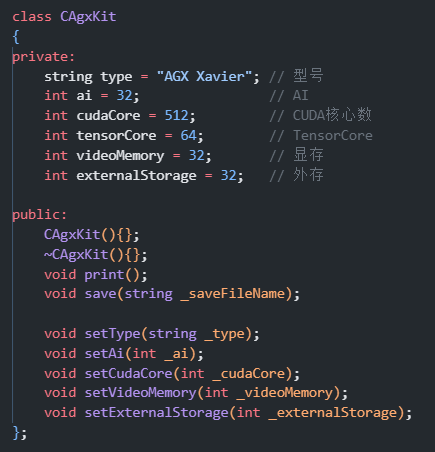


图 1.2-4 CAgxKit

双目摄像头数据结构:



图 1.2-5 CBinocularCamera

多线激光雷达数据结构:



图 2.2-6 CMultiLineLidar

九轴陀螺仪:

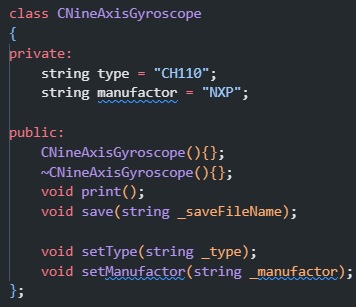


图1.2-7 CNineAxisGyroscope

液晶显示屏:

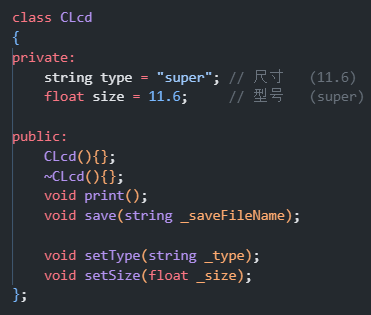


图 3.2-8 CLcd

电池模块:

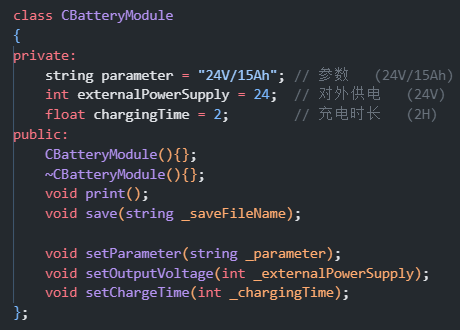


图 4.2-9 CBatteryModule

学生信息:

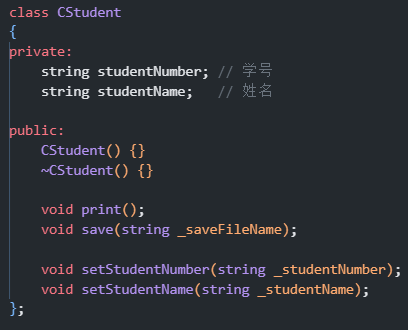


图1.2-10 CStudent

**1.3关键函数**

(1).保存信息

通过C**intelligentCar**类中的**save()**方法读入信息并将分配信息保存到硬盘中。该函数**依次调用**各个模块类中的save方法保存各自的信息。主要方式是通过ofstream流打开指定路径的文件，以追加模式进行写入。

void CIntelligentCar::save()

{

ofstream fw;

string \_saveFileName = this->saveFileName;

// 清空文件

fw.open(\_saveFileName, ios::trunc);

fw.clear();

fw.close();

// 开始写入

fw.open(\_saveFileName, ios::app);

fw << "-----------------智能小车信息------------------" << endl;

fw << "小车编号: " << this->id << endl;

this->chassis.save(\_saveFileName);

this->agxKit.save(\_saveFileName);

this->binocularCamera.save(\_saveFileName);

this->multiLineLidar.save(\_saveFileName);

this->nineAxisGyroscope.save(\_saveFileName);

this->lcd.save(\_saveFileName);

this->batteryModule.save(\_saveFileName);

fw << "-----------------------------------------------" << endl;

this->student.save(\_saveFileName);

fw << endl;

fw.close();

return;

}

(2).读取信息

通过**CIntelligentCar**类中的**print()**方法将硬盘中的文本数据读取到控制台，首先通过ifstream流指定文件路径，以ios::in模式打开，按行读取数据不断存储到string buffer中。

void CChassis::print()

{

cout << "<底盘信息>" << endl;

cout << "编号: " << this->id << endl;

cout << "型号: " << this->type << endl;

cout << "轴距: " << this->wheelbase << endl;

cout << "轮距: " << this->wheeltrack << endl;

cout << "最小离地间隙: " << this->minGroundClearance << endl;

cout << "最小转弯半径: " << this->minTurningRadius << endl;

cout << "驱动模式: " << this->drivingForm << endl;

cout << "最大行程: " << this->maxStroke << endl;

this->tyre.print();

return;

}

1.2.3最后，在**int main()**函数通过while循环不断监听用户输入，当输入n时，读取下一个学生信息，输入p时，读取上一个学生信息。当然，当展示第一组数据时，输入P无效。处于最后一组数据时，输入N无效，以防止数组下标越界异常。

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

#include "../include/CChassis.h"

#include "../include/CTyre.h"

#include "../include/CAgxKit.h"

#include "../include/CBinocularCamera.h"

#include "../include/CMultiLineLidar.h"

#include "../include/CNineAxisGyroscope.h"

#include "../include/CLcd.h"

#include "../include/CBatteryModule.h"

#include "../include/CStudent.h"

#include "../include/CIntelligentCar.h"

int main()

{

int nums;

string carId;

string chassisId;

string studentNumber;

string studentName;

CTyre tyre; // 轮胎

CChassis chassis; // 底盘

CAgxKit agxKit; // Agx套件

CBinocularCamera binocularCamera; // 双目摄像头

CMultiLineLidar multiLineLidar; // 多线激光雷达

CNineAxisGyroscope nineAxisGyroscope; // 9轴陀螺仪

CLcd lcd; // 液晶显示屏

CBatteryModule batteryModule; // 电池

cout << "请输入学生人数： ";

cin >> nums;

CStudent students[nums];

CIntelligentCar \*intelligentCars[nums];

for (int i = 0; i < nums; ++i)

{

cout << "请输入第" << i + 1 << "位学生学号: ";

cin >> studentNumber;

students[i].setStudentNumber(studentNumber);

cout << "请输入第" << i + 1 << "位学生姓名: ";

cin >> studentName;

students[i].setStudentName(studentName);

}

system("clear");

for (int i = 0; i < nums; i++)

{

cout << "<开始构建第" << i + 1 << " 辆小车>" << endl;

cout << "请输入小车编号: ";

cin >> carId;

cout << "请输入底盘编号: ";

cin >> chassisId;

chassis.setId(chassisId);

chassis.setTyre(tyre);

intelligentCars[i] = new CIntelligentCar(carId, students[i], chassis, agxKit, binocularCamera, multiLineLidar, nineAxisGyroscope, lcd, batteryModule);

intelligentCars[i]->setIsAllocated(true);

}

system("clear");

cout << "小车分配完成" << endl;

cout << "按y查看小车信息,按n退出: ";

char ch;

cin >> ch;

if (ch == 'y')

{

intelligentCars[0]->print();

}

else

{

return 0;

}

for (int i = 0; true;)

{

cout << "按n查看下一辆小车,按p查看上一辆小车，否则退出: ";

cin >> ch;

switch (ch)

{

case 'n':

if (i == nums - 1)

{

cout << "无下一辆小车,请重新输入：" << endl;

continue;

}

else

{

++i;

}

break;

case 'p':

if (i == 0)

{

cout << "无上一辆小车,请重新输入：" << endl;

continue;

}

else

{

--i;

}

break;

case 'q':

cout << "查看小车信息结束" << endl;

return 0;

}

system("clear");

intelligentCars[i]->print();

}

return 0;

}

1. **程序效果展示**

**2.1编译源代码**

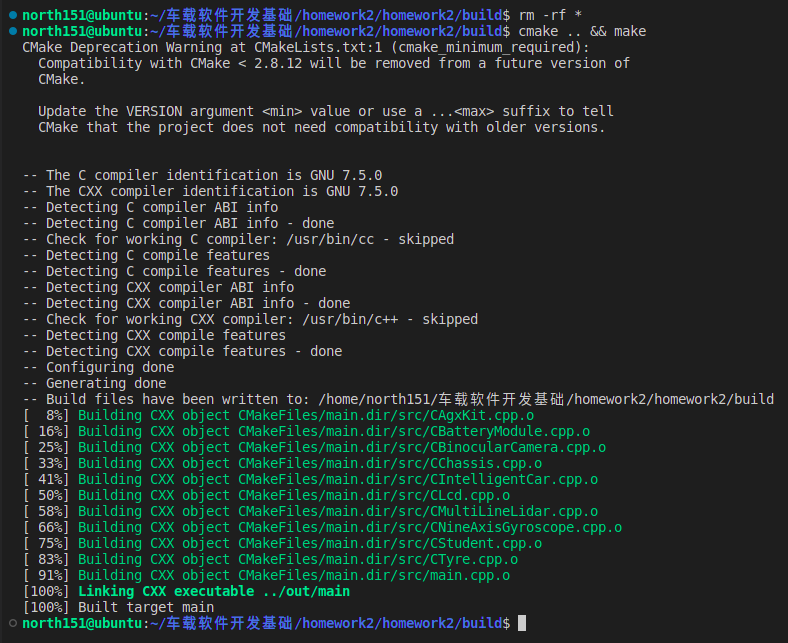


图 2.1-1 编译源代码

**2.2录入效果展示**

程序首先输入学生人数,学生姓名及其学号:

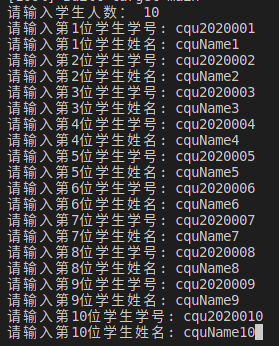


图 2.2-1 输入学生信息

然后开始构建10辆小车:

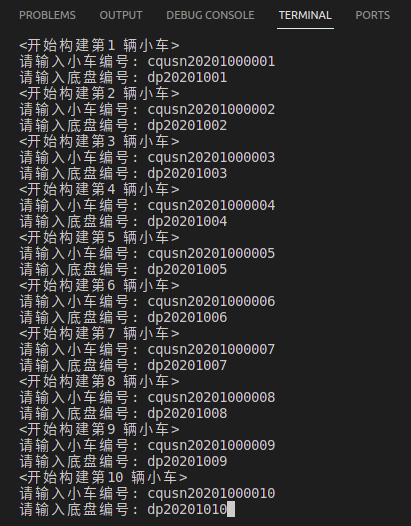


图2.2-2 构建小车

**2.3保存效果展示**

信息录入完毕后,将通过save()函数将分配信息保存至本地硬盘文件中，形式如下：

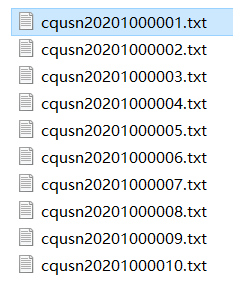


图 2.3-1 carInfo

文件内容:



图 2.3-2 cqusn20201000001.txt

**2.4读取效果展示**

完成10辆小车的构建后，开始读取通过print()方法读取之前保存到硬盘的文本文件，打印输出到命令行窗口，并持续监听用户输入，按下N展示下一组数据，按下P展示上一组数据。小车信息。

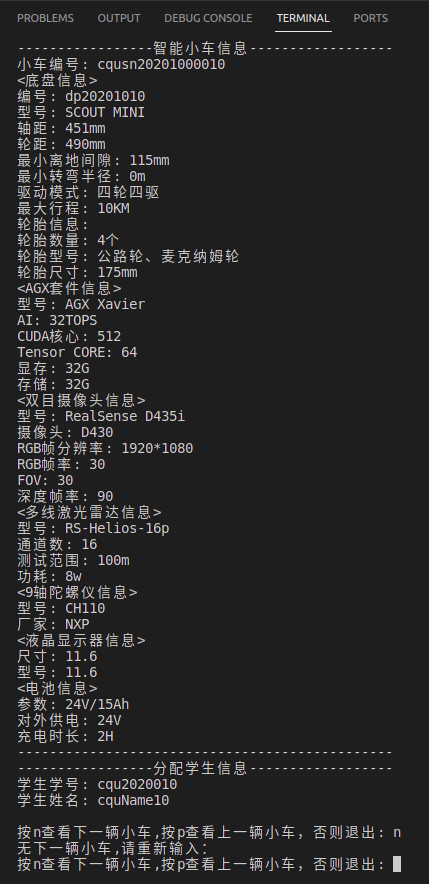


图2.4-1 读取效果展示

1. **总结和分析**

[Github](https://github.com/north151/carTask): https://github.com/north151/carTask

[Gitee](https://gitee.com/north151/homework2): https://gitee.com/north151/homework2

此次实验效果基本达到预期，能够通过程序录入小车信息、学生信息和分配信息并保存至硬盘中进行读写，较好地实现了报告所要求的各个功能。但是在编码过程中也遇到许多问题：

1.类设计阶段采用何种设计模式存在困惑，在学习面向对象编程时最重要三个特征就是封装、继承、多态。那么在类关系设计时第一反应便是采用继承，但经过分析与初步编码测试后，发现继承并不适合该题目要求，编写出的程序也很难达到复用的目的。因为车辆很明显是由底盘、轮胎、摄像头等部件组合而成，是部分与整体的关系，所以最后选择采用组合进行类关系设计，并顺利完成编码。

2.由于小车各个模块的对象变量为private类型,所以不能从main.cpp中直接访问各个模块对象,需要通过小车对象的public方法来间接访问,例如:通过小车的save()方法调用其他各个模块的save方法。

1. **参考文献**

[1]C++ 组合 (Composition)模式详解.http://t.csdn.cn/XUqVc,2021-05-14.

[2]C/C++编写命名规范.http://t.csdn.cn/VzFcz,2022-05-21.

[3]C++ 保存数据的四种方式.http://t.csdn.cn/6g3xK,2021-03-20.

[4]C++面向对象设计原则详解.http://t.csdn.cn/KkfqZ,2020-03-29.

[5][美]Stephen Prata.C++ Primer plus[M].张海龙 袁国忠.北京:人民邮电出版社,2015:96-183.