**湖州师范学院信息工程学院**

**学生实验报告**

课程名称： 面向对象程序设计（C++）

班 级： 20240823

实验内容： 多态与文件读写

指导教师： 高强

实验时间： 2025.6.5

实验地点： 31-409

学生姓名： 李佳诚

得 分：

**湖州师范学院信息工程学院制**

实验题目：类的应用

# **一、实验目的**

1. 掌握类的继承与虚函数机制，理解并实现运行时多态；
2. 学习重载输出运算符 operator<<，统一格式化展示类对象信息；
3. 理解交通监管系统中的对象建模方法；
4. 熟练掌握 C++ 文件读写操作，实现对象信息的持久化存储与恢复。

# **实验内容**

## 题目1要求：设计一个闯红灯处理系统，具体要求如下：

1. **定义抽象基类 Transport**

#### 属性：

* std::string name：车主姓名
* std::string plate：车牌号

#### 成员函数：

* 构造函数与**虚**析构函数；
* std::string getName() const：返回车主姓名；
* std::string getPlate() const：返回车牌号；
* virtual void handleViolation() = 0：处理闯红灯行为（纯虚函数）；
* virtual std::string toString() const：返回格式化字符串。示例：

车主:张三|车牌号:沪A1234

**注意：使用竖线 | 分隔字段，字段内标签和值之间用冒号 : 分隔。**

#### 重载输出运算符：

friend std::ostream& operator<<(std::ostream& os, const **Transport**& v);

调用 toString()，再输出字符串。

1. **定义派生类 Vehicle**

#### 新增属性：

* int points：驾驶积分，初始值为12。

#### 重写虚函数：

* handleViolation()：输出格式：

[沪A1234] 闯红灯，扣除6分，剩余X分

* toString()：调用基类 toString()，前缀添加 "Vehicle|"，末尾追加 "|积分:X"

示例：

Vehicle|车主:张三|车牌号:沪A1234|积分:6

1. **定义派生类 Ebike**

#### 新增属性：

* int times：记录该车主闯红灯累计次数。

#### 重写虚函数：

* handleViolation()：输出格式：

**张三**骑电瓶车闯红灯，已记录并进行批评教育（第X次）

* toString()：调用基类 toString()，前缀添加 "Ebike|"，末尾追加 "|批评次数:X"，示例：

Ebike|车主:李四|车牌号:E123|批评次数:2

1. **实现读写函数**

* **void writeToFile(const string& filename, const vector<Transport\*>& transports) ；**

以写的方式打开文件，遍历transports，将每一个元素转化成实际类型，然后调用<<写入文件。

将对象信息写入文本文件，每个对象占一行，直接使用已重载的 operator<< 输出。

#### 文件存储格式示例：

Vehicle|车主:张三|车牌号:沪A1234|积分:6  
 Ebike|车主:李四|车牌号:E123|批评次数:2

* **vector<Transport\*> readFromFile(const string& filename)；**

以读的方式打开文件，使用循环调用getline读取一行，然后先使用‘|’分割字符串，在使用‘|’:分割，根据读取的内容，创建对象，存入**vector<Transport\*> 中。**

1. **测试**

* 在主函数中创建一个 Ebike 对象和一个 Vehicle 对象；定义

vector<Transport\*> transports;保存。

* 通过**多态指针或引用**调用 handleViolation() 进行处理；

实现。void processViolations(const vector<Transport\*>& transports)；函数。

// 清理内存  
void cleanup(vector<Transport\*>& transports) {  
    for (auto t : transports) {  
        delete t;  
    }  
    transports.clear();  
}  
  
int main() {  
      
    // 创建测试对象  
    vector<Transport\*> transports;  
    Ebike ebike("李四", "E123");  
    Vehicle vehicle("张三", "沪A1234");  
      
    transports.push\_back(&ebike);  
    transports.push\_back(&vehicle);  
      
      
    // 统一处理闯红灯行为  
    processViolations(transports);      
      
    // 写入文件  
    string filename = "violation\_records.txt";  
    writeToFile(filename, transports);  
    cout << "\n已写入文件: " << filename << endl;  
      
    // 从文件读取并验证  
    cout << "\n从文件读取并验证:" << endl;  
    vector<Transport\*> readTransports = readFromFile(filename);  
    for (const auto& t : readTransports) {

   cout << \*t<< endl;  
    }  
      
    // 清理内存  
    cleanup(readTransports);  
      
    return 0;  
}

1. **提示：**

**数字和字符串转换**

* 数字转字符串

#include "string"

std::to\_string(123);

* 字符串转数字

#include <string>

std::string str = "123";

int num = std::stoi(str);

**分割字符串**

#include <sstream>

vector<string> split(const string& s, char delimiter) {

vector<string> tokens;

string token;

istringstream tokenStream(s);

while (getline(tokenStream, token, delimiter)) {

tokens.push\_back(token);

}

return tokens;

}

**逐行读取文件**

#include <fstream>

#include <string> int main() {

std::ifstream file("data.txt");

std::string line;

while (std::getline(file, line)) { // 逐行读取文件

std::cout << line << std::endl;

}

file.close();

return 0;

}

完整代码：

#include <iostream>

#include <string>

#include <vector>

#include <fstream>

#include <sstream>

using namespace std;

// 抽象基类 Transport

class Transport {

protected:

string name;

string plate;

public:

Transport(const string& n, const string& p) : name(n), plate(p) {}

virtual ~Transport() {}

string getName() const { return name; }

string getPlate() const { return plate; }

virtual void handleViolation() = 0; // 纯虚函数

virtual string toString() const = 0; // 纯虚函数

friend ostream& operator<<(ostream& os, const Transport& v) {

os << v.toString();

return os;

}

};

// 分割字符串函数

vector<string> split(const string& s, char delimiter) {

vector<string> tokens;

string token;

istringstream tokenStream(s);

while (getline(tokenStream, token, delimiter)) {

tokens.push\_back(token);

}

return tokens;

}

// 派生类 Vehicle

class Vehicle : public Transport {

private:

int points; // 驾驶积分

public:

Vehicle(const string& n, const string& p, int pts = 12)

: Transport(n, p), points(pts) {}

void handleViolation() override {

points -= 6;

cout << "[" << plate << "] 闯红灯，扣除6分，剩余" << points << "分" << endl;

}

string toString() const override {

return "Vehicle|车主:" + name + "|车牌号:" + plate + "|积分:" + to\_string(points);

}

};

// 派生类 Ebike

class Ebike : public Transport {

private:

int times; // 闯红灯累计次数

public:

Ebike(const string& n, const string& p, int t = 0)

: Transport(n, p), times(t) {}

void handleViolation() override {

times++;

cout << name << "骑电瓶车闯红灯，已记录并进行批评教育（第" << times << "次）" << endl;

}

string toString() const override {

return "Ebike|车主:" + name + "|车牌号:" + plate + "|批评次数:" + to\_string(times);

}

};

// 写入文件函数

void writeToFile(const string& filename, const vector<Transport\*>& transports) {

ofstream outFile(filename);

if (!outFile) {

cerr << "无法打开文件: " << filename << endl;

return;

}

for (const auto& t : transports) {

outFile << t->toString() << endl;

}

outFile.close();

}

// 读取文件函数

vector<Transport\*> readFromFile(const string& filename) {

vector<Transport\*> transports;

ifstream inFile(filename);

if (!inFile) {

cerr << "无法打开文件: " << filename << endl;

return transports;

}

string line;

while (getline(inFile, line)) {

vector<string> tokens = split(line, '|');

if (tokens.empty()) continue;

string type = tokens[0];

string name, plate;

int value = 0;

for (size\_t i = 1; i < tokens.size(); i++) {

vector<string> kv = split(tokens[i], ':');

if (kv.size() < 2) continue;

if (kv[0] == "车主") name = kv[1];

else if (kv[0] == "车牌号") plate = kv[1];

else if (kv[0] == "积分" || kv[0] == "批评次数") value = stoi(kv[1]);

}

if (type == "Vehicle") {

transports.push\_back(new Vehicle(name, plate, value));

} else if (type == "Ebike") {

transports.push\_back(new Ebike(name, plate, value));

}

}

inFile.close();

return transports;

}

// 处理违规行为函数

void processViolations(const vector<Transport\*>& transports) {

for (const auto& t : transports) {

t->handleViolation();

}

}

// 清理内存函数

void cleanup(vector<Transport\*>& transports) {

for (auto t : transports) {

delete t;

}

transports.clear();

}

int main() {

// 创建测试对象

vector<Transport\*> transports;

transports.push\_back(new Ebike("李四", "E123"));

transports.push\_back(new Vehicle("张三", "沪A1234"));

// 统一处理闯红灯行为

cout << "处理闯红灯行为：" << endl;

processViolations(transports);

// 写入文件

string filename = "violation\_records.txt";

writeToFile(filename, transports);

cout << "\n已写入文件: " << filename << endl;

// 从文件读取并验证

cout << "\n从文件读取并验证:" << endl;

vector<Transport\*> readTransports = readFromFile(filename);

for (const auto& t : readTransports) {

cout << \*t << endl;

}

// 清理内存

cleanup(transports);

cleanup(readTransports);

return 0;

}