

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Программирование | | |
| Курс | I | | | Группа | 423 |

Отчёт по контрольной работе №2

Вариант № 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 423 |  |  |  | Агишев Даниил Рустамович |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Макарук Роман Валерьевич |
|  |  |  |  | Федин Алексей Константинович |

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc147756544)

[2 Исходные данные 3](#_Toc147756545)

[3 Особые ситуации 3](#_Toc147756546)

[4 Математические методы и алгоритмы решения задач 3](#_Toc147756547)

[5 Форматы представления данных 3](#_Toc147756548)

[6 Структура программы 5](#_Toc147756549)

[7 Блок-схема алгоритма решения задачи 8](#_Toc147756550)

[8 Описание хода выполнения 9](#_Toc147756551)

[9 Результаты работы программы 9](#_Toc147756552)

[10 Выводы по заданию 11](#_Toc147756553)

[11 Исходный код полученного программного решения 11](#_Toc147756554)

## 1 Постановка задачи

Задан базовый и производный класс. Разработать поля и методы, наследуемые из базового класса, а также собственные. Компоненты производных классов. Базовый класс может быть абстрактным. Реализовать возможность получения списка объектов в контейнере. Базовый класс – средство передвижения. Поля в нем: вес, мощность мотора, скорость. Производный класс – автомобиль; производный класс второго поколения – спортивный автомобиль, грузовой автомобиль.

## 2 Исходные данные

В качестве исходный данных используется, введенные пользователем или считанные из файла данные.

## 3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

– Ввод пользователем некорректного имени файла

– Ввод пользователем некорректных значений переменных

– Выбор пользователем некорректного пункта меню

## 4 Математические методы и алгоритмы решения задач

Согласно постановке задачи для получения необходимых списков будут использоваться операторы “>”, “<” и “==”.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 Форматы представления данных Таблица 1 - Классы, используемые в программе   |  |  | | --- | --- | | Имя | Описание | | transport | Хранит информацию о транспортном средстве | | car | Хранит информацию об автомобиле | | sportsCar | Хранит информацию о спортивном автомобиле | | truck | Хранит информацию о грузовом автомобиле |   Таблица 2 – Поля класса transport   |  |  |  |  | | --- | --- | --- | --- | | Имя | Описание | Модификатор | Тип | | typeOfTransport | Тип транспорта | protected | string | | weight | Вес транспорта | protected | int | | engine\_power | Мощность мотора | protected | int | | speed | Скорость транспорта | protected | Int |   Таблица 3 – Поля класса car   |  |  |  |  | | --- | --- | --- | --- | | Имя | Описание | Модификатор | Тип | | typeOfTransport | Тип транспорта | protected | string | | weight | Вес транспорта | protected | int | | engine\_power | Мощность мотора | protected | int | | speed | Скорость транспорта | protected | int | | brand | Марка автомобиля | protected | string |   Таблица 4 – Поля класса sportsCar   |  |  |  |  | | --- | --- | --- | --- | | Имя | Описание | Модификатор | Тип | | typeOfTransport | Тип транспорта | protected | string | | weight | Вес транспорта | protected | int | | engine\_power | Мощность мотора | protected | int | | speed | Скорость транспорта | protected | int | | brand | Марка автомобиля | protected | string | | model | Модель автомобиля | private | string |   Таблица 5 – Поля класса truck   |  |  |  |  | | --- | --- | --- | --- | | Имя | Описание | Модификатор | Тип | | typeOfTransport | Тип транспорта | protected | string | | weight | Вес транспорта | protected | int | | engine\_power | Мощность мотора | protected | int | | speed | Скорость транспорта | protected | int | | brand | Марка автомобиля | protected | string | | capacity | Грузоподъемность | private | string |   Данные о каждом поле класса должны быть расположены на разных строках. 6 Структура программы Таблица 6 - Модули программы   |  |  | | --- | --- | | Название модуля | Описание | | Classes | Содержит классы и их методы | | Enums | Содержит данные типа enum и типа define | | Files | Содержит функции для работы с файлами | | Functions | Содержит функции добавления и удаления элементов вектора, и вывода всех элементов вектор | | Input | Содержит функции для ввода данных | | main | Содержит функцию main() | | MenuInterface | Содержит функции меню | | Tests | Содержит функции модульных тестов программы |   Таблица 7 – Основные функции модуля Files   |  |  | | --- | --- | | Имя | Описание | | IsDataCorrect | Проверка данных в файле | | CheckFile | Проверка адреса файла | | GetFromFile | Заполнение вектора данными из файла | | SaveToFile | Сохранение вектора в файл |   Таблица 8 – Основные функции модуля Functions   |  |  | | --- | --- | | Имя | Описание | | Add | Добавление элемента в вектор | | Delete | Удаление элемента из вектора | | PrintAll | Вывод всех элементов вектора на экран | |  |
|  |  |

Таблица 9 – Основные функции модуля Input

|  |  |
| --- | --- |
| Имя | Описание |
| GetInt | Считывание целого числа из консоли |
| GetBool | Считывание логического типа данных из консоли |
| GetUnsInt | Считывание неотрицательного целого числа из консоли |
| TypeOfTransportChoice | Считывания пункта меню типов транспортных средств из консоли |
| MainMenuChoice | Считывание пункта основного меню из консоли |
| SecondMenuChoice | Считывание пункта второго меню из консоли |

Таблица 10 – Функции модуля MenuInterface

|  |  |
| --- | --- |
| Имя | Описание |
| StartProgram | Запуск основных функций программы |
| ShowMainMenu | Вывод основного меню на экран |
| ShowSecondMenu | Вывод второго меню на экран |
| ShowSaveMenu | Вывод на экран меню сохранения вектора в файл |

Таблица 8 – Функции модуля Tests

|  |  |
| --- | --- |
| Имя | Описание |
| StartTests | Функция, содержащая тесты |
| CreateTestVector | Функция создания тестового вектора |
| CompareVectors | Функция сравнение векторов |

## 7 Блок-схема алгоритма решения задачи

На рисунке 1 представлена блок-схема алгоритма решения задачи.

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

Рисунок 1 – Блок-схема алгоритма решения задачи

## 8 Описание хода выполнения

Для выполнения задания необходимо изучить наследование классов в языке C++.

После обучения этому приступили к реализации основных задач программы – созданию базового класса и классов-наследников, заполнению полей и разбиению структуры программы на модули.

## 9 Результаты работы программы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2 – Пример вывода на экран всех элементов вектора

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3 – Пример работы функции добавления элемента в вектор

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 4 – Пример работы функции удаления элемента в векторе

## 10 Выводы по заданию

В ходе выполнения задания были изучены методы работы с языком С++ и реализацией Базовых и производных классов. Были устранены недостатки в знаниях библиотек и получены новые навыки программирования.

## 11 Исходный код полученного программного решения

**Classes.h**

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include "Input.h"

using namespace std;

////////////////////////////////////////////////////////////////

// means of transportation

class transport {

protected:

string typeOfTransport;

int weight;

int engine\_power;

int speed;

public:

transport();

transport(string typeOfTransport\_, int weight\_, int engine\_power\_, int speed\_);

void SetType(string typeOfTransport\_);

void SetWeight(int weight\_);

void SetEnginePower(int engine\_power\_);

void SetSpeed(int speed\_);

virtual void SetAll();

string GetType();

int GetWeight();

int GetEnginePower();

int GetSpped();

void ShowWeight();

void ShowEnginePower();

void ShowSpeed();

virtual void ShowAll();

virtual ~transport() {}

};

////////////////////////////////////////////////////////////////

// class cars

class car : public transport {

protected:

string brand;

public:

car();

car(string typeOfTransport\_, int weight\_, int engine\_power\_, int speed\_, string brand\_);

void SetBrand(string brand\_);

void SetAll() override;

string GetBrand();

void ShowBrand();

void ShowAll() override;

};

////////////////////////////////////////////////////////////////

// class sports car

class sportsCar : public car {

private:

string model;

public:

sportsCar();

sportsCar(string typeOfTransport\_, int weight\_, int engine\_power\_, int speed\_, string brand\_, string model\_);

void SetModel(string model\_);

void SetAll() override;

string GetModel();

void ShowModel();

void ShowAll() override;

};

////////////////////////////////////////////////////////////////

// class truck

class truck : public car {

protected:

int capacity = 0;

public:

truck();

truck(string typeOfTransport\_, int weight\_, int engine\_power\_, int speed\_, string brand\_, int capacity\_);

void SetCapacity(int capacity\_);

void SetAll() override;

int GetCapacity();

void ShowCapacity();

void ShowAll() override;

};

**Classes.cpp**

#include "classes.h"

////////////////////////////////////////////////////////////////

// class transport

transport::transport() {

typeOfTransport = "empty";

weight = 0;

engine\_power = 0;

speed = 0;

}

transport::transport(string typeOfTransport\_, int weight\_, int engine\_power\_, int speed\_) {

typeOfTransport = typeOfTransport\_;

weight = weight\_;

engine\_power = engine\_power\_;

speed = speed\_;

}

void transport::SetType(string typeOfTransport\_) {

typeOfTransport = typeOfTransport\_;

}

void transport::SetWeight(int weight\_) {

weight = weight\_;

}

void transport::SetEnginePower(int engine\_power\_) {

engine\_power = engine\_power\_;

}

void transport::SetSpeed(int speed\_) {

speed = speed\_;

}

void transport::SetAll() {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Введите вес: "; weight = GetUnsInt();

cout << "Введите мощность двигателя: "; engine\_power = GetUnsInt();

cout << "Введите максимальную скорость: "; speed = GetUnsInt();

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

string transport::GetType() {

return typeOfTransport;

}

int transport::GetWeight() {

return weight;

}

int transport::GetEnginePower() {

return engine\_power;

}

int transport::GetSpped() {

return speed;

}

void transport::ShowWeight() {

cout << weight;

}

void transport::ShowEnginePower() {

cout << engine\_power;

}

void transport::ShowSpeed() {

cout << speed;

}

void transport::ShowAll() {

cout << "Тип: " << typeOfTransport << endl

<< "Вес: " << weight << endl

<< "Мощность двигателя: " << engine\_power << endl

<< "Скорость: " << speed << endl;

}

////////////////////////////////////////////////////////////////

// class car

car::car() {

weight = 0;

engine\_power = 0;

speed = 0;

brand = "empty";

}

car::car(string typeOfTransport\_, int weight\_, int engine\_power\_, int speed\_, string brand\_)

: transport(typeOfTransport\_, weight\_, engine\_power\_, speed\_) {

brand = brand\_;

}

void car::SetBrand(string brand\_) {

brand = brand\_;

}

void car::SetAll() {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Введите вес: "; weight = GetUnsInt();

cout << "Введите мощность двигателя: "; engine\_power = GetUnsInt();

cout << "Введите максимальную скорость: "; speed = GetUnsInt();

cout << "Введите марку: "; cin >> brand;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

string car::GetBrand() {

return brand;

}

void car::ShowBrand() {

cout << brand;

}

void car::ShowAll() {

cout << "Тип: " << typeOfTransport << endl

<< "Вес: " << weight << endl

<< "Мощность двигателя: " << engine\_power << endl

<< "Скорость: " << speed << endl

<< "Марка: " << brand << endl;

}

////////////////////////////////////////////////////////////////

// class sports car

sportsCar::sportsCar() {

weight = 0;

engine\_power = 0;

speed = 0;

brand = "empty";

model = "empty";

}

sportsCar::sportsCar(string typeOfTransport\_, int weight\_, int engine\_power\_, int speed\_, string brand\_, string model\_)

:car(typeOfTransport\_, weight\_, engine\_power\_, speed\_, brand\_) {

model = model\_;

}

void sportsCar::SetModel(string model\_) {

model = model\_;

}

void sportsCar::SetAll() {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Введите вес: "; weight = GetUnsInt();

cout << "Введите мощность двигателя: "; engine\_power = GetUnsInt();

cout << "Введите максимальную скорость: "; speed = GetUnsInt();

cout << "Введите марку: "; cin >> brand;

cout << "Введите модель:"; cin >> model;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

string sportsCar::GetModel() {

return model;

}

void sportsCar::ShowModel() {

cout << model;

}

void sportsCar::ShowAll() {

cout << "Тип: " << typeOfTransport << endl

<< "Вес: " << weight << endl

<< "Мощность двигателя: " << engine\_power << endl

<< "Скорость: " << speed << endl

<< "Марка: " << brand << endl

<< "Модель: " << model << endl;

}

////////////////////////////////////////////////////////////////

// class truck

truck::truck() {

weight = 0;

engine\_power = 0;

speed = 0;

brand = "empty";

capacity = 0;

}

truck::truck(string typeOfTransport\_, int weight\_, int engine\_power\_, int speed\_, string brand\_, int capacity\_)

:car(typeOfTransport\_, weight\_, engine\_power\_, speed\_, brand\_) {

capacity = capacity\_;

}

void truck::SetCapacity(int capacity\_) {

capacity = capacity\_;

}

void truck::SetAll() {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Введите вес: "; weight = GetUnsInt();

cout << "Введите мощность двигателя: "; engine\_power = GetUnsInt();

cout << "Введите максимальную скорость: "; speed = GetUnsInt();

cout << "Введите марку: "; cin >> brand;

cout << "Введите грузоподъемность:"; capacity = GetUnsInt();

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

int truck::GetCapacity() {

return capacity;

}

void truck::ShowCapacity() {

cout << capacity;

}

void truck::ShowAll() {

cout << "Тип: " << typeOfTransport << endl

<< "Вес: " << weight << endl

<< "Мощность двигателя :" << engine\_power << endl

<< "Скорость: " << speed << endl

<< "Марка: " << brand << endl

<< "Грузоподъемность: " << capacity << endl;

}

**Enums.h:**

#pragma once

#define TestSize 4

#define ControlSize 5

enum MainMenu { file = 1, Manual };

enum SecondMenu { showList = 1, addTransport, deleteTransport, saveList, back, closeProgram };

enum TypeOfTransport { carType = 1, sportsCarType, truckType, transportType };

**Files.h:**

#pragma once

#include <iostream>

#include <fstream>

#include <filesystem>

#include <string>

#include "Functions.h"

using namespace filesystem;

bool IsDataCorrect(ifstream &file); //Проверка корректности данных в файле

string CheckFile(); //Проверка корректности имени файла

void GetFromFile(vector<any> &Arr, string path); //Счиитывание из файла

void SaveToFile(vector<any> &Arr); //Сохранение в файл

**Files.cpp:**

#include "Files.h"

bool IsDataCorrect(ifstream& file) {

string testString = "";

int testInt = 0;

while (!file.eof()) {

getline(file, testString);

if (testString == "Car") {

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

getline(file, testString);

}

else if (testString == "Sport\_car") {

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

getline(file, testString);

getline(file, testString);

}

else if (testString == "Truck") {

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

getline(file, testString);

file >> testInt; if (file.fail()) return false; file.get();

}

else if (testString == "Transport") {

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

file >> testInt; if (file.fail()) return false; if (testInt < 0) return false; file.get();

}

}

file.seekg(0, ios::beg);

return true;

}

string CheckFile()

{

error\_code error{};

string path{};

bool isLoaded = false;

ifstream file;

do

{

cout << "Введите путь к файлу: ";

getline(cin, path);

if (!ifstream(path))

{

cout << "Указанный файл не найден!" << endl;

continue;

}

if (!is\_regular\_file(path, error))

{

cout << "Ошибка, недопустимый адрес файла!" << endl;

continue;

}

file.open(path);

if (!file)

{

cout << "Загрузка запрещена. Повторите ввод." << endl;

file.close();

}

else if (!IsDataCorrect(file))

{

cout << "Файл содержит ненкорректные данные." << endl;

file.close();

}

else

isLoaded = true;

} while (!isLoaded);

cout << "Файл открыт." << endl;

file.close();

return path;

}

void GetFromFile(vector<any>& Arr, string path) {

ifstream file;

file.open(path);

while (!file.eof()) {

int tmpInt = 0;

string tmpString = "";

string typeT = "";

getline(file, typeT);

if (typeT == "Car") {

car tmp;

tmp.SetType(typeT);

file >> tmpInt; file.get();

tmp.SetWeight(tmpInt);

file >> tmpInt; file.get();

tmp.SetEnginePower(tmpInt);

file >> tmpInt; file.get();

tmp.SetSpeed(tmpInt);

getline(file, tmpString);

tmp.SetBrand(tmpString);

Arr.push\_back(tmp);

}

else if (typeT == "Sport\_car") {

sportsCar tmp;

tmp.SetType(typeT);

file >> tmpInt; file.get();

tmp.SetWeight(tmpInt);

file >> tmpInt; file.get();

tmp.SetEnginePower(tmpInt);

file >> tmpInt; file.get();

tmp.SetSpeed(tmpInt);

getline(file, tmpString);

tmp.SetBrand(tmpString);

getline(file, tmpString);

tmp.SetModel(tmpString);

Arr.push\_back(tmp);

}

else if (typeT == "Truck") {

truck tmp;

tmp.SetType(typeT);

file >> tmpInt; file.get();

tmp.SetWeight(tmpInt);

file >> tmpInt; file.get();

tmp.SetEnginePower(tmpInt);

file >> tmpInt; file.get();

tmp.SetSpeed(tmpInt);

getline(file, tmpString);

tmp.SetBrand(tmpString);

file >> tmpInt; file.get();

tmp.SetCapacity(tmpInt);

Arr.push\_back(tmp);

}

else if (typeT == "Transport") {

transport tmp;

tmp.SetType(typeT);

file >> tmpInt; file.get();

tmp.SetWeight(tmpInt);

file >> tmpInt; file.get();

tmp.SetEnginePower(tmpInt);

file >> tmpInt; file.get();

tmp.SetSpeed(tmpInt);

Arr.push\_back(tmp);

}

}

cout << "Список загружен." << endl;

file.close();

}

void SaveToFile(vector<any>& Arr) {

string path = "";

bool isDataSaved = false;

do {

cout << "Сохранить в: " << endl;

cin >> path;

if (ifstream(path)) {

cout << "Файл уже существует." << endl;

cout << "[0] - Перезаписать существующий файл." << endl;

cout << "[1] - Повторить ввод." << endl;

int tryAnotherFile = GetBool();

if (tryAnotherFile) {

continue;

}

}

ofstream myFile(path, ofstream::app);

error\_code ec{};

if (!is\_regular\_file(path, ec)) {

cout << "Адрес содержит недопустимые значения. Повторите ввод." << endl;

continue;

}

if (!myFile) {

cout << "Запись запрещена. Повторите ввод." << endl;

myFile.close();

continue;

}

myFile.close();

myFile.open(path, ofstream::trunc);

for (unsigned int i = 0; i < Arr.size(); i++) {

string typeN = Arr[i].type().name();

if (typeN == "class car") {

car tmpVec = any\_cast<car>(Arr[i]);

myFile << tmpVec.GetType() << endl

<< tmpVec.GetWeight() << endl

<< tmpVec.GetEnginePower() << endl

<< tmpVec.GetSpped() << endl

<< tmpVec.GetBrand() << endl;

}

else if (typeN == "class sportsCar") {

sportsCar tmpVec = any\_cast<sportsCar>(Arr[i]);

myFile << tmpVec.GetType() << endl

<< tmpVec.GetWeight() << endl

<< tmpVec.GetEnginePower() << endl

<< tmpVec.GetSpped() << endl

<< tmpVec.GetBrand() << endl

<< tmpVec.GetModel() << endl;

}

else if (typeN == "class truck") {

truck tmpVec = any\_cast<truck>(Arr[i]);

myFile << tmpVec.GetType() << endl

<< tmpVec.GetWeight() << endl

<< tmpVec.GetEnginePower() << endl

<< tmpVec.GetSpped() << endl

<< tmpVec.GetBrand() << endl

<< tmpVec.GetCapacity() << endl;

}

else if (typeN == "class transport") {

transport tmpVec = any\_cast<transport>(Arr[i]);

myFile << tmpVec.GetType() << endl

<< tmpVec.GetWeight() << endl

<< tmpVec.GetEnginePower() << endl

<< tmpVec.GetSpped() << endl;

}

}

myFile.close();

cout << "Запись завершена." << endl;

isDataSaved = true;

} while (!isDataSaved);

}

**Input.h:**

#pragma once

#include <iostream>

#include "Classes.h"

#include "Enums.h"

int GetInt(); //Проверка ввода целого числа

bool GetBool(); //Проверка ввода булевого значения

int GetUnsInt(); //Проверка ввода целого неотрицательного числа

TypeOfTransport TypeOfTransportChoice(); //Проверка ввода типа транспортного средства

MainMenu MainMenuChoice(); //Проверка ввода пункта первого меню

SecondMenu SecondMenuChoice(); //Проверка ввода пункта второго меню

**Input.cpp:**

#include "Input.h"

using namespace std;

//Создание шаблона с параметром типа Т

template<typename T>

T GetInput() {

T userInput;

cin >> userInput;

while (cin.fail()) { //cin.fail() - определяет соответствует ли введеное значение значению, определенному в переменной

cout << "Повторите ввод." << endl;

cin.clear(); //Сброс ошибки

cin.ignore(INT\_MAX, '\n'); //Извлечение символов из входного потока, извлеченные символы не используются

cin >> userInput;

}

cin.ignore(INT\_MAX, '\n');

return userInput;

}

int GetInt() {

return GetInput<int>();

}

bool GetBool() {

return GetInput<bool>();

}

int GetUnsInt() {

int input = GetInput<int>();

if (input < 0) {

cout << "Взято значение по модулю." << endl;

input = abs(input);

}

return input;

}

TypeOfTransport TypeOfTransportChoice() {

return static\_cast<TypeOfTransport>(GetInt());

}

MainMenu MainMenuChoice() {

return static\_cast<MainMenu>(GetInt());

}

SecondMenu SecondMenuChoice() {

return static\_cast<SecondMenu>(GetInt());

}

**Functions.h:**

#pragma once

#include "Classes.h"

#include <any>

void Add(vector<any> &Arr, bool test); //Добавление объекта в список

void Delete(vector<any>& Arr, bool test); //Удаление объекта из списка

void PrintAll(vector<any>& Arr); //Вывод списка на экран

**Functions.cpp:**

#include "Functions.h"

void Add(vector<any> &Arr, bool test) {

bool exitFlag;

TypeOfTransport choice;

do {

exitFlag = true;

if (!test) {

cout << "Какой тип добавить?" << endl

<< "1 - Автомобиль" << endl

<< "2 - Спортивный автомобиль" << endl

<< "3 - Грузовой автомобиль" << endl

<< "4 - Транспортное средство" << endl;

choice = TypeOfTransportChoice();

}

else {

choice = transportType;

}

if (choice == carType) {

car tmp;

tmp.SetAll();

tmp.SetType("Car");

Arr.push\_back(tmp);

}

else if (choice == sportsCarType) {

sportsCar tmp;

tmp.SetAll();

tmp.SetType("Sport\_car");

Arr.push\_back(tmp);

}

else if (choice == truckType) {

truck tmp;

tmp.SetAll();

tmp.SetType("Truck");

Arr.push\_back(tmp);

}

else if (choice == transportType) {

transport tmp;

if (!test) {

tmp.SetAll();

}

else {

tmp.SetWeight(0);

tmp.SetEnginePower(0);

tmp.SetSpeed(0);

}

tmp.SetType("Transport");

Arr.push\_back(tmp);

}

else {

cout << "Нет такого пункта" << endl;

exitFlag = false;

}

} while (!exitFlag);

}

void Delete(vector<any> &Arr, bool test) {

unsigned int number = 0;

if (test) {

number = ControlSize;

}

else {

cout << "Введите номер элемента, который нужно удалить: ";

number = GetInt();

while (number > Arr.size() || number < 1) {

cout << "Введено некорректное значение." << endl;

number = GetInt();

}

}

for (unsigned int i = 0; i < Arr.size(); i++) {

if (i == number - 1) {

for (i; i < Arr.size() - 1; i++) {

Arr[i] = Arr[static\_cast<std::vector<std::any, std::allocator<std::any>>::size\_type>(i) + 1];

}

Arr.pop\_back();

break;

}

}

}

void PrintAll(vector<any>& Arr) {

for (unsigned int i = 0; i < Arr.size(); i++) {

cout << "№" << (i + 1) << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

string typeN = Arr[i].type().name();

if (typeN == "class car") {

car tmp;

tmp = any\_cast<car>(Arr[i]);

Arr[i] = tmp;

any\_cast<car>(Arr[i]).ShowAll();

}

else if (typeN == "class sportsCar") {

sportsCar tmp;

tmp = any\_cast<sportsCar>(Arr[i]);

Arr[i] = tmp;

any\_cast<sportsCar>(Arr[i]).ShowAll();

}

else if (typeN == "class truck") {

truck tmp;

tmp = any\_cast<truck>(Arr[i]);

Arr[i] = tmp;

any\_cast<truck>(Arr[i]).ShowAll();

}

else if (typeN == "class transport") {

transport tmp;

tmp = any\_cast<transport>(Arr[i]);

Arr[i] = tmp;

any\_cast<transport>(Arr[i]).ShowAll();

}

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

}

**main.cpp:**

#include <iostream>

#include <clocale>

#include "MenuInterface.h"

#include "Tests.h"

using namespace std;

int main() {

setlocale(LC\_CTYPE, "Russian");

cout << "Задается базовый и производный класс. Необходимо разработать поля и методы," << endl

<< "наследуемые из базового класса, а также собственные компоненты базовых классов" << endl

<< "Базовй класс - средство передвижения. Поля в нем: вес, мощность мотора, скорость" << endl

<< "Производный класс - автомобиль" << endl

<< "Производный класс второго поколения - спортивный автомобиль, грузовой автомобиль" << endl << endl;

cout << "Агишев Даниил" << endl

<< "Контрольная работа №2" << endl

<< "Вариант №1" << endl

<< "группа 423" << endl << endl;

if (StartTests()) {

cout << endl << "Тесты пройдены успешно." << endl << endl;

StartProgram();

}

else {

cout << endl << "Тесты провалены." << endl;

}

return EXIT\_SUCCESS;

}

**MenuInterface.h:**

#pragma once

#include "Files.h"

void ShowMainMenu(); //Вывод первого меню

void ShowSecondMenu(); //Вывод второго меню

void ShowSaveMenu(vector<any>& Arr); //Вывод меню сохранения в файл

int StartProgram(); //Запуск основных функций

**MenuInterface.cpp:**

#include "MenuInterface.h"

void ShowMainMenu() {

cout << "Выберите пункт меню:" << endl;

cout << "1 - Загрузить из файла" << endl;

cout << "2 - Ввести вручную" << endl << endl;

}

void ShowSecondMenu() {

cout << "Выберите пункт меню:" << endl;

cout << "1 - Вывести список" << endl;

cout << "2 - Добавить транспорт в список" << endl;

cout << "3 - Удалить транспорт из списка" << endl;

cout << "4 - Сохранить список в файл" << endl;

cout << "5 - Вернуться в основное меню" << endl;

cout << "6 - Закрыть программу" << endl << endl;

}

void ShowSaveMenu(vector<any> &Arr) {

cout << "Сохранить список в файл?" << endl

<< "1 - ДА 2 - НЕТ" << endl;

if (GetBool()) {

SaveToFile(Arr);

}

}

int StartProgram() {

bool exitFlag;

bool exitAllFlag;

do {

string filePath = "";

vector <any> TransportArr;

do {

exitFlag = true;

ShowMainMenu();

MainMenu MainMenuUserChoice = MainMenuChoice();

switch (MainMenuUserChoice)

{

case file:

filePath = CheckFile();

GetFromFile(TransportArr, filePath);

break;

case Manual:

Add(TransportArr, false);

break;

default:

cout << "Нет такого пункта меню" << endl << endl;

exitFlag = false;

break;

}

} while (!exitFlag);

do {

exitFlag = false;

exitAllFlag = false;

ShowSecondMenu();

SecondMenu SecondMenuUserChoice = SecondMenuChoice();

switch (SecondMenuUserChoice)

{

case showList:

if (!TransportArr.empty()) {

PrintAll(TransportArr);

}

else {

cout << "Список пуст." << endl;

}

break;

case addTransport:

Add(TransportArr, false);

break;

case deleteTransport:

if (!TransportArr.empty()) {

Delete(TransportArr, false);

}

else {

cout << "Список пуст" << endl;

}

break;

case saveList:

if (!TransportArr.empty()) {

SaveToFile(TransportArr);

}

else {

cout << "Список пуст" << endl;

}

break;

case back:

exitFlag = true;

break;

case closeProgram:

cout << "Завершение работы...";

exitAllFlag = true;

exitFlag = true;

break;

default:

cout << "Нет такого пункта меню" << endl << endl;

break;

}

} while (!exitFlag);

TransportArr.clear();

} while (!exitAllFlag);

return EXIT\_SUCCESS;

}

**Tests.h:**

#pragma once

#include "Functions.h"

bool StartTests(); //Запуск модульных тестов

void CreateTestVector(vector <any> &vec, int size); //Создание векторов для тестов

bool CompareVectors(vector <any> &testVec, vector <any>& resultVec);//Сравнение тестовых векторов с корректным резултатом

**Tests.cpp:**

#include "Tests.h"

bool StartTests() {

vector <any> testVector;

vector <any> resultVector;

bool flagAdd = false;

bool flagDel = false;

CreateTestVector(resultVector, ControlSize); //Создание вектора с исходными значениями

CreateTestVector(testVector, TestSize); //Создание вектора с результативными значениями

Add(testVector, true);

flagAdd = CompareVectors(testVector, resultVector);

testVector.clear();

resultVector.clear();

CreateTestVector(resultVector, TestSize);

CreateTestVector(testVector, ControlSize);

Delete(testVector, true);

flagDel = CompareVectors(testVector, resultVector);

if (flagAdd && flagDel) {

return true;

}

else {

return false;

}

}

void CreateTestVector(vector <any>& vec, int size) {

transport transportTmp;

car carTmp;

sportsCar sportsCarTmp;

truck truckTmp;

int param = 0;

int num = 0;

string strParam = "empty";

transportTmp.SetType("Transport");

transportTmp.SetWeight(param);

transportTmp.SetEnginePower(param);

transportTmp.SetSpeed(param);

param++;

carTmp.SetType("Car");

carTmp.SetWeight(param);

carTmp.SetEnginePower(param);

carTmp.SetSpeed(param);

carTmp.SetBrand(strParam);

param++;

sportsCarTmp.SetType("Sport\_car");

sportsCarTmp.SetWeight(param);

sportsCarTmp.SetEnginePower(param);

sportsCarTmp.SetSpeed(param);

sportsCarTmp.SetBrand(strParam);

sportsCarTmp.SetModel(strParam);

param++;

truckTmp.SetType("Truck");

truckTmp.SetWeight(param);

truckTmp.SetEnginePower(param);

truckTmp.SetSpeed(param);

truckTmp.SetBrand(strParam);

truckTmp.SetCapacity(param);

while (num < size) {

vec.push\_back(transportTmp);

num++;

if (num >= size)

break;

vec.push\_back(carTmp);

num++;

if (num >= size)

break;

vec.push\_back(sportsCarTmp);

num++;

if (num >= size)

break;

vec.push\_back(truckTmp);

num++;

if (num >= size)

break;

}

}

bool CompareVectors(vector <any>& testVec, vector <any>& resultVec) {

if (testVec.size() == resultVec.size()) {

for (unsigned int i = 0; i < testVec.size(); i++) {

string typeT = testVec[i].type().name();

string typeR = resultVec[i].type().name();

if (typeT == "class transport" && typeR == "class transport") {

transport testTmp = any\_cast<transport>(testVec[i]);

transport resultTmp = any\_cast<transport>(resultVec[i]);

if (testTmp.GetType() == resultTmp.GetType() &&

testTmp.GetWeight() == resultTmp.GetWeight() &&

testTmp.GetEnginePower() == resultTmp.GetEnginePower() &&

testTmp.GetSpped() == resultTmp.GetSpped()) {

continue;

}

else {

return false;

}

}

else if (typeT == "class car" && typeR == "class car") {

car testTmp = any\_cast<car>(testVec[i]);

car resultTmp = any\_cast<car>(resultVec[i]);

if (testTmp.GetType() == resultTmp.GetType() &&

testTmp.GetWeight() == resultTmp.GetWeight() &&

testTmp.GetEnginePower() == resultTmp.GetEnginePower() &&

testTmp.GetSpped() == resultTmp.GetSpped() &&

testTmp.GetBrand() == resultTmp.GetBrand()) {

continue;

}

else {

return false;

}

}

else if (typeT == "class sportsCar" && typeR == "class sportsCar") {

sportsCar testTmp = any\_cast<sportsCar>(testVec[i]);

sportsCar resultTmp = any\_cast<sportsCar>(resultVec[i]);

if (testTmp.GetType() == resultTmp.GetType() &&

testTmp.GetWeight() == resultTmp.GetWeight() &&

testTmp.GetEnginePower() == resultTmp.GetEnginePower() &&

testTmp.GetSpped() == resultTmp.GetSpped() &&

testTmp.GetBrand() == resultTmp.GetBrand() &&

testTmp.GetModel() == resultTmp.GetModel()) {

continue;

}

else {

return false;

}

}

else if (typeT == "class truck" && typeR == "class truck") {

truck testTmp = any\_cast<truck>(testVec[i]);

truck resultTmp = any\_cast<truck>(resultVec[i]);

if (testTmp.GetType() == resultTmp.GetType() &&

testTmp.GetWeight() == resultTmp.GetWeight() &&

testTmp.GetEnginePower() == resultTmp.GetEnginePower() &&

testTmp.GetSpped() == resultTmp.GetSpped() &&

testTmp.GetBrand() == resultTmp.GetBrand() &&

testTmp.GetCapacity() == resultTmp.GetCapacity()) {

continue;

}

else {

return false;

}

}

}

return true;

}

else {

return false;

}

}