Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Автоматизированных систем управления

*(АСУ)*

Классы. Открытые и закрытые уровни доступа.

Конструкторы. Инициализация данных объекта. Определение

методов. Создание объекта в памяти. Стандартные потоки

ввода-вывода.

ООП

Лабораторная работа 1

Вариант 14

|  |  |
| --- | --- |
|  | Выполнила:  Студентка гр. З-433П8-5    Медведева Юлия Евгеньевна    «29» октября 2024г.  (дата) |
|  | Проверил:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность, ученая степень, звание)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/  (подпись) И. О. Фамилия  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.  (дата) |

Томск 2024

Оглавление

[Введение 3](#_Toc190775477)

[Задание 5](#_Toc190775478)

[1 Теоретическая часть 6](#_Toc190775479)

[1.1 Классы C++ 6](#_Toc190775480)

[1.2 Уровни доступа в классах 6](#_Toc190775481)

[1.3 Определение методов 6](#_Toc190775482)

[1.4 Инициализация данных объекта 7](#_Toc190775483)

[1.5 Конструкторы 7](#_Toc190775484)

[1.6 Создание объекта в памяти 7](#_Toc190775485)

[1.7 Стандартные потоки ввода-вывода ЯП С++ 7](#_Toc190775486)

[1.7 Объектно-ориентированное программирование 8](#_Toc190775487)

[1.8 Принципы ООП 9](#_Toc190775488)

[2 Описание хода выполнения работы 10](#_Toc190775489)

[Заключение 13](#_Toc190775490)

[Список литературы 14](#_Toc190775491)

[Приложение 1 15](#_Toc190775492)

[Листинг 1 Файл main.h с описанием интерфейсов 15](#_Toc190775493)

[Листинг 2 Файл main.cpp с реализацией программы 16](#_Toc190775494)

[Приложение 2 18](#_Toc190775495)

[Листинг 3 Файл main.h с описанием интерфейсов при использовании ООП 18](#_Toc190775496)

[Листинг 4 Файл main.cpp с реализацией программы при использовании ООП 19](#_Toc190775497)

# Введение

Цель: изучить основы объектно-ориентированного программирования (ООП) на примере работы с классами, их конструкторами, методами и уровнями доступа, а также освоить работу со стандартными потоками ввода-вывода в языке программирования.

Задачи:

* Изучение теоретических основ:
  + Ознакомиться с понятием классов и объектов в ООП.
  + Изучить уровни доступа (открытые и закрытые) и их значение для инкапсуляции данных.
  + Понять, что такое конструкторы и как они используются для инициализации объектов.
* Практическая реализация:
  + Создать класс с открытыми и закрытыми полями, реализовать методы доступа (геттеры и сеттеры).
  + Реализовать конструктор для инициализации данных объекта при его создании.
  + Определить несколько методов для работы с данными объекта (например, метод для отображения информации об объекте).
* Работа со стандартными потоками ввода-вывода:
  + Реализовать ввод данных для инициализации объекта с использованием стандартного ввода.
  + Реализовать вывод данных об объекте с использованием стандартного вывода.
* Тестирование и отладка:
  + Провести тестирование созданного класса и его методов, убедиться в корректности работы.
  + Отладить код, если будут обнаружены ошибки.

Сборка проекта осуществлялась в IDE Microsoft Visual Studio 2022 для x82 разрядной системы Windows.

# Задание

Изучить классы и их возможности в C++. Написать программу с использованием парадигм ООП.

Предметной областью ТЗ является автомобиль. Обязательные характеристики и типы, которые должны быть использованы при написании программы:

* марка — char\*;
* мощность — int;
* стоимось — float.

# 1 Теоретическая часть

## 1.1 Классы C++

Классы — это ключевой элемент объектно-ориентированного программирования в C++. Они служат для создания новых типов данных, которые могут включать как данные (члены класса), так и функции (методы). Класс определяет шаблон или «чертеж» для создания объектов, которые представляют экземпляры этого класса. Объявляется класс с помощью ключевого слова ***class***, а тело класса заключено в фигурные скобки и заканчивается точкой с запятой.

Пример объявления класса:

class MyClass {

// Данные и методы класса

};

## 1.2 Уровни доступа в классах

В C++ существует три уровня доступа, которые регулируют видимость членов класса для других частей кода:

* public: открытые члены, доступные из любого места кода, имеющего доступ к объекту класса;
* protected: защищенные члены, доступные только в пределах класса, в котором они объявлены, а также в классах-наследниках;
* private: закрытые члены, доступные только внутри самого класса.

## 1.3 Определение методов

Методы — это функции, объявленные внутри класса. Их можно определить как внутри класса, так и за его пределами, указав имя класса и оператор разрешения области ***::*** .

class MyClass {

public:

// объявление метода

void setValue(int v);

private:

int value;

};

// Определение метода вне класса

void MyClass::setValue(int v) {

value = v;

}

## 1.4 Инициализация данных объекта

Данные объекта можно инициализировать несколькими способами:

* Через список инициализации в конструкторе;
* Путем присвоения значений после создания объекта.

## 1.5 Конструкторы

Конструкторы — это методы, которые автоматически вызываются при создании объекта. Основные особенности конструкторов:

* Имя конструктора совпадает с именем класса;
* Конструкторы не возвращают значения;
* Конструкторы могут быть перегружены для создания объектов с разными начальными значениями.

class MyClass {

public:

// Конструктор по умолчанию

MyClass() { value = 0; }

// Конструктор с параметром

MyClass(int v) : value(v) {}

private:

int value;

};

## 1.6 Создание объекта в памяти

Объекты можно создавать как в стеке, так и в куче. При создании в стеке память освобождается автоматически, когда объект выходит из области видимости. Для создания объектов в куче используется оператор ***new***, а для удаления — ***delete***.

// Объект в стеке

MyClass obj1;

// Объект в куче

MyClass\* obj2 = new MyClass();

delete obj2; // ручное освобождение памяти

## 1.7 Стандартные потоки ввода-вывода ЯП С++

C++ предоставляет стандартные потоки для ввода и вывода:

* cin — стандартный поток ввода из консоли;
* cout — стандартный поток вывода в консоль;
* cerr — поток для вывода ошибок, используется для вывода сообщений об ошибках.

Для работы с потоками используется библиотека ***<iostream>.***

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Введите число: ";

cin >> number;

cout << "Вы ввели: " << number << endl;

return 0;

}

## 1.7 Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) — это парадигма программирования, основанная на концепции "объектов", которые могут содержать как данные, так и код: данные в виде полей (атрибутов), а код в виде процедур (методов). ООП позволяет моделировать реальные объекты и их взаимодействия, что делает разработку программ более интуитивной и структурированной.

ООП необходимо для упрощения разработки и сопровождения программного обеспечения. Основные преимущества ООП включают:

* Модульность: Код разбивается на независимые модули (классы), что упрощает его понимание и тестирование.
* Повторное использование кода: Классы и методы могут быть повторно использованы в различных частях программы или в других проектах.
* Упрощение сопровождения: Изменения в одном классе минимально влияют на другие части программы, что облегчает поддержку и обновление кода.
* Улучшение читаемости: Код становится более понятным благодаря использованию объектов, которые отражают реальные сущности.

## 1.8 Принципы ООП

Полиморфизм — это способность объектов разных классов обрабатывать данные через один и тот же интерфейс. Это означает, что один и тот же метод может иметь разные реализации в разных классах. Полиморфизм позволяет использовать один и тот же код для работы с объектами различных типов, что делает код более гибким и расширяемым. Существует два основных типа полиморфизма:

* Полиморфизм времени компиляции (или статический) — достигается через перегрузку методов и операторов.
* Полиморфизм времени выполнения (или динамический) — достигается через переопределение методов в наследуемых классах.

Инкапсуляция — это принцип ООП, который подразумевает скрытие внутреннего состояния объекта и предоставление доступа к нему только через определенные методы. Это позволяет защитить данные от некорректного использования и изменения. Инкапсуляция способствует созданию "черного ящика", где внутренние детали реализации скрыты, а взаимодействие с объектом происходит только через его публичные интерфейсы.

Наследование — это механизм, который позволяет создавать новые классы на основе существующих. Новый класс (наследник) наследует свойства и методы родительского класса (базового класса), что позволяет повторно использовать код и расширять функциональность. Наследование способствует созданию иерархий классов и упрощает организацию кода, позволяя создавать более специализированные классы на основе общих.

# 2 Описание хода выполнения работы

Вариант для выполнения предполагает написать программу, где главное действующее лицо – автомобиль. Основные его характеристики – марка, номер (год выпуска) и стоимость. Тип данных для марки - char\*, предназначенный для хранения букв и цифр. Тип данных для номера – int, целочисленный тип с шириной не менее 16 бит и диапазоном чисел от -2 147 483 648 до 2 147 483 647. Для стоимости предназначен тип float, c минимальной шириной в 32 бита и в 6 значащих цифр.

Отдельно стоит отметить, что атрибуты имеют закрытый уровень доступа по условию.

Первая итерация, где код отражает задание без усложнений и дополнений представлена в листинге 1. Файл main.h содержит описание интерфейсов консольного приложения. Листинг 2 содержит файл main.cpp, который реализует логику работы программы: инициализацию классов, вызов методов ввода информации и вывода информации.

Приложение 1 включает файлы для базовой реализации задания: объявить класс с характеристиками и использовать его на практике в виде программы. Описанное является определением одного из пунктов объектно-ориентированного программирования – инкапсуляции. То есть ситуация, когда вся информация, нужная для работы конкретного объекта, хранится внутри этого объекта (свойства бренд, номер и цена, и методы ввода и вывода информации).

Займемся вторым принципом ООП – наследованием. То есть создание нового класса происходит на основе существующего класса и все атрибуты и методы последнего переходят к наследнику. Наследник может использовать их все, отбросить часть или дописать свое.

Поскольку основная тема лабораторной работы – автомобили, было решено усложнить проект в сторону вариативности классов машин. Под классом в контексте машины понимается категория, в которую можно объединить несколько автомобилей в зависимости от их размеров, характеристик, и предназначения. Они помогают покупателям выбрать машину, которая наилучшим образом соответствует их потребностям, а также стандартизируют подход производителей к выпуску различных моделей.

Согласно европейской классификации, есть девять классов. В работе будет использован класс A – мини-автомобили, который наследует класс Car. В родительский класс так же внесем изменение в метод Print – сделаем его виртуальным. В класс для машин A добавим еще атрибут mainInfo. Его значением будет краткая информация о классе авто.

Из-за желания использовать кириллицу при вводе данных было заменены некоторые типы и методы. Например, вместо char используется wchar\_t, а вместо cout используется далее wcout. Это помогло избежать ошибки при вводе и выводе значений.

В файле с реализацией (main.cpp) снова опишем логику работы консольной программы. При запуске программы так же будет выводиться информация о двух машинах, значение бренда для которых будет браться из конструкторов (по умолчанию и с параметрами). После этого выводится на экран сообщение с предложением выбрать дальнейшее действие и варианты этих действий. Нужно ввести номер выбранного пункта:

1. Статическое выделение памяти с параметрами по умолчанию
2. Статическое выделение памяти с переданными параметрами и наследованием
3. Динамическое выделение памяти, используется конструктор по умолчанию
4. Динамическое выделение памяти с переданными параметрами

Если пользователь выбрал первый пункт, то произойдет вывод информации о машине, через статическое выделение памяти с параметрами по умолчанию, то есть пользователь не вводит никаких данных. Тоже самое поведение осуществится при выборе второго пункта, но в этом случае информация расширится описанием машины. При выборе третьего пункта пользователю предлагается вручную ввести информацию и бренде машины, ее номере и стоимости. После ввода последней характеристики на экран выводит введенная информация. Для последнего способа память выделяется динамически. Параметры передаются при создании инстанса.

При работе с вводимыми данными добавлена обработка. Так для ввода цены нельзя передать буквенные значения. Программа попросит ввести корректное значение. При вводе марки машины нельзя ввести значение, состоящее только из цифр, но можно ввести смешанное (буквы + цифры) или только буквенное.

# Заключение

В лабораторной работе продемонстрированы базовые принципы объектно-ориентированного программирования (ООП) на примере создания и использования классов в языке С++.

В процессе работы были изучены и применены такие ключевые аспекты ООП, как инкапсуляция, наследование и полиморфизм. В лабораторной были созданы базовый класс Car и его наследник ACar, что позволило наглядно реализовать принцип наследования. С помощью методов Input и Print был достигнут контроль над вводом и выводом информации об объектах, а применение виртуального метода Print позволило продемонстрировать полиморфизм, благодаря которому каждый класс мог иметь уникальный вывод информации.

Реализация различных типов автомобилей с разными характеристиками помогает понять, как объектно-ориентированный подход облегчает создание сложных систем с возможностью добавления новых функций и модификаций без изменения основного кода.

# Список литературы

1. Страуструп, Б.  *C++* *Programming Language (или Язык программирования C++).* — 4-е изд. — Бином, 2022.
2. Дейтел, П.Дж. Дейтел. *Как программировать на С++: Пятое издание*. Пер. с англ. - М.: ООО «Би- ном-Пресс», 2008 г.
3. Липпман, С., Лажой, Ж., Муллиган, Б. Э. *Язык программирования C++. Базовый курс.* — 5-е изд. — Диалектика-Вильямс, 2018.
4. Кнут, Д. Р. *Искусство программирования*. — Т. 1. Алгоритмы. — 3-е изд. — М.: Вильямс, 2018.
5. Лоспинозо Дж. *C++ для профи*. — М.: Питер, 2021.

# Приложение 1

## Листинг 1 Файл main.h с описанием интерфейсов

// подключаем возможность работы со строками

#include <cstring>

// подключаем возможность работы с вводом и выводом

#include <iostream>

// объявляем класс

class Car

{ // объявляем приватные члены класса

private:

// указатель на строку для хранения бренда автомобиля

char\* brand\_;

// целочисленная переменная для хранения номера автомобиля

int number\_;

// переменная с плавающей точкой для хранения цены автомобиля

float price\_;

// объявляем члены класса, доступные извне

public:

// конструктор класса по умолчанию

// (нужно при создании объекта класса без параметров)

Car();

// объявление конструктора класса с параметрами (указаны в скобках)

// при создании объекта класса в месте использования

// нужно передвать три аргумента для создания нового инстанса

Car(const char\* brand\_, int number\_, float price\_);

// объявление метода печати

void Print();

// объявление метода ввода

void Input();

};

## Листинг 2 Файл main.cpp с реализацией программы

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "main.h"

using namespace std;

// Конструктор по умолчанию

Car::Car() : number\_(0), price\_(0)

{

// Выделяем память для строки brand\_ и копируем в нее строку "unknown"

brand\_ = new char[8];

strcpy(brand\_, "unknown");

}

// Конструктор с параметрами

Car::Car(const char\* brand\_, int number\_, float price\_)

: brand\_(new char[15]), number\_(number\_), price\_(price\_)

{

// Копируем строку "unknownbrand" в строку brand\_

strcpy(this->brand\_, "unknownbrand");

}

// Метод вывода информации о машине

void Car::Print()

{

cout << "Машина:" << endl;

cout << "Марка: " << brand\_ << endl;

cout << "Номер: " << number\_ << endl;

cout << "Цена: " << price\_ << endl;

}

// Метод ввода информации о машине

void Car::Input()

{

// Выделяем память для строки brand\_ и читаем строку из консоли

brand\_ = new char[100];

cout << "Введите марку машины: ";

cin >> brand\_;

cout << "Введите номер машины: ";

cin >> number\_;

cout << "Введите цену машины: ";

cin >> price\_;

}

int main()

{

// Создаем объект машины с использованием конструктора с параметрами

Car car1("Рено", 1994, 100.0);

cout << "Информация о машине car1:" << endl;

car1.Print();

// Создаем динамический объект машины

Car\* car2 = new Car("Рено", 2000, 1000.0);

cout << "Информация о машине car2:" << endl;

car2->Print();

delete car2;

// Создаем объект машины с использованием конструктора по умолчанию

Car car3;

// Выводим сообщение, приглашающее ввести информацию о машине

cout << "\nВведите информацию о машине car3:" << endl;

// Вызов метода для ввода инофрмации

car3.Input();

// Выводим сообщение с информацией о машине

cout << "\nИнформация о машине car3:" << endl;

// Вызов метода для вывода информации

car3.Print();

return 0;

}

# Приложение 2

## Листинг 3 Файл main.h с описанием интерфейсов при использовании ООП

#pragma once

#pragma warning(disable : 4996)

#include <cstring>

#include <iostream>

#include <string>

class Car {

private:

std::wstring brand\_; // Используем std::wstring

int number\_;

float price\_;

public:

Car(); // Конструктор по умолчанию

Car(const wchar\_t\* brand, int number, float price); // Конструктор с параметрами

virtual ~Car(); // Деструктор

virtual void Print() const; // Метод вывода информации

void Input(); // Метод ввода информации

// Методы доступа (геттеры и сеттеры)

const wchar\_t\* GetBrand() const { return brand\_.c\_str(); }

int GetNumber() const { return number\_; }

float GetPrice() const { return price\_; }

void SetBrand(const wchar\_t\* brand);

void SetNumber(int number);

void SetPrice(float price);

};

class AdditionalCar : public Car {

private:

const wchar\_t\* mainInfo\_;

public:

AdditionalCar(); // Конструктор по умолчанию

AdditionalCar(const wchar\_t\* brand, const wchar\_t\* mainInfo, int number, float price);

virtual ~AdditionalCar(); // Деструктор

void Print() const override; // Переопределяем метод Print

};

## Листинг 4 Файл main.cpp с реализацией программы при использовании ООП

#define NOMINMAX

#include "main.h"

#include <Windows.h>

#include <cctype> // Для функции isalpha

#include <cstring>

#include <iostream>

#include <limits> // Для numeric\_limits

#include <stdlib.h>

#include <io.h>

#include <fcntl.h>

using namespace std;

// Конструктор по умолчанию для Car

Car::Car() : number\_(0), price\_(0), brand\_(L"unknown") {}

// Конструктор с параметрами для Car

Car::Car(const wchar\_t\* brand, int number, float price)

: number\_(number), price\_(price), brand\_(brand) {

}

// Деструктор для Car

Car::~Car() {}

// Метод вывода информации о машине

void Car::Print() const {

wcout << L"Машина:" << endl;

wcout << L"Марка: " << (brand\_.empty() ? L"неизвестно" : brand\_.c\_str()) << endl;

wcout << L"Номер: " << number\_ << endl;

wcout << L"Цена: " << price\_ << endl;

}

void Car::Input() {

// Ввод марки машины с валидацией

while (true) {

wcout << L"Введите марку машины: ";

wstring brand;

wcin >> brand;

bool hasLetter = false;

bool hasOnlyDigits = true;

for (wchar\_t ch : brand) {

if (iswalpha(ch)) {

hasLetter = true;

}

if (!iswdigit(ch)) {

hasOnlyDigits = false;

}

}

// Условие для проверки валидности:

// Проверяем, есть ли хотя бы одна буква и не только цифры

if (hasLetter && !hasOnlyDigits) {

brand\_ = brand; // Запоминаем марку

break; // Ввод корректен, выходим из цикла

}

else {

wcout << brand << endl;

wcout << L"Ошибка: марка должна содержать хотя бы одну букву и не может состоять только из цифр." << endl;

}

}

// Ввод номера машины с валидацией

while (true) {

wcout << L"Введите номер машины: ";

wcin >> number\_;

if (wcin.fail() || number\_ < 0) {

wcin.clear(); // Очищаем флаг ошибки

wcin.ignore(numeric\_limits<streamsize>::max(), L'\n'); // Игнорируем оставшиеся символы в буфере

wcout << L"Ошибка: введите корректный номер машины." << endl;

}

else {

break; // Ввод корректен, выходим из цикла

}

}

// Ввод цены машины с валидацией

while (true) {

wcout << L"Введите цену машины: ";

wcin >> price\_;

if (wcin.fail() || price\_ < 0) {

wcin.clear(); // Очищаем флаг ошибки

wcin.ignore(numeric\_limits<streamsize>::max(), L'\n'); // Игнорируем оставшиеся символы в буфере

wcout << L"Ошибка: введите корректную цену машины." << endl;

}

else {

break; // Ввод корректен, выходим из цикла

}

}

}

// Реализация метода записи марки SetBrand

void Car::SetBrand(const wchar\_t\* brand) {

brand\_ = brand;

}

// Реализация метода записи года выпуска SetPrice

void Car::SetNumber(int number) {

number\_ = number;

}

// Реализация метода записи стоимости SetPrice

void Car::SetPrice(float price) {

price\_ = price;

}

// Конструктор по умолчанию для AdditionalCar

AdditionalCar::AdditionalCar()

: Car(L"unknown", 0, 0), mainInfo\_(L"description") {

}

// Конструктор для класса AdditionalCar

AdditionalCar::AdditionalCar(const wchar\_t\* brand, const wchar\_t\* mainInfo, int number, float price)

: Car(brand, number, price), mainInfo\_(mainInfo) {}

// Деструктор для AdditionalCar

AdditionalCar::~AdditionalCar() {}

// Метод вывода информации о машине

void AdditionalCar::Print() const {

Car::Print(); // Вызов метода Print базового класса

cout << "Краткое описание о машине:" << endl;

wcout << mainInfo\_ << endl;

}

int main() {

#ifdef \_WIN32

system("chcp 1251 > nul");

#endif

setlocale(LC\_ALL, "Russian");

SetConsoleOutputCP(866);

int numberChoice;

// Статическое выделение памяти с использованием конструктора по умолчанию

Car car1; // Использует конструктор по умолчанию

wcout << L"Информация о машине car1 (по умолчанию):" << endl;

car1.Print();

// Используем сеттеры для изменения значений

car1.SetBrand(L"Тойота");

car1.SetNumber(2021);

car1.SetPrice(15000.0);

wcout << L"\nОб новленная информация о машине car1:" << endl;

car1.Print();

// Создаем динамический объект машины

Car\* car2 = new Car(L"Рено", 2000, 1000.0);

wcout << L"Информация о машине car2:" << endl;

car2->Print();

delete car2;

wcout << L"Выберите, что вы хотите сделать (введите номер):" << endl;

wcout << L"1. Статическое выделение памяти с параметрами по умолчанию" << endl;

wcout << L"2. Статическое выделение памяти с переданными параметрами и наследованием" << endl;

wcout << L"3. Динамическое выделение памяти с параметрами по умолчанию" << endl;

wcout << L"4. Динамическое выделение памяти с переданными параметрами" << endl;

cin >> numberChoice;

if (numberChoice == 1) {

// Статическое выделение памяти с параметрами по умолчанию

Car car1; // Использует конструктор по умолчанию

wcout << L"Информация о машине car1 (по умолчанию):" << endl;

car1.Print();

}

else if (numberChoice == 2) {

// Статическое выделение памяти с переданными параметрами

// Использование наследования

AdditionalCar car2(L"MiniCar",

L"MiniCar A1 — это компактный и экономичный мини-автомобиль, идеально подходящий для городских условий. С его небольшими размерами и маневренностью, A1 легко справляется с узкими улицами и плотным трафиком. Оснащен современным 1.2-литровым двигателем, он обеспечивает отличную экономию топлива, что делает его идеальным выбором для ежедневных поездок.",

2022,

15000.0

); // Использует конструктор с параметрами

wcout << L"Информация о машине car2 (с параметрами и наследованием):" << endl;

car2.Print();

}

else if (numberChoice == 3) {

// Динамическое выделение памяти с использованием конструктора по умолчанию

Car\* car3 = new Car(); // Создаем объект с использованием конструктора по умолчанию

wcout << L"\nВведите информацию о машине car3:" << endl;

car3->Input(); // Ввод информации о машине

wcout << L"\nИнформация о машине car3:" << endl;

car3->Print(); // Вывод информации о машине

delete car3; // Освобождаем память

}

else if (numberChoice == 4) {

// Динамическое выделение памяти с переданными параметрами

Car\* car4 = new Car(L"Форд", 2020, 20000.0); // Использует конструктор с параметрами

wcout << L"Информация о машине car4 (с параметрами):" << endl;

car4->Print();

delete car4; // Освобождаем память

}

else {

wcout << L"Введено неизвестное число" << endl;

}

// Добавлено ожидание ввода перед закрытием

wcout << L"Нажмите Enter, чтобы выйти...";

cin.ignore(); // Игнорируем символ новой строки, если он остался в буфере

cin.get(); // Ждем ввода пользователя

return 0;

}