Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное 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](#_Toc181550505)

[Теоретическая часть 4](#_Toc181550506)

[1. Классы C++ 4](#_Toc181550507)

[2. Уровни доступа в классах 4](#_Toc181550508)

[3. Определение методов 4](#_Toc181550509)

[4. Инициализация данных объекта 5](#_Toc181550510)

[5. Конструкторы 5](#_Toc181550511)

[6. Создание объекта в памяти 6](#_Toc181550512)

[7. Стандартные потоки ввода-вывода ЯП С++ 6](#_Toc181550513)

[Код и комментарии 7](#_Toc181550514)

[Заключение 15](#_Toc181550515)

# Введение

Лабораторная работа выполнена в редакторе коде Visual Studio. Целью работы является программа на языке С++, написанная в парадигме ООП и реализующая вывод информации об автомобилях.

Для написания программы потребовалось изучить:

* классы C++;
* уровни доступа в них;
* определение методов;
* инициализация данных объекта;
* конструкторы;
* создание объекта в памяти;
* стандартные потоки ввода-вывода ЯП С++.

# Теоретическая часть

## 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.

class MyClass {

// Данные и методы класса

};

## 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.

class MyClass {

public:

int publicVar; // доступен всем

protected:

int protectedVar; // доступен только наследникам

private:

int privateVar; // доступен только внутри класса

};

## 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:

void setValue(int v); // объявление метода

private:

int value;

};

// Определение метода вне класса

void MyClass::setValue(int v) {

value = v;

}

## 4. Инициализация данных объекта

Данные объекта можно инициализировать несколькими способами:

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

class MyClass {

public:

MyClass(int v) : value(v) {} // инициализация через список инициализации

private:

int value;

};

## 5. Конструкторы

Конструкторы — это специальные 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;

};

## 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

## 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;

}

# Код и комментарии

Файл main.h содержит описание интерфейсов консольного приложения. 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.

main.h

// 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();

};

Файл main.cpp содержит логику работы программы: инициализацию классов, вызов методов ввода информации и вывода информации.

main.cpp

#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;

}

Выше представлены 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 втором 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 усложнять 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:

* A — мини-автомобили;
* B — маленькие автомобили;
* C — средне размерные автомобили;
* D — полноразмерные автомобили;
* E — автомобили бизнес-класса;
* F — представительские автомобили;
* S — спортивные купе;
* M — минивэны и коммерческие автомобили;
* J — кроссоверы и внедорожники.

В работе будут использованы классы A и C .Для каждого будет объявлен класс, который наследует общий 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.

main.h

#pragma once

#pragma warning(disable : 4996)

// 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 с возможностью перезаписать его

virtual void Print();

// объявление метода ввода

void Input();

virtual ~Car(); // Деструктор

};

// объявление класса ACar, наследующего от класса Car

class ACar : public Car

{

private:

std::string mainInfo\_;

public:

ACar(const char\* brand\_, const std::string& mainInfo\_, int number\_, float price\_);

void Print() override; // переопределяем метод Print

};

// объявление класса CCar, наследующего от класса Car

class CCar : public Car {

public:

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

void Print() override; // переопределяем метод Print

};

Перейдем к файлу с реализацией (main.cpp) и 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.

Варианты для выбора:

1. Посмотреть машину класса A
2. Посмотреть машину класса C
3. Ввести и посмотреть информацию о машине без класса

Если пользователь выбрал первый пункт, то произойдет вывод информации о машине класса А, которая вносится в код. Тоже самое поведение осуществится при выборе второго пункта. При выборе последнего, третьего, пункта, пользователю предлагается вручную ввести информацию и бренде машины, ее номере и стоимости. После ввода последней характеристики на экран выводит введенная информация и программа завершает свое выполнение.

main.cpp

#include <iostream>

#include <string>

#include <cstring>

#include "main.h"

using namespace std;

// Конструктор по умолчанию

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

brand\_ = new char[8]; // Выделяем память для строки

strcpy(brand\_, "unknown"); // Копируем строку по умолчанию

}

// Конструктор с параметрами

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

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

brand\_ = new char[strlen(brand) + 1]; // Выделяем память под строку

strcpy(brand\_, brand); // Копируем переданную строку

}

Car::~Car() {

delete[] brand\_; // Освобождаем память

}

// Метод вывода информации о машине

void Car::Print() {

cout << "Машина:" << endl;

cout << "Марка: " << (brand\_ ? brand\_ : "неизвестно") << endl; // Проверка на nullptr

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

cout << "Цена: " << price\_ << endl;

}

// Метод ввода информации о машине

void Car::Input() {

// Освобождаем старую память

if (brand\_ != nullptr) {

delete[] brand\_;

}

// Выделяем память для новой строки

brand\_ = new char[100];

cout << "Введите марку машины: ";

cin >> brand\_;

cout << "Введите номер машины: ";

cin >> number\_;

cout << "Введите цену машины: ";

cin >> price\_;

}

// Конструктор для класса ACar

ACar::ACar(const char\* brand, const string& mainInfo, int number, float price)

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

// Метод вывода информации о машине с дополнительными данными для класса A

void ACar::Print() {

cout << "Машина класса A:" << endl;

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

cout << "Краткая информация о классе машин:" << endl;

cout << mainInfo\_ << endl;

}

// Конструктор для класса CCar

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

: Car(brand, number, price) {}

// Метод вывода информации о машине с дополнительными данными для класса C

void CCar::Print() {

cout << "Машина класса C:" << endl;

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

}

int main() {

int numberChoice;

#ifdef \_WIN32

system("chcp 1251 > nul"); // Установка кодировки для Windows

#endif

// 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;

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

cout << "1. Посмотреть машину класса A" << endl;

cout << "2. Посмотреть машину класса C" << endl;

cout << "3. Ввести и посмотреть информацию о машине без класса" << endl;

cin >> numberChoice;

if (numberChoice == 1) {

// Создаем объект машины класса A

ACar\* carA = new ACar("Фиат 500",

"Это маленькие, юркие городские авто с небольшими моторами (чаще объемом от 0,8 до 1,4 л), с механическими, роботизированными либо автоматическими коробками передач и багажниками символических размеров.",

2007, 50000.00);

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

carA->Print();

delete carA; // Освобождаем память

}

else if (numberChoice == 2) {

// Создаем объект машины класса C

CCar\* carC = new CCar("Форд Фокус", 2003, 20000.5);

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

carC->Print();

delete carC; // Освобождаем память

}

else if (numberChoice == 3) {

// Создаем объект машины с использованием конструктора по умолчанию

Car car3;

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

car3.Input(); // Ввод информации о машине

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

car3.Print(); // Вывод информации о машине

}

else {

cout << "Введено неизвестное число" << endl;

}

// Добавлено ожидание ввода перед закрытием

cout << "Нажмите Enter, чтобы выйти...";

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

cin.get(); // Ждем ввода пользователя

return 0;

}

# 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 наследники ACar и CCar, 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.