

**Контрольная работа**

**по дисциплине   
«Объектно-ориентированное программирование»**

**Вариант № 3**

**Выполнил:** *студент гр. ИО-17 Бондаренко М.А.*

**Преподаватель:***доцент каф ИТУС* Исаева Г.Н.

# Введение

**Объектно-ориентированное программирование** (в дальнейшем ООП) — парадигма программирования, в которой основными концепциями являются понятия объектов и классов. В центре ООП находится понятие объекта.

*Объект* — это сущность, экземпляр класса, которой можно посылать сообщения, и которая может на них реагировать, используя свои данные. Данные объекта скрыты от остальной программы. Сокрытие данных называется *инкапсуляцией*.

*Инкапсуляция* — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя.

*Наследование* — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом

*Полиморфизм* — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Целью выполнения контрольной работы являются:

* расширение и углубление теоретических знаний по изучаемой дисциплине;
* получение дополнительных практических навыков по тематике контрольных работ;
* развитие творческих способностей и самостоятельности в ходе проведения исследований.

# Теоретическое описание исследуемого объекта

**Постановка задачи**

*Создать абстрактный класс CVehicle. На его основе реализовать классы CPlane, ССаг и CShip. Классы должны иметь возможность задавать и получать координаты, параметры средств передвижения (цена, скорость, год выпуска). Для самолета должна быть определена высота, для самолета и корабля — количество пассажиров. Для корабля — порт приписки.*

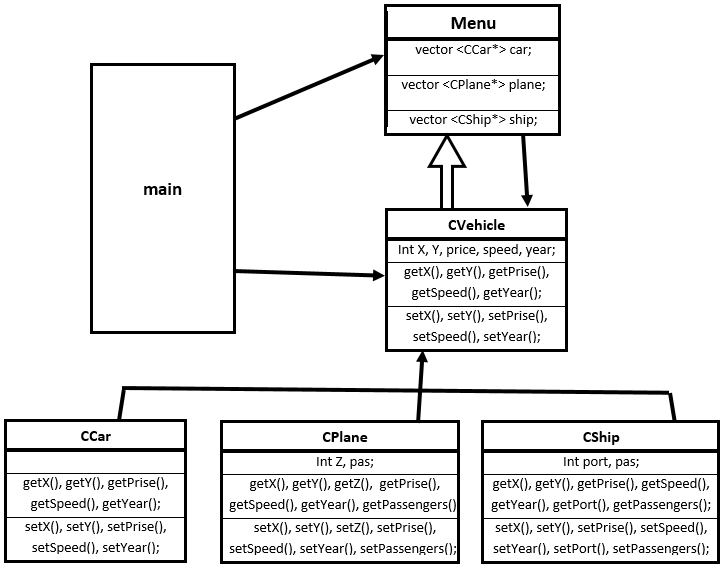
*Написать программу, создающую список объектов этих классов в динамической памяти. Программа должна содержать меню, позволяющее осуществить проверку всех методов классов.*

**Теоретическое описание**

Программа должна содержать меню, позволяющее осуществить проверку всех методов классов. Функция MainMenu() в этой программе выводит список выбираемых пунктов с номерами и после ввода пользователем номера пункта возвращает это значение главной функции.

Изменяемой сущностью в данной задаче является класс, который реализован на основе абстрактного класса CVehicle. Следовательно, требуется инкапсулировать эту изменяемую сущность так, чтобы класс Menu имел дело с некоторой абстрактной функцией.

Отсюда вывод: необходим абстрактный класс CVehicle, обеспечивающий единый унифицированный интерфейс для всех его производных классов, в данном случае — для классов CPlane, ССаг и CShip. В результате мы приходим к диаграмме классов, показанной на рис.1.



**Рис 1**. *Диаграмма классов программы.*

# Методика решения рассматриваемой задачи

|  |  |  |
| --- | --- | --- |
| **Выбор типов входных/ выходных переменных** | | |
| **CCar** | | |
| **x, y** | int | Координаты машины |
| **price** | int | Стоимость машины |
| **speed** | int | Скорость машины |
| **year** | int | Год выпуска машины |
|  | | |
| **CPlane** | | |
| **x, y, z** | int | Координаты самолёта  (Z – высота) |
| **price** | int | Стоимость самолёта |
| **speed** | int | Скорость самолёта |
| **year** | int | Год выпуска самолёта |
| **pas** | int | Количество пассажиров |
|  | | |
| **CShip** | | |
| **x, y** | int | Координаты корабля |
| **price** | int | Стоимость корабля |
| **speed** | int | Скорость корабля |
| **year** | int | Год выпуска корабля |
| **pas** | int | Количество пассажиров |
| **port** | int | Порт приписки корабля |

В абстрактном классе CVehicle конструктор отсутствует, так как объекты этого класса создаваться не будут. Класс содержит виртуальный деструктор, в связи с тем, что все методы класса — чисто виртуальные, в проекте отсутствует модуль реализации класса.

Изменение и получение переменных производится через функции Get и Set. Те же функции наследуют и остальные классы CPlane, ССаг и CShip.

Классы CPlane, ССаг и CShip наследуют параметры абстрактного класса CVehicle. Так как в абстрактном классе прописаны чисто виртуальные функции, то и другие классы унаследуют это. Их функции тоже будут чисто виртуальными, поэтому нет необходимости в каждом унаследованном классе объявлять функции чисто виртуальными.

В заголовке Menu.h описано меню программы, которое работает через switch. В этом же заголовке объявлены векторы, в которых хранятся объекты классов CPlane, ССаг и CShip. Это необходимо для того, чтобы хранить несколько объектов этих классов. Через меню пользователь может выполнять методы объектов: создание и изменение параметров. Так же пользователь может выбирать с каким объектом работать.

# Реализация решения задачи

Используя разработанный алгоритм было осуществлено тестирование программного кода (Приложение 2) в интегрированной среде разработки приложений Visual Studio.NET.

Результаты тестов представлены для различных входных данных в Приложении 1

# Заключение

В данной КР была решена задача «*Создать абстрактный класс CVehicle. На его основе реализовать классы CPlane, ССаr и CShip…*» методом ООП в интегрированной среде разработки приложений Visual Studio.NET. В ходе решения было сделано:

1. Исследована и изучена литература, указанная в списке литературы;
2. Получены дополнительные практические навыки по тематике контрольных работ;
3. Развиты творческие способности и самостоятельность в ходе проведения исследований.

# Список литературы

1. Материалы лекций и практических занятий
2. Герберт Шилдт. С++ Руководство для начинающих. Второе издание. 2005 г.
3. Герберт Шилдт. С++Базовый курс. Третье издание. 2010 г
4. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб.: Питер, 2003. —461 с: ил. ISBN 5-94723-568-4
5. Devcolibri. Образовательная IT-академия для начинающих Android-разработчиков. https://devcolibri.com

# Содержание

[Введение 2](#_Toc9782985)

[1. Теоретическое описание исследуемого объекта 2](#_Toc9782986)

[2. Методика решения рассматриваемой задачи 3](#_Toc9782987)

[3. Реализация решения задачи 4](#_Toc9782988)

[Заключение 5](#_Toc9782989)

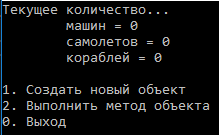
[Список литературы 5](#_Toc9782990)

[Содержание 5](#_Toc9782991)

[Приложение 1 5](#_Toc9782992)

[Приложение 2 6](#_Toc9782993)

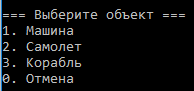
# Приложение 1



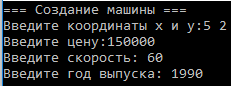
*Рис. 2. Меню программы*



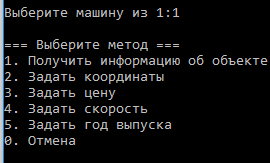
*Рис. 3. При выборе пункта 2 программа выведет сообщение о том, что объекты еще не созданы, поэтому нельзя выполнить метод объекта.*



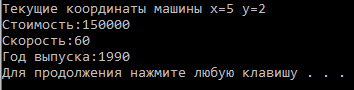
*Рис. 4. При выборе пункта 1 появляется меню выбора объекта.*



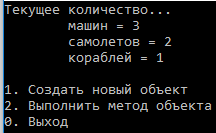
*Рис. 5. Пользователь выбирает объект, который желает создать и вводит параметры этого объекта.*



*Рис. 6. Теперь, когда имеется объект, пользователь может выполнить его методы.*



*Рис. 7. Пример одного из методов объекта.*



*Рис. 8. Программа может создавать и хранить несколько объектов разного типа.*

# Приложение 2

**CVehicle.h**

// Абстрактный класс CVehicle

class CVehicle

{

protected:

virtual ~CVehicle() {}

int X, Y; // Координаты

int price, speed, year; // Цена, скорость, год выпуска

virtual int getX() = 0;

virtual int getY() = 0;

virtual int getPrise() = 0;

virtual int getSpeed() = 0;

virtual int getYear() = 0;

// Чисто виртуальные функции:

virtual void setX(int X) = 0;

virtual void setY(int Y) = 0;

virtual void setPrise(int price) = 0;

virtual void setSpeed(int speed) = 0;

virtual void setYear(int year) = 0;

};

**CCar.h**

#include "CVehicle.h"

class CCar : public CVehicle

{

public:

CCar() {};

~CCar() {};

int getX() { return this->X; }

int getY() { return this->Y; }

int getPrise() { return this->price; }

int getSpeed() { return this->speed; }

int getYear() { return this->year; }

void setX(int X) { this->X = X; }

void setY(int Y) { this->Y = Y; }

void setPrise(int price) { this->price = price; }

void setSpeed(int speed) { this->speed = speed; }

void setYear(int year) { this->year = year; }

};

**CPlane.h**

#include "CVehicle.h"

class CPlane : public CVehicle

{

int Z, pas; // Доп. координата (высота) и пассажиры

public:

CPlane() {};

~CPlane() {};

int getX() { return this->X; }

int getY() { return this->Y; }

int getPrise() { return this->price; }

int getSpeed() { return this->speed; }

int getYear() { return this->year; }

void setX(int X) { this->X = X; }

void setY(int Y) { this->Y = Y; }

void setPrise(int price) { this->price = price; }

void setSpeed(int speed) { this->speed = speed; }

void setYear(int year) { this->year = year; }

void setZ(int Z) { this->Z = Z; };

void setPassengers(int pas) { this->pas = pas; };

int getZ() { return this->Z; };

int getPassengers() { return this->pas; };

};

**CShip.h**

#include "CVehicle.h"

class CShip : public CVehicle

{

private:

int pas, port; // Пассажиры и порт

public:

CShip() {};

~CShip() {};

int getX() { return this->X; }

int getY() { return this->Y; }

int getPrise() { return this->price; }

int getSpeed() { return this->speed; }

int getYear() { return this->year; }

void setX(int X) { this->X = X; }

void setY(int Y) { this->Y = Y; }

void setPrise(int price) { this->price = price; }

void setSpeed(int speed) { this->speed = speed; }

void setYear(int year) { this->year = year; }

void setPort(int port) { this->port = port; };

void setPassengers(int pas) { this->pas = pas; };

int getPort() { return this->port; };

int getPassengers() { return this->pas; };

};

**Menu.h**

using namespace std;

#include <vector>

#include "CVehicle.h"

#include "CCar.h"

#include "CPlane.h"

#include "CShip.h"

vector <CCar\*> car;

vector <CPlane\*> plane;

vector <CShip\*> ship;

void Menu() {

cout << "Текущее количество...\n";

cout << "\tмашин = " << car.size() << endl;

cout << "\tсамолетов = " << plane.size() << endl;

cout << "\tкораблей = " << ship.size() << "\n\n";

cout << "1. Создать новый объект\n";

cout << "2. Выполнить метод объекта\n";

cout << "0. Выход\n";

}

void Objects() {

cout << "\n=== Выберите объект ===\n";

cout << "1. Машина\n";

cout << "2. Самолет\n";

cout << "3. Корабль\n";

cout << "0. Отмена\n";

}

void Methods() {

cout << "\n=== Выберите метод ===\n";

cout << "1. Получить информацию об объекте\n";

cout << "2. Задать координаты\n";

cout << "3. Задать цену\n";

cout << "4. Задать скорость\n";

cout << "5. Задать год выпуска\n";

}

void MainMenu(bool exit = 0) {

while (exit != 1) {

int v, n, o, m, c, p, s, a, x, y, z;

Menu();

cin >> v;

system("cls");

switch (v) { // Создание нового объекта

case 1: {

Objects();

cin >> o;

system("cls");

switch (o) {

case 1: {

cout << "=== Создание машины ===" << endl;

car.push\_back(new CCar);

n = car.size() - 1;

cout << "Введите координаты x и y:";

cin >> x >> y;

car.at(n)->setX(x);

car.at(n)->setY(y);

cout << "Введите цену:";

cin >> a;

car.at(n)->setPrise(a);

cout << "Введите скорость: ";

cin >> a;

car.at(n)->setSpeed(a);

cout << "Введите год выпуска: ";

cin >> a;

car.at(n)->setYear(a);

system("cls");

}

break;

case 2: {

cout << "=== Создание самолёта ===" << endl;

plane.push\_back(new CPlane);

n = plane.size() - 1;

cout << "Введите координаты x, y и z:";

cin >> x >> y >> z;

plane.at(n)->setX(x);

plane.at(n)->setY(y);

plane.at(n)->setZ(z);

cout << "Введите цену:";

cin >> a;

plane.at(n)->setPrise(a);

cout << "Введите скорость: ";

cin >> a;

plane.at(n)->setSpeed(a);

cout << "Введите год выпуска: ";

cin >> a;

plane.at(n)->setYear(a);

cout << "Введите кол-во пассажиров: ";

cin >> a;

plane.at(n)->setPassengers(a);

system("cls");

}

break;

case 3: {

cout << "=== Создание корабля ===" << endl;

ship.push\_back(new CShip);

n = ship.size() - 1;

cout << "Введите координаты x и y:";

cin >> x >> y;

ship.at(n)->setX(x);

ship.at(n)->setY(y);

cout << "Введите цену:";

cin >> a;

ship.at(n)->setPrise(a);

cout << "Введите скорость: ";

cin >> a;

ship.at(n)->setSpeed(a);

cout << "Введите год выпуска: ";

cin >> a;

ship.at(n)->setYear(a);

cout << "Введите кол-во пассажиров: ";

cin >> a;

ship.at(n)->setPassengers(a);

cout << "Введите порт приписки: ";

cin >> a;

ship.at(n)->setPort(a);

system("cls");

}

break;

case 0: system("cls"); break; break;

default: cout << "Такой команды нет!\n"; system("pause"); system("cls");

}

break;

}

case 2: {// Выбор метода

if (car.size() + plane.size() + ship.size() == 0) {

cout << "\nНет существующих объектов!\n";

system("pause");

system("cls");

break;

}

Objects();

cin >> c;

system("cls");

switch (c) {

//машина

case 1: {

if (car.size() == 0) {

cout << "\nМашин нет!\n";

system("pause");

system("cls");

break;

}

do {

cout << "\nВыберите машину из " << car.size() << ":";

cin >> n;

if (n > car.size() || n <= 0) {

cout << "\nТакой машины нет!";

system("pause");

system("cls");

}

} while (n > car.size() || n <= 0);

n--;

Methods();

cout << "0. Отмена\n";

cin >> m;

system("cls");

switch (m) {

case 1:

cout << "Текущие координаты машины x=" << car.at(n)->getX() << " y=" << car.at(n)->getY()

<< "\nСтоимость:" << car.at(n)->getPrise() << "\nСкорость:" << car.at(n)->getSpeed()

<< "\nГод выпуска:" << car.at(n)->getYear() << endl;

system("pause");

system("cls");

break;

case 2:

cout << "Введите новые координаты x и y:";

cin >> x >> y;

car.at(n)->setX(x);

car.at(n)->setY(y);

system("cls");

break;

case 3:

cout << "Введите новую цену:";

cin >> a;

car.at(n)->setPrise(a);

system("cls");

break;

case 4:

cout << "Введите новую скорость: ";

cin >> a;

car.at(n)->setSpeed(a);

system("cls");

break;

case 5:

cout << "Введите новый год выпуска: ";

cin >> a;

car.at(n)->setYear(a);

system("cls");

break;

case 0: system("cls"); break; break;

default:

cout << "Такой команды нет!\n";

system("pause");

system("cls");

} break; }

case 2: { //самолет

if (plane.size() == 0) {

cout << "\nСамолётов нет!\n";

system("pause");

system("cls");

break; }

do {

cout << "\nВыберите cамолет из " << plane.size() << ":";

cin >> n;

if (n > plane.size() || n <= 0) {

cout << "\nТакого самолёта нет!";

system("pause");

system("cls");

}

} while (n > plane.size() || n <= 0);

n--;

Methods();

cout << "6.Задать кол-во пассажиров\n0. Отмена\n";

cin >> p;

system("cls");

switch (p) {

case 1:

cout << "Текущие координаты самолёта xyz: (" << plane.at(n)->getX() << ";" << plane.at(n)->getY()

<< ";" << plane.at(n)->getZ() << ")" << endl;

cout << "Стоимость:" << plane.at(n)->getPrise() << "\nСкорость:" << plane.at(n)->getSpeed()

<< "\nГод выпуска:" << plane.at(n)->getYear() << "\nКол-во пассажиров:" << plane.at(n)->getPassengers();

system("pause");

system("cls");

break;

case 2:

cout << "Введите новые координаты x, y и z:";

cin >> x >> y >> z;

plane.at(n)->setX(x);

plane.at(n)->setY(y);

plane.at(n)->setZ(z);

system("cls");

break;

case 3:

cout << "Введите новую цену:";

cin >> a;

plane.at(n)->setPrise(a);

system("cls");

break;

case 4:

cout << "Введите новую скорость: ";

cin >> a;

plane.at(n)->setSpeed(a);

system("cls");

break;

case 5:

cout << "Введите новый год выпуска: ";

cin >> a;

plane.at(n)->setYear(a);

system("cls");

break;

case 6:

cout << "Введите новое кол-во пассажиров: ";

cin >> a;

plane.at(n)->setPassengers(a);

system("cls");

break;

case 0: system("cls"); break; break;

default: cout << "Такой команды нет!\n"; system("pause"); system("cls");}

break;

}

case 3: { //корабль

if (ship.size() == 0) {

cout << "\nКораблей нет!\n";

system("pause");

system("cls");

break;

}

do {

cout << "\nВыберите корабль из " << ship.size() << ":";

cin >> n;

if (n > ship.size() || n <= 0) {

cout << "\nТакого корабля нет!";

system("pause");

system("cls");

}

} while (n > ship.size() || n <= 0);

n--;

Methods();

cout << "6.Задать кол-во пассажиров\n7.Задать порт приписки\n0. Отмена\n";

cin >> s;

system("cls");

switch (s) {

case 1:

cout << "Текущие координаты корабля x и y: (" << ship.at(n)->getX() << ";" << ship.at(n)->getY() << ")" << endl;

cout << "Стоимость:" << ship.at(n)->getPrise() << "\nСкорость:" << ship.at(n)->getSpeed()

<< "\nГод выпуска:" << ship.at(n)->getYear() << "\nКол-во пассажиров:" << ship.at(n)->getPassengers()

<< "\nПорт приписки:" << ship.at(n)->getPort();

system("pause");

system("cls");

break;

case 2:

cout << "Введите новые координаты x и y:";

cin >> x >> y;

ship.at(n)->setX(x);

ship.at(n)->setY(y);

system("cls");

break;

case 3:

cout << "Введите новую цену:";

cin >> a;

ship.at(n)->setPrise(a);

system("cls");

break;

case 4:

cout << "Введите новую скорость: ";

cin >> a;

ship.at(n)->setSpeed(a);

system("cls");

break;

case 5:

cout << "Введите новый год выпуска: ";

cin >> a;

ship.at(n)->setYear(a);

system("cls");

break;

case 6:

cout << "Введите новое кол-во пассажиров: ";

cin >> a;

ship.at(n)->setPassengers(a);

system("cls");

break;

case 7:

cout << "Введите новый порт приписки: ";

cin >> a;

ship.at(n)->setPort(a);

system("cls");

break;

case 0: system("cls"); break; break;

default: cout << "Такой команды нет!\n"; system("pause"); system("cls");

} break;

}

case 0: break; break;

default: cout << "Такой команды нет!\n"; system("pause"); system("cls");}break;

}

case 0: exit = 1; break;

default: cout << "Такой команды нет!\n"; system("pause"); system("cls");

}

}

}

**Main.cpp**

#include "stdafx.h"

#include <iostream>

#include "Menu.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

MainMenu();

return 0;

}