Лабораторная работа № 23

«Разработка алгоритмов и программ с использованием классов, внешних методов класса, конструкторов, свойств и деструкторов»

Один из авторов-разработчиков языка С++ Бьёрн Страуструп, говоря о разнице между языками высокого уровня Си и Си++подчеркивал, что именно использование классов в коде программы делает данную программу относящейся к языку Си++, в то время как программа в синтаксисе Си++, но без использования классов, скорее относится к бесклассовому языку Си. Многие наши программы, написанные в предыдущих лабораторных работах, по данному критерию можно отнести скорее к языку высокого уровня Си. Эти программы нужно отнести к функциональному, структурному, алгоритмическому программированию (ФП) – то есть парадигме (идеологии, философии, методологическому подходу) программирования, при котором программист придумывает алгоритм по достижении требуемого заказчику результата, создает в коде программы структуры данных (это переменные, константы, массивы (коллекции), структуры struct, enum, union), необходимые для хранения данных и манипуляций над ними для достижения требуемой заказчику цели. Обычно функционального программирования достаточно, чтобы решать небольшие задачи промышленности, бизнеса и общества. Грамотно, профессионально продуманные и реализованные программы в парадигме алгоритмического программирования обычно занимают небольшой объем оперативной памяти (до 1000 строк кода), требуют меньше вычислительных мощностей (ресурсов ПК), а потому работают даже быстрее, чем их программы-аналоги, написанные в парадигме объектно-ориентированного программирования (ООП). Но для решения больших, комплексных задач для бизнеса и социума функциональный подход может требовать очень высококлассных специалистов (математиков, алгоритмистов, программистов), много времени на анализ задачи заказчика и реализацию ее оптимального решения в коде. Для решения таких сложных комплексных задач теоретически должен лучше подойти объектно-ориентированный подход. ООП позволяет использовать большой коллектив разработчиков для работы над одним большим комплексным проектом (решением). Кода при этом нередко пишется разработчиками и генерируется «роботами» (вспомогательными программами (утилитами)) довольно много, например, 10 000 строк кода и больше не являются пределом для ООП, соответственно, требуется больше времени на компиляцию и исполнение программы, а потому требуется больше вычислительных мощностей (то есть более мощные, современные и, как следствие этого, более дорогие компьютеры и сервера), а также тестирование возможных ошибок в работе программы. Последнее потребовало выделение категории работников-тестировщиков, поскольку, чем больше кода – тем больше вероятность ошибок; чем больше компонентов – тем больше соединений между ними, а потому тем больше вероятность рассогласованности в их совместной работе (это тоже ошибки).

Как вывод, у ФП и ООП есть свои плюсы и минусы, которые надо учитывать. Самым грамотным подходом будет анализ задачи заказчика и подбор наилучшей подходящей парадигмы для ее решения. Но в наше время ООП находится в тренде, а потому им обязательно надо уметь пользоваться. Как и в случае с алгоритмизацией, ООП достаточно универсален для различных современных языков программирования, а потому освоение алгоритмизации и ООП на языке С++ означает фактическое их освоение для большинства современных объектно-ориентированных языков программирования (не только С++, но и С# (Си шарп), Java, Object C, SWIFT, Pyton (Пайтон – «Питон») и т.д.). Абсолютное большинство всех современных высокоуровневых языков программирования будут объектно-ориентированными.

Суть парадигмы ООП в том, что программисты рассматривают окружающий их мир (предметную область: промышленность, бизнес, общество, природу и т.д.) как совокупность взаимодействующих между собой объектов (людей, продавцов, покупателей, автомобилей, велосипедов, домов, квартир, смартфонов, компьютеров, кораблей, стульев, деревьев, собак, кошек, и т.д.). Мы встречаем (видим) реальные объекты (экземпляры), например, Андрея Иванова и Алесю Петрову. В человеческом языке это обычно существительные. Но можно и у Андрея Иванова и Алеси Петровой выделить некие общие черты, характеристики, особенности – например, вес, возраст, рост, пол, наличие имени и фамилии, возможность говорить, ходить, есть, спать и т.д. При этом сами характеристики имеют конкретное проявление и могут различаться в некотором диапазоне. Например, рост Андрея – 167 см, а Алеси – 159 см, но у каждого из них есть какой-то рост. Если рассмотреть многие похожие объекты, то можно выделить эти характеристики и даже указать, в каком примерно диапазоне они усреднённо варьируются, например, рост людей будет приблизительно от 15 до 250 см. Объекты, обладающие одинаковым набором характеристик можно отнести к одной группе. Андрей и Алеся могут быть отнесены к группе «люди», но рост имеют и животные, насекомые и т.д., но это другие группы, поскольку кроме физических характеристик при отнесении объекта к группе нужно учитывать и его поведение – что он может делать. Человек может разговаривать, ходить на двух ногах, готовить пищу, а вот животные и насекомые этого не умеют, поэтому у нас есть отдельные группы Люди, Животные, Насекомые. Эти группы еще можно назвать сущностями или классами. Помните, что сущность-класс – это собирательный набор характеристик и поведения, это название группы объектов со сходными характеристиками и поведением в определенном диапазоне усредненных крайних значений. Класс почти всегда является существительным. Класс – это всегда собирательный набор характеристик и поведения, это идея, это усреднённый собирательный образ всех его конкретных проявлений (объектов, экземпляров класса). Если объект (экземпляр класса) обычно можно потрогать (например, Андрея Иванова), то потрогать класс невозможно: Андрей Иванов относится к классу Люди (Человек), но потрогать собственно Человека нельзя, Человек – это идея, сущность, собирательный образ, усредненный образ всех конкретных людей.

Когда заказчик формулирует задачу, которую должна уметь выполнять программа, то он как правило указывает в своем тексте объекты и(или) усредненные сущности (классы). Разработчик должен добиться от заказчика или представителя заказчика (business owner – владелец бизнеса или доверенный представитель бизнесмена-заказчика) письменного описания задачи, обычно называемого документом Vision (вИдение заказчика) – документ, текст с описанием предметной области (бизнеса заказчика) и указанием задач, требований к заказываемому программному продукту. Документ Vision анализируется разработчиком с целью выделения объектов и сущностей (классов), которые затем будут реализованы в коде программы.

Если вникнуть в суть понятия класс, то он будет очень сильно напоминать структуру struct. Это не случайно. И структура struct и класс class как бы происходят из одного источника, а потому обладают схожими наборами свойств, характеристик, возможностей, ограничений. Но традиционно для ООП используют классы, а в рамках ФП используют структуры struct. Но по умолчанию (если в коде явно программистом не указано иное) содержимое полей экземпляров структуры struct открытое (public), то есть можно из функции main() прочитать и записать новые данные в поля любого экземпляра структуры struct. А содержимое полей и вообще внутренних членов экземпляра класса закрытое (private), то есть к ним (внутренним членам класса) нельзя обратиться извне (из функции main()) напрямую и прочитать или записать новые данные в поля, если в коде декларации класса (определения класса) явно программистом не указано иное (если программистом явно не разрешен открытый доступ к этим полям из-вне, но для последнего случая было проще объявить не класс, а структуру, поскольку классы по парадигме ООП должны выполнять 3 главных принципа ООП: инкапсуляцию (сокрытие внутреннего содержимого экземпляра класса от прямого изменения из-вне; этот принцип нарушается), наследование и полиморфизм (многоформие, многообразие поведения объекта в зависимости от типа конкретного экземпляра класса). Объект класса и экземпляр класса – это выражения-синонимы.

Класс и его члены

Класс является абстрактным типом данных, определяемым программистом, и представляет собой модель реального объекта в виде данных и функций для работы с ними.

Данные класса называются полями (по аналогии с полями структуры), а функции класса — методами. Поля и методы называются элементами класса (членами класса). Создание (объявление) класса напоминает объявление структуры:

```
class  <имя>
{
        [ private: ]
                <описание скрытых элементов>
        public:
                <описание доступных (открытых) элементов>
} ; //описание класса заканчивается точкой с запятой
```

Спецификаторы доступа **private** (частный, закрытый), **public** (общедоступный, открытый), **protected** (открытый для классов-наследников, и закрытый для всех других внешних объектов) управляют видимостью элементов класса.

Элементы, описанные после служебного слова private, видимы только внутри класса. Этот вид доступа принят в классе по умолчанию (то есть если не написать никакого спецификатора доступа, все члены класса по умолчанию получат этот спецификатор доступа, причем визуально такого слова не будет в коде описания класса видно). Рекомендуется явно в коде писать спецификатор private, чтобы визуально видеть «настройки» членов класса, на основе чего можно быстрее анализировать код, редактировать его и, главное, исправлять ошибки при их возникновении. Принцип инкапсуляции (сокрытия содержимого экземпляра класса от прямого доступа извне его, недопущения изменения внутреннего состояния объекта класса без его ведома и «согласия») требует, чтобы внутреннее содержимое объекта класса было максимально скрыто от любых других объектов какого-бы то ни было класса, что проще всего сделать, используя спецификатор private там, где это возможно.

Конкретные переменные типа «класс» называются экземплярами класса, или объектами. После объявления нового типа данных в программе – класса, можно создавать переменные его типа, указатели и ссылки на них, массивы экземпляров класса (то есть массивы из элементов, каждый из которых является объектом типа класс). Аналогично можно работать и со структурами struct, но содержимое структуры по умолчанию открытое (неявно имеет спецификатор public), а потому не поддерживает принципа инкапсуляции, поэтому в рамках ООП используют именно классы, содержимое которых по умолчанию private. Итак, структура struct, как и класс, может иметь внутри себя «свои» функции, и эти индивидуальные внутренние функции экземпляра класса называются методами. Метод – это функция, объявленная в классе, а потому относящаяся только к этому классу. Это означает, что у объектов данного класса можно вызвать эту функцию (этот метод), причем только у них. Объект класса состоит из своих внутренних членов (полей, методов, конструкторов и т.д.). При любом спецификаторе доступа внутренние члены объекта класса видят друг друга и могут напрямую обращаться друг к другу, даже если имеют спецификатор private. Метод вызывается у объекта класса, а потому имеет доступ к содержимому всех остальных членов этого экземпляра класса (например, содержимому полей этого объекта класса, причем именно этого одного объекта, что даже можно явно указать ключевым словом this (этот объект)). Стало быть, мы можем писать (объявлять) в классе методы, посредством вызова

которых можно манипулировать внутренним содержимым объекта класса (например, данными, хранимыми в полях экземпляра класса).

Каждый класс может состоять из членов. Члены класса, это:

1) **поле** – переменная, массив, структура для хранения данных определенного типа:

int number;

double weight;

2) **свойство** – метод, предназначенный для обеспечения доступа к полю класса (то есть через свойство можно получить значение конкретного поля (get) и/или присвоить конкретному полю значение (set). Получить значение конкретного поля можно почти всегда, но если поле константное или его значение вычисляется, зависит от других значений и условий, то присвоить ему значение будет нельзя (то есть свойство set в таком случае создавать программисту нельзя). Не надо создавать свойства для присвоения значений полям, которые получают конкретные смысловые значения, а не любые (в классе Калькулятор поле Результат нет смысла делать со свойством, устанавливающим ему значение, поскольку лучше, чтобы значение поля Результат устанавливал метод, вычисляющий результат из двух чисел (см. ниже)).

3) **конструктор** – особый метод, который создает экземпляры своего одноименного класса (объекты своего класса, переменные типа своего класса). Класс – это общая конструкция, схема, идея, общий шаблон, «матрица» некого явления, процесса или предмета (машина, компьютер, человек). Мы описываем класс подобно структуре struct, а после можем создавать объекты данного класса (например, конкретная машина Honda Accord III 1989 года выпуска в кузове седан с двигателем B20A I4 весом 1255 кг и регистрационным номером BY6426IP-1, принадлежащая водителю Ильину И.К. – то есть конкретный автомобиль, реально существовавший, существующий сейчас или могущий быть созданным (обнаруженным) в будущем; объект обычно можно потрогать руками, а сам класс как таковой – нет). Конструктор класса Машина создает этот объект – конкретный автомобиль Honda Accord III, экземпляр класса Машина. Конструктор пишется программистом. Конструктор имеет имя, всегда **совпадающее** с именем его класса. Конструктор – это метод, поэтому пишется со скобками**()**, может принимать ни одного или несколько входных параметров и никогда **ничего не возвращает** (не имеет return'а). Конструктором можно создавать переменные типа своего класса (объекты), а также указатели и ссылки типа своего класса. Переменные типа некого одного класса можно объединять в массивы. То есть класс – это пользовательский (программистский) тип данных, которым можно пользоваться как базовым типом данных, но создаваемая переменная будет иметь свою внутреннюю структуру из своих полей и у нее можно вызвать ее личные методы.

4) **метод** – член класса, представляющий собой функцию, которая находится в данном классе (принадлежит ему и его объектам, может быть вызвана только у экземпляров данного класса). Метод, подобно функции, может принимать ни одного или несколько входных параметров и одно значение возвращать return'ом или ни одного (void). Метод видит содержимое поля своего экземпляра класса (даже закрытые от других экземпляров классов – private, поскольку метод сам находится в этом же (своем) экземпляре класса) и **может изменять** значения **не**константных полей своего класса.

Методы обычно оформляются как внешние: в классе описывается **прототип метода**, а **за** классом с указанием пространства имен данного класса (ИмяКласса**::**) дается полное определение метода (тело метода).

5) **деструктор** – особый метод класса, который уничтожает объекты своего класса, когда они уже, по мнению программиста, не нужны. Деструктор очищает память от неиспользуемых объектов своего класса, аналогично коду delete. Если вызов конструктора «рождает» объект класса, то вызов для него деструктора – уничтожает этот объект, то есть означает смерть этого объекта.

При создании каждого объекта класса конструктором (иных способов нет, при создании экземпляра класса конструктор вызывается либо неявно, либо явно в коде) выделяется память, достаточная для хранения всех полей объекта класса. Обычно выгодно использовать не конструктор без параметров, который только выделить память под новый создаваемый объект класса, а написать свой конструктор с параметрами, который принимаемые входные значения поместит в соответствующие по типам поля создаваемого объекта. Сколько полей – столько и входных значений для конструктора нужно указать. Если конструктор задаст значения для всех полей создаваемого объекта класса, то можно сразу распечатать содержимое объекта и посмотреть, находятся ли в нем требуемые значения. Для последней цели логично писать в каждом классе метод, распечатывающий содержимое всех полей экземпляра класса на консоль.

Доступ к элементам объекта аналогичен доступу к полям структуры struct. Для этого используются операция “.” (оператор доступа «точка») при обращении к элементу через имя объекта (прямая адресация) и операция -> при обращении через указатель (косвенная адресация), например:

double rez = k.getR();//свойство у объекта k возвращает значение и мы это значение сохраняем в переменной соответствующего типа

mas[4].getR();//обращаемся к свойству у объекта №4 в массиве из объектов

ptrCar->setNumber(123456);// ptrCar – указатель, а потому к его полю получаем доступ, используя ->

(*ptrCar).setNumber(123456);//если указатель разыменовать, то обращаемся к его полю через операцию «.»

(&mas[4])->getR();//mas[4] – пятый элемент массива, фактически, отдельная переменная, но если у нее взять адрес оператором амперсанд &, то потребуется использовать косвенную адресацию ->, а не «точку».

Обратиться таким образом можно только к элементам со спецификатором public. Получить или изменить значения элементов (полей) со спецификатором private можно только через обращение к соответствующим методам (они называются свойствами) со спецификатором public. Поскольку конструктор вызывается извне объекта класса (этот объект еще просто-напросто не создан), то конструктор всегда должен быть public. Все свойства также логично создавать со спецификатором public, ведь ими будут пользоваться извне экземпляра класса, чтобы узнать значения полей класса или предложить установить новые значения для полей (а свойство может эти значения проверять на корректность, релевантность и либо присваивать полям, либо отклонять запрос на присвоение неподходящих значений внутренним полям объекта). Хотя внутренние члены объекта класса имеют прямой доступ друг к другу независимо от спецификатора, все же логично даже им пользоваться свойствами, чтобы устанавливать значения полям объекта класса, ведь свойства могут содержать проверки на корректность присваиваемых значений, а это увеличивает безопасность кода, корректность работы программы.

Задание 1.

Наберите код и поработайте с классами и объектами. Прочитайте комментарии к коду и ответьте на вопросы, которые там встречаются:

```cpp
#include <iostream>//Класс по умолчанию содержит шесть специальных функций (то есть они в нем уже есть даже если вы их не напишете; с другой стороны, их все или некоторые из них по
#include <Windows.h>//вашему усмотрению можно написать явно в коде самому, и код их тел можно написать в соответствии с вашими целями; также некоторые из функций-членов класса
//можно явно в коде определить как недоступные - присвоить им явно спецификатор delete). Эти 6 членов класса: 1) конструктор по умолчанию, 2) конструктор копирования,
//3) конструктор перемещения, 4) деструктор, 5) оператор присваивания копированием, 6) оператор присваивания перемещением.
//#include <functional>//подключение библиотеки для создания указателей на функции, посредством которых тоже можно вызвать функции класса (методы класса). Работает и без нее
using namespace std;

class Human//начало объявления класса Human
{//фигурные скобки, открывающие тело класса Human
private://спецификатор доступа private:, означающий, что все, описанное за ним, является ВНУТРЕННИМИ членами класса Human, скрытыми от других классов и объектов, до тех пор, пока
//не будет ниже объявлен другой спецификатор доступа к членам класса Human. Все поля: age, weight, sex, name являются СКРЫТЫМИ (закрытыми, недоступными для других объектов, но
//безусловно доступными для всех других членов данного класса. Каждый член класса видит все другие члены данного класса (этого же объекта) вне зависимости от их спецификаторов
    int age;//на самом деле private: int age;//закрытое целочисленное поле ВОЗРАСТ класса Человек. У любого объекта класса Human в поле age можно поместить целое число, например 17
    double weight;///на самом деле private: double weight;//закрытое вещественное поле ВЕС класса Человек
    bool sex;///на самом деле private: bool sex;//закрытое логическое поле ПОЛ класса Человек
    char name[20];///на самом деле private: char name[20];//закрытое поле ИМЯ класса Человек типа статического символьного одномерного массива до 20 символов
//в среде Visual Studio можно проверить спецификатор любого члена класса, если навести на него в коде курсор мыши. Например, если навести курсор мыши на поле age, то всплывет
//контекстная подсказка в виде пиктограммы "кирпич и навесной замОк", где кирпич означает ПОЛЕ, а замок - его СПЕЦИФИКАТОР private (ЗАКРЫТЫЙ, ЗАЩИЩЕННЫЙ от доступа извне). Также в
//контекстной подсказке будет написано "(поле) int Human::age", то есть что это за член класса (поле), его тип (int), его пространство имен/область видимости (Human::) и имя самого
//данного члена класса (age). Да, у нас уже не просто целочисленная перменная (int age;), а целочисленное поле в классе Человек (int Human::age;). Запомните порядок записи слов,
//она иерархическая: сначала записывается имя класса, потом символы-операторы :: и потом имя члена. Оператор :: - это оператор разрешения (указания) пространства имен (в нашем
//случае указание, в каком именно классе должен быть член, написанный справа от ::). ИмяКласса::членЭтогоКласса; Если слева не указано никакое имя/пространство имен, а далее идет
//оператор разрешения пространства имен :: и некоторое имя справа, то это означает обращение в члену, определенному в глобальном пространстве имен, например ::alpha означает
//обращение к члену alpha, который определен глобально (он видим всеми при условии, что в данной локальной области он не закрывается объектом с точно таким же именем; собственно,
//для того, чтобы явно обратиться к глобальному объекту (члену), а не его локальной одноименной "копии" (копии с точки зрения имени) и нужно написать ::имяЧлена, тем самым явно в
//коде указав, что мы обращаемся к глобальному ::alpha, а не локальному alpha в данном коде, который (alpha) теоретически так может быть назван в этой локальной области)
public://спецификатор доступа public (ПУБЛИЧНЫЙ, ОБЩЕДОСТУПНЫЙ, ОТКРЫТЫЙ для обращения извне данного класса Человек). Поскольку далее новых спецификаторов в коде тела класса
//Человек не написано, то все нижележащие члены класса являются ОТКРЫТЫМИ (мы их такими сделали в коде) вплоть до окончания описания тела класса Человек
    Human()//конструктор класса Человек. Если в классе находится функция(), с точно таким же именем, как имя самого данного класса и она НИЧЕГО НЕ возвращает (не написан даже void
    {//перед именем функции Human() и НЕТ никакого return'а), то эта функция-член класса является КОНСТРУКТОРОМ экземпляров данного класса. Данный конструктор ничего не принимает,
        cout << "Конструктор по умолчанию создал экземпляр класса Human.\n";//ведь круглые скобки () пустые - это конструктор ПО УМОЛЧАНИЮ, который самим компилятором создается
    }//внутри каждого класса, но тут я его написал явно в коде и в его тело поместил код печати строки символов на консоль, чтобы видеть, когда данный конструктор будет работать,
    //то есть когда он будет создавать новый объект класса Человек. Без вызова конструктора невозможно создать новый экземпляр класса. Это особый член любого класса, он является
    //функцией, которая вызывается и работает при создании нового объекта конкретного класса, причем только один раз для одного объекта, потом один раз для другого объекта данного
    //класса и т.д. Имя конструктора предопределено именем его класса (как именно?), ничего не может вернуть, поэтому в конструкторе запрещено писать void и return;

    //Human() = default;//это современная запись конструктора по умолчанию в коде, вместо тела ему присваивается зарезервированное слово default. Мы выше уже написали конструктор
    //по умолчанию, поэтому этот, точно такой же по имени и сигнатуре конструктор, нужно закомментировать (скрыть от компилятора). Или наоборот, этот раскомментировать, а
    //вышенаписанный конструктор по умолчанию закомментировать. Нельзя в классе создать два одинаковых члена, две одинаковых по имени и сигнатуре функции, хотя их тела разные, (а
    // конструктор - это особая функция). Конструктор по умолчанию без входных параметров должен быть в классе только один. Конструктор без параметров резервирует (занимает) место
    //в оперативной памяти для хранения объекта класса (значений всех его полей), но ведь он не принимает никакие значения, а потому не может заполнить поля объекта класса
    //оригинальными значениями (например, значениями пользователя). Технически можно присваивать полям конкретные константные или рандомные значения. Какие в этом плюсы и минусы?
    Human(int a, double w, bool s, char n[])//выгодно написать в классе полноценный конструктор с параметрами. Мы его обязаны написать явно в коде, иначе компилятор его не создаст.
    {//В нашем классе 4 поля, соответственно, пусть наш конструктор с параметрами принимает 4 входных значения типов, соответствующих типам полей нашего класса Человек
        SetAge(a);//присваиваем каждое входное значение соответсвующему полю объекта класса Человек: age = a; weight = w; sex = s; name = n; (так можно, поля класса хоть и скрытые
        SetWeight(w);//(private), но конструктор - член класса Человек, а потому имеет доступ ко ВСЕМ членам данного объекта класса Человек). Но в классе Человек написаны функции
        SetSex(s);//(их определение смотри ниже), которые выполняют роль свойств - специальных функций, предназначенных для контроля и фильтрации доступа к полям класса, чтобы в
        SetName(n);//поля не поместили значения, логически (по смыслу) не подходящие для данного поля (например age = -23; или age = 9876;). Написав такие свойства, я их использую
    }//даже в конструкторе, который по определению имеет доступ к полям напрямую, поскольку мне выгодно иметь контроль (фильтрацию) присваиваемых полям значений. В классе Human уже
    //есть конструктор Human();, но тут мы написали конструктор с другой сигнатурой Human(int, double, bool, char[]);, а потому для компилятора это 2 разных конструктора (2 разных
    //функции). Поскольку у этих двух функций одинаковые имена, но разные сигнатуры, то они называются перегруженными функциями (пергруженными конструкторами).
    ~Human()//декструктор объектов класса Human. Деструктор уничтожает конкретный объект класса Human, для которого однократно вызывается деструктор, если надо уничтожить этот
    {//объект класса Human (освободить от него оперативную память). При этом объект помечается как "удаленный (очищенный)", его уже нет в оперативной памяти, а потому вызвать для
        cout << "Деструктор класса Human уничтожает объект " << GetName() << endl;//него декструктор повторно будет ошибкой. Для конкретного объекта деструктор срабатывает единожды
    }//и это должен быть первый и последний вызов деструктора для данного объекта. Имя деструктора всегда начинается с тильды ~ и содержит точное имя класса и не имеет входных
    //параметров (круглые скобки пусты () ). Для класса можно написать только один деструктор, то есть деструкторы нельзя перегружать. Деструктор неявно создается в каждом классе
    //компилятором, но тут я его написал в коде явно, а в тело его поместил код печати фразы на консоль, чтобы видеть, когда будут срабатывать деструкторы для объектов класса Human
    int GetAge()//это функция в классе Человек, которая ничего не принимает и возвращает целочисленное значение - значение, хранимое в поле age объекта класса Человек. То есть эта
    {//функция является по сути свойством - специальной функцией для доступа к конкретному полю. Эта функция - свойство getter (get), то есть она позволяет получить (get) значение,
        return age;//хранимое в закрытом поле age.Спецификатором private я запретил прямой доступ к значению поля age, но разрешил через данное свойство GetAge(), которое находится
    }//в разделе public:. Если бы я был против какого-то ни было доступа извне к полю age, то я бы не написал данное свойство-геттер. Но в функции main() может понадобиться узнать
    //значение, хранимое в закрытом защищенном поле age, и для таких случаев я заранее в классе прописываю свойства для каждого поля. Если закрытое поле не будет иметь свойства
    //доступа к нему извне, то его знавение нельзя будет узнать другим объектом, что неудобно и должно быть серьезно аргументированно разработчиком. Поскольку поле Возраст закрытое
    //то его имя начинается с маленькой буквы age, а поскольку конструкторы, деструкторы, свойства и функции (методы класса) находятся в разделе public, то есть они открытые, то их
    //имена пишутся с большой буквы (GetAge()), как и имя класса Human, поскольку он тоже виден извне.
    int SetAge(int a)//обявим функцию SetAge(int a) в классе Человек. Эта общедоступная (public) функция нужна как свойство для присвоения значения полю ВОЗРАСТ, то есть через это
    {//открытое свойство (set, сеттер) можно присвоить значение закрытому полю age. В свойства удобно и полезно помещать проверки, которые не позволят поместить в поле логически
        if (a > -1 && a < 150)//(по смыслу) не подходящее значение. Например, возраст человека в годах может находиться в диапазоне от 0 до примерно 149 лет, а потому напишем код,
        {//который будет это условие проверять и не позволит присвоить объекту класса Человек некорректное значение типа -27 или 9999
            age = a;//если переданное в свойство значение прошло проверку на корректность успешно, то присвоим его полю Human::age
            //this->age = a;//аналогичная по результату запись через указатель на текущий ("этот") объект, то есть обратимся к текущему, этому экземпляру касса Человек (этот объект
            return age;//будет создан в будущем), обратимся у этого объекта к полю age (поскольку this - это указатель, то обращение косвенное: доступ через стрелку ->) и присвоим
        }//этому полю значение входного параметра a (а прошел проверку, и, значит, хранит "адекватное" значение от 0 до 149 включительно). Указатель this внутри класса не
        else//обязателен, но может пригодиться для уточнения имен, например, если бы входной параметр мы назвали не (int a), а (int age), то код присвоения выглядел бы как
```

```
        {// age = age;, что неоднозначно, непонятно, какой именно переменной или полю присваивается какое именно значение (из входной переменной или из поля класса), но такая
            return -1;//коллизия легко исправляется с помощью уточняющего использования указателя: this->age = age; где понятно, что справа входное значение, а слева обращение к
        }//полю экземпляра этого класса Human. Если входной параметр проверку не прошел, то полю не присваиваем некорректное значение и возвращаем -1 как сигнал о неудачной попытке
    }//присвоить некорректное значение, а если удачно, то вернем новое, присвоенное полю age, значение. Это грамотный подход для написания большинства функций (и свойств)
    double GetWeight()//свойство возврата значения из поля Human::weight
    {//обясните, почему написано weight с маленькой буквы и GetWeight с большой буквы?
        return weight;//аналогично коду: return this->weight; любой класс имеет по умолчанию внутри себя указатель на текущий экземпляр данного класса, в случае класса Human этот
    }//указатель можно явно объявить как Human* this;, но не нужно, поскольку он неявно уже создан компилятором и им можно сразу пользоваться
    double SetWeight(double w)//свойство для установки значения веса человека, то есть присвоения значения полю Human::weight
    {
        if (w > 0 && w < 1000)//проверка, чтобы присваиваемый человеку вес был в "адекватном" диапазоне от 0 до 1000 кг
        {
            weight = w;
            return w;//можно эти 2 строки кода уместить в одну с тем же результатом: return weight = w;, то есть сначала присвоить значение из w в поле weight и вернуть значение из
        }//weight, в котором уже должно быть новое присвоенное значение
        else
        {
            return -1;//иначе возвратим -1 как указание, что входное значение-кандидат на присвоение было некорректным согласно нашей проверке и мы его не присвоили полю weight, в
        }//результате чего данное поле сохранило "старое" значение (если оно там было к настоящему моменту) или так и осталось непроинициализированным (если до этого момента
    }//данному полю значений не присваивали или присваивали неудачно (они тоже не прошли проверки в данном свойстве-сеттере)
    bool GetSex()//открытое (public) свойтство-геттер для возврата значения, хранимого в закрытом (private) поле ПОЛ класса Человек
    {
        return sex;
    }
    bool SetSex(bool s)//открытое (public) свойтство-сеттер для установки значения в закрытое (private) поле ПОЛ класса Человек. В этом свойстве мы присваиваем значение из входного
    {//параметра s в поле Human::sex. Проверку на логическую корректность значения не пишем, хотя можно увязать значение пола с именем человека
        sex = s;
        return sex;//код, заменяющий эти 2 строки кода с аналогичным результатом: return this->sex = s;
    }
    char* GetName()//открытое (public) свойтство-геттер для возврата значения, хранимого в закрытом (private) поле ИМЯ класса Человек. Тип возвращаемого значения - указатель на
    {//символьный динамический одномерный массив (или одну символьную динамическую переменную, формально по записи тут неизвестно): char*
        return name;//return this->name;
    }
    char* SetName(char n[])//открытое (public) свойтство-сеттер для установки значения в закрытое (private) поле ИМЯ класса Человек. Свойство входным параметром принимает
    {//символьный статический одномерный массив
        if (strlen(n)>0)//если входной массив символов содержит хотя бы один символ
        {
            //cout << n << '_' << name << '_' << strlen(n) << '_' << strlen(name) << '_' << sizeof(n) << '_' << sizeof(name) << endl;//просмотр значений и размеров массивов
            strcpy_s(name, strlen(n)+1, n);//то помещаем его в поле ИМЯ экземпляра класса Человек, причем входной массив может не заканчиваться ноль-терминатором, поэтому указываем
            return name;//в функции strcpy_s() размер на 1 символ больший, чтобы хватило места. Возвращаем новое значение из поля ИМЯ текущего объекта класса Человек
        }
        else
        {
            return name;//если результат проверки негативный, то не присваиваем полю ИМЯ никакое значение и возвращаем то значение, которое хранится в поле ИМЯ на данный момент,
        }//хотя можно было вернуть и значение пустого указателя: return NULL;
    }
    int CelebrateBirthday()//открытая (public) функция ОтметитьДеньРождения() класса Человек, которая увеличивает текущий возраст данного объекта класса Человек на 1 год. Все
    {//функции, которые объявлены ВНУТРИ класса, а потому могут быть вызваны только у экземпляров данного класса, называеются в ООП методами. Метод - это функция, объявленная
    //внутри класса и реализующая некоторый функционал данного класса, то есть поведение экземпляров данного класса
        int a = GetAge();//int a = this->GetAge();//узнаем текущий возраст данного объекта класса Человек (сделаем это через свойство-геттер)
        SetAge(++a);//this->SetAge(++a);//установим (присвоим) этому объекту класса Человек возраст, на один год больший (сделаем это через свойство-сеттер)
        return GetAge();//вернем новый возраст данного экземпляра класса Человек, который был установлен работой данного метода
    }//конец тела метода int Human::CelebrateBirthday();
    double EatFood(double f)//открытая (public) функция СъестьЕду() - метод, который принимает в качестве своего входного праметра вещественное значение - вес съеденной еды и
    {//увеличивает вес данного экземпляра класса Человек на 75% от веса съденной еды (будем считать, что при переваривании пищи ее вес уменьшается)
        double w = GetWeight();//посредством свойства узнаем текущий вес этого объекта класса Человек
        SetWeight(w + f * 0.75);//присвоим данному человеку новый вес, прибавив к его исходному весу 75% от веса съеденной пищи, то есть 0.75*f
        return GetWeight();//вернем посредством свойства-геттера новое значение веса данного человека
    }//конец тела открытого метода double Human::EatFood(double f);
    void Show()//открытый (public) метод Показать() для печати на консоль значений всех полей данного экземпляра класса Человек. Задания могут и не требовать написания данного
    {//метода, но он обычно нужен для демонстрации результатов работы программы и очень полезен для тестирования и отладки программы, поэтому выгодно этот метод писать сразу после
        cout << GetName() << ": " << GetAge() << " лет, " << GetWeight() << " кг, ";//декларации полей и свойств класса. Выгодно писать этот метод первым методом в любом классе, а
        if (GetSex() == true)//назвать данный метод можно как Show(), Print(), View() и т.д. Метод печати значений полей класса обычно ничего не принимает и ничего не возвращает, и
        {//для данного метода быть "войдовским" (то есть ничего не возвращать, void) уместно, в то время как другие методы обычно имеют потребность принимать те значения, которые
            cout << "мужской пол.\n";//им нужны и возвращают значение переменной (массива, указателя, поля и т.д.), которое они изменили в ходе своей работы
        }
        else
        {
            cout << "женский пол.\n";
        }
    }//конец тела метода void Human::Show();
};//конец объявления (декларации) класса Human. Декларация класса (его тела) всегда дожна завершаться фигурными скобками и символом "точка с запятой" };

class Array//пример декларации класса Array с членами по умолчанию, которые ниже в коде явно прописаны программистом. В программе можно создать и использовать много классов
{
protected://ниже объявляются 2 закрытых (защищенных от доступа из других объектов) поля класса Array
    int len{ 0 }; //аналогично int len = 0; декларация и инициализация поля начальным значением, которое можно потом поменять
    double* val{ nullptr };//аналогично double* val = nullptr; или double* val = NULL; декларация поля-указателя на вещественный одномерный массив и инициализация его "пустотой"
public://ниже определены (созданы) открытые (доступные из других объектов) члены класса Array
    Array() = default;//аналогично Array(){} конструктор по-умолчанию без параметров
    Array(int _len)//определяем конструктор с параметром, который инициализирует одно поле объекта и выделяет память под второе поле-массив
    {
        len = _len;
        val = new double[_len];
        //проверка на выделение памяти в ДООП под val
    }
    Array(const Array& a) = delete;//конструктор копирования явно в коде удалён программистом (сделан недоступным для данного класса Array)
    Array(Array&& a)//конструктор перемещения с пустым телом
    { }
    ~Array()//деструктор (уничтожитель) объекта класса Array
    {
        delete[] val;//удаляем массив val из ДООП
    }
    Array& operator=(const Array& rhs);//оператор присваивания копированием
    Array& operator=(Array&& rhs);//оператор присваивания перемещением
    double& operator[](int i)//оператор [] позволяет возвращать значение с определенным индексом (номером) из массив val
    {
        return val[i];
    }
    const double& operator[](int i) const//оператор [] позволяет возвращать значение с определенным индексом (номером) из массив val без возможности изменения индекса при этом
    {
        return val[i];
    }
};//конец определения (декларации) класса Array
//выше мы продекларировали класс Human, то есть мы его объявили, но пока еще не создали НИ ОДНОГО человека (ни одного экземпляра класса Human). Создадим объекты класса Человек в
int main()//основном методе этой программы
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    Human h;//нельзя написать Human h0();, хотя неявно вызывается открытый конструктор по умолчанию, который создает объект h класса Human, резервирует (занимает, бронирует) под
```

```
//него оперативную память, но не заполняет поля объекта h конкретными значениями
h.Show();//вызовем у экземпляра h класса Человек его метод Show(), чтобы посмотреть значения полей этого экземпляра класса Человек. Поля содержат "мусорные" значения
char a[] = "Радомир";
h.SetName(a);//установим (присвоим) полю ИМЯ объекта h типа класса Человек значение из массива a. Сделаем это посредством вызова открытого свойства-сеттера, ведь само поле ИМЯ
//закрытое (защищенное, private), а потому недоступное для использования из main'а напрямую, но доступное косвенно через общедоступное (public) свойство SetName()
h.SetAge(56);//аналогично установим значение поля Возраст у объекта h посредством свойства-сеттера SetAge()
h.SetWeight(60.77);//аналогично установим значение поля Вес у объекта h посредством свойства-сеттера SetWeight()
h.SetSex(true);//аналогично установим значение поля Пол у объекта h посредством свойства-сеттера SetSex()
h.Show();//теперь вызовем метод Show() у объекта h, чтобы посмотреть, изменились ли значения в полях данного объекта. Да теперь поля содержат значения (Jack, 56, 60.77, true),
//которые мы им присвоили посредством использования свойств-сеттеров у объекта h
h.CelebrateBirthday();//вызовем метод ПраздникованияДняРождения() у объекта h. Теперь его возраст (значение, хранимое в поле age) должен увеличиться на единицу
h.Show();//проверяем
h.EatFood(0.23);//вызовем метод EatFood() у объекта h, "накормив" этого человека 230-ю граммами пищи
h.Show();//проверим, изменился ли вес объекта h. Помните, что вес увеличивается только на 75% от съеденной еды
//h.~Human();//так вызывается деструктор у объекта h, но обычно сборщик мусора Garbage Collector сам вызывает деструкторы в нужное время, когда заканчивается область видимости
//у данного объекта, в нашем случае это закрывающая фигурная скобка функции main()
Human* h0 = new Human();//создадим указатель на динамический объект - экземпляр класса Человек, создаваемый в Динамической области оперативной памяти. Явно вызовем конструктор
//без параметров Human() для резервирования места под объект h0. Поля объекта при этом значениями не заполняются
if (h0 == NULL)//если в коде написано new, то надо проверка на выделение оперативной памяти
{
    cout << "Ошибка создания экземпляра класса Human. Оперативная память под объект не выделена.\n";
    system("pause");
    return 0;
}
h0->Show();//проверим, какие значения хранятся в полях объекта h0. Там "мусорные" значения. h0 - указатель, поэтому адресация косвенная ->
char b[20];
cout << "Введите имя человека: ";
cin.getline(b, 20);
h0->SetName(b);//заполним поля объекта h0 значениями, определяемыми пользователем с клавиатуры
int c;
cout << "Введите возраст человека (полных лет): ";
cin >> c;
h0->SetAge(c);
double d;
cout << "Введите вес человека (кг.граммы): ";
cin >> d;
h0->SetWeight(d);
bool e;
cout << "Укажите пол человека (0-женский, 1-мужской): ";
cin >> e;
h0->SetSex(e);
h0->Show();//проверим, что в полях те значения, которые мы вводили с клавиатуры

Human* h1;//или Human* h1 = NULL;//создан указатель типа класса Human, но этот указатель ни на что не указывает (ни содержит в себе адреса никакого экземпляра класса Human)
//h1->Show();//нельзя, будет Ошибка C4700 "использована неинициализированная локальная переменная "h1"", ведь указатель не ссылается пока ни на один объект
h1 = new Human();//присвоим указателю адрес динамической переменной типа класса Human, которую создаем вызовом конструктора без параметров Human()
if (h1 == NULL)//если в коде написано new, то надо проверка на выделение оперативной памяти
{
    cout << "Ошибка создания экземпляра класса Human. Оперативная память под объект не выделена.\n";
    system("pause");
    return 0;
}
h1->Show();//проверяем значения полей экземпляра h1. Ошибок нет, но в полях "мусорные" значения. Заполним теперь эти поля значениями, вводимыми пользователем с клавиатуры
cout << "Введите имя человека: ";
cin.ignore();//потребовалось для корректного считывания строки с клавиатуры
cin.getline(b, 20);
h1->SetName(b);
cout << "Введите возраст человека (полных лет): ";
cin >> c;
h1->SetAge(c);//присваиваем значение полю Возраст объекта h1
cout << "Введите вес человека (кг.граммы): ";
cin >> d;
h1->SetWeight(d);
cout << "Укажите пол человека (0-женский, 1-мужской): ";
cin >> e;
h1->SetSex(e);
h1->Show();//проверим, что в полях объекта h1 те значения, которые мы вводили с клавиатуры

h0->Show();//проверим, что значения полей объекта h0 НЕ изменились. Так и есть.

char n[] = "Франческа";
Human h2(12, 34.9, false, n);//создадим экземпляр h2 типа класса Human, вызвав конструктор с четырьмя параметрами Human(int, double, bool, char[]), который мы определили в
//классе Human
h2.Show();//распечатаем значения, находящиеся в полях. Поля содержат корректные значения, которые в них поместил конструктор с параметрами сразу, когда резервировал оперативную
//память под экземпляр h2 класса Human. Всегда при создании класса определяйте в нем конструктор с параметрами, который заполнит уникальными значениями все поля объектов класса
cout << "Введите имя человека: ";
cin.ignore();
cin.getline(b, 20, '\n');
cout << "Введите возраст человека (полных лет): ";
cin >> c;
cout << "Введите вес человека (кг.граммы): ";
cin >> d;
cout << "Укажите пол человека (0-женский, 1-мужской): ";
cin >> e;
Human* h3 = new Human(c, d, e, b);//для создания динамических объектов тоже удобно вызывать конструктор с параметрами и создавать экземпляр класса и одновременно заполнять его
//поля уникальными значениями, которые, например, определяет пользователь с клавиатуры
if (h3 == NULL)//если в коде написано new, то надо проверка на выделение оперативной памяти
{
    cout << "Ошибка создания экземпляра класса Human. Оперативная память под объект не выделена.\n";
    system("pause");
    return 0;
}
h3->Show();//проверим, что в полях объекта h3 хранятся значения, которые мы вводили с клавиатуры. Так и есть.
//есть возможность создавать указатели на функции и вызывать эти функции через указатели на них. Также можно создавать указатели на методы в классах, что позволяет создавать и
//использовать события - ситуации, когда работа метода в одном классе вызывает работу методов тех классов, которые "подписаны" на получение сообщения о срабатывании этих
//методов. Это связано с механизмом делегатов. Ниже пропробуем создать указатели на методы и вызвать их посредством данных указателей
typedef void (Human::* method)();//создадим указатель на метод, который ничего не возвращает (void) и ничего не принимает в качестве входных параметров (). Назовем этот
//указатель на метод именем method и пусть он указывает на метод в классе Human. Ключевым словом typedef определим данный указатель как новый тип данных для данной программы
method m = &Human::Show;//создадим экземпляр m типа нашего указателя method. Выше мы указали явно в коде, что наш указатель может указывать только на метод в классе Human,
//причем такой метод, который никаких параметров не принимает ничего не возвращает (void). В классе Human под такое описание подпадает только метод Human::Show(), поэтому на
//него и будет указывать наш указатель на метод. Для этого присвоим нашему указателю адрес имени метода &Human::Show; Это тот редкий случай, когда имя метода пишется без
//круглых скобок (). Итак, теперь у нас есть указатель на метод Human::Show(), вызовем его у объектов класса Human
(h.*m)();//вызовем метод Human::Show() у объекта h класса Human. h - обычный нединамический объект, поэтому доступ к его методу Human::Show() прямой, "через точку"
(h0->*m)();//вызовем метод Human::Show() у объекта h0 класса Human. h - указатель на динамический объект, поэтому доступ к его методу Human::Show() косвенный, через стрелку ->

typedef int (Human::* function)(int);//создадим указатель на метод int Human::SetAge(int), который принимает целочисленное значение и возвращает целочисленное значение, назовем
//этот указатель function и укажем перед ним слева тип возвращаемого значения и за ним справа в круглых скобках тип входного значения метода int Human::SetAge(int)
function m0 = &Human::SetAge;//создадим экземпляр m0 типа нашего указателя function и присвоим ему адрес метода &Human::SetAge; КРУГЛЫЕ СКОБКИ НЕ пишем за именем метода тут
(h2.*m0)(50);//вызовем метод Human::SetAge(50) у объекта h2 класса Human. h2 - обычный нединамический объект, поэтому доступ к его методу Human::SetAge(50) прямой, через точку
//напомню, что метод Human::SetAge(50) является свойством-сеттером, которое присваивает полю Human::age новое значение (если оно пройдет проверку на корректность).Само свойство
(h2.*m)();//ничего не печатает на консоль, поэтому для проверки его работы вызовем метод Human::Show(), причем сделаем это через указатель на метод m
```

```cpp
    (h1->*m0)(33);//вызовем метод Human::SetAge(33) у объекта h1 класса Human. h1 - указатель на динамический объект, поэтому доступ к его методам косвенный через "стрелку" ->
    (h1->*m)();//чтобы проверить результат работы метода Human::SetAge(33) вызовем через указатель m метод Human::Show() у объекта h1

    typedef int (Human::* func)();//создадим указатель с именем func на метод Human::CelebrateBirthday(), который ничего не принимает и возвращает целочисленное значение.
    //Соответсвенно, слева от func пишем тип возвращаемого значения int, а справа в круглых скобках ничего не пишем (), поскольку никакие аргументы метод Human::CelebrateBirthday()
    //не принимает
    func m1 = &Human::CelebrateBirthday;//создадим экземпляр m1 типа указателя на метод func и присвоим ему адрес метода &Human::CelebrateBirthday; Круглые скобки НЕ пишем
    (h2.*m1)();//посредством указателя m1 вызовем метод Human::CelebrateBirthday() у объекта h2. Адресация прямая (почему?)
    (h2.*m)();//проверим, сработал ли метод Human::CelebrateBirthday() у объекта h2, вызванный через указатель на метод
    (h3->*m1)();//посредством указателя m1 вызовем метод Human::CelebrateBirthday() у объекта h3. Адресация косвенная (почему?)
    (h3->*m)();//проверим, сработал ли метод Human::CelebrateBirthday() у объекта h3, вызванный через указатель на метод
    system("pause");
    return 0;
}//конец тела функции main()
```

Тестируем программу, использующую ООП. Найдите на консоли места, которые показывают, когда срабатывают конструкторы, деструкторы, методы, свойства и для каких именно объектов из кода (h, h0, h1, h2, h3). Где работают методы установки возраста человека SetAge(), увеличения текущего возраста человека на 1 год методом CelebrateBirthday(), печати значений полей методом Show(), которые вызываются не напрямую (как их вызвать у объекта напрямую?), а через вызов указателей на методы? Как создавать указатели на методы?

```
Консоль отладки Microsoft Visual Studio
Конструктор по умолчанию создал экземпляр класса Human.
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
ММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММММ
—№х|Ц: -858993460 лет, -9.25596e+61 кг, женский пол.
Радомир: 56 лет, 60.77 кг, мужской пол.
Радомир: 57 лет, 60.77 кг, мужской пол.
Радомир: 57 лет, 60.9425 кг, мужской пол.
Конструктор по умолчанию создал экземпляр класса Human.
“КЭ?: -842150451 лет, -6.27744e+66 кг, женский пол.
Введите имя человека: Филипп
Введите возраст человека (полных лет): 10
Введите вес человека (кг.граммы): 39.45
Укажите пол человека (0-женский, 1-мужской): 1
Филипп: 10 лет, 39.45 кг, мужской пол.
Конструктор по умолчанию создал экземпляр класса Human.
чКЭННННННННННННННННННННэээЭЭЭЭЭЭЭЭЭЭЭЭЦ
: -842150451 лет, -6.27744e+66 кг, женский пол.
Введите имя человека: Рогнеда
Введите возраст человека (полных лет): 13
Введите вес человека (кг.граммы): 41.23
Укажите пол человека (0-женский, 1-мужской): 0
Рогнеда: 13 лет, 41.23 кг, женский пол.
Филипп: 10 лет, 39.45 кг, мужской пол.
Франческа: 12 лет, 34.9 кг, женский пол.
Введите имя человека: Добрыня
Введите возраст человека (полных лет): 15
Введите вес человека (кг.граммы): 45.99
Укажите пол человека (0-женский, 1-мужской): 1
Добрыня: 15 лет, 45.99 кг, мужской пол.
Радомир: 57 лет, 60.9425 кг, мужской пол.
Филипп: 10 лет, 39.45 кг, мужской пол.
Франческа: 50 лет, 34.9 кг, женский пол.
Рогнеда: 33 лет, 41.23 кг, женский пол.
Франческа: 51 лет, 34.9 кг, женский пол.
Добрыня: 16 лет, 45.99 кг, мужской пол.
Для продолжения нажмите любую клавишу . . .
Деструктор класса Human уничтожает объект Франческа
Деструктор класса Human уничтожает объект Радомир
```

Иллюстрация создания указателей и ссылок на «простые» объекты и «динамические» объекты. В код прежней программы добавлены еще строки кода. Допишите и протестируйте:

```
    //методов. Это связано с механизмом делегатов. Ниже пропробуем создать указатели на методы и вызвать их посредством данных указателей
    Human* pH = &h;//указатель на объект h класса Human
    pH->Show();//вызовем метод Show() у объекта h через указатель pH
    Human& rH = h;//создадим ссылку rH на объект h класса Human
    rH.Show();//вызовем метод Show() у объекта h через ссылку rH
    Human** pH3 = &h3;//создадим указатель на адрес динамического объекта h3. Поскольку h3 является указателем первого уровня*, то на него надо создать указатель второго уровня**
    (*pH3)->Show();//pH3 - указатель второго уровня, поэтому мы берем значение* объекта, на который ссылается этот указатель, а вторую * заменяет стрелка косвенной адресации ->
    (**pH3).Show();//тут мы обратились к самому объекту, применив **, поэтому метод Show() вызывается уже напрямую через оператор прямого доступа "точка"
    Human& rH3 = (*h3);//создадим ссылку на динамический объект h3. Ссылка может указывать только на сам объект, поэтому присваиваем ей значение объекта, на который указывает h3
    rH3.Show();//вызовем метод Show() у объекта, на который указывает указатель h3, но вызовем тот метод через ссылку rH3
    typedef void (Human::* method)();//создадим указатель на метод, который ничего не возвращает (void) и ничего не принимает в качестве входных параметров (). Назовем этот
```

Деструкторы можно вызывать только у объектов, к которым мы не планируем больше обращаться, поскольку деструкторы уничтожают объекты (указывают сборщику мусора Garbage Collector, что место оперативной памяти, ранее выделенной под данные объекты, можно пометить как «свободное», а значит доступное для записи сюда новых объектов с ликвидацией прежних данных, записанных в этих ячейках оперативной памяти.

```
h.~Human();//вызов деструктора у объекта h класса Human
h3->~Human();//вызов деструктора у динамического объекта h3 класса Human
```

Класс может содержать полное определение своих членов внутри себя (с их телами, смотри пример выше), но можно внутри класса только описать члены данного класса (написать их прототипы), а полное определение членов класса (это относится к методам, свойствам, конструкторам, деструктору, но НЕ полям) написать рядом с данным классом, но ВНЕ его тела. Такой подход более грамотный и позволяет не писать длинные тела классов, поскольку в классе описываются только поля и прототипы его методов (свойств, конструкторов, деструктора).

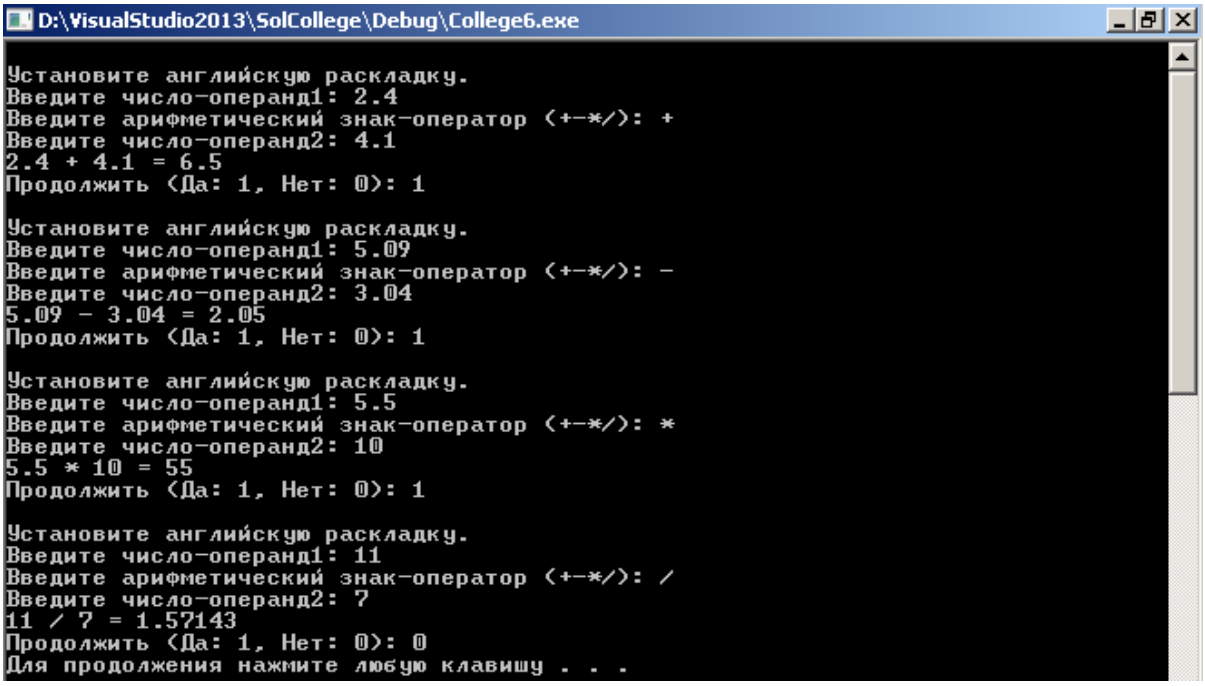
Задание 2.

Наберите код программы, в ходе чего освойте запись определений членов класса с их телами вне самого класса. Помните, что при определении (декларации и инициализации) члена класса вне этого класса нужно ОБЯЗАТЕЛЬНО **указать пространство имен** класса, которому этот член принадлежит. Например, если мы описываем в коде некий член класса с его телом, то пишем не int CelebrateBirthday(){…}, а int Human:: CelebrateBirthday(){…}.

```
#include <iostream>
using namespace std;

class Calculator//описываем класс ПЕРЕД функцией main(), в которой будем создавать экземпляры класса Calculator
{
private://спецификатор доступа делает все поля за ним видимыми только внутри этого класса. Теряет силу тогда, когда объявляется другой спецификатор за ним ниже (у нас public: )
    double a;//полностью private double a;//поле для хранения значения переменной типа double
    double b;
    char c;//полностью private char c;//поле для хранения значения переменной типа char (одного символа)
    double r;
public://спецификатор доступа делает все поля за ним видимыми из всех классов. Теряет силу тогда, когда объявляется другой спецификатор за ним ниже (у нас такого нет)
    double getA();//полностью public double getA();//прототип метода (свойства) для получения значения поля private double a; Полное определение метода написано ЗА классом
    double getB();
    char getC();
    double getR();
    void setA(double a0);//полностью public void setA(double a0);//прототип метода (свойства) для установки (присвоения значения) полю private double a;//полное определение ЗА классом
    void setB(double a0);
    //свойства для установки значения поля C и Z не создаем: Z - это поле результата вычисления, оно (и C) вычисляется (инициализируется) с помощью метода vychislenie(char x)
    Calculator(double a1, double b1);//прототип конструктора, принимающего 2 значения для инициализации двух полей экземпляра класса Calculator//полное определение написано ЗА классом
    //конструктор всегда public, его имя всегда совпадает с названием его класса и он всегда ничего не возвращает//конструктор - это особый метод, создающий экземпляр своего класса
    void vychislenie(char x);//прототип метода, принимающий символ типа char и ничего не возвращающий, поскольку все поля (даже private) СВОЕГО класса ему доступны для изменения
};//конец описания класса Calculator//ниже пишем полное определения методов (свойств, конструкторов), прототипы которых указанны в классе

Calculator::Calculator(double a1, double b1)//сначала указываем пространство имен - имя класса и знаки ::пишем полное описание конструктора, принимающего 2 переменные типа double
{
    a = a1;//значение первой переменной конструктор присваивает полю private double a;//конструктор находится в пространстве имен своего класса и ему доступны все поля своего класса
    b = b1;//значение второй переменной конструктор присваивает полю private double b;//конструктор находится в пространстве имен своего класса и ему доступны все поля своего класса
}//конец описания конструктора Calculator::Calculator(double a1, double b1)//у класса может быть один или больше конструкторов, их прототипы должны быть в классе

double Calculator::getA()//метод (свойство) для получения значения поля a//метод находится в пространстве имен класса Calculator
{
    return a;
}

double Calculator::getB()//метод (свойство) для получения значения поля b//метод находится в пространстве имен класса Calculator
{
    return b;
}

char Calculator::getC()//метод (свойство) для получения значения поля c//метод находится в пространстве имен класса Calculator
```

```
{
    return c;
}

double Calculator::getR()//метод (свойство) для получения значения поля r (результата арифметической операции)//метод находится в пространстве имен класса Calculator
{
    return r;
}

void Calculator::setA(double a0)//метод (свойство) для установки значения поля a//метод находится в пространстве имен класса Calculator
{
    a = a0;
}

void Calculator::setB(double b0)//метод (свойство) для установки значения поля b//метод находится в пространстве имен класса Calculator
{
    b = b0;
}

void Calculator::vychislenie(char x)//метод принимает символ типа char, поля класса ему доступны, поскольку он находится в пространстве имен класса Calculator
{//метод по символу определяет, какую операцию над полями a и b надо совершить и делает ее//результат вычисления метод помещает в поле r, а полученный символ - в поле c
    c = x;
    switch (x)
    {
    case'+':
    {
              r = a + b;
              break;
    }
    case'-':
    {
              r = a - b;
              break;
    }
    case'*':
    {
              r = a * b;
              break;
    }
    case'/':
    {
              r = a / b;
              break;
    }
    default://если введен неучтенный символ, сообщить пользователю
    {
              cout << "Арифметическое действие отсутсвует.\n";
              break;
    }
    }
    return;//метод изменил значения полей экземпляра своего класса, поэтому возвращать ему ничего не надо (void)
}//конец описания метода void Calculator::vychislenie(char x)

int main()//функция main() находится ВНЕ клааса Calculator, поэтому поля private класса Calculator функции main() не доступны
{
    setlocale(0, "Russian");
    double a1, b1;
    char x0;
    bool f;
    do
    {
        cout << "\nУстановите английскую раскладку.\nВведите число-операнд1: ";//ввсести число, можно вещественное и нажать клавишу Enter
        cin >> a1;
        cout << "Введите арифметический знак-оператор (+-*/): ";//ввсести символ +, -, * или / и нажать клавишу Enter
        cin >> x0;
        cout << "Введите число-операнд2: ";//ввсести число, можно вещественное и нажать клавишу Enter
        cin >> b1;
        Calculator k(a1, b1);//работает конструктор класса Calculator, который принимает 2 числа и создает экземпляр класса Calculator//k - это переменная типа класса Calculator
        k.vychislenie(x0);//у объекта (экземпляра класса) Calculator вызываем метод vychislenie, которому передаем символ типа char
        double rez = k.getR();//переменной rez присваиваем значение, возвращаемое свойством getR() у объекта k
        cout << k.getA() << ' ' << k.getC() << ' ' << k.getB() << " = " << k.getR() << endl;//печатаем значения полей объекта k, для чего используем свойства класса Calculator,
        cout << "Продолжить (Да: 1, Нет: 0): ";//которые возвращают значения private полей класса Calculator
        cin >> f;
    }
    while (f == true);//цикл бесконечно работает, пока не введут 0 (false)
    system("pause");
    return 0;
}
```

Протестируйте работу калькулятора:



Задание 3

Напишите для класса Calculator еще один метод, который для двух значений полей a и b в качестве результата своей работы присваивает переменной-результату r рандомное (случайное) значение в диапазоне от a до b. Внутри метода предусмотрите сначала проверку: если a > b, то поменять местами значения этих полей, а потом генерировать случайное число в диапазоне этих значений. Поля a и b имеют вещественный тип, а потому для генерации рандомного значения требуется явное приведение их типа к целочисленному (int)a, (int)b. Может потребоваться подключение нужных для рандома библиотек.

Задание 4

Задания по варианту:

1	1	Создать класс Дом, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий соотношение жилой площади дома к общей площади дома. Метод принимает значение жилой площади дома (метры квадратные) и значение общей площади дома (метры квадратные). Если величина жилой площади дома больше общей площади дома, то вернуть – 1 (как сообщение о неверных входных данных), иначе рассчитать по формуле: ЖилаяДоляВДоме = жилаяПлощадьДома / общаяПлощадьДома.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4.

		Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
2	1	Создать класс Завод, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий величину рентабельности завода. Метод принимает значение годового дохода (рублей), затрат на производство, зарплаты, сырье, энергию (рублей), величину налогов с прибыли (%). Рентабельность = ((годовойДоход – затратыГодовые) * X) / годовойДоход. Если разность (годовойДоход – затратыГодовые) больше нуля, то X = (1 – величинаНалоговВПроцентах / 100), иначе X = 1.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
3	1	Создать класс Государство, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий среднюю плотность населения (человек / км^2) по формуле: СредняяПлотностьНаселения = общееКоличествоНаселенияСтраны (человек) / общуюПлощадьСтраны (км^2). Если нет таких полей, то метод принимает значения общегоКоличестваНаселения и общейПлощадиСтраны. Исключите возможность деления на ноль.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения,

		ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
4	1	Создать класс Город, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий среднюю этажность домов города. Если нет таких полей, то метод принимает целочисленный массив, в котором индексы означают этажность домов, а хранимое по этому индексу значение – количество домов соответствующей этажности и метод принимает размер массива. Далее в цикле метод проходится по массиву и подсчитывает общее количество этажей всех домов в городе и количество всех домов в городе. СредняяЭтажность = общееКоличествоЭтажей / общееКоличествоЗданий. Исключить возможность деления на ноль. Этажность – величина вещественного типа.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
5	1	Создать класс Кинофильм (или киносериал), имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий доходность фильма в кинопрокате (долларов). Метод

		принимает значение затрат на киносъемку, зарплату актеров и иные расходы (долларов), кассовые сборы фильма (долларов), величину налога на прибыль (процентов). Доходность = (кассовыеСборы – затратыНаКиносъемку) * Р. Если кассовыеСборы меньше затратНаКиносъемку, то Р = 1, иначе Р = (1 – налогНаПрибыльВПроцентах / 100).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
6	1	Создать класс Самолет, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий объем потребленного за полет керосина. Метод принимает общееВремяПолета (часов), времяВзлета (часов), расходКеросинаДляДанногоТипаСамолета (литров / час). ОбъемПотребленногоКеросина = времяВзлета * расходКеросина * 1.6 + (времяПолета – времяВзлета) * расходКеросина.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
7	1	Создать класс Фирма, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что

		проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий сумму налогов, необходимых к выплате фирмой за месяц. Метод принимает объем выручки (рублей), сумму затрат на зарплаты, энергию, сырье, аренду (рублей), налог в ФСЗН (проценты), налог на прибыль (проценты), НДС (проценты). Х (рублей) = (объемВыручки – суммаЗатрат) * ((ФСЗНвПроцентах + НДСвПроцентах) / 100)). Если разность А = (объемВыручки – суммаЗатрат – Х) больше нуля, то СуммаНалогов = Х +А * (налогНаПрибыльВПроцентах / 100), иначе СуммаНалогов = Х. Вернуть СуммуНалогов (рублей).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
8	1	Создать класс Наручные часы, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий отображаемое время, которое будут показывать часы, по прошествии определенного периода времени, по сравнению с эталонным временем. Метод принимает величину погрешности (секунд в сутки, вещественное число, знак минус означает отставание), время работы часов (суток). ОтображаемоеВремя (секунд) = 60*60*24 * (времяРаботы + времяРаботы * погрешность).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.

9	1	Создать класс Растение, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий объем кислорода, продуцируемого растением за год. Метод принимает объем кислорода, производимого растением в светлое время суток (литров / час * $м^3$), объем кислорода, потребляемого растением в темное время суток (литров / час * $м^3$), площадь поверхности растения ($м^3$). Если полагать количество темного и светлого времени суток усредненно за год равным, то ОбъемКислорода = (объемКислородаВСветлоеВремя – объемКислородаВТемноеВремя) * площадьПоверхностиРастения * 365 * 12.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
10	1	Создать класс Автомастерская, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий средний доход с одного отремонтированного автомобиля. Метод принимает количество отремонтированных автомобилей за год (штук), объем выручки (рублей), объем затрат на запчасти, зарплату, энергию, аренду (рублей), величину налогов (процент). ДоходСАвтомобиля (рублей) = ((выручка – затраты) * (1 – налогиВПроцентах / 100)) / отремонтированныеЗаГодАвтомобили. Исключить возможность деления на ноль.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем

		вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
11	1	Создать класс Аэропорт, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий прогнозируемое количество самолетов в аэропорту через определенный промежуток времени. Метод принимает средний интервал между вылетами самолетов из аэропорта (минут), среднее время между прилетами самолетов в аэропорт (минут), текущее количество самолетов в аэропорту (штук), промежуток времени, на который надо сделать прогноз (минут). СамолетовБудет = текущееКоличествоСамолетов + промежутокВремени / времяПрилета – промежутокВремени / времяВылета. Если СамолетовБудет меньше нуля по расчетам, то реально СамолетовБудет ноль. Исключить возможность деления на ноль.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
12	1	Создать класс Работник, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий размер заработной платы работника за отработанный им

		месяц. Метод принимает количество отработанных работником часов, стоимость одного часа работы (рублей / час), процент премии. РазмерЗарплаты = (отработанныеЧасы * стоимостьОдногоЧаса) * (1 + премияВПроцентах / 100).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
13	1	Создать класс Корабль, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий объем потраченного на плавание топлива. Метод принимает общую длину пути (км), длину пути в сложных условиях (против течения, непопутный ветер), длину пути в благоприятных условиях (км), расход топлива ($м^3$ на километр пути). Топливо ($м^3$) = расходТоплива * (длинаПутиВСложныхУсловиях * 1.4 + длинаПутиВБлагоприятныхУсловиях * 0.7 + (общаяДлинаПути – длинаПутиВСложныхУсловиях – длинаПутиВБлагоприятныхУсловиях)).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
14	1	Создать класс Обувь, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что

		проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий стоимость 1 пары обуви. Метод принимает значение текущего года, год создания данной модели обуви, стоимость материалов (рублей), оплата труда (рублей), скидка (%). Стоимость (рублей) = (– 0.1 * (текущийГод – годСоздания) + (стоимостьМатериалов + оплатаТруда) * 2.5) * (1 + скидкаВПроцентах / 100).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
15	1	Создать класс Детская игрушка, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий вес пластика, необходимый для изготовления кубика-рубика. Метод принимает длину грани кубика (см), количество составных кубиков в одном ребре, плотность пластика (кг / см^3). ВесПластика (кг) = количествоКубиковВРебре^3 * 6 * (длинаГраниКубика / количествоСоставныхКубиков)^3 * 0.01 * плотностьПластика. Исключить возможность деления на ноль.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
16	1	Создать класс Одежда, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и

		их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий площадь ткани, необходимой для пошива одного изделия конкретного размера. Метод принимает площадь ткани, необходимой для пошива изделия 28-о размера этой же модели, размер данного изделия, необходимость дополнительного выделения ткани (да / нет). Т = (1 + (текущийРазмер – 28) * 0.1) * площадьТканиДля28Размера. Если необходимостьДополнительногоВыделенияТкани == Да, то ПлощадьТкани = Т * 1.3, иначе ПлощадьТкани = Т.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
17	1	Создать класс Бизнесмен, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий прибыль бизнесмена за год. Метод принимает годовой доход (рублей), расходы на аренду (рублей), сырье и оборудование (рублей), зарплату сотрудникам (рублей), величину страховки (%), общую ставку налогов (%). Прибыль (рублей) = (Доход – Расходы – Зарплаты) * (1 – (налогиВПроцентах + страховкаВПроцентах) / 100)).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с

		параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
18	1	Создать класс Смартфон, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий количество пикселей на дюйм для экрана смартфона (ppi). Метод принимает размер диагонали экрана смартфона (дюймов), высоту экрана (пикселей), ширину экрана (пикселей) и рассчитывает результат по формуле: КоличествоПикселейНаДюйм = (корень квадратный из (высотаЭкрана^2 + ширинаЭкрана^2)) / диагональЭкрана.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
19	1	Создать класс Животное (зверь), имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий количество потомства от данного животного за несколько поколений (включая самого данного животного). Метод принимает среднее количество потомства у одного животного, количество поколений. Метод суммирует количество потомства в цикле, возводя среднееКоличествоПотомства в степень 0, потом 1, потом 2 и так далее и прибавляя каждый промежуточный результат в сумму-накопитель. Например, если среднее количество потомства 3 детеныша, для 4-х поколений будет суммарно $3^0+3^1+3^2+3^3+3^4 = 121$ зверь.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего

		экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
20	1	Создать класс Материк (континент), имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий плотность населения материка (человек /км^2). Метод принимает площадь материка (км^2), одномерный массив, в котором каждый элемент – количество жителей одной страны на континенте, размер массива. Метод суммирует количество жителей всех стран на континенте и делит эту сумму на площадь всего континента. Исключить возможность деления на ноль.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
21	1	Создать класс Ноутбук, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса?

		Напишите метод, рассчитывающий и возвращающий название разрешения экрана ноутбука (символьный одномерный массив). Метод принимает количество пикселей по горизонтали и по вертикали экрана и проверяет, если разрешение 640x480 пикселей, то вернуть «VGA», иначе если разрешение 1024x768, то вернуть «XGA», если 1280x720, то вернуть «HD», если 1920x1080, то вернуть «FullHD», если 2048x1080, то вернуть «2К», если 2560x1440, то вернуть «QuadHD», если 3072x1620, то вернуть «3К», если 3840x2160, то вернуть «4K-UltraHD», иначе – вернуть «AnyResolution».
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
22	1	Создать класс Улица, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий среднюю этажность домов на данной улице. Метод принимает целочисленный массив, в котором индексы означают этажность домов, а хранимое по этому индексу значение – количество домов соответствующей этажности, и метод принимает размер массива. Далее в цикле метод проходится по массиву и подсчитывает общее количество этажей всех домов улицы и само количество всех домов. СредняяЭтажностьУлицы = общееКоличествоЭтажей / общееКоличествоДомов. Исключить возможность деления на ноль. Этажность – величина вещественного типа.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
23	1	Создать класс Магазин, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю

		значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий прибыль магазина за месяц. Метод принимает себестоимость реализованных товаров (рублей), расходы на аренду и закупку оборудования (рублей), зарплату продавцам (рублей), величину торговой надбавки (%), общую ставку налогов (%). Прибыль (рублей) = (СебестоимостьРеализованныхТоваров * (1 + торговаяНадбавкаВПроцентах / 100) – Расходы – Зарплаты) * (1 – (налогиВПроцентах / 100)).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
24	1	Создать класс Планета, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий объем планеты. Число пи методу известно. Метод принимает радиус планеты (км) и возвращает объем планеты по формуле ОбъемПланеты = (4 / 3) * пи * радиус^3.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2.

		Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
25	1	Создать класс Музыкальная композиция (песня), имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий статус музыкальной композиции (пластинки, диска). Метод принимает количество проданных дисков (штук) и электронных экземпляров данной песни (штук), суммирует их и возвращает символьный массив с указанием статуса песни. Метод определяет статус песни по принципу: если продажи составили больше 100 000 копий, то вернуть «Серебряный» статус, если продажи составили больше 500 000, то вернуть «Золотой», если больше 1 000 000, то – «Платиновый», если больше 10 000 000, то – «Бриллиантовый», иначе – «Нет статуса». Исключите возможность передачи в метод отрицательного количества продаж.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
26	1	Создать класс Спортивная команда, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий коэффициент относительной успешности данной спортивной команды на олимпиаде. Метод принимает количество спортсменов в команде, количество золотых, количество серебряных и количество бронзовых медалей, заработанных командой. Далее, количество золотых медалей умножается на 3 и складывается с количеством серебряных медалей, умноженных на 2, и складывается с количеством бронзовых медалей. Полученная сумма делится на количество спортсменов в команде и вещественный результат возвращается методом. Исключите возможность деления на ноль.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в

		полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
27	1	Создать класс Автомобиль, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий потребленное текущим автомобилем топливо (в литрах) при проезде им некоторого расстояния (входной параметр) с учетом среднего расхода топлива (входной параметр) на данном типе дороги по формуле ОбъемПотребленногоТоплива = расстояние * среднийРасходТоплива.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
28	1	Создать класс Спортсмен, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего

		экземпляра класса? Напишите метод, рассчитывающий и возвращающий ИндексМассыТела по формуле: ИндексМассыТела = вес (кг) / квадрат роста (метры).
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
29	1	Создать класс Мотоцикл, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий суммарное пройденное данным мотоциклом расстояние. Метод принимает значение потребленного топлива (литров) и средний расход топлива на 1 километр пути (литров / км). По формуле ОчередноеПройденноеРасстояние = объемПотребленногоТоплива / среднийРасходТоплива. Далее поле СуммарноеПройденноеРасстояние надо увеличить на величину ОчередногоПройденногоРасстояния. Вернуть из метода новое значение СуммарногоПройденногоРасстояния.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.
30	1	Создать класс Компьютер, имеющий закрытые (private) поля типа int, double, char, bool, статический строковый массив (из символов) и статический массив числового типа. Придумать подходящие по смыслу имена полей и их типы. Для каждого поля класса (а они должны быть закрытыми) написать открытые (public) свойства (геттер и сеттер). В свойстве-сеттере, по возможности, предусмотрите проверку, которая не позволит присвоить полю значение, которое ему не подходит с точки зрения здравого смысла. К полям класса обращайтесь только через свойства. Напишите для вашего класса явно в коде конструктор по умолчанию без параметров и конструктор с параметрами. Последний принимает необходимые значения и инициализирует ими все поля экземпляра своего класса. Поместите в конец каждого конструктора код, который печатает на консоль информацию, что

		проработал такой-то конструктор. Все конструкторы класса должны быть открытыми (public). Напишите в классе деструктор, который сообщает на консоль об уничтожении экземпляра вашего класса. Напишите общедоступный (public) метод (функцию в вашем классе) с именем Print(), который ничего не принимает и не возвращает, но печатает на консоль значения всех полей экземпляра класса с пояснениями (например, кг, литров, №, рублей, метров и т.д.). Почему данный метод «знает» значения, хранимые в полях текущего экземпляра класса? Напишите метод, рассчитывающий и возвращающий количество операций с плавающей запятой, выполняемых за 1 секунду данных компьютером. Метод принимает значение рабочей частоты процессора (в ГигаГерцах) и количество тактов (целое, штук), которые требуются процессору, чтобы осуществить одно вычисление над числом с плавающей запятой. КоличествоОпераций СПлавающейЗапятой = частотаПроцессора / количествоТактовДляОдногоВычисленияСПлавающейЗапятой.
	2	Создайте экземпляр типа вашего класса конструктором без параметров. Распечатайте хранимые значения в полях вашего экземпляра с помощью вызова у него метода Print(). Проинициализируйте поля вашего экземпляра класса посредством его свойств-сеттеров. Снова распечатайте содержимое полей вашего экземпляра посредством вызова у него метода Print().
	3	Создайте объект вашего класса посредством вызова конструктора с параметрами, который примет значения, ранее введенные пользователем с клавиатуры. Распечатайте значения полей вашего объекта класса путем вызова у него метода Print().
	4	Создайте динамический объект типа вашего класса путем вызова конструктора без параметров. Через свойства-сеттеры заполните его поля, запросив значения у пользователя. Вызовите у вашего объекта методы Print() и метод из пункта 1. Создайте динамический объект типа вашего класса путем вызова конструктора с параметрами. Вызовите у вашего объекта методы Print() и метод из пункта 1.
	5	Создайте указатель на объект, созданный в пункте 2. Создайте указатель на объект, созданный в пункте 4. Вызовите у этих объектов метод Print() через эти указатели. Создайте ссылку на объект, созданный в пункте 2. Создайте ссылку на объект, созданный в пункте 4. Вызовите метод пункта 1 у этих объектов через ссылки. Вызовите деструкторы у ваших объектов перед завершением программы.