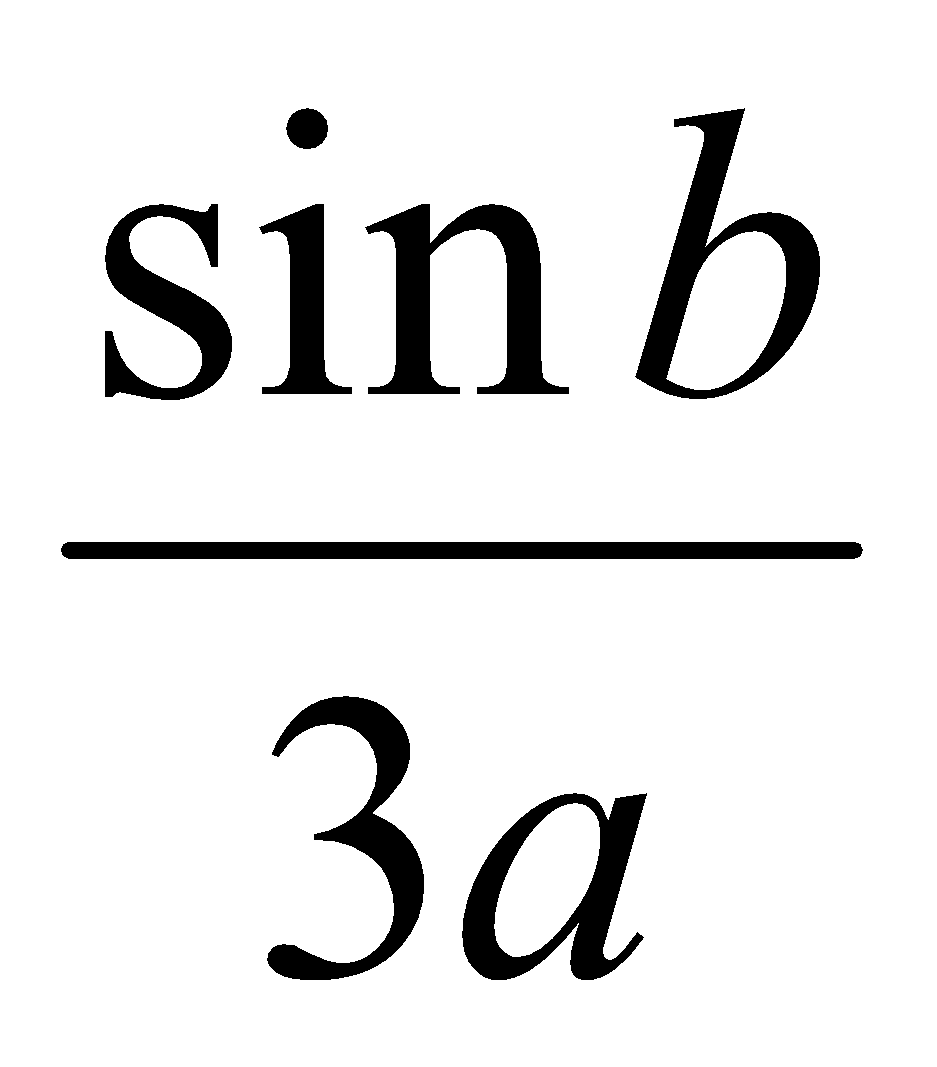
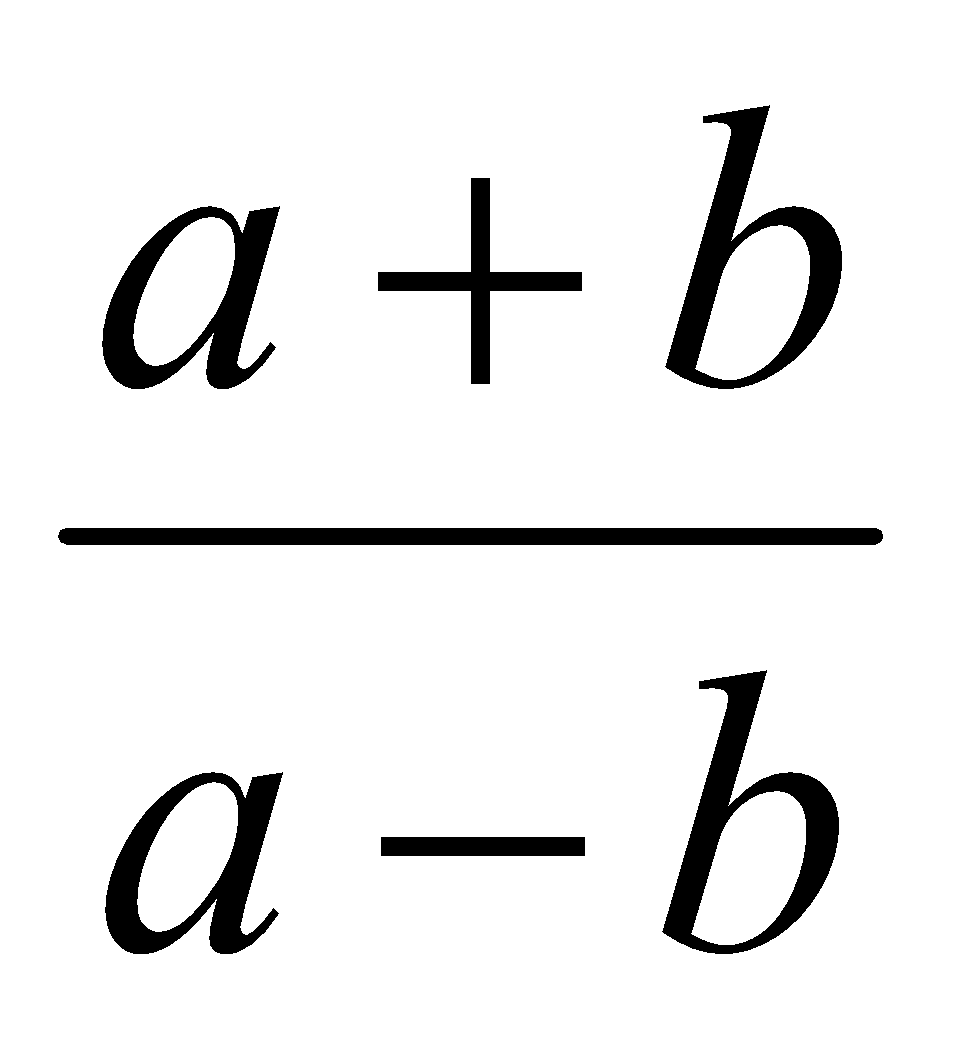
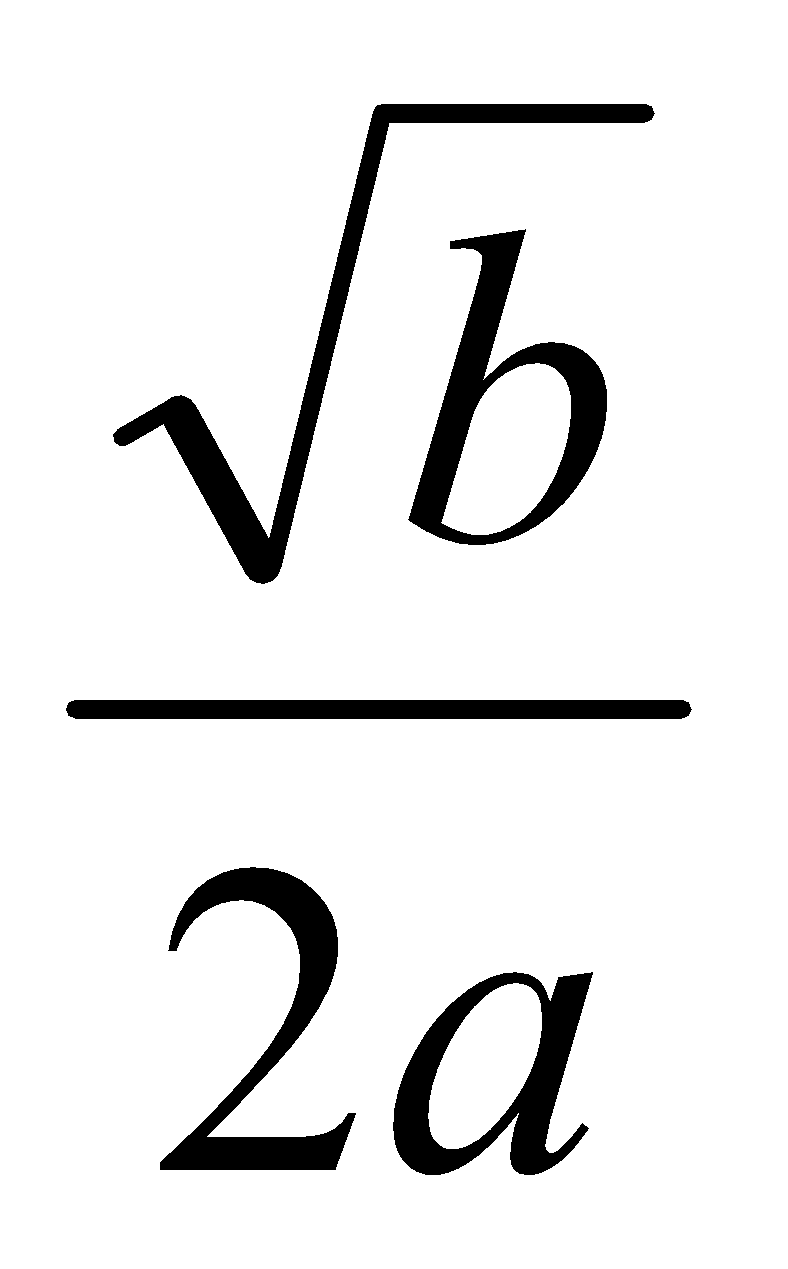
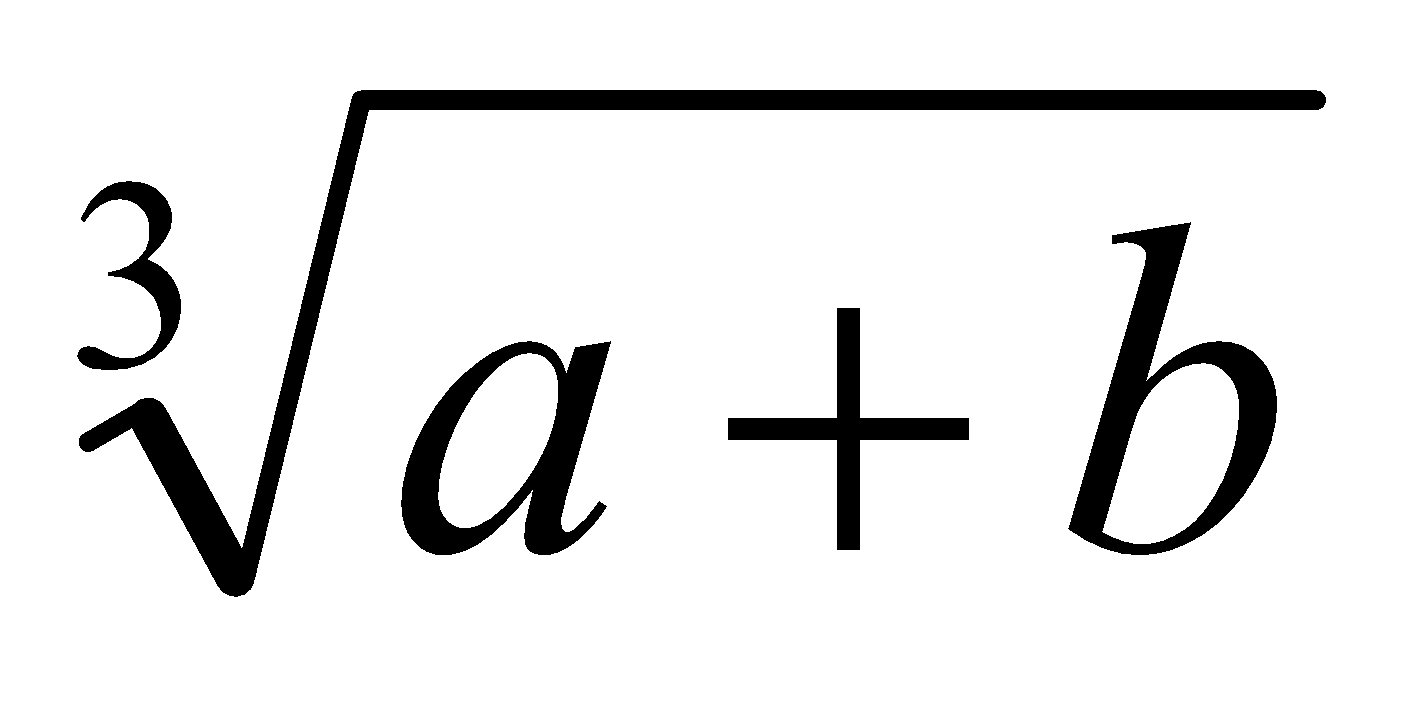
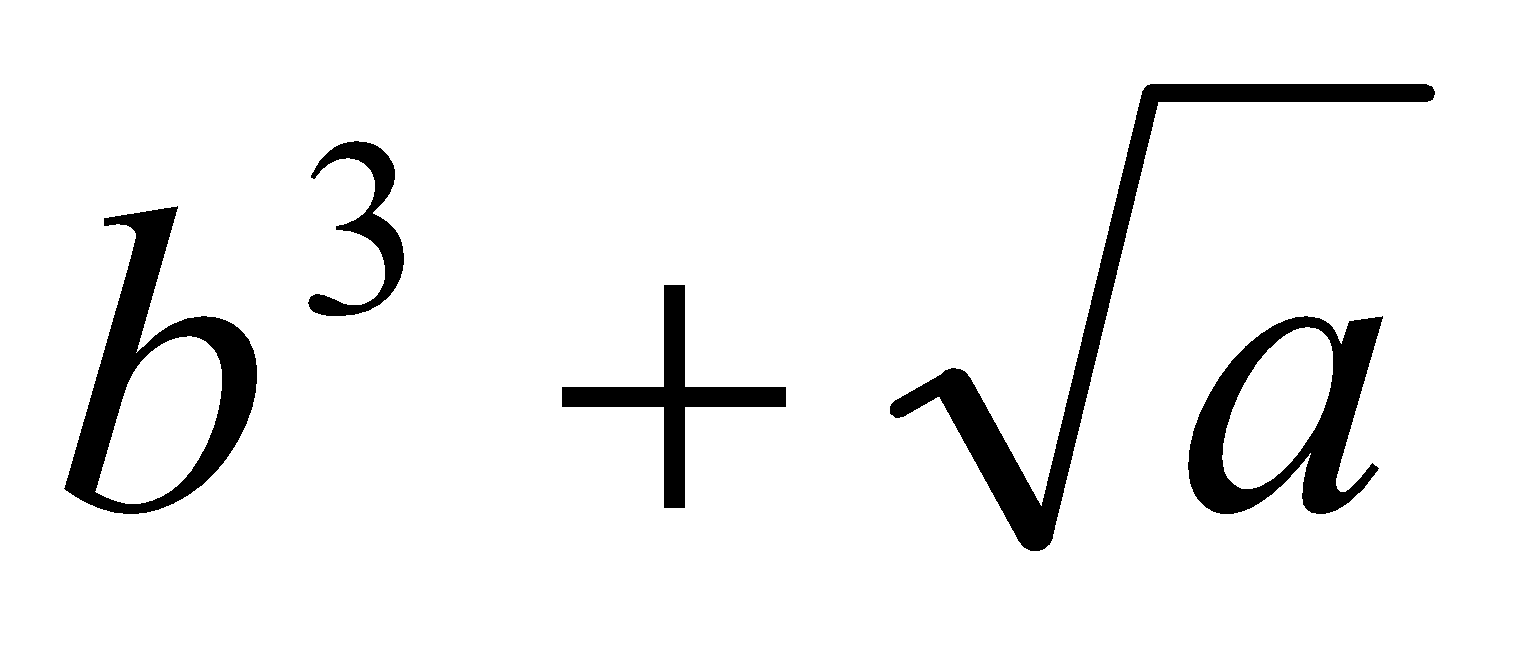
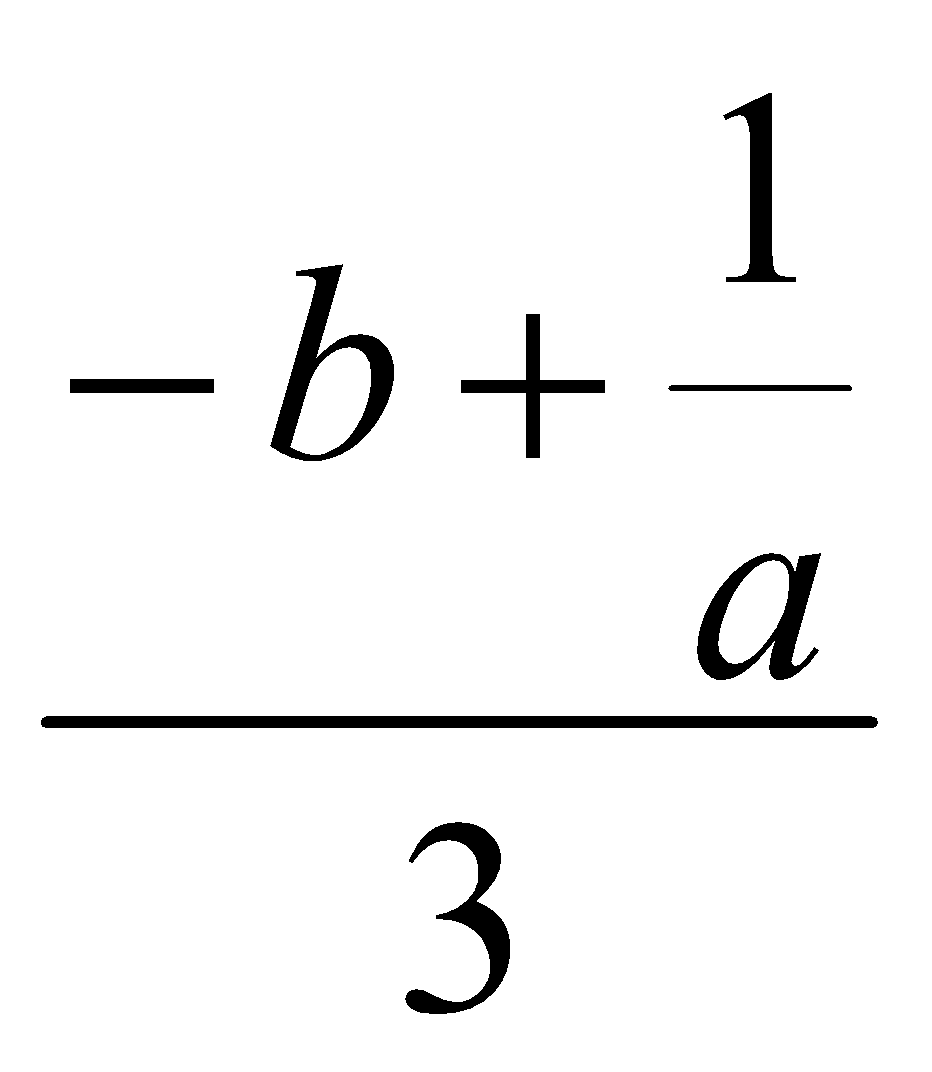
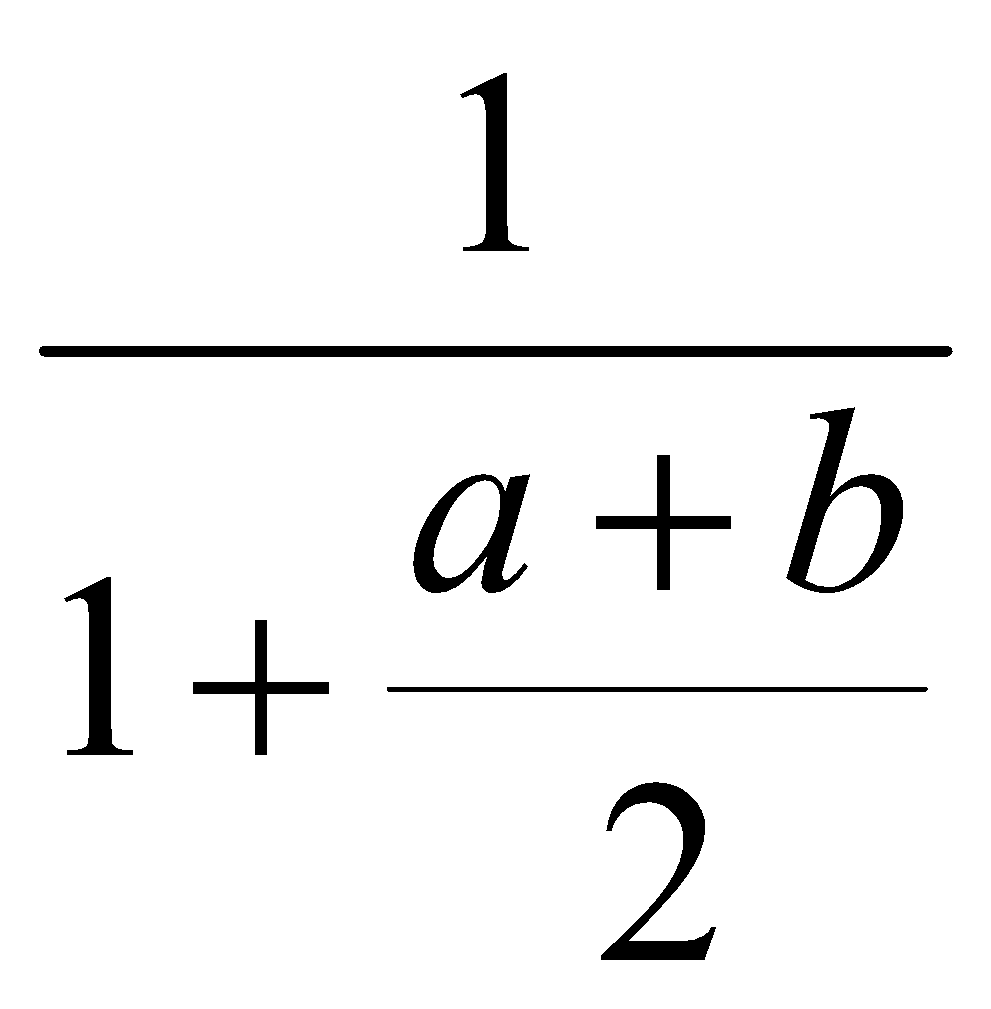
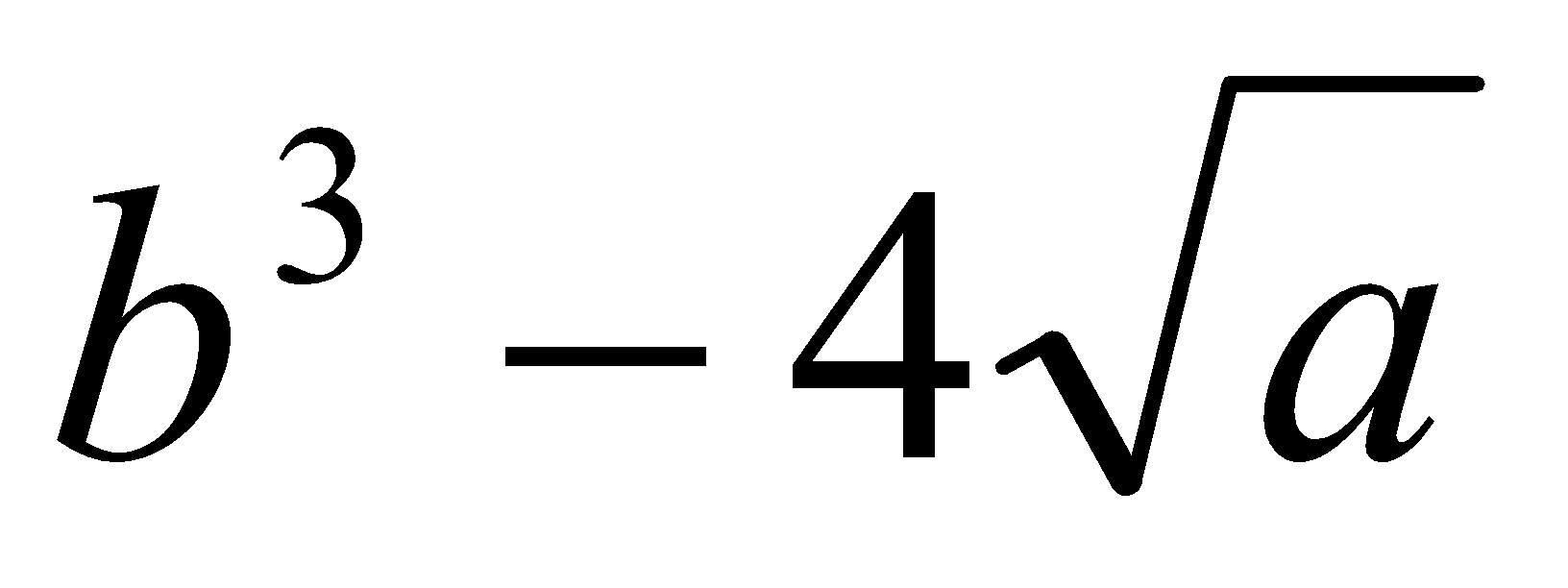
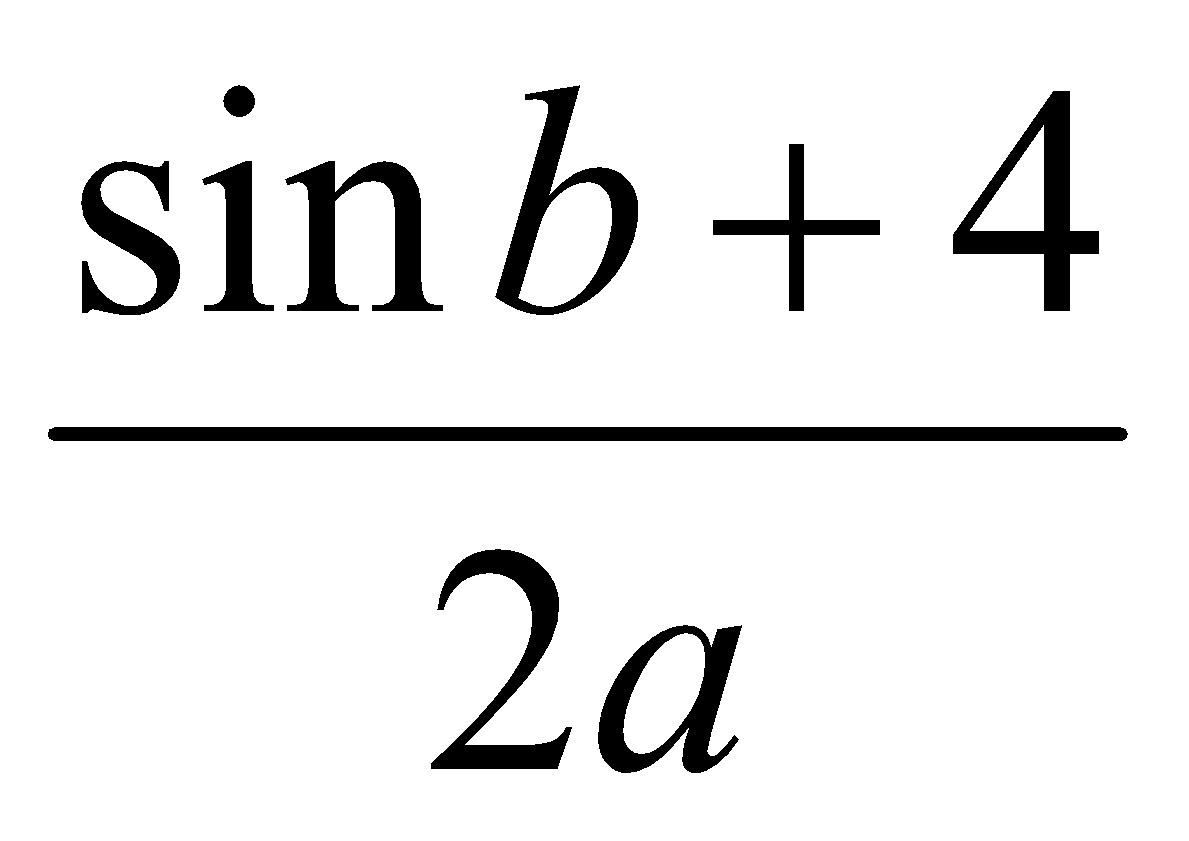
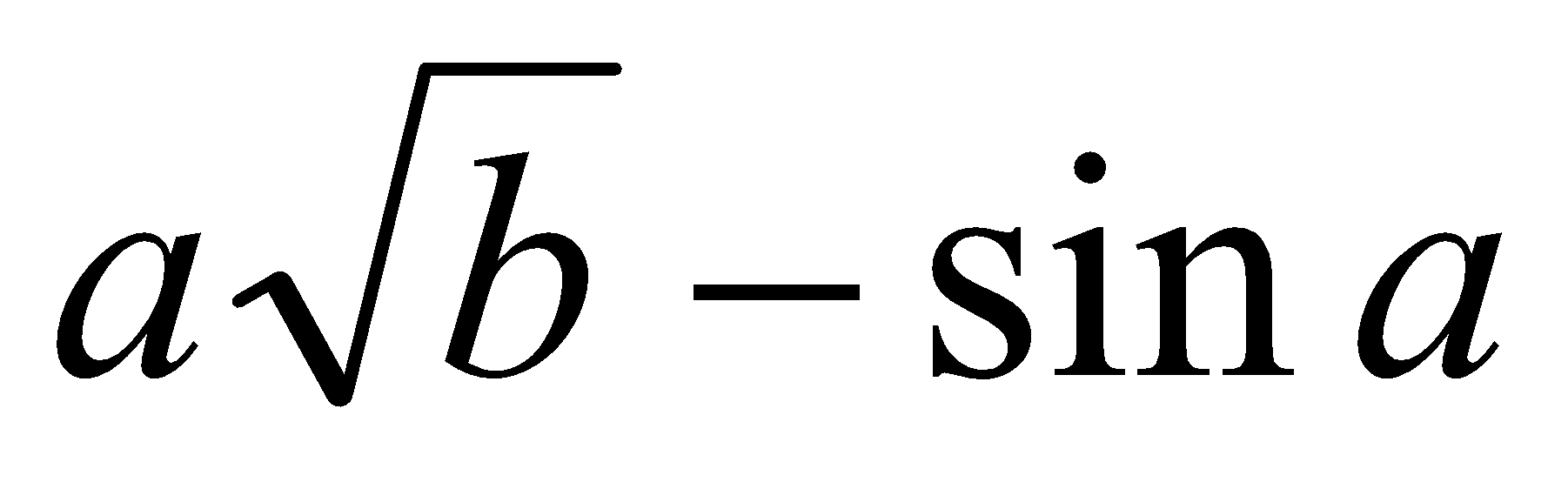
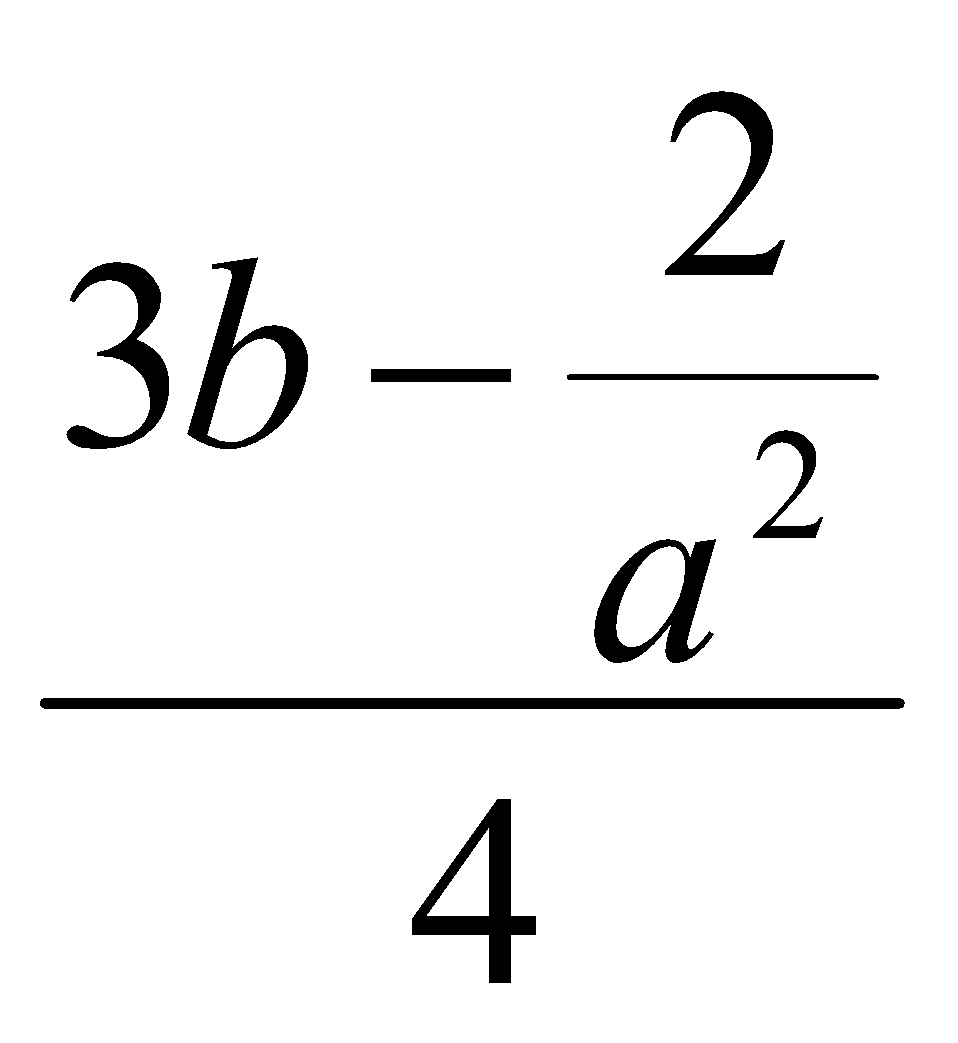
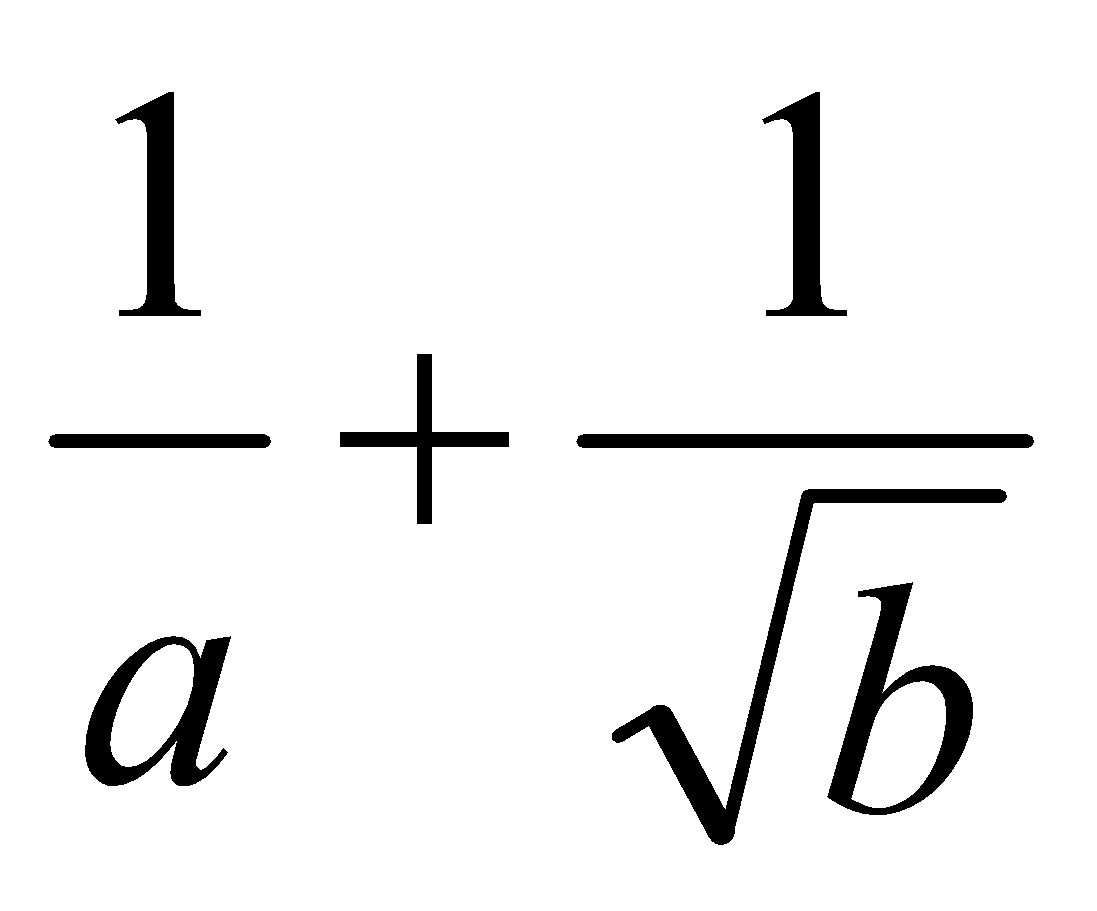
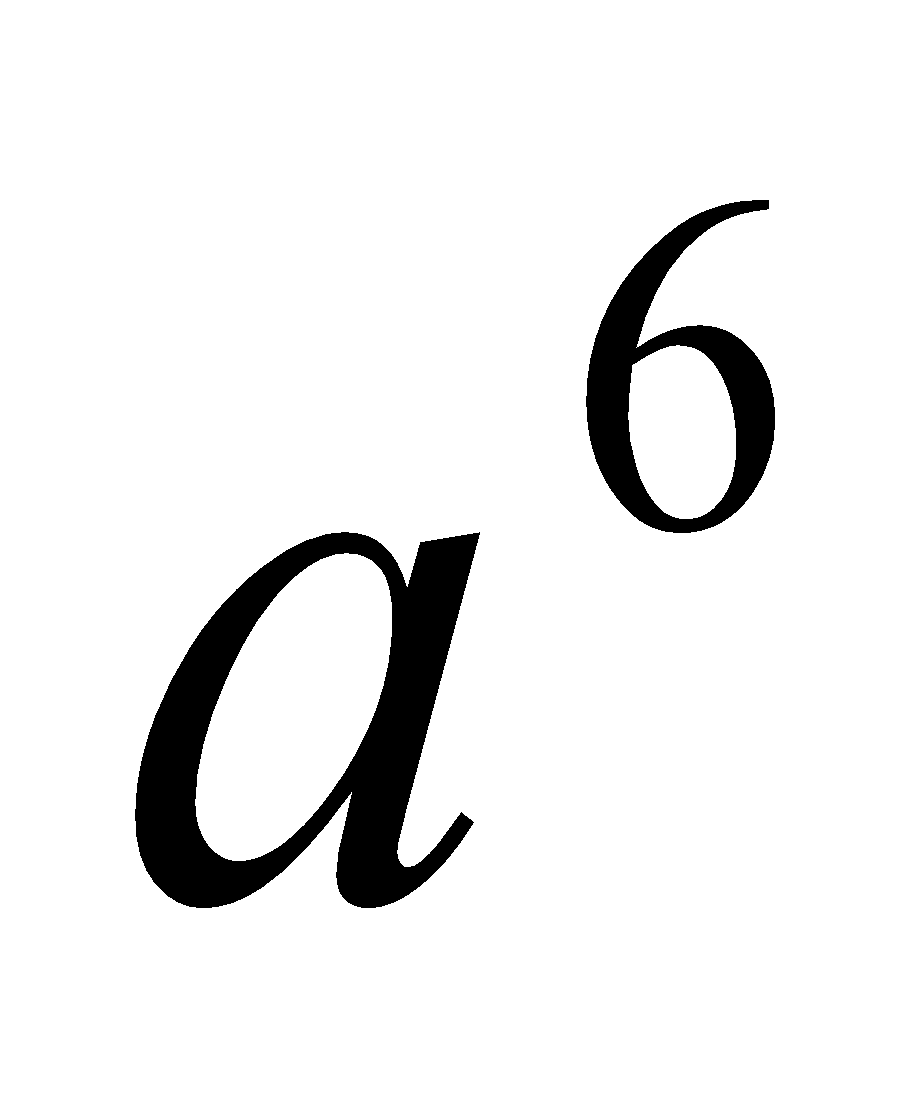
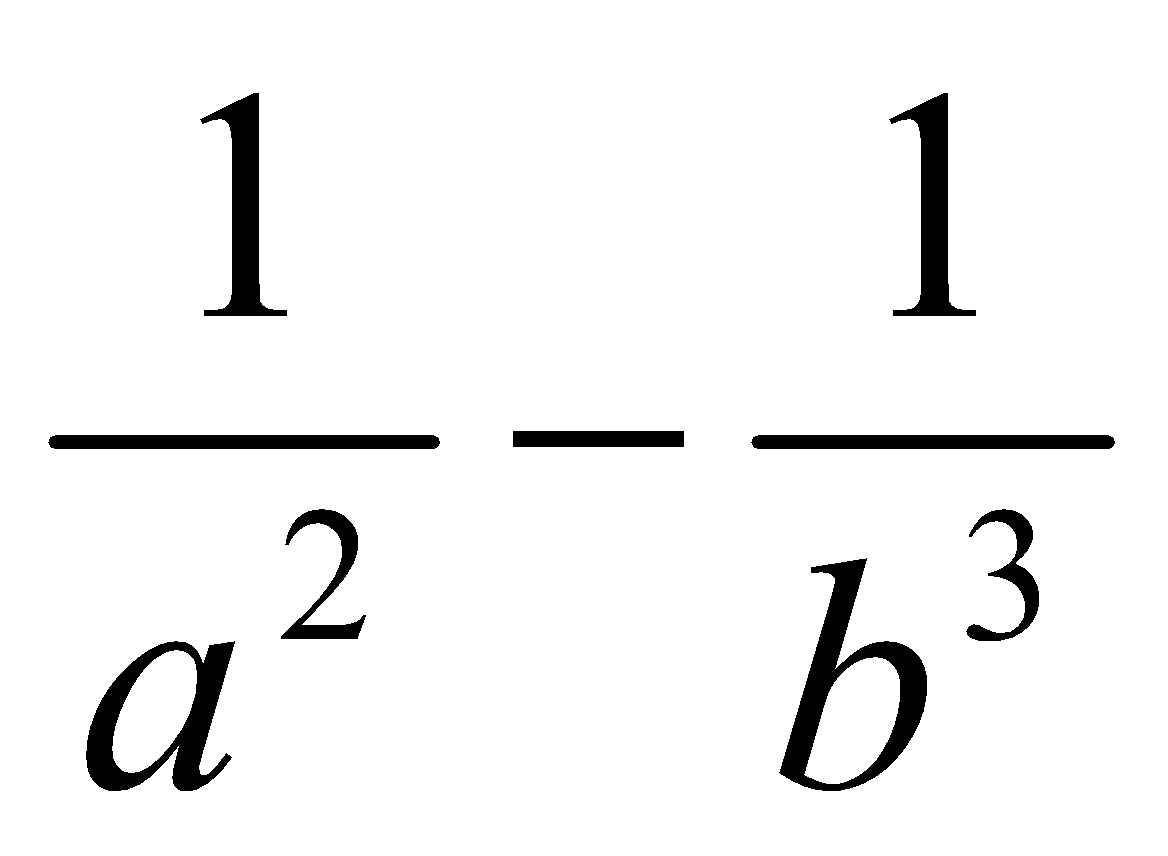
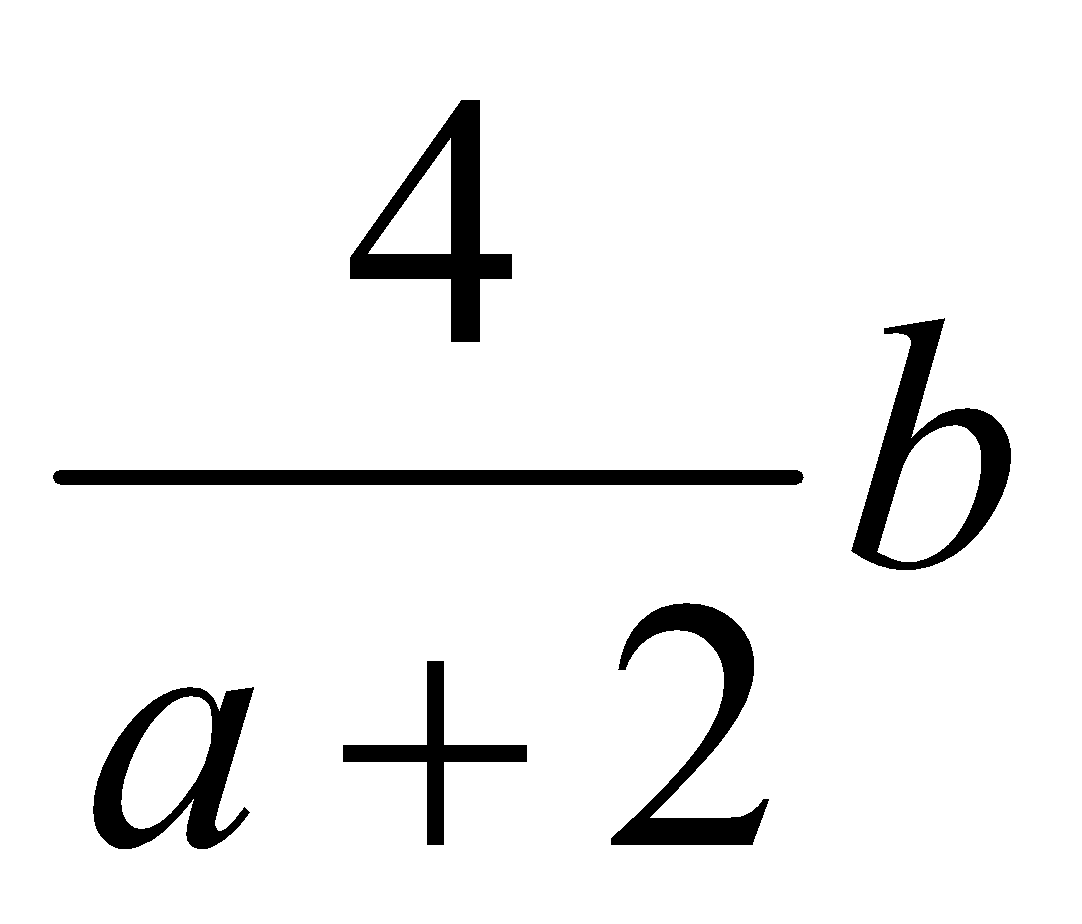
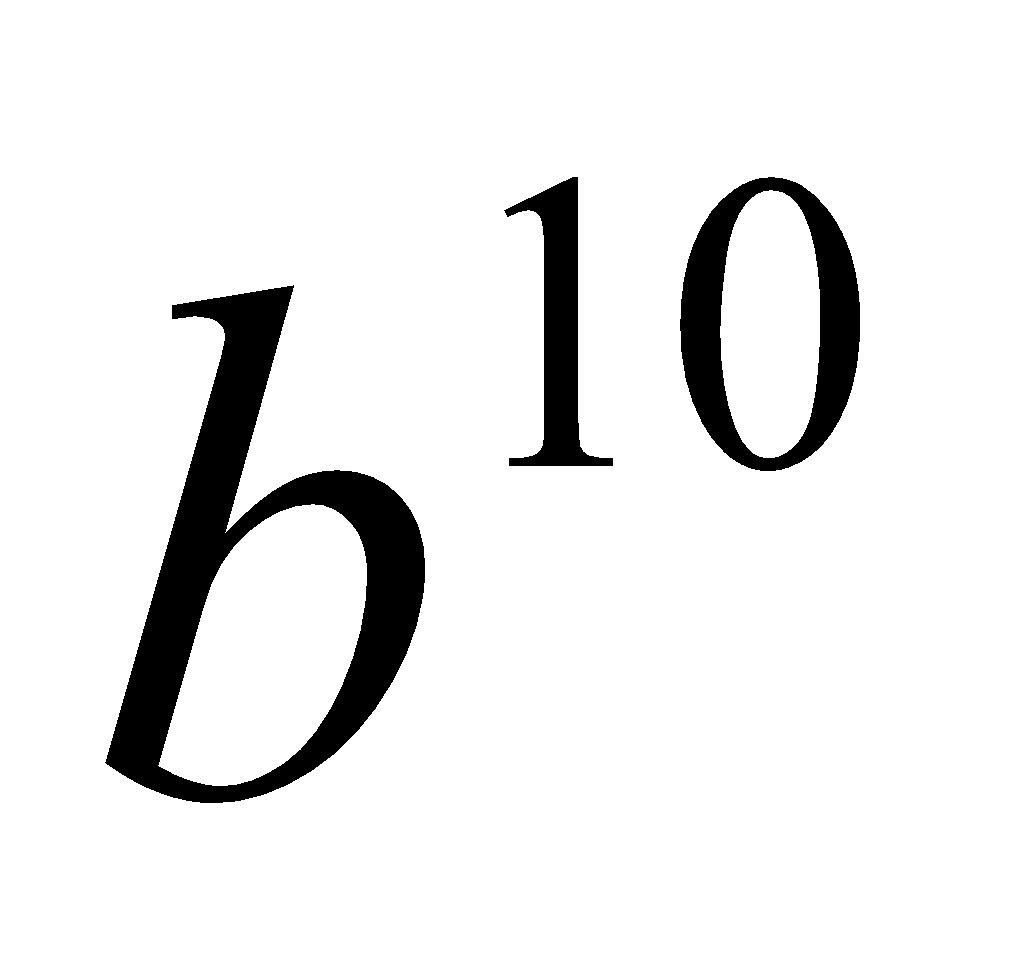
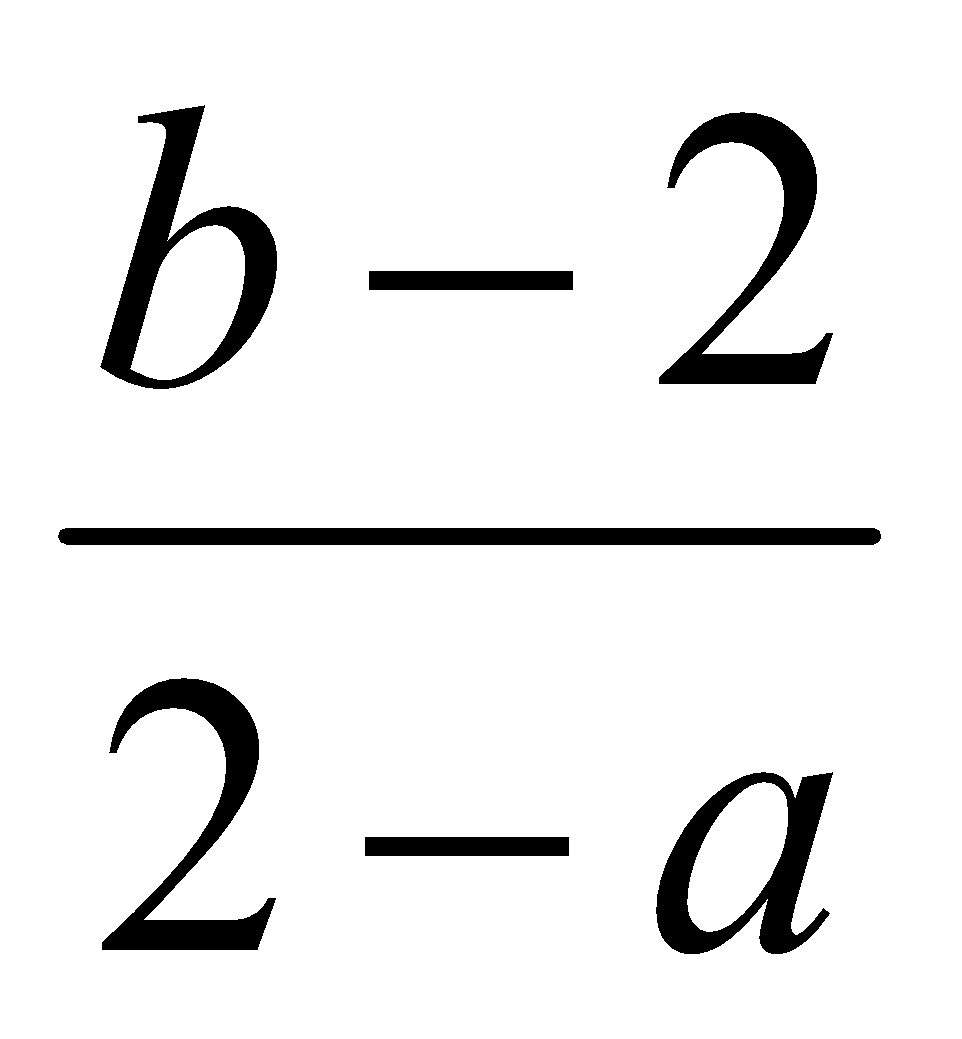
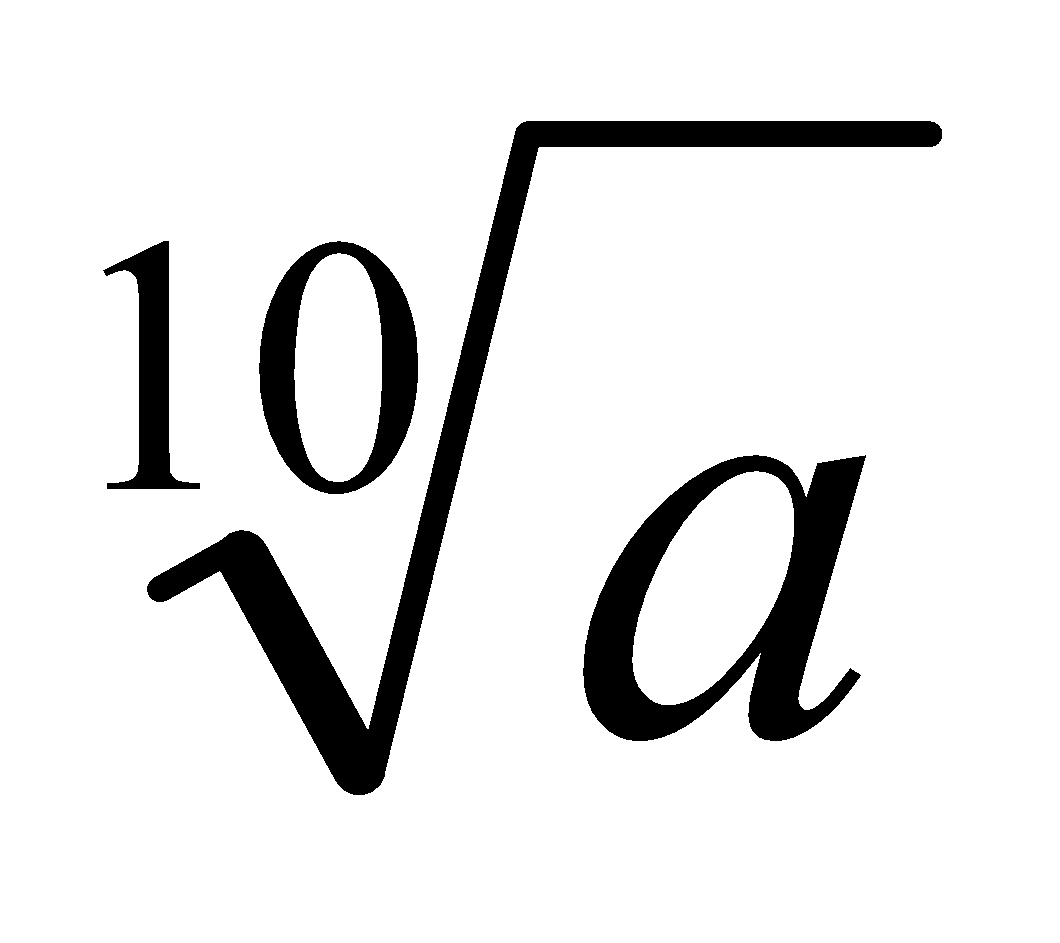
**Тема: “Классы в .NET. Специальные типы классов”**

**Задание 1.**

Создать класс А с целочисленными полями а и b и двумя методами согласно варианту.Внутри класса реализовать конструктор для инициализации a и b. Создать объект класса и продемонстрировать работу со всеми элементами класса.

1. Метод вычисления суммы a и b, метод вычисления значения выражения .
2. Метод вычисления разности a и b, метод вычисления значения выражения .
3. Метод вычисления произведения a и b, метод вычисления значения выражения .
4. Метод вычисления частного a и b, метод вычисления значения выражения .
5. Метод нахождения среднего арифметического a и b, метод вычисления значения выражения .
6. Метод вычисления значения выражения , метод возведения в куб суммы a и b.
7. Метод вычисления значения выражения , метод возведения в квадрат разности a и b.
8. Метод вычисления значения выражения , метод возведения в квадрат частного a и b.
9. Метод вычисления значения выражения, метод возведения в квадрат суммы a и b.
10. Метод вычисления значения выражения , метод возведения в куб произведения a и b
11. Метод вычисления значения выражения , метод возведения в куб частного a и b.
12. Метод вычисления значения выражения , метод вычисления .
13. Метод вычисления значения выражения , метод возведения в куб суммы a и b..
14. Метод вычисления значения выражения , метод вычисления .
15. Метод вычисления значения выражения , метод вычисления .

**Задание 2: Работа с static классами**

Создайте static классы, которые выполняют различные операции с массивами объектов, включая сортировку, фильтрацию, вычисление статистики и генерацию данных.

**Задачи:**

1. Создать static класс ArrayUtils, содержащий метод GetMaxValue, который находит максимальный возраст среди массива объектов класса Person.
2. Добавить в ArrayUtils метод SortByName, который сортирует массив Person по имени в алфавитном порядке.
3. Реализовать static метод FilterByAge, возвращающий всех людей старше 30 лет.
4. Добавить метод CalculateAverageSalary, который вычисляет среднюю зарплату среди массива Employee.
5. Реализовать метод GenerateRandomPersons, создающий массив случайных объектов Person заданного размера.
6. Создать static класс MathOperations, содержащий метод Sum, который складывает числа в переданном массиве double.
7. Реализовать метод Product, который перемножает все элементы массива double.
8. Создать метод FindMostPopularCity, который определяет, в каком городе проживает больше всего людей.
9. Реализовать static метод ShuffleArray, который перемешивает элементы массива Person.
10. Добавить метод FindOldestEmployee, который находит самого старшего работника из списка Employee.
11. Создать static класс StringProcessor, содержащий метод ConcatenateNames, который объединяет имена всех объектов Person через запятую.
12. Реализовать метод CountPeopleInCity, который подсчитывает количество людей в каждом городе.
13. Добавить метод ReverseArray, который меняет порядок элементов в массиве Person на обратный.
14. Реализовать static метод FindEmployeesByDepartment, который возвращает список сотрудников из заданного департамента.
15. Реализовать static метод GroupByAge, который группирует людей по возрасту и возвращает словарь, где ключ — возраст, а значение — список людей.

**Задание 3: Работа с sealed и abstract классами**

В этом задании вам предстоит разработать иерархию классов с использованием abstract и sealed классов. В каждой задаче у вас должен быть абстрактный класс (родительский), sealed класс (конкретная реализация), модельный класс с массивом объектов, а также два метода бизнес-логики.

**Задачи**

**1. Система управления транспортом**

Разработайте систему учета транспорта на предприятии.

**Структура классов:**

* **abstract класс Transport** (базовый) с полями Model, MaxSpeed, FuelConsumption.
* **sealed классы Car и Truck** (конкретные реализации).
* **TransportManager** (модельный класс) с массивом Transport[] для хранения всех транспортных средств.

**Методы бизнес-логики:**

1. **GetMostEfficientVehicle()** – находит транспорт с наименьшим расходом топлива.
2. **GetFastestVehicle()** – определяет самый быстрый транспорт.

**2. Система управления банковскими счетами**

Разработайте систему управления счетами клиентов в банке.

**Структура классов:**

* **abstract класс BankAccount** с полями AccountNumber, Balance, OwnerName.
* **sealed классы SavingsAccount и CheckingAccount** (сохранения и расчетный счет).
* **Bank** (модельный класс) с массивом BankAccount[] для хранения всех счетов.

**Методы бизнес-логики:**

1. **GetRichestClient()** – находит клиента с самым большим балансом.
2. **GetTotalBankBalance()** – вычисляет общий баланс всех клиентов.

**3. Система управления персоналом**

Создайте систему учета сотрудников в компании.

**Структура классов:**

* **abstract класс Employee** с полями Name, Age, Salary.
* **sealed классы Manager и Developer** (руководитель и разработчик).
* **Company** (модельный класс) с массивом Employee[] для хранения сотрудников.

**Методы бизнес-логики:**

1. **GetHighestPaidEmployee()** – находит сотрудника с самой высокой зарплатой.
2. **GetAverageAge()** – вычисляет средний возраст сотрудников.

**4. Система учета студентов**

Разработайте систему для управления студентами в университете.

**Структура классов:**

* **abstract класс Person** с полями FullName, Age.
* **sealed классы Student и Teacher**.
* **University** (модельный класс) с массивом Person[].

**Методы бизнес-логики:**

1. **GetBestStudent()** – находит студента с самым высоким средним баллом.
2. **GetTeachersByAge(int age)** – возвращает всех преподавателей старше указанного возраста.

**5. Система складского учета**

Создайте систему управления товарами на складе.

**Структура классов:**

* **abstract класс Product** с полями Name, Price, Quantity.
* **sealed классы Electronics и Clothing** (электроника и одежда).
* **Warehouse** (модельный класс) с массивом Product[].

**Методы бизнес-логики:**

1. **GetTotalStockValue()** – вычисляет общую стоимость всех товаров.
2. **FindMostExpensiveProduct()** – находит самый дорогой товар.

**6. Система управления билетами**

Разработайте систему учета билетов на различные мероприятия.

**Структура классов:**

* **abstract класс Ticket** с полями EventName, Price, SeatNumber.
* **sealed классы ConcertTicket и TheaterTicket** (концерт и театр).
* **TicketOffice** (модельный класс) с массивом Ticket[].

**Методы бизнес-логики:**

1. **GetTotalRevenue()** – вычисляет общую сумму проданных билетов.
2. **GetMostExpensiveTicket()** – находит самый дорогой билет.

**7. Система управления книгами в библиотеке**

Разработайте систему учета книг в библиотеке.

**Структура классов:**

* **abstract класс Book** с полями Title, Author, Pages.
* **sealed классы FictionBook и NonFictionBook**.
* **Library** (модельный класс) с массивом Book[].

**Методы бизнес-логики:**

1. **GetMostPagesBook()** – находит книгу с наибольшим количеством страниц.
2. **GetBooksByAuthor(string author)** – возвращает все книги указанного автора.

**8. Система управления медицинскими пациентами**

Разработайте систему учета пациентов в больнице.

**Структура классов:**

* **abstract класс Person** с полями FullName, Age, HealthStatus.
* **sealed классы Patient и Doctor**.
* **Hospital** (модельный класс) с массивом Person[].

**Методы бизнес-логики:**

1. **GetMostCriticalPatient()** – находит пациента с самым тяжелым состоянием.
2. **GetDoctorsBySpecialty(string specialty)** – возвращает врачей с указанной специализацией.

**9. Система управления заказами в магазине**

Создайте систему учета заказов.

**Структура классов:**

* **abstract класс Order** с полями OrderId, CustomerName, TotalAmount.
* **sealed классы OnlineOrder и InStoreOrder**.
* **Store** (модельный класс) с массивом Order[].

**Методы бизнес-логики:**

1. **GetLargestOrder()** – находит заказ с самой высокой суммой.
2. **GetOrdersByCustomer(string customerName)** – возвращает заказы конкретного клиента

**10. Система учета спортсменов**

Разработайте систему управления спортивными командами.

**Структура классов:**

* **abstract класс Athlete** с полями Name, Age, Sport, Medals.
* **sealed классы Runner и Swimmer** (разные виды спортсменов).
* **SportsClub** (модельный класс) с массивом Athlete[].

**Методы бизнес-логики:**

1. **GetMostAwardedAthlete()** – находит спортсмена с наибольшим количеством медалей.
2. **GetAthletesBySport(string sport)** – возвращает список спортсменов по указанному виду спорта.

**11. Система учета фильмов**

Разработайте систему управления фильмами в кинотеатре.

**Структура классов:**

* **abstract класс Movie** с полями Title, Genre, Duration.
* **sealed классы ActionMovie и ComedyMovie**.
* **Cinema** (модельный класс) с массивом Movie[].

**Методы бизнес-логики:**

1. **GetLongestMovie()** – находит фильм с самой длинной продолжительностью.
2. **GetMoviesByGenre(string genre)** – возвращает список фильмов указанного жанра.

**12. Система управления интернет-магазином**

Создайте систему учета товаров в интернет-магазине.

**Структура классов:**

* **abstract класс Product** с полями Name, Category, Price, StockQuantity.
* **sealed классы ElectronicsProduct и ClothingProduct**.
* **OnlineStore** (модельный класс) с массивом Product[].

**Методы бизнес-логики:**

1. **GetOutOfStockProducts()** – находит все товары, которых нет в наличии.
2. **GetMostExpensiveProduct()** – находит самый дорогой товар.

**13. Система управления домашними животными**

Разработайте систему учета домашних животных.

**Структура классов:**

* **abstract класс Pet** с полями Name, Breed, Age, OwnerName.
* **sealed классы Dog и Cat**.
* **VeterinaryClinic** (модельный класс) с массивом Pet[].

**Методы бизнес-логики:**

1. **GetOldestPet()** – находит самое старое домашнее животное.
2. **GetPetsByOwner(string ownerName)** – возвращает всех питомцев указанного владельца.

**14. Система управления музыкальными альбомами**

Разработайте систему учета музыкальных альбомов.

**Структура классов:**

* **abstract класс MusicAlbum** с полями Title, Artist, ReleaseYear, TrackCount.
* **sealed классы RockAlbum и PopAlbum**.
* **MusicLibrary** (модельный класс) с массивом MusicAlbum[].

**Методы бизнес-логики:**

1. **GetNewestAlbum()** – находит самый новый альбом.
2. **GetAlbumsByArtist(string artist)** – возвращает альбомы указанного исполнителя.

**15. Система управления книгами в книжном магазине**

Разработайте систему учета книг в книжном магазине.

**Структура классов:**

* **abstract класс Book** с полями Title, Author, Price, StockQuantity.
* **sealed классы FictionBook и NonFictionBook**.
* **Bookstore** (модельный класс) с массивом Book[].

**Методы бизнес-логики:**

1. **GetCheapestBook()** – находит самую дешевую книгу.
2. **GetBooksInStock()** – возвращает список книг, которые есть в наличии.

**Задание 4: Работа с partial классами**

**1. Система управления студентами**

Разработайте систему учета студентов в университете.

**Структура классов:**

* **partial класс Student**:
  + В одной части (Student.Data.cs) хранятся поля Name, Group, GPA.
  + В другой (Student.Methods.cs) – методы работы со студентами.
* **University** (модельный класс) с массивом Student[].

**Методы бизнес-логики:**

1. **GetTopStudents()** – находит студентов с GPA выше 4.5.
2. **GetStudentsByGroup(string group)** – возвращает список студентов указанной группы.

**2. Система учета заказов интернет-магазина**

**Структура классов:**

* **partial класс Order**:
  + В первой части (Order.Data.cs) – OrderID, CustomerName, TotalAmount.
  + В другой (Order.Methods.cs) – методы обработки заказов.
* **Store** (модельный класс) с массивом Order[].

**Методы бизнес-логики:**

1. **GetHighValueOrders(decimal minAmount)** – находит заказы дороже указанной суммы.
2. **GetOrdersByCustomer(string customerName)** – возвращает все заказы конкретного клиента.

**3. Система управления автомобилями в автопарке**

**Структура классов:**

* **partial класс Car**:
  + В первой части (Car.Data.cs) – Brand, Model, Year, Mileage.
  + В другой (Car.Methods.cs) – методы управления автопарком.
* **Fleet** (модельный класс) с массивом Car[].

**Методы бизнес-логики:**

1. **GetNewestCars(int year)** – находит машины, выпущенные после указанного года.
2. **GetCarsByBrand(string brand)** – возвращает автомобили указанной марки.

**4. Система учета сотрудников в компании**

**Структура классов:**

* **partial класс Employee**:
  + В первой части (Employee.Data.cs) – Name, Position, Salary.
  + В другой (Employee.Methods.cs) – методы анализа сотрудников.
* **Company** (модельный класс) с массивом Employee[].

**Методы бизнес-логики:**

1. **GetHighestPaidEmployees()** – находит сотрудников с самой высокой зарплатой.
2. **GetEmployeesByPosition(string position)** – возвращает сотрудников указанной должности.

**5. Система управления медицинскими записями**

**Структура классов:**

* **partial класс Patient**:
  + В первой части (Patient.Data.cs) – Name, Age, Diagnosis.
  + В другой (Patient.Methods.cs) – методы работы с пациентами.
* **Hospital** (модельный класс) с массивом Patient[].

**Методы бизнес-логики:**

1. **GetPatientsByDiagnosis(string diagnosis)** – находит всех пациентов с указанным диагнозом.
2. **GetOldestPatient()** – находит самого старшего пациента.

**6. Система учета продуктов в магазине**

**Структура классов:**

* **partial класс Product**:
  + В первой части (Product.Data.cs) – Name, Category, Price, Stock.
  + В другой (Product.Methods.cs) – методы управления товарами.
* **Warehouse** (модельный класс) с массивом Product[].

**Методы бизнес-логики:**

1. **GetOutOfStockProducts()** – находит товары, которых нет в наличии.
2. **GetMostExpensiveProduct()** – находит самый дорогой товар.

**7. Система учета книг в библиотеке**

**Структура классов:**

* **partial класс Book**:
  + В первой части (Book.Data.cs) – Title, Author, Pages, Genre.
  + В другой (Book.Methods.cs) – методы работы с книгами.
* **Library** (модельный класс) с массивом Book[].

**Методы бизнес-логики:**

1. **GetLongestBook()** – находит книгу с наибольшим количеством страниц.
2. **GetBooksByGenre(string genre)** – возвращает книги указанного жанра.

**8. Система учета клиентов банка**

**Структура классов:**

* **partial класс BankClient**:
  + В первой части (BankClient.Data.cs) – Name, AccountBalance.
  + В другой (BankClient.Methods.cs) – методы анализа счетов.
* **Bank** (модельный класс) с массивом BankClient[].

**Методы бизнес-логики:**

1. **GetClientsWithLowBalance(decimal minBalance)** – находит клиентов с балансом ниже указанного.
2. **GetRichestClient()** – находит клиента с максимальным балансом.

**9. Система учета фильмов**

**Структура классов:**

* **partial класс Movie**:
  + В первой части (Movie.Data.cs) – Title, Director, Duration, Genre.
  + В другой (Movie.Methods.cs) – методы работы с фильмами.
* **Cinema** (модельный класс) с массивом Movie[].

**Методы бизнес-логики:**

1. **GetLongestMovie()** – находит фильм с самой длинной продолжительностью.
2. **GetMoviesByDirector(string director)** – возвращает фильмы указанного режиссера.

**10. Система управления ресторанами**

**Структура классов:**

* **partial класс Dish**:
  + В первой части (Dish.Data.cs) – Name, Category, Price, Weight.
  + В другой (Dish.Methods.cs) – методы работы с блюдами.
* **Restaurant** (модельный класс) с массивом Dish[].

**Методы бизнес-логики:**

1. **GetMostExpensiveDish()** – находит самое дорогое блюдо.
2. **GetDishesByCategory(string category)** – возвращает все блюда указанной категории (например, "Десерты", "Горячие блюда").

**11. Система управления турами**

**Структура классов:**

* **partial класс Tour**:
  + В первой части (Tour.Data.cs) – Destination, Duration, Price, Hotel.
  + В другой (Tour.Methods.cs) – методы поиска туров.
* **TravelAgency** (модельный класс) с массивом Tour[].

**Методы бизнес-логики:**

1. **GetLongestTour()** – находит тур с самой длительной поездкой.
2. **GetToursByDestination(string destination)** – возвращает список туров в указанное место.

**12. Система учета аренды автомобилей**

**Структура классов:**

* **partial класс RentalCar**:
  + В первой части (RentalCar.Data.cs) – Brand, Model, Year, PricePerDay, IsAvailable.
  + В другой (RentalCar.Methods.cs) – методы управления прокатом авто.
* **RentalService** (модельный класс) с массивом RentalCar[].

**Методы бизнес-логики:**

1. **GetAvailableCars()** – находит автомобили, доступные для аренды.
2. **GetCarsByBrand(string brand)** – возвращает список автомобилей указанной марки.

**13. Система учета спортивных соревнований**

**Структура классов:**

* **partial класс Athlete**:
  + В первой части (Athlete.Data.cs) – Name, Sport, Country, Records.
  + В другой (Athlete.Methods.cs) – методы анализа спортсменов.
* **SportsEvent** (модельный класс) с массивом Athlete[].

**Методы бизнес-логики:**

1. **GetAthletesBySport(string sport)** – возвращает всех спортсменов в указанной дисциплине.
2. **GetRecordHolders()** – находит спортсменов с установленными рекордами.

**14. Система управления пользователями в социальной сети**

**Структура классов:**

* **partial класс User**:
  + В первой части (User.Data.cs) – Username, FollowersCount, PostsCount, LastActiveDate.
  + В другой (User.Methods.cs) – методы работы с пользователями.
* **SocialNetwork** (модельный класс) с массивом User[].

**Методы бизнес-логики:**

1. **GetMostPopularUsers(int minFollowers)** – находит пользователей с количеством подписчиков выше указанного.
2. **GetInactiveUsers(int days)** – возвращает пользователей, которые не заходили в соцсеть указанное количество дней.

**15. Система учета домашних животных в ветеринарной клинике**

**Структура классов:**

* **partial класс Pet**:
  + В первой части (Pet.Data.cs) – Name, Species, Age, OwnerName.
  + В другой (Pet.Methods.cs) – методы учета животных.
* **VeterinaryClinic** (модельный класс) с массивом Pet[].

**Методы бизнес-логики:**

1. **GetOldestPet()** – находит самое старое животное.
2. **GetPetsByOwner(string ownerName)** – возвращает всех животных указанного владельца.