Вопрос 1.

1. **Метод** - это …
2. **Статическая система типов** - это …
3. **Динамическая система типов** - это …
4. **Конструктор** - это …
5. **Перегрузка** (overloading) - это …
6. **Переопределение** (overriding) - это …
7. **Статический метод** - это …
8. **Выявление типов** (type inference) - это …
9. **Композиция** (containment) - это …
10. **Ad hoc полиморфизм** - это …
11. **Инкапсуляция**- это …
12. **Универсальный полиморфизм** - это …
13. **Обобщенное программирование** (generics) - это …
14. **Исключение** (exception) - это …
15. **Инварианты** - это …
16. **Базовый абстрактный класс** - это …
17. **Обработчик событий** (event handler) - это …
18. Принцип **сокрытия данных** - это …
19. **Динамическое связывание** (dynamic binding) - это …
20. **Полиморфизм** - это …
21. **Наследование** - это …
22. **Абстрактный тип данных** - это …
23. **Интерфейс** - это …
24. **Класс** - это …
25. **Объект** - это …
26. **Состояние объекта** - это …
27. Перечислите **виды конструкторов**, которые вам известны (относительно типов их аргументов).
28. Перечислите **стратегии обработки ошибок**, которые вам известны.
29. Приведите примеры **сценариев**, в которых необходима **обработка ошибок**.
30. **Шаблоны типов** - это …
31. **Поток данных** - это …
32. Принцип **единственной ответственности** (single responsibility principle) - это …
33. Принцип **открытости/закрытости** (open-closed principle) - это …
34. Принцип **подстановки Лисков** (Liskov substitution principle) - это …
35. Принцип **разделения интерфейса** (interface segregation principle) - это …
36. Принцип **инверсии зависимостей** (dependency inversion principle) - это …
37. **Интерфейс программирования приложений** (API) - это …
38. Сравните **процедурное**, **ОО** и **обобщенное** программирование.
39. Перечислите **основные концепции** ООП.

Вопрос 2.

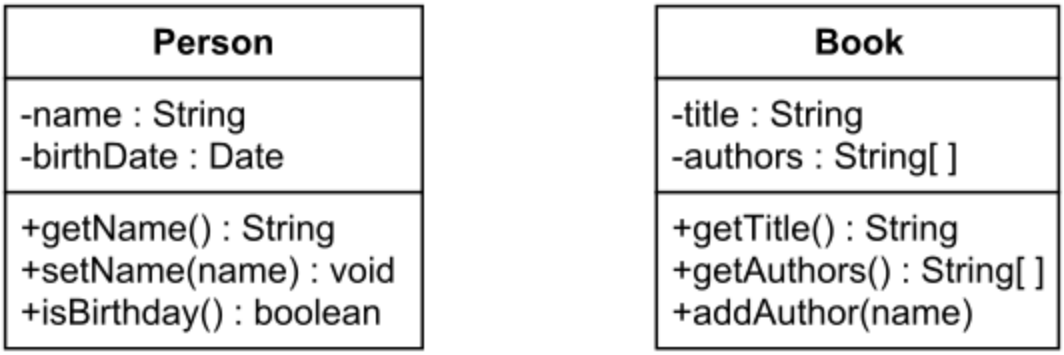
1. Объясните особенности применения функции std::move в C++
2. Приведите примеры реализации обобщенных функций в C++
3. Объясните особенности перегрузки операторов ввода/вывода в C++
4. Опишите способ(ы) реализации динамического полиморфизма в C++
5. Опишите способ(ы) реализации статического полиморфизма в C++",
6. Опишите базовые элементы ввода/вывода в C++
7. Объясните назначение виртуального наследования в C++
8. Объясните назначение виртуальных функций в C++",
9. Объясните механизм и назначение переопределения (override) функций базового класса в классе-потомке в C++
10. Опишите виды отношений между классами (ассоциации, зависимости и т.д.) и их реализацию в C++
11. Объясните механизм множественного наследования в C++
12. Объясните назначение модификаторов наследования (public/private/protected) в C++
13. Объясните особенности перегрузки оператора присваивания в C++
14. Объясните особенности перегрузки двоичных операторов в C++",
15. Объясните механизм и назначение перегрузки операторов в C++
16. Объясните механизм ссылочных значений в C++
17. Опишите основные конструкторы классов в C++ и их назначение
18. Объясните разницу между статическими функциями класса и методами класса в C++
19. Объясните необходимость дружественных функций в C++
20. Объясните механизм синтезирования компилятором C++ функций/методов класса
21. Объясните необходимость методов-геттеров и сеттеров в C++
22. Объясните назначение модификаторов доступа public/private/protected в C++
23. Объясните разницу между функциями и методами в C++",
24. Объясните связь между классами и объектами в C++
25. Объясните механизм перегрузки функций в C++

Вопрос 3.

Для заданного фрагмента кода покажите **диаграмму классов**.

http://www.cs.utsa.edu/~cs3443/uml/uml.html

Диаграммы классов в UML состоят из прямоугольников, обозначающих отдельные классы, и связей между ними, обозначающих зависимости.



Например, классы Person и Book могут быть изображены так: верхняя часть содержит название класса (и дополнительную информацию, как в случае с интерфейсом), средняя часть – скрытые поля, нижняя часть – открытые поля (как правило, методы класса).

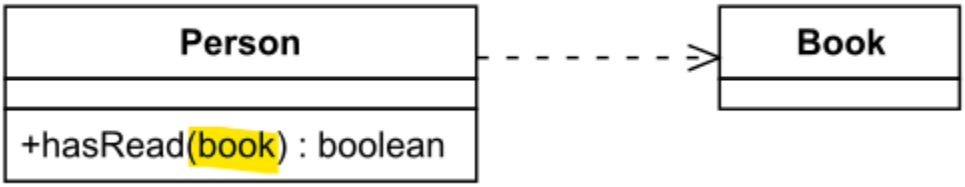
Отношения между классами обычно разделяют на две категории: включение (use) и наследование (inherit).

Включение

В свою очередь каждая категория разделяется на разные формы согласно степени связанности.

Зависимость (dependency)

Объект одного класса использует объект другого класса в своем методе или другим способом, не включая его как поле с данными.

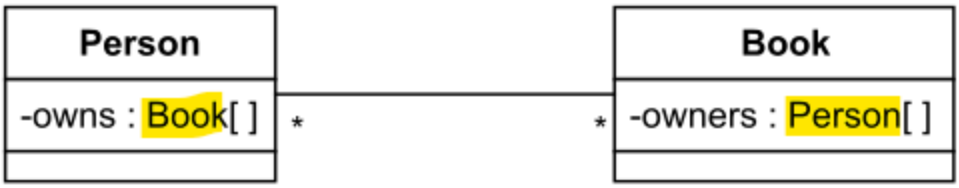


Односторонняя ассоциация (unidirectional association)

Объект одного класса использует объект другого класса в качестве поля с данными.

Двухсторонняя ассоциация (bidirectional association)

Два объекта разных классов используют друг-друга в качестве поля с данными.



Агрегация (aggregation)

Объект одного класса А владеет объектом другого класса Б, при этом объект класса Б является составной частью А.

(на рисунке – двухсторонняя агрегация, но чаще бывает односторонняя)

Композиция (composition)

Является более связанной версией агрегации: объект класса А не просто владеет объектом класса Б, но и контролирует его жизненный цикл (если объект А исчезает, объект Б тоже перестает существовать).

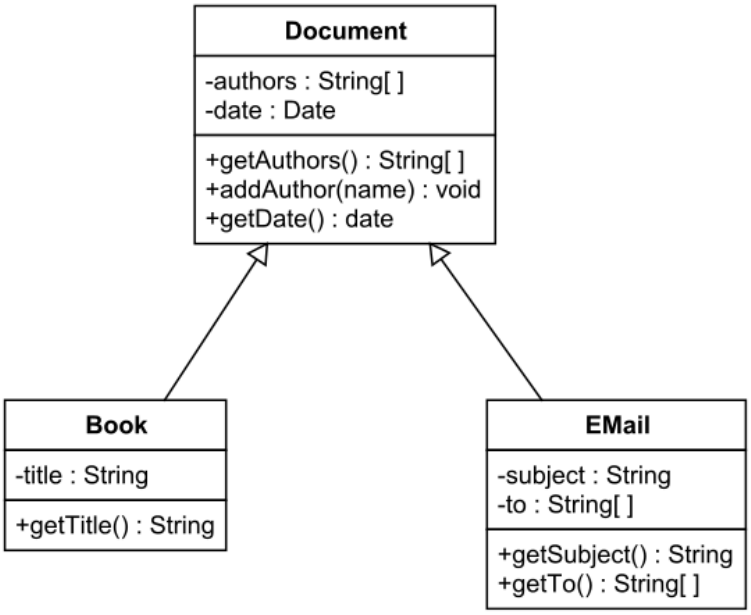
(на рисунке – двухсторонняя композиция, но чаще бывает односторонняя)

Наследование (inheritance)

Разделяется на две формы: обобщение и реализацию.

Обобщение (generalization)

Класс расширяет другой класс, класс-потомок наследует все открытые и защищенные поля своего родительского класса. Потомок может переопределять методы родителя и добавлять новые поля.



Реализация (realization)

Класс-потомок реализует интерфейс – все методы базового абстрактного класса.