**Примеры вопросов к экзамену по дисциплине «Объектно-ориентированное программирование»**

1. Понятие модуля. Принципы модульного программирования. Понятие объекта как динамического модуля.

2. Понятие класса. Понятие метода. Представление метода в виде обычной процедуры. Понятие конструктора и деструктора.

3. Понятие свойства. Методы получения и установки значений свойств. Свойства-массивы (в некоторых языках программирования). Индексаторы (в некоторых языках программирования).

4. Расширение класса путем создания производного класса. Термин «наследование». Существование «прародителя» всех классов. Перекрытие элементов класса в производных классах. Совместимость объектов различных классов. Контроль и преобразование типов. Информация о типе времени выполнения программы.

5. Классы в программных модулях. Атрибуты доступа к элементам объектов. Термин «инкапсуляция».

6. Понятие виртуального метода. Перекрытие виртуального метода в производном классе. Абстрактный виртуальный метод. Механизм вызова виртуального метода. Методы обработки сообщений. Термин «полиморфизм».

7. Понятие ссылки на метод объекта (или делегата – в зависимости от языка программирования). Понятие события. Применение ссылок на методы для расширения объектов.

8. Понятие метакласса (в некоторых языках программирования). Методы, применяемые к классам. Виртуальные конструкторы (в некоторых языках).

9. Ошибки и исключительные ситуации. Классы исключительных ситуаций. Создание и обработка исключительных ситуаций. Защита от утечки ресурсов в случае возникновения исключительных ситуаций. Приемы надежного программирования.

10. Понятие интерфейса. Описание интерфейса. Поддержка интерфейса классом. Механизм подсчета ссылок в интерфейсах. Расширение интерфейса. Глобально-уникальный идентификатор интерфейса. Совместимость интерфейсов и классов. Получение интерфейса через другой интерфейс. Представление интерфейса в памяти. Механизм вызова метода объекта через интерфейс. Применение интерфейса для доступа к объекту динамически-подключаемой библиотеки.

11. Понятие компонента. Понятие визуального программирования. Инструментальные средства визуального компонентного программирования. Современные библиотеки компонентов.

12. Понятие расширяемой программы и расширяемого программирования. Принципы расширяемого программирования. Применение методологии объектно-ориентированного программирования для построения расширяемых программ. Языки программирования Никлауса Вирта: Oberon, Component Pascal.

13. Имитация модульного программирования в языке C++. Понятие пространства имен.

14. Классы в языке C++. Наследование. Конструкторы и деструкторы. Стандартные конструкторы. Создание объектов по значению (на стеке) и по ссылке (в динамической памяти). Операторы new и delete. Размещающий оператор new. Порядок конструирования и разрушения объектов. Вложенные определения классов. «Друзья» класса. Статические члены класса.

15. Множественное наследование. Проблема повторяющихся базовых классов. Замена множественного наследования наследованием от интерфейсов в других языках объектно-ориентированного программирования. Типовой пример применения множественного наследования — «наблюдатель» (observer).

16. Виртуальные методы в языке C++. Недостатки синтаксиса определения и перекрытия виртуальных методов в языке C++. Понятие константного метода. Проблемы, порождаемые наличием константных методов. Операторы приведения типа в языке C++: const\_cast, reinterpret\_cast, static\_cast, dynam-ic\_cast.

17. Ссылки в языке C++. Рекомендации по работе со ссылками. Типичные ошибки при работе со ссылками.

18. Обработка исключительных ситуаций в языке C++. Защита от утечки ресурсов. Имитация оператора try-finally. Понятие автоматического указателя (auto\_ptr). Использование автоматических указателей для защиты от утечки ресурсов.

19. Перегрузка бинарных операторов. Перегрузка унарных операторов. Перегрузка операторов преобразования типа.

20. Шаблоны функций. Перегрузка шаблонов функций. Шаблоны классов. Специализации шаблонов. Стандартная библиотека шаблонов Standard C++ Library. Строки. Итераторы. Потоки ввода-вывода.

21. Деление типов данных на типы-«значения» (value-types) и типы-«ссылки» (reference-types) в языке C#. Автоматическое управление памятью ссылочных данных. Упаковка и разупаковка данных. Типы данных со значением null.

22. Классы в языке C#. Отличие структур (записей) в языке C# от классов. Поля. Методы. Конструкторы и деструкторы. Свойства. Индексаторы. Атрибуты доступа. Сборки. Пространства имен. Частично определяемые классы и их назначение.

23. Делегаты и события в языке C#. Механизм вызова событий.

24. Обобщенные классы в языке C# и их отличие от шаблонов классов в языке C++. Установка ограничений на параметры обобщенных классов. Обощенные делегаты.

25. Понятие итератора в языке C#. Оператор foreach. Оператор yield.

26. Понятие атрибутов в языке C#. Создание пользовательских атрибутов. Анализ атрибутов во время выполнения программы. Понятие рефлексии (reflection) в языке C#.

Сериализация объектов.

27. Типовые приемы программирования (design patterns): «одиночка» (singleton), «заместитель» (proxy/surrogate), «обертка» (decorator/wrapper). Привести примеры применения этих приемов программирования.

28. Типовые приемы программирования (design patterns): «мост» (bridge), «компоновщик» (composite), «адаптер» (adapter/thank). Привести примеры применения этих приемов программирования.

29. Типовые приемы программирования (design patterns): «фабричный метод» (factory method/virtual constructor), «фабрика классов» (factory). Привести примеры применения этих приемов программирования.

30. Типовые приемы программирования (design patterns): «наблюдатель» (observer), «посетитель» (visitor), «медиатор» (mediator). Привести примеры применения этих приемов программирования.

31. Типовые приемы программирования (design patterns): «пул объектов» (object pool), «публикации-подписчики» (publish-subscribe). Привести примеры применения этих приемов программирования.

32. Принципы расширяемого программирования, известные как SOLID (Single responsibility, Open/closed, Liskov substitution, Interface segregation, Dependency inversion) и SOA (Service-Oriented Architecture).

33. Сравнение языков объектно-ориентированного программирования: Delphi (Component Pascal), C++, C#, Java.

34. Предложить свой наиболее простой и функциональный язык расширяемого программирования.