Вопросы к экзамену по дисциплине «Основы алгоритмизации и программирования» для студентов 2-го курса специальности 09.02.07 «Информационные системы и программирование»

1. Понятие объектно-ориентированного программирования:

подразумевающая организацию программного кода, ориентируясь на данные и объекты, а не на функции и логические структуры.  Обычно объекты в подобном коде представляют собой полноценные блоки с данными, которые имеют определенный набор характеристик и возможностей.

1. Понятия класса и объекта в ООП

Классы в **С++** — это абстракция описывающая методы, свойства, ещё не существующих **объектов**. **Объекты** — конкретное представление абстракции, имеющее свои свойства и методы.

1. Принципы ООП

Четыре основных принципа объектно-ориентированного программирования следующие.

Абстракция. Моделирование требуемых атрибутов и взаимодействий сущностей в виде классов для определения абстрактного представления системы

Инкапсуляция. Скрытие внутреннего состояния и функций объекта и предоставление доступа только через открытый набор функций

Наследование. Возможность создания новых абстракций на основе существующих

Полиморфизм. Возможность реализации наследуемых свойств или методов отличающимися способами в рамках множества абстракций

1. Принцип инкапсуляции в ООП

**Инкапсуляция** (от лат. in capsule — в оболочке) — это заключение данных и функционала в оболочку. В объектно-ориентированном программировании в роли оболочки выступают классы: они не только собирают переменные и методы в одном месте, но и защищают их от вмешательства извне (сокрытие). Важно! **Инкапсуляция** не является сокрытием. Сокрытие — это часть **инкапсуляции**.

1. Принцип полиморфизма в ООП

Полиморфизм подтипов — свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта

1. Принцип наследования в ООП

Наследование — принцип повторного использования кода, при котором один объект может приобретать свойства другого

1. Понятие исключения, виды исключений, способы обработки исключений

**Исключение** – это проблема, возникающая во время выполнения программы. **Исключение** **C++** – это реакция на особые обстоятельства, возникающие во время выполнения программы, такие как попытка деления на ноль. **Исключения** позволяют передавать управление из одной части программы в другую. **Обработка** **исключений** **C++** построена на трех ключевых словах: try, catch и throw. throw-программа выдает **исключение**, когда появляется проблема.

1. Понятие оптимизации программного кода, способы оптимизации

Оптимизация — это процесс преобразования части кода в другую функционально эквивалентную часть для улучшения одной или более характеристик кода. Две самые важные характеристики — это скорость работы и размер кода.

Вообще, оптимизация бывает трех типов:

1. Оптимизация уже готового, проверенного и работающего кода.
2. Изначально написание оптимального кода.
3. Просто использование оптимальных конструкций
4. Понятие рефакторинга программного кода, способы рефакторинга

Рефакторинг — это процесс изменения кода, призванный упростить его обслуживание, понимание и расширение, при этом не изменяя его поведение.

 Переименование (Rename)

 Извлечение функции (Extract Function)

 Генерация заглушек чисто виртуальных методов (Implement Pure Virtuals)

 Генерация объявлений/заглушек методов (Create Declaration/Definition)

 Перемещение объявлений функций (Move Function Definition)

 Преобразование в Raw-String (Convert to Raw-String Literal)

1. Понятие тестирования программного обеспечения. Виды тестирования

**Тестирование** **программного** **обеспечения**-это исследование, проводимое с целью предоставления заинтересованным сторонам информации о качестве тестируемого **программного** продукта или услуги. **Тестирование** **программного** **обеспечения** также может обеспечить объективный, независимый взгляд на **программное** **обеспечение**, позволяющий бизнесу оценить и понять риски внедрения **программного** **обеспечения**.

Обычно тестирование подразделяется на три категории:

1.Функциональное тестирование Модульное (компонентное) Интеграционное 2.Системное Регрессионное Приемочное Смоук

3.Тестирование производительности Тестирование отказоустойчивости 4.Нагрузочное Объемное Тестирование масштабируемости

Обслуживание (регресс и обслуживание) Регрессионное Тестирование технического обслуживания

1. Понятие отладки программного обеспечения. Способы отладки

**Отладка** - это процесс поиска и удаления ошибок из **программное** **обеспечение**. Ошибки возникают в программах, когда строка кода или инструкция конфликтует с другими элементами кода.

1. Понятие алгоритма. Свойства алгоритмов

**Алгоритм** — это всякая система вычислений, выполняемых по строго определѐнным правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи. Общее в этих определениях то, что **алгоритм** — это предписание. Единого «истинного» определения **понятия** «**алгоритм**» нет, так как **понятие** **алгоритма** является фундаментальным и не может быть выражено через другие, поэтому его следует рассматривать как неопределяемое.



1. Способы записи алгоритма
2. Базовые алгоритмические структуры, их описание
3. Понятие интегрированной среды разработки (ИСР). Примеры ИСР. Состав ИСР
4. Язык программирования C#. Возникновение, основное назначение, особенности
5. Понятие интерфейса. Виды интерфейсов
6. Принципы разработки интерфейса
7. Понятие жизненного цикла программного обеспечения. Модели жизненного цикла
8. Этапы разработки приложений
9. Понятие строки в C#. Методы строк
10. Понятие массива в C#. Одномерные и многомерные массивы, принципы работы с массивами
11. Понятие множества в C#. Класс HashSet – свойства, методы
12. Понятие списка в C#. Класс List<T> - свойства, методы
13. Понятие файла в C#. Классы StreamWriter и StreamReader – свойства, методы
14. Принцип модульности при разработке приложений. Добавление класса в проект и его использование
15. Создание и использование библиотеки класса