Нулевая группа вопросов

1. Для решения Лабораторной работы номер 2, связанной с созданием потоков, вы можете использовать следующие функции WinAPI:
   - CreateThread: Для создания нового потока.
   - WaitForSingleObject: Для ожидания завершения выполнения потока.
   - CloseHandle: Для закрытия дескриптора потока.
   - GetCurrentThread: Для получения дескриптора текущего потока.
   - ExitThread: Для завершения текущего потока.
2. Процесс в операционной системе Windows - это независимая и изолированная единица выполнения программы. Он включает в себя исполняемый код, данные, стек и другие ресурсы, необходимые для выполнения программы. Процессы в Windows имеют свой собственный адресное пространство и работают в изоляции друг от друга. ОС управляет процессами, назначает им ресурсы, контролирует их выполнение и обеспечивает безопасность.
3. Критическая секция - это участок кода в многозадачной программе, который должен выполняться без прерываний другими потоками для предотвращения конфликтов и гонок данных. Критическая секция обычно защищает общие ресурсы, такие как переменные, от доступа нескольких потоков одновременно. В Windows, критические секции могут быть созданы с использованием функций InitializeCriticalSection, EnterCriticalSection и LeaveCriticalSection.
4. Семафор - это синхронизационный объект, который используется для управления доступом к общим ресурсам в многозадачных приложениях. Семафор может иметь счетчик, который указывает на количество доступных ресурсов. Потоки могут уменьшать счетчик семафора при запросе ресурса и увеличивать его при освобождении ресурса. Это позволяет управлять доступом к ресурсам. В Windows, семафоры можно создавать с использованием функции CreateSemaphore.
5. Сравнительный анализ стандарта C++98 с и без применения библиотеки Boost (в контексте лабораторных):

C++98 является старым стандартом C++, который не включает в себя многие современные функции и возможности, такие как многопоточность, сетевое программирование, и др. В лабораторных работах, требующих такие функциональности, вам может потребоваться написать собственный код или использовать сторонние библиотеки.
Boost - это сторонняя библиотека для C++, предоставляющая множество расширенных функций и инструментов, включая поддержку многопоточности, парсинг XML, работу с сетью, и многое другое. Использование Boost может значительно упростить разработку, позволяя использовать готовые компоненты и функции для решения задач.

Вторая группа вопросов

1. Процедурная декомпозиция - это методология разработки программ, которая основывается на разделении программы на небольшие, самодостаточные процедуры или функции. Каждая процедура выполняет конкретную задачу и может вызываться из других частей программы. Это помогает упростить код, делает его более читаемым и поддерживаемым, и позволяет повторно использовать процедуры.
2. Динамический полиморфизм - это концепция объектно-ориентированного программирования, которая позволяет объектам разных классов реагировать на одинаковые сообщения (методы) специфичным для самих объектов образом. В контексте языка программирования, динамический полиморфизм обычно реализуется через механизм виртуальных функций. Когда объект вызывает виртуальную функцию, фактический код, который будет выполнен, определяется во время выполнения программы на основе типа объекта, что позволяет объектам разных классов иметь различную реализацию одного и того же метода.
3. Инкапсуляция - это один из принципов объектно-ориентированного программирования, который представляет собой механизм, позволяющий объединить данные (переменные) и методы (функции), работающие с этими данными, в единый объект. Основная идея инкапсуляции заключается в том, чтобы скрыть детали реализации объекта и предоставить только интерфейс (публичные методы), через который можно взаимодействовать с объектом. Это позволяет обеспечить

безопасность данных и упростить их использование, а также обеспечивает модульность и упрощает разработку и поддержку кода.

Третья группа вопросов

1. Creation Method:

Паттерн Creation Method относится к порождающим паттернам проектирования и используется для инкапсуляции создания объектов. Он предоставляет абстрактный метод (интерфейс или базовый класс), в котором определен метод-фабрика для создания экземпляров конкретных классов. Этот метод делегирует создание объектов на подклассы, что позволяет скрыть детали создания объектов от клиентского кода.

Пример использования:
Допустим, что есть иерархия классов для различных форм, таких как круг, квадрат, и треугольник. Вместо того, чтобы клиентский код создавал объекты этих классов напрямую, вы можете использовать паттерн Creation Method. Вы создадите базовый класс `Shape` с абстрактным методом `CreateShape()`, а затем подклассы (например, `Circle`, `Square`, `Triangle`) реализуют этот метод, чтобы создавать свои экземпляры. Теперь клиентский код может вызвать `CreateShape()` без знания конкретных классов и их создания.

2. Visitor:

Паттерн Visitor позволяет инкапсулировать операции, выполняемые над объектами, внешне, в отдельных классах. Он предоставляет способ добавления новых операций, не изменяя классы объектов, которые эти операции могут выполнять. В паттерне Visitor создается класс "посетитель" (Visitor), который определяет методы для каждого типа объектов, которые можно посетить. Классы объектов имеют метод, принимающий посетителя в качестве аргумента и вызывающий соответствующий метод посетителя.

Пример использования:
Рассмотрим систему обработки документов. У нас есть классы `TextDocument`, `ImageDocument`, и `AudioDocument`, представляющие разные типы документов. Вместо того, чтобы добавлять новые функции (например, печать, конвертацию и т.д.) в каждый класс документа, мы можем использовать паттерн Visitor. Мы создаем класс `DocumentVisitor`, содержащий методы для обработки каждого типа документа. Каждый класс документа имеет метод `Accept`, который принимает посетителя и вызывает соответствующий метод посетителя. Это позволяет добавить новые функции для обработки документов, не изменяя сами классы документов.