https://rsdn.org/article/com/introcom.xml

1. **Что такое COM? COM-программирование?**

COM – модель программного обеспечения. Модель разработана Microsoft.

**COM-программирование**: разработка программного обеспечения, имеющего модель COM

- **COM (Component Object Model**) - это технология, разработанная Microsoft, для создания и использования компонентов (или объектов) в программных приложениях.

**COM-программирование** - это методология разработки программ, основанная на использовании COM-компонентов для создания расширяемых и переиспользуемых приложений.

1. Что такое COM-объект(компонент)? CLSID?

**COM-объект:** специализированныйобъект времени исполнения (экземпляр)**.**

**CLSID (Class Identifier)** - это уникальный идентификатор, который идентифицирует конкретный COM-класс (или объект) в системе.

1. Что такое GUID? Где применяется GUID? Размер GUID-идентификатора?

GUID (Globally Unique Identifier) - это уникальный идентификатор, используемый для уникальной идентификации ресурсов, объектов и компонентов в системе. GUID применяются в COM для идентификации классов и интерфейсов. Размер GUID-идентификатора составляет 128 бит (16 байт).

1. Какие типы COM-контейнеров бывают?

Существуют два основных типа COM-контейнеров:

- Локальные COM-контейнеры: Эти контейнеры работают на том же компьютере, где выполняется клиентское приложение.

- Удаленные COM-контейнеры: Эти контейнеры находятся на удаленной машине и позволяют клиентским приложениям взаимодействовать с удаленными COM-объектами.

1. Что является клиентом и сервером в COM?

**COM-сервер:** программный модуль, реализующий COM-объект

**COM-клиент**: программный модуль, создающий COM-объект и использующий его методы.

1. Поясните понятия «однокомпонентный» и «многокомпонентный» COM-сервер.

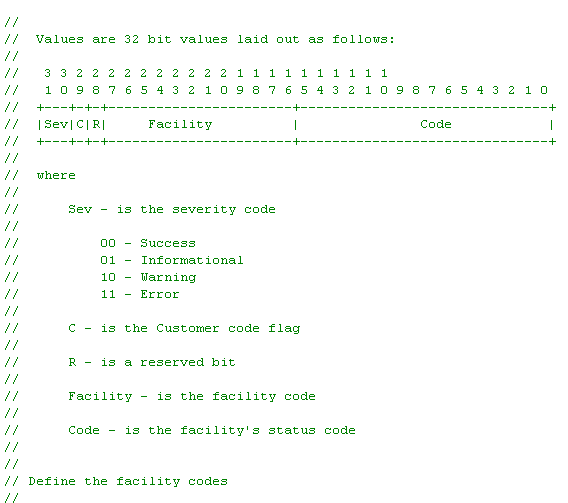
**COM-сервер:** может быть **однокомпонентным** (реализующим один тип объектов) или **многокомпонентным** (реализующим несколько типов объектов

1. **Поясните типы COM-серверов: CLSCTX\_INPROC\_SERVER, CLSCTX\_LOCAL\_SERVER, СLSCTX\_REMOTE\_SERVER**.

**COM-сервер может иметь тип**: **CLSCTX\_INPROC\_SERVER** (DLL внутрипроцессный сервер); **CLSCTX\_LOCAL\_SERVER** (EXE-сервер за границами процесса, но та том же компьютере); **СLSCTX\_REMOTE\_SERVER** (EXE-сервер на удаленном компьютере

1. **Как называется имя библиотеки, обеспечивающей работу COM-приложений**.

**COM:** все константы и прототипы функций необходимые для поддержки COM-программирования становятся доступными с помощью **#include <objbase.h>**

**9. Поясните назначение типа и структуру HRESULT**. ****

HRESULT - это стандартный тип данных в Windows для представления ошибок и статусов выполнения функций. Он используется для передачи информации о результате выполнения операции между компонентами

Он может быть использован в качестве альтернативы булевому результату прохождения/провала. Код HRESULT состоит из следующих полей 1:

Severity (серьезность): указывает, успешно ли завершилась функция.

флаг кода клиента, зарезервированный вит

Facility (область): идентифицирует часть системы, к которой относится статус.

Code (код): идентифицирует конкретное состояние в контексте области.

1. **Что такое COM-интерфейс?**

COM-интерфейс - это набор методов, предоставляемых COM-объектом, которые могут быть вызваны клиентским приложением. Интерфейсы определяют контракты для взаимодействия между клиентом и сервером.

1. Интерфейс представляет собой набор абстрактных функций, имеющий аналогично COM-компонентам свой GUID-идентификатор. каждый интерфейс включает один или несколько методов.

2. Интерфейс: каждый интерфейс имеет идентификатор, который имеет тип GUID (как и идентификатор объекта).

**11. Чем характеризуется COM-интерфейс?**

каждый интерфейс имеет идентификатор, который имеет тип GUID (как и идентификатор объекта).

- COM-интерфейс характеризуется своими методами, которые определяют функциональность, доступную клиенту. Каждый метод имеет уникальное имя и сигнатуру (параметры и типы возвращаемых значений).

**12. Что значит «стандартный» COM-интерфейс?**

Стандартные COM-интерфейсы - это интерфейсы, определенные Microsoft или другими стандартными организациями, которые широко используются в COM для общих задач, таких как управление объектами и доступ к объектам через систему COM.

За стандартными интерфейсами закреплены предопределенные GUID-идентификаторы. Важнейшим среди стандартных интерфейсов является интерфейс IUnknown. Все остальные интерфейсы являются производными (наследуют все методы) от IUnknown. Каждый компонент должен поддерживать (часто говорят «реализовывать») как минимум стандартный ин терфейс IUnknow

1. **Назовите два стандартных COM-интерфейса.**

IUnknown, IClassFactory

Этот интерфейс предоставляет базовые методы для управления объектом, включая управление ссылками и получение доступа к другим интерфейсам объекта.

предоставляет методы для создания экземпляров COM-объектов. Когда клиент запрашивает создание объекта, **IClassFactory** отвечает за создание и возврат экземпляра объекта.

1. **Перечислите методы интерфейса IUnknown и поясните их назначение.**
2. **AddRef (Add Reference)**: Этот метод увеличивает счетчик ссылок объекта. Каждый раз, когда клиент запрашивает указатель на интерфейс объекта (например, вызывая **QueryInterface**), клиент вызывает **AddRef** для увеличения счетчика ссылок. Этот метод гарантирует, что объект не будет уничтожен, пока на него существуют активные ссылки.
3. **Release (Release Reference)**: Этот метод уменьшает счетчик ссылок объекта. Каждый раз, когда клиент больше не нуждается в объекте или завершает работу с ним, он вызывает **Release** для уменьшения счетчика. Если счетчик достигнет нуля, объект освобождается и уничтожается. Это позволяет освободить ресурсы, занимаемые объектом, когда он больше не используется.
4. **QueryInterface**: Этот метод используется для запроса указателя на другие интерфейсы, поддерживаемые объектом. Клиент передает идентификатор интерфейса (IID) в метод **QueryInterface**, и если объект поддерживает этот интерфейс, метод возвращает указатель на него. Это позволяет клиентам работать с различными интерфейсами объекта, не зная его конкретного типа.
5. Что такое «фабрика классов» и для чего она нужна?

"Фабрика классов" (Class Factory) - это объект, который используется в технологии Component Object Model (COM) для создания экземпляров COM-объектов. Она представляет собой реализацию интерфейса **IClassFactory**. Фабрика классов выполняет следующие основные функции:

1. **Динамическое создание объектов**: Фабрика классов позволяет клиентскому коду динамически создавать экземпляры COM-объектов без знания их конкретных типов и реализаций. Клиенты запрашивают экземпляры объектов у фабрики классов, и фабрика создает объекты соответствующего типа.
2. **Управление временем жизни объектов**: Фабрика классов обычно реализует методы **AddRef** и **Release** для управления временем жизни созданных объектов. Это важно, потому что при создании объектов клиент получает указатель на интерфейс объекта, и это приводит к учету ссылок, которые должны быть корректно управляемыми.
3. **Поддержка многопоточности**: Фабрика классов может обеспечивать создание объектов с учетом требований многопоточности. Например, она может гарантировать, что созданные объекты будут потокобезопасными.
4. **Управление блокировкой сервера**: Фабрика классов может управлять блокировкой (загрузкой в память) COM-сервера. Это важно для оптимизации использования ресурсов, поскольку фабрика может решать, следует ли загружать или выгружать COM-сервер в зависимости от активных клиентов.
5. Перечислите методы интерфейса IClassFactory и поясните их назначение.

**CreateInstance**: Этот метод создает новый экземпляр COM-объекта, реализуемого фабрикой классов. Метод принимает два параметра: **pUnkOuter** (указатель на аггрегирование, который используется для создания составных объектов, и обычно передается как **NULL**) и **riid** (идентификатор интерфейса, который клиент хочет получить).

**LockServer**: Этот метод используется для управления блокировкой (загрузкой в память) COM-сервера. Метод принимает булево значение (флаг) **fLock**. Если **fLock** равно **TRUE**, это означает, что COM-сервер должен быть загружен и оставаться в памяти. Если **fLock** равно **FALSE**, это означает, что COM-сервер может быть выгружен из памяти, если на него больше нет активных ссылок.

1. Что такое «счетчик ссылок на интерфейсы»? Для чего он нужен? Каким образом и когда этот счетчик увеличивается и уменьшается?

Счетчик ссылок на интерфейсы представляет собой механизм управления временем жизни COM-объектов. Позволяет автоматически управлять временем жизни объекта, избегая утечек памяти и обеспечивая эффективное использование ресурсов. отслеживает количество активных экземпляров (инстансов) компонентов в рамках данного COM-сервера. Этот счетчик увеличивается и уменьшается в следующих местах:

**AddRef** увеличивает счетчик, а **Release** уменьшает его. Когда счетчик ссылок достигает нуля, фабрика классов уничтожается

1. **Какое соглашение о вызове и возврате должен обеспечивать метод COM-объекта? Какие методы являются исключением?**

\_\_stdcall

Метод COM-объекта может быть исключением из этого правила, если он соответствует одному из следующих условий:

* Метод возвращает указатель на COM-объект. В этом случае метод должен возвращать значение void и использовать соглашение о вызове cdecl.
* Метод вызывает другой метод COM-объекта. В этом случае метод должен возвращать значение HRESULT и использовать соглашение о вызове stdcall.
* Метод вызывает функцию Win32 API. В этом случае метод должен возвращать значение HRESULT и использовать соглашение о вызове stdcall.

Методы COM-объекта должны следовать соглашению о вызове и возврате "Стандартный вызов" (StdCall). Обычно используется для обеспечения корректного взаимодействия COM-объектов с клиентским кодом и операционной системой. Некоторые методы COM могут использовать соглашение о вызове **Cdecl** и генерировать исключения C++.

1. Что должен «знать» COM-клиент, чтобы использовать COM-объект?

**CLSID:** Уникальный идентификатор класса объекта.

**Интерфейсы:** Идентификаторы интерфейсов и их методы, которые клиент может вызывать.

**Правила управления временем жизни:** Правила подсчета ссылок (**AddRef** и **Release**) для избежания утечек памяти.

**Библиотека типов:** Если доступна, для получения информации о структуре объекта.

**Соглашение о вызове и возврате:** Необходимо для корректного взаимодействия.

**Обработка ошибок:** Знание о значениях **HRESULT** и правилах обработки ошибок.

**Информация из документации:** Дополнительные детали и рекомендации от создателя объекта.

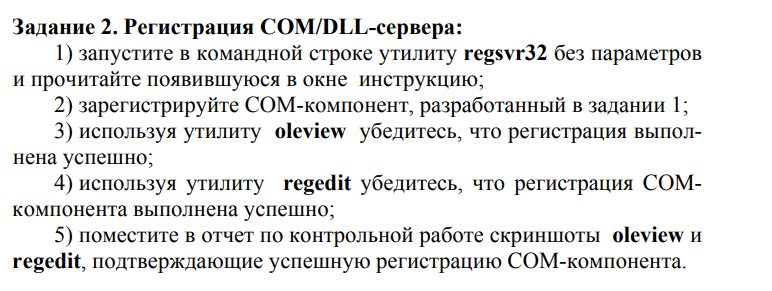
Чтобы COM-клиент мог использовать COM-объект:

1. Знание интерфейса COM-объекта и его CLSID.
2. Понимание основных концепций COM: жизненный цикл, подсчет ссылок, RPC.
3. Доступ к библиотеке типов для описания объекта.
4. Умение создать объект с использованием функций, таких как **CoCreateInstance**.
5. Обработка ошибок и исключений.
6. Работа с интерфейсами: получение, вызов методов, управление ссылками.
7. Понимание правил высвобождения ресурсов после завершения работы с объектом.

CreanteInstace. Функция принимает пять параметров, причем первые четыре являются входными, последний – выходной. Первый параметр (CLSID\_CA) передает значение GUID-идентификатора компонента, а третий – тип сервера (в примере INPROC-сервер). Назначение второго параметра (в примере его значение равно NULL) будет пояснено в четвертой практической работе. В четвертом параметре передается значение GUID-интерфейса (в примере передается идентификатор интерфейса IUnknown), указатель на который возвращается с помощью последнего, пятого параметра.

1. Объясните в чем заключается процесс регистрации COM-объекта?

Обычно процесс регистрации выполняется с использованием инструмента **regsvr32**, который вызывает функцию **DllRegisterServer** объекта.



Процесс регистрации COM-объекта включает создание объекта, добавление записей в реестр (с CLSID и путями), использование информационных файлов, установку библиотеки типов, и возможно, добавление объектов в COM-каталог. Регистрация позволяет системе обнаруживать и взаимодействовать с объектом.

Начало формы

1. Поясните назначение утилиты regsvr32 и принцип ее работы.

**Назначение**: **regsvr32** предназначена для регистрации и дерегистрации библиотек (DLL) и контейнеров классов COM (Component Object Model) в операционной системе Windows.

**Принцип работы**:

* 1. **Регистрация DLL**: Вызов **regsvr32** с аргументом, указывающим на DLL-файл, приводит к вызову функции **DllRegisterServer** внутри этой DLL. Эта функция обновляет записи в реестре операционной системы, регистрируя классы COM и другую информацию о DLL, что позволяет клиентам COM обращаться к объектам, предоставляемым этой DLL.
  2. **Дерегистрация DLL**: Если вызвать **regsvr32** с параметром **/u**, то это приведет к вызову функции **DllUnregisterServer** внутри DLL, и соответствующие записи в реестре будут удалены. Это дерегистрирует классы COM и другую информацию, связанную с DLL.
  3. **regsvr32** самостоятельно не занимается созданием или удалением DLL-файлов, она просто обновляет информацию в реестре.

1. Поясните назначение утилиты regedit.

**Назначение**: **regedit** представляет собой Реестр Windows - инструмент для работы с реестром операционной системы. Реестр Windows - это центральное хранилище конфигурации и настроек Windows, включая информацию о программных и аппаратных ресурсах, настройках операционной системы и установленных приложениях.

**Принцип работы**:

* 1. **regedit** предоставляет графический интерфейс для просмотра, редактирования и управления содержимым реестра. Пользователь может переходить по разделам, создавать, изменять и удалять ключи и значения, а также выполнять ряд других операций.

1. Перечислите пять функций, которые экспортируются COM/DLL-контейнером. Поясните назначение этих функций.

**DllGetClassObject**: **Эта функция вызывается для получения указателя на фабрику классов**. Она используется для создания экземпляров COM-объектов данного класса. Контейнер возвращает указатель на объект фабрики классов, который может быть использован клиентами для создания объектов данного класса.

**DllCanUnloadNow**: **Эта функция позволяет определить, можно ли выгрузить COM-сервер из памяти.** Если нет активных объектов или блокировок сервера, функция возвращает S\_OK, что указывает на то, что сервер может быть безопасно выгружен.

**DllRegisterServer**: **Эта функция регистрирует информацию о COM-сервере в реестре операционной системы.** Это включает в себя информацию о CLSID (идентификаторе класса), ProgID (идентификаторе программы), путях к файлам, интерфейсах и других параметрах, необходимых для создания и использования COM-объектов.

**DllUnregisterServer**: **Эта функция удаляет информацию о COM-сервере из реестра операционной системы.** Она выполняет обратную операцию по отношению к **DllRegisterServer**. Это может потребоваться при деинсталляции или обновлении COM-сервера.

**DllInstall**: **Эта функция используется для установки COM-сервера.** Она может выполнять различные операции в зависимости от потребностей установки, включая регистрацию, де-регистрацию и другие задачи. Эта функция не является обязательной и может отсутствовать в контейнере.

1. Назовите функцию COM-контейнера, которая вызывается OLE32 для получения указатель на фабрику классов.

**DllGetClassObject**

1. Назовите функцию фабрики классов, в которой создается объект компонента.

Функция фабрики классов, в которой создается объект компонента, называется **CreateInstance**

1. Поясните назначение «счетчика экземпляров компонент». Где этот счетчик увеличивается и где уменьшается?

"Счетчик экземпляров компонент" - это общий счетчик, который отслеживает количество активных экземпляров (инстансов) компонентов в рамках данного COM-сервера. Этот счетчик увеличивается и уменьшается в следующих местах:

**AddRef** увеличивает счетчик, а **Release** уменьшает его. Когда счетчик ссылок достигает нуля, фабрика классов уничтожается.

1. Назовите условие, при котором объект компонента удаляется.

метод Release уничтожает экземпляр компонента, если обнаруживает, что после уменьшения счетчика ссылок на 1, его значение стало равным 0.

1. Объясните на механизм блокировки COM-сервера (функция LockServer фабрики классов).

Механизм блокировки COM-сервера через функцию **LockServer** в интерфейсе **IClassFactory** предназначен для управления временем жизни сервера. При создании объекта клиентом счетчик блокировки увеличивается, предотвращая выгрузку сервера из памяти. При завершении использования клиентом счетчик уменьшается, и если становится равным нулю, сервер может быть выгружен. Это важно для серверов, предоставляющих объекты, используемые в разных процессах (out-of-process).

1. **LockServer(true)**: Когда клиент запрашивает фабрику классов о создании экземпляра COM-объекта и включает флаг блокировки (**true**), это увеличивает счетчик блокировки сервера. Это означает, что сервер не будет выгружен, даже если все клиенты завершат работу с созданными объектами. Сервер останется активным до тех пор, пока счетчик блокировки не станет равен нулю.
2. **LockServer(false)**: Когда клиент завершает работу с COM-объектами и передает **false** флаг блокировки в функцию **LockServer**, это уменьшает счетчик блокировки. Если счетчик блокировки достигнет нуля и при этом счетчик ссылок на все созданные объекты также равен нулю, то COM-сервер будет выгружен.

Таким образом, блокировка COM-сервера позволяет сохранить сервер в работе, пока есть активные клиенты, которые могут создавать и использовать COM-объекты. Это важно, чтобы не перезапускать сервер каждый раз, когда клиент заканчивает работу. Благодаря этой функциональности COM-серверы могут эффективно обслуживать несколько клиентов в течение длительного времени.