MINISTERUL EDUCAȚIEI, CULTURII

ȘI CERCETĂRII AL REPUBLICII MOLDOVA

UNIVERSITATEA DE STAT „ALECU RUSSO” DIN BĂLȚI

FACULTATEA DE ŞTIINŢE REALE, ECONOMICE ŞI ALE MEDIULUI

CATEDRA DE MATEMATICĂ ȘI INFORMATICĂ

Реферат

Тема: Инкапсуляция

Научный руководитель:

Г-н Кабак Евгений Валерьевич

Выполнено студентом:

Гр. IS41R

Алексеева Инна

Бельцы – 2021

**«Инкапсуляция** (**encapsulation**) - **это** механизм, который объединяет данные и код, манипулирующий этими данными, а также защищает и то, и другое от внешнего вмешательства или неправильного использования ... Другими **словами**, объект - **это** то, что поддерживает **инкапсуляцию»**. (Википедия. Доступно в интернете: <https://www.google.com/search?q=+%D0%B8%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F+%D1%8D%D1%82%D0%BE+%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%8B%D0%BC%D0%B8+%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D0%BC%D0%B8&ei=qE-UYJ-KCYSvrgS3sq-gBA&oq=+%D0%B8%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F+%D1%8D%D1%82%D0%BE+%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%8B%D0%BC%D0%B8+%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D0%BC%D0%B8&gs_lcp=Cgdnd3Mtd2l6EAMyBggAEBYQHjoHCAAQRxCwAzoECAAQHlDzP1jSRGDBZWgBcAJ4AIABrwGIAYEHkgEDMC43mAEAoAEBqgEHZ3dzLXdpesgBCMABAQ&sclient=gws-wiz&ved=0ahUKEwif_t3j8LXwAhWEl4sKHTfZC0QQ4dUDCA4&uact=5> )

**Введение в объектно-ориентированное программирование.**

Объектно-ориентированное программирование - это немного более автоматизированный способ программирования. Объектно-ориентированные программы - это не просто процедурные программы, переведенные в новый синтаксис. Они должны основываться на новой философии дизайна. Они требуют новой стратегии программирования, которую часто трудно освоить.

Основная идея ООП: программа состоит из группы объектов, часто связанных друг с другом. В C ++ объекты описываются с использованием нового типа данных класса. Класс включает в себя набор переменных (данных) и операций (методов или функций-членов), которые действуют на эти переменные. Полученными объектами можно управлять с помощью сообщений. В ООП объекты включают не только данные (элементы данных), но также методы (функции-члены) воздействия на эти данные. Эти две части объединены в функциональную единицу программы. Другими словами, объекты содержат данные и методы для работы с этими данными. Ниже приведены три основных преимущества объектно-ориентированных программ по сравнению с аналогичными программами, работающими сверху вниз.

Сопровождение программы. Программы легче читать и понимать, ООП позволяет управлять сложностью программы, оставляя для программиста видимыми только основные детали.

Модификация программы (добавление или удаление функций). Можно часто вносить в программу дополнения или исключения, например, при работе с базой данных, просто добавляя и исключая объекты. Новые объекты могут наследовать все свойства базовых объектов, вам просто нужно добавить или удалить различные свойства. Повторное использование. Вы можете сохранить хорошо спроектированный объект в наборе полезных программ, а затем вставить его в новую программу с небольшими изменениями или без них.

**Основные термины и положения ООП. Инкапсуляция данных.**

Этот термин включает логическую связь данных с определенной операцией. Это также означает, что они доступны не глобально для всей программы, а локально - доступны только для небольшой ее части. Инкапсуляция также автоматически подразумевает защиту данных. Для этого и предназначена структура классов в C ++. В классе управление функциональными деталями объекта осуществляется с помощью спецификаторов private, public, protected.

Иерархия классов. В общем, можно рассматривать иерархию классов как родословную в генеалогическом дереве, где класс C ++ является шаблоном для создания классов-потомков. Объекты, полученные из описания класса, называются экземплярами этого класса. Можно создать иерархию классов с родительским классом и несколькими классами-потомками. Основой для этого являются производные классы.

**Наследование.**

Наследование ООП позволяет классу получать свойства другого класса объектов. Родительский класс служит шаблоном для производного класса; этот шаблон можно изменять разными способами. Наследование важно, поскольку оно позволяет повторно использовать определение класса без внесения серьезных изменений в код.

**Полиморфизм.**

Полиморфизм - построен на описанной выше концепции наследования. Программа отправляет одно и то же сообщение как объекту родительского класса, так и всем объектам производных классов. И родительский класс, и классы-потомки ответят на сообщение соответствующим образом. Полиморфизм дает возможность дополнять уже существующие части программы.

Виртуальные функции. Виртуальные функции определены в родительском классе, а в производных классах эти функции переопределяются, и для них создаются новые реализации. При работе с виртуальными функциями сообщения передаются как указатели, указывающие на объект, вместо того, чтобы быть переданными непосредственно объекту. Виртуальные функции используют таблицу для адресной информации. Эта таблица инициализируется во время выполнения с помощью конструктора. Конструктор вызывается каждый раз, когда создается объект его класса. Задача конструктора в этом случае - связать виртуальную функцию с таблицей адресной информации. Во время компиляции адрес виртуальной функции неизвестен; вместо этого ему назначается позиция в адресной таблице.

ООП полностью принадлежит миру C ++, поскольку у C нет основного ядра - класса абстрактных типов данных. Следовательно, переписать процедурно-ориентированную программу как объектно-ориентированную намного сложнее, чем просто заменить одно ключевое слово другим.

ООП - это метод программирования, позволяющий рассматривать основные идеи как несколько объектов. Используя объекты, представить задачи, которые необходимо выполнить, их взаимодействие и любые заданные условия, которые необходимо выполнить. Структура данных часто составляет основу объектов; таким образом, в C или C ++ тип структуры может образовывать примитивный объект. Связь с объектом можно организовать с помощью сообщений. Использование сообщений похоже на вызов функций в процедурной программе.

Когда объект получает сообщение, вступают в действие методы, содержащиеся в объекте. Методы (их иногда используют функции) аналогичные функции процедурно-ориентированного программирования. Тем не менее метод является частью объекта, как было бы в процедурном аналоге.

**С ++ -язык предметно-ориентированного программирования.**

Объектно-ориентированное программирование - это новый способ подхода к программированию. Такое программирование, взяло лучшие черты структурного программирования. Наиболее важное понятие языков объектно-ориентированного программирования –это понятие объекта (объекта). Объект - это логическая единица, которая содержит данные и правила (методы) обработки этих данных. В языке С ++ в таких правилах обработки выступают функции, т. е. объект в Borland C ++ объединяет в себе данные и функции, обрабатывающие эти данные. Одним из самых понятий языка С ++ является главным понятием класса. На языке С ++ для того, чтобы определить объект, надо сначала определить его форму с помощью ключевого слова Ближайшей аналогией класса является структура. Память выделяется объекту только тогда, когда класс используется для его создания. Любой объект языка С ++ имеет одинаковые атрибуты и функциональность с другими объектами того же класса. За создание своих классов и поведение этих классов несет ответственность сам программист. Работая в некоторой среде, программист получает доступ к обширной библиотеке стандартных классов. Обычно объект находится в некотором уникальном состоянии, установленном текущими значениями его атрибутов. Функциональность объектного класса определяется возможными операциями над экземплярами этого класса.

**Структуры.**

Структура-группа данных различных типов и (или) назначения, которые представляют собой единый информационный элемент.

{

КООРД \* t; // точка

int PCount; //

};

Каждый элемент данных, называемый полем, имеет различное назначение. Одно поле содержит точки, другое расстояние между собой.

Мы в объекте всегда также можно хранить группы разнотипных данных, то они тоже содержат структуры. Отличные структуры от класса в том, что в структуре нет закрытых и защищенных членов. Все члены структуры открыты.

Объявление структуры аналогично объявлению класса, но вместо ключевого слова класс ставится слово Структура:

struct идентификатор

{

Объявление данных или объектов

};

Объявлений не подразумевает создание объекта или конструкции. Объявление-это просто описание будущего объекта. Чтобы использовать объект класса или переменный определенного типа их сначала необходимо объявить в качестве объекта этого класса или переменного типа. Аналогично для использования структурной схемы необходимо объявить переменную структуры.

**ИНКАПСУЛЯЦИЯ ДАННЫХ**

Инкапсуляция данных - это процесс, в котором некоторая дополнительная информация добавляется к элементу данных, чтобы добавить к нему некоторые функции. К примеру: в нашей сети мы используем модель OSI или TCP / IP, и в этих моделях передача данных происходит через разные уровни. Инкапсуляция данных добавляет к данным информацию протокола, так что передача данных может происходить должным образом. Эта информация может быть добавлена ​​в верхний колонтитул (верхний колонтитул) или в конец (нижний колонтитул или трейлер) данных.

Данные инкапсулируются на стороне отправителя, от уровня приложения до физического уровня. Каждый уровень берет инкапсулированные данные из предыдущего слоя и добавляет некоторую дополнительную информацию, чтобы инкапсулировать их и некоторые другие функции с данными. Эти функции могут включать согласованность данных, мониторинг и обнаружение ошибок, управление потоком, управление перегрузкой, информацию о маршрутизации и т. Д.

**ДЕИНКАПСУЛЯЦИЯ ДАННЫХ**

Деинкапсуляция данных - это процесс, обратный инкапсуляции данных. Инкапсулированная информация удаляется из полученных данных для получения исходных данных. Этот процесс происходит на стороне получателя. Данные деинкапсулируются на том же уровне на стороне получателя, что и инкапсулированный уровень на стороне отправителя. Добавленная информация верхнего и нижнего колонтитула удаляется из данных в этом процессе.

Данные инкапсулируются на каждом уровне на стороне отправителя и деинкапсулируются на том же уровне на стороне получателя модели OSI или TCP / IP.

**ПРОЦЕСС ИНКАПСУЛЯЦИИ (НА СТОРОНЕ ОТПРАВИТЕЛЯ)**

Шаг 1. Уровень приложения, представления и сеанса в модели OSI принимает пользовательские данные в форме потоков данных, инкапсулирует их и пересылает данные на транспортный уровень. Нет необходимости добавлять к данным какой-либо верхний или нижний колонтитул - это зависит от приложения.

Шаг 2. Транспортный уровень берет поток данных с верхних уровней и разбивает его на несколько частей. Транспортный уровень инкапсулирует данные, добавляя соответствующий заголовок к каждой части. Эти фрагменты данных теперь называются сегментами данных. Заголовок содержит информацию о последовательности, так что сегменты данных могут быть повторно собраны на стороне получателя.

Шаг 3. Сетевой уровень берет сегменты данных с транспортного уровня и инкапсулирует их, добавляя дополнительный заголовок к сегменту данных. Этот заголовок данных содержит всю информацию о маршрутизации для правильной доставки данных. Здесь инкапсулированные данные называются пакетом данных или дейтаграммой.

Шаг 4: Канальный уровень берет пакет данных или дейтаграмму с сетевого уровня и инкапсулирует его, добавляя дополнительный заголовок и нижний колонтитул. Заголовок содержит всю информацию о переключении для правильной доставки данных в соответствующие компоненты оборудования, а нижний колонтитул содержит всю информацию, относящуюся к обнаружению и мониторингу ошибок. Здесь инкапсулированные данные называются фреймом данных.

Шаг 5: Физический уровень берет кадры данных с уровня канала данных и инкапсулирует их, преобразуя их в соответствующие сигналы данных или биты, соответствующие физической среде.

**ПРОЦЕСС ДЕИНКАПСУЛЯЦИИ (НА СТОРОНЕ ПОЛУЧАТЕЛЯ)**

Шаг 1. Физический уровень принимает инкапсулированные сигналы данных или биты от отправителя и деинкапсулирует их в форме кадра данных, который будет перенаправлен на верхний уровень, то есть на уровень канала данных.

Шаг 2: Канальный уровень берет кадры данных с физического уровня. Он деинкапсулирует фреймы данных и проверяет заголовок фрейма, чтобы увидеть, переключен ли фрейм данных на правильное оборудование или нет. Если фрейм попадает не в тот пункт назначения, он отбрасывается, в противном случае он проверяет информацию в нижнем колонтитуле. Если есть какая-либо ошибка в данных, запрашивается повторная передача данных, если нет, то они деинкапсулируются, и пакет данных пересылается на верхний уровень.

Шаг 3. Сетевой уровень получает пакет данных или дейтаграмму от уровня канала данных. Он деинкапсулирует пакеты данных и проверяет заголовок пакета, если пакет направлен в правильное место назначения или нет. Если пакет отправляется неправильному адресату, пакет отбрасывается, если все в порядке, он деинкапсулируется, и сегмент данных пересылается на верхний уровень.

Шаг 4: Транспортный уровень берет сегменты данных с сетевого уровня и деинкапсулирует их. Сначала он проверяет заголовок сегмента, а затем повторно собирает сегменты данных для формирования потоков данных, а затем эти потоки данных пересылаются на более высокие уровни.

Шаг 5: Уровень приложения, представления и сеанса в модели OSI берет инкапсулированные данные с транспортного уровня, деинкапсулирует их, и данные приложения отправляются в приложения.

А теперь вернемся в сегодняшний день к современным чудесам автопрома с автоматической коробкой передач. На самом деле, по сути, ничего не изменилось. Бензонасос по-прежнему подает бензин в двигатель, дифференциалы обеспечивают поворот колес под разными углами, коленчатый вал преобразует поступательное движение поршня во вращательное движение колес. Прогресс бывает разным. Теперь все эти действия скрыты от пользователя и позволяют ему крутить руль и нажимать педаль газа, не задумываясь о том, что в это время происходит с форсункой, дроссельной заслонкой и распределительным валом. Именно скрытие внутренних процессов, происходящих в автомобиле, позволяет эффективно использовать его даже тем, кто не является профессиональным автомехаником с двадцатилетним стажем работы. Это сокрытие в ООП называется **инкапсуляцией**.

**Заключение**

**Инкапсуляция - это свойство системы, которое позволяет объединять данные и методы, которые работают с ней, в классе и скрывать детали реализации от пользователя. Инкапсуляция неразрывно связана с концепцией интерфейса класса. Фактически, все, что не входит в интерфейс, инкапсулируется в класс.**