**1. ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ**

**Лекция 1**

**Тема лекции**: абстракция, инкапсуляция, наследие и полиморфизм. Объекты в свете абстракции и инкапсуляция . Создание объектов и инициализации данных состояния. Протокол описания класса: поля данных i функции-члены

# Основные определения ООП

Существует 5 ключевых компонентов любой объектно-ориентированного языка программирования, в том числе и языка С ++. Это следующие понятия:

• **Объект** - инкапсулированная абстракция, содержащая информацию о состоянии и четко определенную множество протокола доступа (сообщения, которые обрабатывает объект).

• **Сообщение** - это специальный символ, идентификатор или ключевое слово с параметрами или без них, представляющее действие, которое выполняет объект.

• **Класс** - это определенный тип объектов. Он задается с помощью описания класса, где определяются переменные состояния и протокол доступа к объектам данного класса. Классы могут иметь иерархическую или контейнерную компанию.

• **Экземпляр** **объекта**. Все объекты принадлежат какому-либо классу. Свойства экземпляра объекта определяются описанием класса.

• **Метод**. Метод существует для каждого сообщения, определенного для какого-то класса. Метод определяет реакцию объекта на сообщения. Обычно состоит из ряда выражений и может использовать протокол другого класса.

Таким образом, все объекты принадлежат каким-то классам. Объекты обрабатывают сообщения согласно методам, которые были определены в описании класса. Эти переменные могут иметь одинаковые или разные значения в разных экземплярах класса.

В языке С ++ принята следующая терминология:

• **Класс** - ключевое слово class - новый тип данных (расширение структурного типа данных).

• **Объект** - переменная типа ClassName, где ClassName - определенный ранее класс.

• **Данные** **состояния** - закрытые данные или переменные экземпляра объекта. Объявляется в описании класса и называются данными члены класса.

• **Сообщение** указывается с помощью прототипов функций в описании класса. Прототипы функций содержат тип возвращаемого значения, имя функции и список параметров. Список параметров должен содержать типы параметров и необязательные имена параметров.

• **Метод** - это определение (реализация) функций. Прототипы функций и их определения вместе - это сообщения, которые может обрабатывать объект. Вместе они называются функциями-членами класса. К членам класса относятся функции-члены и данные-члены.

**Абстракция, инкапсуляция, наследственность и полиморфизм**

• **Абстракция** - формирование представления о качествах и свойствах предмета путем мысленного удаления каких-то материальных объектов.

• **Инкапсуляция** - результат хранения в капсуле, размещение в ограниченном пространстве. Единица инкапсуляции - это объект. Капсула или объект, включает в себя данные состояния и протоколы сообщений, которые обрабатывает объект.

• **Наследственность** - это передача свойств, полученные от предшественника. Получаемые данные состояния и протоколы сообщений из протокола родительского класса.

• **Полиморфизм** - это условия, в которых вид имеет две различные морфологические формы. Это означает, что отдельные сообщения могут вызывать различные действия на этапе выполнения программы. Конкретная форма реализации сообщения определяется и связывается с объектом во время выполнения программы.

# Объекты в свете инкапсуляции и абстракции

Объект содержит в себе значение, отражающие его внутреннее состояние. Это переменные состояния для объектов данного класса. Кроме того, он содержит в себе данные о средстве сообщений, которые к нему поступают. В объекте инкапсулированные все свойства абстракции, включая значение абстракций данных и функциональные абстракции (сообщения и методы). Две основные компоненты объекта как инкапсуляции абстракций задаются с помощью определения абстракций (поля данных и определения функций-членов, заданных в протоколе описания класса) и внутренних состояний (фактические значения полей данных для конкретного объекта).

Для объектов желательны следующие свойства:

**• Четкая граница.** Она определяет область видение инкапсуляции. В С ++ область видение инкапсуляции для объекта распространяется на описание его класса и описание родительских классов.

• **Хорошо разработанный интерфейс**. Интерфейс описывает как объект взаимодействует с другими или сегментами программы. В С++ он задается с помощью прототипов функций-членов. Другие объекты могут осуществлять доступ к объекту только через такой контролируемый интерфейс.

• **Защищенная внутреннее представление**. Внутренняя представления объекта невидимое для других объектов и не может быть ими уничтожено. В С ++ эта защита является гибким и может управляться благодаря использованию концепции закрытых, защищенных и открытых секций описания класса. Для той же цели используется принцип дружественных функций и классов.

**Классы и объекты**

Вывод: объект обладает состоянием, поведением, индивидуальностью;

структура и поведение подобных объектов определяет общий для них класс;

термины "экземпляр класса" и "объекты" взаимозаменяемы.

# Состояние объекта

Состояние объекта характеризуется перечнем всех возможных свойств данного объекта (обычно статических) и текущими значениями каждого из этих свойств (обычно динамическими).

К числу свойств объекта относятся присущие ему и приобретенные им характеристики, черты, качества или возможности, которые делают данный объект самим собой.

Все свойства объекта характеризуются значениями их параметров. Эти значения могут быть простыми количественными характеристиками, а могут означать другой объект.

Внутри каждого объекта хранятся в защищенном виде элементы, отражающие его состояние, и состояние всей системы в целом, распределенный между объектами.

# Поведение

Поведение характеризует то, как объект влияет или подлежит действия других объектов с точки зрения изменения состояния этих объектов и передачи сообщений.

Определенное влияние одного объекта на другой с целью вызвать соответствующую реакцию называется операцией. Например, объект пользователь может активизировать операции add и pop для того, чтобы дать очередном объекта прирост или сократить его.

В объектных и объектно-ориентированных языках операции, выполняемые над данным объектом, называются методами и входят составной частью в описания класса. В С ++ с этой целью используется также термин функция-элемент или функция-член.

Существует пять основных видов операций над объектами:

1.Модификатор - операция, изменяет состояние объекта путем записи или доступа.

2.Селектор - операция, дает доступ для определения состояния объекта без его изменения (операция чтения).

3.Итератор - операция доступа к содержанию объекта по частям (в определенной последовательности).

4.Конструтор - операция создания и инициализации объекта.

5.Деструктор - операция разрушения объекта и освобождения памяти, которую они занимали.

Последние два вида операций применяются в языке C ++. Таким образом, любой метод является операцией, но не любая операция является методом.

Совокупность всех методов и общедоступных процедур, касающихся конкретного объекта, образуют протокол этого объекта. Таким образом протокол объекта определяет оболочку поведения объекта охватывает его внутренний статический и внешний динамический проявление.

# Объекты как автоматы

Наличие внутреннего состояния объекта означает, что порядок выполнения операций имеет важное значение. Это позволяет представить объект в качестве сложного независимого автомата. Объекты могут быть активными и пассивными. Активным является тот объект, который расположен в канале управления. Активный объект автономен, то есть он может реализовать свое поведение без воздействия со стороны других объектов. Пассивный объект может изменять свое состояние только под воздействием других объектов. Таким образом, активные объекты системы составляют канал управления. Если система имеет несколько каналов управления, то и активных объектов может быть множество.

Свойства объекта, отличающие его от всех других объектов, называются индивидуальностью. В большинстве языков программирования для отличия временных объектов, их взаимной адресации и идентификации используются имена переменных. Источником многих ошибок в объектно-ориентированном программировании является невозможность отделить имя объекта от самого объекта.

# Время существования объектов

Началом времени существования любого объекта является момент его создания (отведение участка памяти), а окончанием - момент изъятия отведенного участка памяти. В языке С ++ создания объекта автоматически приводит к вызову конструктора для данного класса объектов. В С ++ можно переопределить средства создания объектов. Для создания объектов в свободном участке памяти применяется процедура new. Объект продолжает существовать до тех пор, пока он занимает место в памяти, даже если будет потеряно ссылка на этот объект. Объекты, созданные в программном стеке, ликвидируются автоматически при переходе управления за пределы области действия переменной, связанной с данным объектом. При ликвидации объекта автоматически вызывается деструктор.

# Отношение между объектами

Сами по себе объекты не представляют никакого интереса, только в процессе взаимодействия объектов между собой реализуется цель системы. Отношение двух любых объектов основываются на предположении, что каждый объект имеет информацию о другой объект, об операциях, которые над ним можно выполнять и о ожидаемое поведение. Существуют два типа иерархических соотношений объектов:

1.Видношення использования.

2.Видношення включения.

Отношение использования. При использовании одного объекта другим, они обмениваются сообщениями. Пересылки сообщений между объектами однонаправленное, но возможны и двунаправленные связи. Каждый объект, включенный в отношение использования, может выполнять следующие три роли:

1.Вплив. Объект может влиять на другие объекты, но сам никогда не подверженной воздействию других объектов; в определенном смысле соответствует понятию активный объект.

2. Использование. Объект в этом случае может только подвергаться управлению со стороны других объектов, но никогда не выступает в роли объекта влияет на другие.

3.Посередництво. Такой объект может выступать как в роли влияющий, так и в роли исполнителя. Как правило, объект-посредник создается для выполнения операций в интересах какого-то активного объекта или другого посредника.

# Понятие синхронизации

При передаче сообщений от одного объекта к другому оба взаимодействующих объекты должны определенным образом синхронизовуватись. Для последовательных систем такая синхронизация реализуется через вызовы подпрограмм. В параллельных системах для многоканального управления необходимо решить проблему исключительных ситуаций. Отсюда следует еще один способ классификации объектов.

1. Объект-транслятор - пассивный объект, имеющий только один канал управления.

2.Блокований объект - пассивный объект, имеющий несколько каналов управления.

3.Паралельний объект - активный объект, имеющий несколько каналов управления.

Отношение включения. Объекты находятся в отношении включения, если одни объекты являются элементами состояния других объектов.

Включение одних объектов в другие предпочтительнее использовать в том плане, что при этом уменьшается число объектов с которыми нужно оперировать на данном уровне описания. С другой стороны, использование одних объектов другими имеет преимущество: не возникает сильной зависимости между объектами.

# Понятие "класс"

В то время, как объект означает конкретную сущность, определенную во времени и пространстве, класс определяет лишь абстракцию, "выжимку" из объекта. Следовательно, с точки зрения объектно-ориентированного программирования, определением класса являются: "класс - множество объектов, связанных общностью структуры и поведения". Любой объект просто экземпляром класса. Объект не является классом, хотя класс может быть объектом. Объекты, которые не связаны общностью структуры и поведения, не могут образовать класс.

В то время, как объект существует конкретно и играет определенную роль, класс содержит описание структуры и поведение всех объектов, связанных отношением общности. То есть класс выполняет роль соглашения о связях в отношениях абстракции и всех ее реализации. Это достигается за счет того, что в интерфейсной части описания класса содержатся необходимые данные о проектные решения. Интерфейсная часть описания класса соответствует его внешнему проявлению, подчеркивает его абстрактность, но скрывает структуру и особенности поведения. Интерфейсная часть класса состоит из перечня действий, предполагает описание других классов, констант, переменных и особенностей, необходимых для полного определения данной абстракции.

Реализация класса составляет его внутреннюю поведение и определяет особенности поведения. В этой части раскрывается реализация тех операций, которые перечислены в интерфейсной части описания. Интерфейсная часть описания класса может быть разделена на 3 составные части: общедоступная, защищена, отделена.

Структура состояния объекта также требует определенного описания. Он состоит в объявлении констант и переменных в уединенной части описания интерфейса. Это делает общую часть структуры объектов закрытой для доступа и внесение в этой части изменений, не отражается на функционировании объектов-пользователей. В С ++ допускаются следующие определения как в уединенной так и в общедоступной части интерфейса класса.

Взаимосвязь классов и объектов. Отношение между классами и объектами

В большинстве практических случаев классы статичны, то есть все их особенности и содержание определены в процессе компиляции программы. Из этого следует, что любой созданный объект относится к фиксированной класса. Объекты, наоборот, в процессе выполнения программы непрерывно создаются и разрушаются.

На этапе анализа и ранних стадиях проектирования решаются две основные задачи:

- Выявление классов и объектов, составляющих словарь предметной области;

- Построение структур, обеспечивающих совместную взаимодействие объектов, при которой достигаются заданные требования.

В первом случае говорят о ключевых абстракции задачи, а во втором - о механизмах реализации.

В первой фазе проекта внимание проектировщика сосредоточивается на внешних проявлениях ключевых абстракций и механизмов. Такой подход создает логический каркас системы, состоящей из структуры классов и структуры объектов. На последующих фазах проекта, включая и реализацию, внимание переключается на внутреннюю поведение ключевых абстракций и механизмов и охватывает вопросы их физического представления. Решения, принимаемые в процессе проектирования составляют архитектуру модулей и архитектуру процессов в системе.

Для оценки качества классов и объектов, выделяемых в системе существует 5 критериев:

1. Взаимозависимость.
2. Связанность.
3. Достаточность.
4. Полнота.
5. Простота.

Взаимозависимость - это степень глубины связей между отдельными модулями. Фрагменты системы, которые сильно зависят от других, гораздо сложнее воспринимать, заменить и модифицировать. Для улучшения качества системы следует по возможности избегать сильной зависимости между отдельными модулями.

Связанность - это степень взаимодействия между элементами отдельного модуля, класса или объекта. Наименее желательным является связанность по случайному признаку, когда в одном модуле собираются совершенно независимые абстракции. Желательным является функциональная связанность, при которой все элементы модуля тесно взаимодействуют для достижения цели.

Достаточность - наличие в классе или модули программы всего необходимого для реализации логической и эффективного поведения. То есть компоненты должны быть полностью пригодны к использованию. Например, рассмотрим класс множество. В этот класс необходимо включить операции извлечения и добавления элемента. Будет ошибкой не включить в класс одной из указанных операций. Нарушение требований достаточности оказывается, как только создается класс пользователь.

Полнота - наличие в интерфейсной части класса всех требуемых характеристик абстракции. Идея достаточности предъявляет к интерфейсу минимальные требования, а идея полноты охватывает все существенные аспекты абстракции. Полнота является субъективным фактором и разработчики часто выносят на верхний уровень такие операции, можно реализовать на более низком уровне. Из этого вытекает требование простоты.

Простыми являются только такие операции, обеспечивающие реализацию эффективного действия абстракции. В примере с множественной операция добавления элемента является примитивной, а операция сложения 4-х элементов не будет примитивной, поскольку эффективно реализуется через операцию добавления одного элемента.

Эффективность тоже субъективный фактор.