**Get-методы** Для получения информации о характеристиках объекта (в том числе о защищённых полях).

**Private-** члены класса, доступные только разработчикам класса (т.е. только при реализации методов этого класса).

**Protected**- члены класса, доступные разработчикам класса и разработчикам классов-потомков.

**Public** - члены класса, доступные как разработчикам, так и пользователям класса.

**Set-методы** Для изменения характеристик объекта (в том числе защищённых полей).

**Абстрактный-**Класс, содержащий хотя бы один абстрактный виртуальный метод (чисто виртуальный метод). Если класс, производный от абстрактного, не определяет все чисто виртуальные функции, он также является **абстрактным.**

**Нельзя создать объект** абстрактного класса**//**абстрактный класс нельзя использовать при явном приведении типов, для описания типа параметра и типа возвращаемого функцией значения**//**формальным параметром функции может быть указатель абстрактного класса**//**допускается объявлять указатели и ссылки на абстрактный класс, если при инициализации не требуется создавать временный объект; впоследствии указатель на абстрактный класс можно связать с объектом реального класса- наследника.

**Базовый** (порождающий, родительский) - Класс, на основе которого создается новый класс.

**Базовый класс-** **Прямой базовый класс** явно указывается при объявлении производного класса. **Косвенный базовый класс** явно не указывается, он наследуется через 2 или более уровней иерархии классов.

**Виртуальные методы-** методы, содержащиеся в разных классах одной иерархии, но имеющие общее имя и объявленные с ключевым словом virtual.

**Виртуальный метод** – это метод, будучи описанным в потомках, замещающий собой соответствующий метод везде, даже в методах, описанных для предка, если он вызывается для потомка. Виртуальный метод представляет собой метод базового класса, который переопределяется в производном классе.

**Внешние методы** реализованы как отдельные функции. Чтобы показать, что это не обычная функция, а метод конкретного класса, к имени метода добавляется имя класса с помощью операции :: (расширение области видимости). Текст внешних методов может быть скрыт от пользователей класса.

**Внутренние методы** реализованы в рамках описания класса; их текст доступен для чтения пользователям класса.

**Включение** – отношение между объектами. (управление именами; управление доступом.) **Вложенный класс**-класс, объявленный внутри другого класса,

**Возможности наследования** **МОЖНО**: Добавлять в производный класс данные (поля)**//** Дополнять в производном классе функциональные возможности базового класса (вводить новые методы) **//**Модифицировать в производном классе методы базового класса **НЕЛЬЗЯ**: Модифицировать в производном классе поля, представленные базовым классом (сохранив их идентификаторы)

**//**Удалять компоненты из базового класса

**Деструктор** – специальный метод, который неявно вызывается при корректном уничтожении объекта. **Дружественная функция**, которая может быть обычной функцией или методом другого класса, должна быть описана внутри класса, к скрытым полям которого она должна иметь доступ, с описателем friend, и получать в качестве параметра объект класса или ссылку на этот объект.

**Иерархия классов -** При наследовании формируется иерархия классов, а при объявлении объектов и **иерархия объектов**. Определяет новые классы на основе уже имеющихся. В иерархии объектов производный объект имеет возможность доступа элементам данных и методам объекта, типизированных базовым классом.

**Исключения** – это механизм обработки ошибок сиспользованием систематического, объектно- ориентированного подхода.

**Инкапсуляция-** Хранение в одной структуре как данных (констант и переменных), так и функций их обработки (методов). / Объединение в одной конструкции полей и методов.

**Класс** – определенный пользователем проблемно-ориентированный тип данных, описывающий внутреннюю структуру объектов, которые являются его экземплярами / Определенный программистом прототип программируемого объекта с набором атрибутов (переменных и методов), которые описывают данный объект. Доступ к атрибутам и методам осуществляется через точку **Косвенный базовый класс** явно не указывается, он наследуется через 2 или более уровней иерархии классов.

**Конкретные классы -** Классы, которые могут быть реализованы.

**Константные методы-**методы, не изменяющие значения полей объекта, для которого эти методы применяются.

**Конструктор** – специальный метод, который неявно вызывается при создании нового объекта.

**Конструктор копирования** – специальный конструктор, который получает в качестве параметра константную ссылку на объект этого же типа.

**Конструктор по умолчанию-**это конструктор без параметров, который не выполняет никаких дополнительных действий.

**Константные поля** -Поле класса может быть объявлено константным с помощью модификатора const. Константныt поля могут содержать различные значения для различных объектов. Эти значения задаются в конструкторе и в дальнейшем не могут быть изменены.

**Методы или функции-члены**- Особые функции, определенные внутри класса.

**Назначение деструктора** – выполнение дополнительных действий по освобождении ресурсов, захваченных при создании объекта или в процессе работы с ним.

**Назначение конструктора** – выполнение дополнительных действий по инициализации объекта (например, задание начальных значений полей создаваемого объекта).

**Наследование** – Передача данных и методов от родительских классов производным / Отношение между двумя классами (**предком** и **потомком**), когда объект-потомок повторяет элементы структуры и поведения предка / Механизм создания нового класса на основе уже существующего.

**Непосредственным базовым классом** называется такой класс, который входит в список базовых классов при определении класса.

**Операторы приведения типов:**

**const\_cast** служит для удаления модификатора const;**dynamic\_cast** применяется для преобразования указателей родственных классов иерархии, в основном — указателя базового класса в указатель на производный класс; **reinterpret\_cast** выполняет преобразование указателей на пустой тип (void \*) в указатели на более конкретные типы (int \*, char \* и т.д.) ; **static\_cast** обеспечивает преобразование родственных типов указателей или объектов. Этот оператор можно использовать для подавления выдачи предупреждений при компиляции в случае замены типа, занимающего больший объем памяти, на тип, занимающий меньший объем памяти.

**Оператор присваивания по умолчанию** копирует все поля объекта из левой части всоответствующие поля объекта из правой части.

**Объект** (экземпляр класса) находится в таком же отношении к своему классу, в каком переменная находится по отношению к своему типу.

**Объектно-ориентированное программирование** (ООП) - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы могут образовывать иерархию наследования.

**Одиночное наследование**- это когда класс наследует свои атрибуты от одного родительского класса.

**Множественное наследование**- класс наследует свои атрибуты от нескольких родительских классов

**Очередью** называется структура данных, содержащая последовательность однотипных элементов и позволяющая эффективно выполнять следующие **операции-** вставка нового элемента в хвост очереди; просмотр элемента, находящегося в голове очереди; удаление из очереди элемента, находящегося в голове; определение количества элементов в очереди (или определение того, пуста ли очередь)

**Перегрузка оператора вызова функции** - Изменение работы метода, унаследованного дочерним классом от родительского класса. Запись- *тип\_возврата operator()* (формаль\_параметры);Класс, в котором определён хотя бы один оператор вызова функции, **называетсяфункциональным классом**. Функциональныйкласс может не иметь других полей и методов.Функциональные классы используются валгоритмах из библиотеки STL.

**Перегрузка оператора приведения типа** имеет следующий прототип- *operator тип().* Тип – задает тип данных, к которому приводится объект. Этот тип может бытькак стандартным, так и пользовательским. Нет возвращаемого значения.

**Перегрузка операторов** - Определение работы стандартных операторов с экземплярами данного класса (с объектами пользовательских классов). **Каждый оператор** в С++ реализован как вызов функции со следующим прототипом- *Тип operator имя-оператора(список\_параметров).*

**Переопределение** (замещение) метода – ситуация, когда в классе-потомке определяется метод с таким же названием и набором параметров, как и у предка. Для предка и потомка работают разные методы.

**Перегрузка** – определение в одном и том же классе нескольких методов с различным набором параметров.

**Полиморфизм** – Способность объекта отреагировать на некоторый запрос соответственно своему типу (или возможность реализации одного и того же метода несколькими классами иерархии различными способами). В переводе с греческого означает многообразие форм (poly - много, morphos -форм) / Свойство, которое позволяет использовать одно имя для обозначения действий, общих для родственных классов. **Понижающee (downcast)** –преобразование из базового класса в производный; **Повышающее (upcast)** – приведение из производного класса в базовый; **Перекрестное (crosscast)** – приведение между производными классами одного базового или, наоборот, между базовыми классами одного производного . **Позднее (или динамическое) связывание** (применяется только для методов классов) – во время выполнения приложения определяется действительный класс объекта, адрес которого находится в указателе, и вызывается метод нужного класса. **Полиморфизм реализован**: механизмом перегрузки функций и операций**//**механизмом шаблонов (обобщенных функций)**//**механизмом переопределения методов класса.

**Пользователи класса** создают объекты предоставленного в их распоряжение класса и работают с этими объектами, используя интерфейс, предоставленный разработчиками класса.

**Поля или данные-члены** -данные, входящие в класс.

При построении иерархии классов классы, находящиеся на верхних уровнях иерархии, могут не быть реализованы. Это означает, что нельзя создать ни одного объекта, принадлежащего этому классу. Такие классы называются **абстрактными**. **Объявление** абстрактного метода и класса- *class abstract\_class {… virtual void DoSomething() = 0;};*

**Производный (порожденный**), **класс-потомок, наследник** - Новый класс, формируемый на основе базового / Подтип базового класса.

**Прямой базовый класс** явно указывается при объявлении производного класса.

**Раннее (или статическое) связывание** – адрес функции назначается во время компиляции, и именно этот адрес используется при вызове функции;

**Разработчики класса** определяют назначение класса, его интерфейс, реализуют интерфейс класса в рамках предоставленных им средств.

**Список инициализации обязателен**: если в качестве поля задаётся объект другого класса, для которого должен быть запущен конструктор; для вызова конструктора предка при использовании механизма наследования

**Статические методы**- Предназначены для обращения к статическим полям класса. Могут обращаться непосредственно только к статическим полям и вызывать только другие статические методы класса, потому что им не передается скрытый указатель this.

**Статические поля**: Поле класса может быть объявлено статическим с помощью модификатора static. При реализации класса необходимо определить поле, общее для всех объектов этого класса. Если статическое поле объявлено как public, к нему можно обращаться и пользователям класса.

**Страж включения**- Для предотвращения ошибок, связанных с повторным описанием класса, логично обрамлять описание класса стражем включения.

**Типы наследования – одиночное (простое) наследование-** базовым является один класс**; – множественное наследование-** базовых классов должно быть несколько**. Указатель this**- При работе методов класса содержит адрес объекта, для которого вызывается этот метод. **Функциональный класс** - класс, в котором определён хотя бы один оператор вызова функции. Может не иметь других полей и методов. Используются в алгоритмах из библиотеки STL. **Функция-оператор может быть определена как**: метод класса, дружественная функция класса ,обычная функция. **Члены класса-**данные и функции, входящие в состав класса. **Преимущества использования ООП**:уменьшение сложности программного обеспечения; повышение надежности программного обеспечения; обеспечение возможности модификации отдельных компонентов программного обеспечения без изменения остальных его компонентов; обеспечение возможности повторного использования отдельных компонентов программного обеспечения. **Не могут быть перегружены операторы**: (.) - доступ к члену класса; (.\*) - доступ к члену класса через указатель; (::) - разрешения области видимости;(?:) - условный оператор;( #, ##) - препроцессорные операторы; static\_cast, const\_cast, dynamic\_cast, reinterpret\_cast - операторы приведения типов (преобразования выражения из одного типа в другой) ; sizeof – определения размера; typeof – определения типа. **Ограничения на перегрузку** **операторов**: нельзя вводить обозначения собственных операторов; не допускается перегрузка операторов для стандартных типов //при перегрузке операторов должно сохраняться число аргументов;//не изменяются приоритеты перегруженных операторов; //при перегрузке операторов нельзя задавать аргументы по умолчанию.