**Динамический полиморфизм**

В C++ введено понятие **виртуальных функций (методов)**. Механизм виртуальных методов заключается в том, что результат вызова виртуального метода с использованием указателя или ссылки зависит от типа объекта, на который указывает этот указатель.

Виртуальная функция объявляется описателем ***virtual***.

Тип данных (класс), содержащий хотя бы одну виртуальную функцию или виртуальный деструктор, называется **полиморфным типом (классом)**, а объект этого типа – **полиморфным объектом**.

Во всех производных классах наследуемая виртуальная функция остается таковой (виртуальной), даже если слово ***virtual*** не указано в явном виде. Таким образом, все типы-наследники полиморфного типа являются полиморфными типами.

Виртуальные функции могут быть перегружены в классах-наследниках. По объявлению перегруженные в производных классах виртуальные функции должны полностью совпадать с базовой виртуальной функцией в части типа возврата, имени, числа и типа аргументов. Как правило отличаются такие функции только кодом тела.

Если в производном классе для виртуальной функции базового класса будут изменены тип или число аргументов, то класс-наследник будет иметь две виртуальные функции – с базовой реализацией и свою собственную невиртуальную или виртуальную функцию.

Если для виртуальной функции в классе наследнике изменен код возврата и она не объявлена как виртуальная, то компилятор выдаст ошибку. Если такая функция будет определяться как виртуальная, то в производном классе появится собственная виртуальная функция.

Если виртуальная функция базового класса не объявлена в производном классе, то производный класс наследует ее базовую реализацию. В случае вызова такой функции по адресу объекта производного класса будет вызван ее базовый вариант.

Отличительной особенностью вызова виртуальной и невиртуальной функции является время ее связывания с кодом реализации. Для невиртуальных функций такое связывание реализуется на этапе редактирования связей (компиляции), для виртуальной – на этапе выполнения программы (так называемое позднее связывание). Иными словами, при вызове виртуальной функции через указатель на полиморфный объект осуществляется динамический выбор тела функции в зависимости от текущего тела объекта, а не от типа указателя. В этом и проявляется **динамический полиморфизм.**

Включение в класс виртуальной функции сопровождается созданием и включением в структуру класса нового невидимого программисту члена – указателя на таблицу виртуальных функций. В этой таблице содержатся адреса всех виртуальных методов класса. Каждой функции на этапе компиляции выделяется участок памяти, первый элемент этого участка и является адресом функции. По сути таблица виртуальных функций – массив указателей. Каждой виртуальной функции присваивается номер в порядке определения в классе. При наследовании производный класс получает таблицу виртуальных функции базового класса. Если в производном класса какой-либо метод переопределяется, он будет заменен на новый. При добавлении новых виртуальных методов в производный класс его таблица расширяется, а таблица базового класса остается такой же.

Через адрес объекта производного класса можно вызвать базовую реализацию виртуальной функции. Для этого необходимо специфицировать имя базового класса при вызове этой функции.

Обычно виртуальные функции производных классов вызывают через указатель через базовый класс, который инициализирован адресом производного класса. В случае вызова виртуальной функции от имени объекта базового класса, который инициализирован объектом производного класса, произойдет вызов его базовой реализации, так как в этой ситуации произойдет неявное преобразование объекта производного класса к объекту базового класса.

Не могут быть виртуальными статические функции, а также функции – не члены класса. Конструкторы, в отличии от деструкторов также не могут быть объявлены виртуальными.

***Виртуальный деструктор*** – важная часть аппарата динамического полиморфизма. Если указатель типа базового класса указывает на объект производного класса, то при удалении объекта с использованием данного указателя в случае невиртуальности деструкторов сработает деструктор того типа, который был использован при объявлении указателя. При описании конструкторов и деструкторов уже было указано, что при удалении объекта во вторую очередь срабатывает деструктор базового типа, удаляя информационные члены базового типа, унаследованные в типе-наследнике, а сначала срабатывает деструктор текущего объекта, удаляя дополнительные члены типа-наследника. Таким образом, деструктор базового типа применяется к объектам производного типа. Но при его локальном срабатывании не будут удалены дополнительные члены типа-наследника.

Таким образом, при создании и удалении объектов производных типов с использованием указателей необходимо описывать деструкторы как виртуальные, если типы-наследники в своем составе имеют динамические структуры. Однако отметим следующее:

1. виртуальный деструктор необходим и для объекта без динамических структур в случае наличия динамических структур у типа-наследника, так как деструктор, автоматически генерируемый системой по умолчанию, является невиртуальным;
2. несмотря на то, что имя деструктора производного класса отличается от имени деструктора базового класса, достаточно объявления деструктора виртуальным только в базовом классе;
3. конструктор, в отличие от деструктора, нельзя описывать как виртуальный, так как всегда срабатывает конструктор именно того типа, который используется при создании объекта, и только после создания объекта его адрес передается для присвоения указателю.

Добавление виртуального деструктора в класс также увеличит размер объекта класса на размер указателя на таблицу виртуальных функций.

## Абстрактные классы

Виртуальная функция называется ***чистой***, если ее код объявляется равным 0. Чистая виртуальная функция имеет следующий вид:

*virtual тип\_возвращаемого\_значения имя\_функции* (*формальные параметры*) = 0;

Такая форма записи функции означает, что данная функция (точнее – метод класса) не имеет тела, описывающего ее алгоритм.

**Абстрактный класс – это класс, содержащий хотя бы одну чистую виртуальную функцию или чистый виртуальный деструктор.**

Нельзя создавать объекты на основе абстрактных классов, так как последние, имея в своем составе чистые виртуальные функции, не являются полноценными типами данных. Однако указатели на абстрактные классы создавать можно.

Абстрактные классы необходимы для наследования базового интерфейса в производном классе, а также адресации производных реализаций их виртуальных функций.

Несмотря на то что абстрактный класс не является полноценным типом, таблица виртуальных функций для него создается. При этом в таблице виртуальных функций перечисляются все виртуальные функции, в том числе и чистые виртуальные функции.

В классе-наследнике чистая виртуальная функция может быть переопределена обычной виртуальной функцией с соответствующей заменой пустого значения на адрес данной функции в таблице виртуальных функций класса-наследника.

Класс-наследник абстрактного класса может также быть абстрактным классом, если в нем осталась (или была дополнительно введена) хотя бы одна чистая виртуальная функция.