Полиморфизм

Пахаев Х.Х.

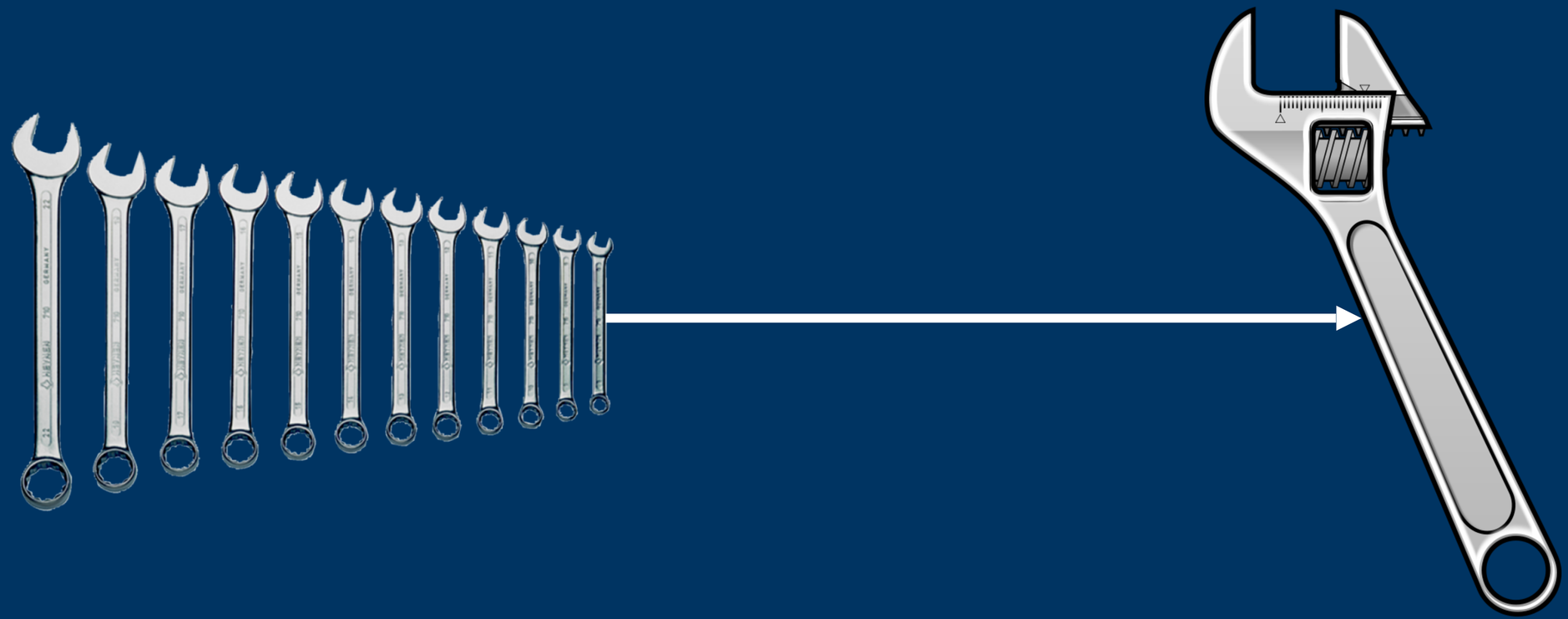
Что это такое?

Полиморфизм - это один из ключевых принципов объектно-ориентированного программирования (ООП), который позволяет объектам разных классов иметь общий интерфейс и вести себя по-разному.

В контексте C++, существует два основных вида полиморфизма: компилируемый (статический) полиморфизм и выполнение (динамический) полиморфизм.

Компилируемый полиморфизм (статический полиморфизм)

- Проявляется при использовании перегрузки функций и операторов, а также при шаблонах (templates).
- При компиляции компилятор выбирает, какая версия функции или оператора должна быть вызвана на основе аргументов, переданных в функцию.



Выполнение (динамический полиморфизм)

- Проявляется при использовании виртуальных функций и механизма наследования.
- Объекты классов-наследников могут переопределять виртуальные функции, и вызов такой функции происходит на основе реального типа объекта во время выполнения (ранее привязанный к объекту тип).

Виртуальные функции

- Виртуальная функция - это функция, объявленная в базовом классе с ключевым словом virtual, и которая может быть переопределена в производных классах.
- Виртуальные функции позволяют реализовать динамический полиморфизм, что означает, что вызов функции определяется на основе реального типа объекта во время выполнения (runtime), а не на основе типа указателя или ссылки во время компиляции.

Абстрактные классы

- Абстрактный класс - это класс, который содержит одну или несколько абстрактных (чистых виртуальных) функций.
- Абстрактные функции не имеют реализации в базовом классе и должны быть переопределены в производных классах. Абстрактные классы создаются с целью определения интерфейса, который должен быть реализован в производных классах.

Интерфейсы

Интерфейс - это абстрактный контракт или набор правил, который определяет, как объекты или компоненты могут взаимодействовать друг с другом. Интерфейсы используются в программировании, чтобы обеспечить структурированное и стандартизированное взаимодействие между разными частями программы или между различными программами.

Протоколы (интерфейсы) в Swift

В языке программирования Swift интерфейсы реализованы с использованием протоколов (protocols). Протоколы представляют абстрактные интерфейсы, которые определяют набор методов, свойств и других требований, которые классы или структуры могут принимать и реализовывать. Протоколы в Swift позволяют достичь полиморфизма и обеспечивают гибкость взаимодействия между различными типами.

Зачем нужен полиморфизм?

- Полиморфизм позволяет писать более гибкий и поддерживаемый код, так как он разрешает действовать с объектами разных классов с использованием общего интерфейса, не зная конкретного типа объекта.