Парадигма программирования — это совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию).

Парадигмы программирования

Императивная, или процедурная парадигма программирования является наиболее известной. Она развилась на базе низкоуровневых языков (машинные коды, ассемблер), основанных на архитектуре фон Неймана. Императивная программа состоит из последовательно выполняемых команд и вызовов процедур, которые обрабатывают данные и изменяют значения переменных программы. Переменные при этом рассматриваются как некоторые контейнеры для данных, подобно ячейкам памяти компьютера.

Функциональная парадигма программирования является менее традиционной, и в тоже время более древней, поскольку получила развитие из вычислений по алгебраическим формулам. Функциональная программа состоит из набора взаимосвязанных и, как правило, рекурсивных функций. Каждая функция определяется выражением, которое задает правило вычисления её значения в зависимости от значений ее аргументов. Выполнение функциональной программы заключается в последовательном вычислении значений функциональных вызовов.

В еще менее традиционной и необычной логической парадигме программа рассматривается как множество логических формул: аксиом (фактов и правил), описывающих свойства некоторых объектов, и теоремы, которую необходимо доказать. В свою очередь, выполнение программы – это доказательство теоремы, в ходе которого строится объект с описанными свойствами.

Основные различия указанных парадигм касаются не только концепции программы, но и роли переменной. В отличие от императивных программ, в функциональных и логических программах отсутствует явное присваивание значений переменным и, как следствие, побочные эффекты. Переменные в таких программах подобны переменным в математике: они являются обозначением функциональных аргументов или объектов, конструируемых в процессе доказательства. Еще одна яркая особенность функциональной и логической парадигм – использование рекурсии вместо циклов.

В получающей все большее распространение объектно-ориентированной парадигме программа описывает структуру и поведение вычисляемых объектов и классов объектов. Объект обычно включает некоторые данные (состояние объекта) и операции с этими данными (методы), описывающие поведение объекта. Классы представляют множество объектов со схожей структурой и схожим поведением. Обычно описание классов имеет иерархическую структуру, включающую полиморфизм операций. Выполнение объектно-ориентированной программы представляет собой обмен сообщениями между объектами, в результате которого они меняют свои состояния.

Характерные свойства основных парадигм программирования представлены в Таблице 1.

Таблица 1.

Свойства парадигм программирования

Парадигма	**Ключевой концепт**	**Программа**	**Выполнение программы**	**Результат**
Императивная	Команда	Последовательность команд	Исполнение команд	Итоговое состояние памяти
Функциональная	Функция	Набор функций	Вычисление функций	Значение главной функции
Логическая	Предикат	Логические формулы	Логическое доказательство	Результат доказатель-тва
Объектно-ориентированная	Объект	Набор классов объектов	Обмен сообщениями между объектами	Результирующее состояние объектов

Парадигмы и языки программирования

Поскольку парадигмы существуют и развиваются в рамках языков программирования, которые призваны обеспечивать удобные механизмы решения различных прикладных задач, неизбежна

интеграция парадигм. Так, уже в первых императивных алгоритмических языках, таких как Фортран, существовали элементы функционального стиля. В то же время применялись и языки, сохраняющие чистоту своей парадигмы, например, объектно-ориентированный Smalltalk и логический Пролог.

Большинство современных языков аккумулируют в себе элементы и приемы нескольких стилей, и тем не менее их можно классифицировать по основному их ядру, реализующему приёмы определенной парадигмы программирования, в частности:

- Императивная парадигма: языки Паскаль, Си, Ада;
- Функциональная парадигма: языки Лисп, Рефал, Плэнер, Schema, Haskel;
- Логическая парадигма: языки Пролог и Datalog;
- Объектно-ориентированная парадигма: Smalltalk, Eiffel.

Хотя Smalltalk, являющийся первым чисто объектно-ориентированным языком программирования, не получил должной известности, его основные идеи (абстрактные типы данных, инкапсуляция данных, полиморфизм операций) легко интегрировались в языки программирования других парадигм, в частности, императивной. Действительно, метод объекта может представлять из себя процедуру или функцию, а отправка сообщения – соответственно вызов процедуры или функции. Именно поэтому объектно-ориентированная парадигма программирования получила широкое распространение с тех пор, как ее средства были включены в популярные императивные языки Си и Паскаль, породив тем самым языки C++ и объектный Паскаль.

Подобным образом из объединения объектно-ориентированной с другими парадигмами возникли объектно-ориентированные варианты других языков программирования, например, язык CLOS – объектно-ориентированный Лисп. В настоящее время существует целый ряд современных языков программирования, в которых объединены две парадигмы:

- Императивная и объектно-ориентированная: C++, C#, Delphi, Java и др.;
- Функциональная и объектно-ориентированная: CLOS, Erlang и др.;
- Логическая и объектно-ориентированная: Object Prolog.

В целом, сравнивая разные языки и парадигмы, следует отметить существенное отличие средств и приёмов традиционной императивной парадигмы программирования от средств и приёмов нетрадиционных парадигм – функциональной и логической. Языки программирования, основанные на нетрадиционных парадигмах, отличаются ещё и методом реализации: программы на таких языках обычно интерпретируются, а не компилируются, как в императивных и императивных объектно-ориентированных языках. Еще одно важное их отличие – они ориентированы на символьную обработку данных, поскольку средства этих языков включают:

- древесные структуры данных, такие как списки Лиспа или термы Пролога;
- встроенный механизм сопоставления с образцом (Рефал, Плэнер, Пролог) и бэктрекинга (Плэнер, Пролог);
- функционалы, или функции высшего порядка (Лисп, Schema, Haskel);
- механизм частичных вычислений (Haskel).

Различия во встроенных структурах и механизмах языков программирования определяют различия в классах решаемых с их помощью прикладных задач. Тем самым, каждая из парадигм имеет свою область применения. Императивные языки удобны как для числовых, так и для символьных вычислений, однако они проигрывают функциональным языкам и Прологу в реализации обработки сложных символьных структур. Функциональное программирование хорошо подходит для символьных преобразований, логическое – для построения дедуктивных баз данных и экспертных систем, но в то же время они неудобны для программирования интерактивных задач или событийно-управляемых приложений. В свою очередь, объектно-ориентированная парадигма подходит для создания интерактивных программ со сложным поведением и интерфейсом, обрабатывающих данные различных типов.