Понятие языка программирования

- ***Язык программирования* (*ЯП*)** – это инструмент для планирования поведения некоторого устройства-исполнителя. Планы, управляющие поведением компьютеров, называются компьютерными программами.

Уточненное определение: ***язык программирования*** - **это нотация для записи компьютерных программ.**

Отличие универсальных ***ЯП*** – алгоритмическая полнота, т.е. возможность описания на таком языке любого вычисления (алгоритма). Например, языки программирования ***SQL*** (язык программирования баз данных) и ***HTML*** (язык разметки гипертекста) не является универсальным в отличие от языков ***Java***, ***C#*** (универсальные языки), на которых можно реализовать практически любой алгоритм, включая программу просмотра гипертекста и СУБД.

Языки и основные парадигмы программирования

- Все программы (как и деятельность по их созданию) можно разделить на два больших типа: ***программы «для себя» и программы «для других»***. Создание программ ***«для себя»*** назовем (несколько условно) ***научно-развлекательным программированием***. Основной критерий их качества – удобство применения для соответствующих целей (учебы, науки, развлечения). Самые известные и широко используемые до сих пор языки – Фортран, Бейсик, Паскаль.
- Создание программ ***«для других»*** ориентированы на эксплуатацию пользователями, не имеющими отношения к авторам. Такие программы называют программными продуктами, а процесс их создания – ***индустриальным программированием.*** Языки индустриального программирования отличаются от языков научно-развлекательного программирования, они сложны в изучении и реализации, включают в себя большое число концепций и понятий, обладают объемными библиотеками. Важным свойством индустриальных языков является наличие изобразительных средств, поддерживающих различные системы программирования.
- Совокупность идей и понятий, определяющих стиль программирования, называется ***парадигмой программирования.***

- В настоящее время в индустриальном программировании активно используются ***императивная*** и ***объектная парадигмы***. Есть основания полагать, что в ближайшее время начнет активно использоваться ***функциональная парадигма***.

Императивная парадигма

- ***Императивная (процедурная) парадигма*** основана на ***фон-неймановской модели*** (основатель математик Дж. фон Нейман). Эта модель до сих пор является основой большинства современных архитектур, что обусловило популярность и доминирование императивной парадигмы. Напомним, что модель содержит три основных компонента: центральное процессорное устройство (ЦПУ), оперативную память (ОП), устройство ввода-вывода (УВВ).

- ***Основные понятия императивных языков программирования (ИЯП)*** представляют собой абстракции основных понятий фон-неймановской модели. Любой ***ИЯП*** включает в себя понятие переменой (например, в языке паскаль – var x: integer, в языке С – int x), понятие операции (A * B – в любом языке), понятие оператора (оператор цикла, оператор присваивания и др.).
- ***Понятие простой переменной*** абстрагирует понятие ячейки памяти. Кроме простых переменных в императивном языке содержатся составные (т.е. состоящие из других переменных) массивы и записи (структуры).
- ***Понятие операции*** обобщает арифметико-логические команды. Почти для любой операции в ИЯП можно найти прототип – команду в машинном языке.

- ***Понятие оператора*** абстрагирует общее понятие команды. Операторы в императивном языке делятся на три группы: оператор присваивания; операторы управления; операторы ввода-вывода. ***Основным оператором*** в любом императивном языке является ***оператор присваивания: V:=E***, где V – переменная; E – выражение. Выполнение оператора присваивания состоит в вычислении значения выражения E и пересылке вычисленного значения в ячейку (или ячейки) ОП, соответствующей переменной V. Т.о, оператор присваивания в ***ИЯП*** может представляться последовательностью команд пересылки и арифметико-логических команд (и даже команд перехода).
- ***Операторы управления (циклы, операторы выбора, перехода и т.п.)*** абстрагируют машинные команды перехода.
- ***Операторы ввода-вывода*** обобщают машинные команды ввода-вывода.

- ***Императивные языки*** концептуально близки машинной архитектуре, поэтому программирование на таких языках позволяет весьма эффективно управлять поведением компьютеров. В индустриальном программировании в настоящее время доминируют либо чисто императивные языки (такие, как C), либо языки со смешанной объектно-императивной парадигмой (C++, Java, C# и др.).

- ***Объектная парадигма*** основана на понятии объекта. ***Объект*** обладает состоянием и поведением. Поведение состоит в посылке сообщений себе и другим объектам. Для каждого вида сообщения существуют «обработчики», которые могут модифицировать состояние объекта и посылать сообщения другим объектам.

- Объекты с одинаковым поведением и набором состояний объединяются в ***классы***. ***Между классами*** могут существовать следующие ***отношения***:
- - ***включение – «объект-подобъект»*** - включение объекта класса X в объект другого класса Y, т.е. говорят, что объект класса Y владеет объектом класса X;
- - ***наследование – «суперкласс-подкласс»*** - объект подкласса ***Derived*** обладает всеми свойствами объекта суперкласса ***Base***, а также, возможно, дополнительными свойствами (специфичными для класса ***Derived***). Таким образом, все объекты класса Derived одновременно принадлежат и классу ***Base***, но не наоборот;
- - ***ссылка*** – объект класса W содержит (но не владеет) ссылку на объект класса Ref. Также существуют и другие отношения.

- ***Объективная парадигма*** достаточно просто сочетается с ***императивной*** парадигмой.
- Состояние описывается набором переменных, а обработчики сообщений представляют собой процедуры или функции, имеющие доступ к состоянию.
- Посылка сообщения сводится к вызову соответствующего обработчика.
- В результате большинство современных языков индустриального программирования сочетает в себе обе парадигмы.
- ***Поэтому будем говорить об объектно-императивной парадигме.***

Функциональная парадигма

Основные понятия функциональных языков – ***функция и выражение. Выражение*** – это комбинация вызовов функций. Наряду с большим числом стандартных (встроенных в язык) функций программист может определять свои функции. ***Определение новой функции включает в себя имя функции, список аргументов и выражение – тело функции. Вызов функции состоит из имени функции и списка выражений – фактических параметров.*** Каждый из фактических параметров соответствует аргументу в определении функции. ***Основная операция – вызов функции.*** При вызове функции сначала вычисляются выражения – фактические параметры, а затем их значения подставляются вместо соответствующих им аргументов в выражение – тело функции. Наконец, вычисляется значение тела, которое и будет значением вызова. Первым языком программирования, в котором была реализована функциональная парадигма, был язык –***Лисп***. Программировать в функциональном стиле можно и на ИЯП.

Схема рассмотрения языков программирования

- Конструкции языков программирования рассматривают по следующей схеме: ***базис, средства развития и средства защиты.***
- ***Базис –*** это понятия и конструкции, встроенные в язык программирования, иначе говоря, это то, что «понимает» транслятор. ***Базис*** подразделяется на ***скалярный*** и ***структурный***.
- ***В скалярный базис*** входят элементарные (неделимые) типы данных и элементарные операции. ***К структурному базису*** относятся встроенные в язык конструкции, которые имеют внутреннюю структуру, т.е. включают в себя другие конструкции языка.
- ***В структурный базис*** императивных языков входят основные типы, например массивы и записи (структуры), большинство операторов языка (за исключением совершенно тривиальных типа ***break*** или ***continue*** в C++).

- ***Базисы императивных языков программирования*** похожи друг на друга. И то, что появляются новые языки, а старые продолжают жить, обусловлено различиями не в базисе, а в средствах развития и защиты.
- ***Средства развития*** – это аппарат, позволяющий добавлять в программы новые понятия (абстракции), которых не было в базисе. Уже в самом первом языке программирования ***Фортране*** появилось такое средство развития, как подпрограмма. Основное требование к средствам развития – возможность определять новые типы данных. Одними из самых мощных средств развития являются ***классы***. Но мало иметь средства создания новых типов данных. Необходимо определять их таким образом, чтобы соответствующие абстракции можно было легко употреблять, а также контролировать их цельность и корректность поведения. Эти функции выполняют ***средства защиты*** в языках программирования. ***К средствам защиты относятся***, например, средства обработки исключительных ситуаций, механизм абстракции данных и др.
- Современные языки программирования отличаются от более ранних языков, прежде всего тем, что у первых значительно усилены именно средства защиты.