## Лекция 2.

### *Раздел 2.* Классификация и сравнительное описание языков функционального и логического программирования

**Тема 4. Логическое программирование в широком и в узком смысле. Обзор языков Prolog, Gödel, Mercury.**
**Тема 5. Функциональное программирование. Терминология, основные понятия. Обзор языков HOPE, FP, ML, HASKEL, CURRY, LISP, SCHEME.**
**Тема 6. Интеграция функциональной и логической модели вычисления.**

### План лекции.

1. Логическое программирование.
2. Обзор языков логического программирования.
3. Функциональное программирование.
4. Обзор языков функционального программирования.

### Основная часть.

### Обзор языков Prolog, Gödel, Mercury

| Язык программирования | Особенности ЯП |
|---|---|
| **Prolog** | Появившись в начале 70-х годов в качестве экспериментальной разработки лабораторий искусственного интеллекта университетов Марселя и Эдинбурга ПРОЛОГ на протяжении почти десяти лет оставался известным лишь узкому кругу специалистов. Название языка "Пролог" происходит от слов ***ЛОГ***ическое ***ПРО***граммирование В 1973 году "группа искусственного интеллекта" во главе с Аланом Колмероэ создала в Марсельском университете программу, предназначенную для доказательства теорем. Эта программа использовалась при построении систем обработки текстов на естественном языке. Программа доказательства теорем получила название Prolog. Она послужила прообразом Пролога.<br>Пролог основывается на таком разделе математической логики, как исчисление ***предикатов.*** Точнее, его базис составляет процедура доказательства теорем методом резолюции для хорновских дизъюнктов. Основные приложения, которые связывают с языком Пролог это, как правило, разработки в области искусственного интеллекта и экспертных систем. |
| **Gödel** | Экспериментальный логический язык со строгим контролем типов. Попытка создать декларативную альтернативу Прологу. Значительное внимание уделено средствам метапрограммирования.<br>Замечательное средство для изучения логического программирования. К сожалению, поддержки для Windows нет и не планируется. |
| **Mercury** | Язык разработан в Мельбурнском университете. Первую версию выпустили Fergus Henderson, Thomas Conway и Zoltan Somogyi 8 апреля 1995 года.<br>Функционально-логический язык программирования. Синтаксис Mercury частично унаследован от Пролога, система типов похожа на Haskell. Это чисто декларативный язык, разработчики полностью убрали из него все императивные возможности, что позволило улучшить встроенные в компилятор возможности оптимизации. Название Mercury дано в честь бога скорости Меркурия и отражает |

| | |
|---|---|
| | направленность на получение быстродействующих программ. Операции, при реализации которых обычно отказываются от чисто декларативного подхода, такие как ввод-вывод, выражаются в Mercury с помощью декларативных конструкций, используя линейные типы. Предназначен для создания прикладных программ |

Функциональное программирование использует математическое понятие функции для выражения концепции действия.

**Обзор языков HOPE, FP, ML, HASKEL, CURRY, LISP, SCHEME**

| Язык программирования | Особенности ЯП |
|---|---|
| **HOPE** | Функциональный язык программирования разработанный в 1970-х годах в Эдинбургском университете.<br>***Основная цель разработки*** была создание очень простого языка программирования, который позволит составлять четкие и модернизируемые программы. Язык Hope не включает в себя оператор присваивания, что является важным его упрощением. Пользователь может свободно задавать свои собственные типы данных без потребности разрабатывать некое сложное кодирование с точки зрения типов низкого уровня. Язык очень сильно типизирован, он обеспечивает выполнение проверки типов, например обрабатывает полиморфные типы и перегруженные операторы. Функции определены рядом рекурсивных уравнений; левая сторона каждого уравнения включает в себя шаблон, который используется для того, чтобы определить какое уравнение необходимо использовать для данного аргумента. Наличие произвольных типов высшего порядка позволяет собирать функции в рекурсивные пакеты. Лениво-оцененные списки обеспечивают возможность использовать бесконечные списки, которые могут использоваться для обеспечения интерактивного ввод/вывод и параллелизма. Язык Hope также включает простое средство для модуляризации, которое может использоваться для защиты реализации абстрактного типа данных. |
| **FP** | FP был изобретен Джоном Бэкусом (John Backus) и опубликован в 1977 году в статье под «Возможно ли освободить программирование от стиля фон Неймана? Алгебра программ в функциональном стиле». Описание языка, данное в статье, является фактическим стандартом, и дальнейшей стандартизации FP не подвергался.<br>Алгебраический язык программирования. FP был придуман как модель языка, поддерживающего стиль программирования на уровне функций. Он был задуман скорее, как математическая модель, чем как действующий язык. FP использует парадигму программирования на уровне функций в качестве основной парадигмы, но имеет всего несколько любительских реализаций (достаточно сильно различающихся между собой), и не используется при разработке реальных программных комплексов. |
| **ML** | Развитая система типов и модулей. Есть и типично императивные средства: присваивания, циклы, обработка исключений. |
| **HASKEL** | Разрабатывается международным комитетом как чисто функциональный "индустриальный" язык программирования.<br>Строгая функциональность (отсутствуют императивные операторы), ленивые вычисления, полиморфная система типов и классы типов |
| **CURRY** | Функционально-логический язык, созданный на основе Haskell. Наследует большинство его свойств, но добавляет логические особенности. Всё ещё в стадии разработки. |

| | |
|---|---|
| **LISP** | Язык ЛИСП (LISP) был разработан в 1958 году американским ученым Джоном Маккарти как функциональный язык, предназначенный для обработки списков. (LISt Processing). Lisp - означает "лепетать". С появлением этого языка машина стала пока лепетать, а не говорить по-человечески.<br>Широкое распространение язык получил в конце 70-х - начале 80-х годов с появлением необходимой мощности вычислительных машин и соответствующего круга задач.<br>В основу языка положен математический аппарат:<br>• лямбда-исчисление Черча<br>• алгебра списочных структур<br>• теория рекурсивных функций<br>Основные особенности языка Лисп<br>• Представление программы и данных производится одинаково - через списки.<br>• Лисп как правило является интерпретирующим языком.<br>• Лисп безтиповый язык, это значит, что символы не связываются по умолчанию с каким-либо типом.<br>• Лисп имеет необычный синтаксис. Из-за большего числа скобок LISP расшифровывают как Lots of Idiotic Silly Parentheses.<br>• Программы, написанные на Лиспе во много раз короче, чем написанные на процедурных языках. Имеет множество диалектов.<br>Лисп одно из главных инструментальных средств систем искусственного интеллекта. Он принят как один из двух основных ЯП для министерства обороны США. Система AutoCAD разработана на Лиспе. |
| **SCHEME** | Диалект языка программирования Лисп. Scheme был разработан Гаем Стилом (Guy L. Steele) и Джеральдом Сассменом (Gerald Jay Sussman) в 1975 году.<br>Основные особенности:<br>- минимализм языка, основанный на лямбда-исчислении, которое используется для получения значительной части синтаксиса языка (11 из 23 синтаксических конструкций) из более примитивных конструкций.<br>- статическая лексическая область видимости: имя переменной относится к самой локальной области видимости; таким образом, код можно читать и интерпретировать вне зависимости от того, в каком контексте он будет вызываться.<br>- блоки, выражающиеся тремя конструкциями let, let* и letrec.<br>- “правильная” хвостовая рекурсия, позволяющая записывать итеративные алгоритмы более идиоматично, через рекурсию, и при этом оптимизирующая их так, чтобы поддерживать неограниченное количество активных вызовов.<br>- продолжения (абстрактные представления состояний программы) как объекты первого класса (процедура call-with-current-continuation).<br>- общее пространство имен для переменных и процедур. |