ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Фахриддинов Жамолиддин Шамсиддин угли
Наманганский инженерно-технологический институт Наманган, Узбекистан

АННОТАЦИЯ

Функциональное программирование — это парадигма программирования, которая рассматривает вычисления как оценку математических функций и избегает изменения состояния и переменных данных. Основные концепции функционального программирования включают неизменяемость, функции высшего порядка и рекурсию. Он способствует написанию декларативного, компактного и простого для понимания кода. Языки функционального программирования, такие как Haskell и Clojure, обеспечивают встроенную поддержку принципов функционального программирования. Функциональное программирование может привести к созданию кода, который станет более модульным, пригодным для повторного использования и менее подверженным ошибкам.

ВВЕДЕНИЕ

Функциональное программирование объясняется на примере диалекта Common Lisp языка LISP. Этот диалект наиболее распространен и имеет официальный стандарт. Common Lisp может работать не только в пакетном режиме (когда он запускается как обычная программа), но и в режиме диалога.

LISP - вероятно, первый из практически реализованных языков1, который основывался на серьезном теоретическом фундаменте и пытался поднять практику программирования до уровня концепций, а не наоборот - опустить концепции до уровня существовавшей на момент создания языка практики.

В настоящий момент функциональное программирование представлено целым семейством языков, но LISP свои позиции не сдает.

В некоторых случаях осознанное усвоение концепций даже на самом низком уровне нереально без базовых теоретических сведений. А знакомство с таким базисом, в свою очередь, стимулирует значительно более глубокий интерес к теории и способствует пониманию того, что на высшие уровни знаний и умений не подняться без овладения теорией.

Теоретической основой языка LISP является логика функциональности: комбинаторная логика или (по наименованию одного из основных понятий в наиболее популярной из нынешних ее формализаций) λ-исчисление.

Списки и функциональные выражения

Атомами в языке lisp являются числа, имена, истина T. Ложью служит пустой список nil, который в принципе атомом не является, но в языке lisp при проверке на то, является ли он атомом, выдается истина. Точно так же выдается истина и при проверке, является ли он списком. Однако все списковые операции применимы к nil, а те, которые работают с атомами, часто к нему неприменимы. Например, попытка присваивания значения выдает ошибку.

Основная операция для задания списков (list a b . . . z). Она вычисляет свои аргументы и собирает их в список. Для этой операции без вычисления аргументов есть скоропись ‘(a b . . . z). Она является частным случаем функции quote (сокращенно обозначаемой ‘), которая запрещает всякие вычисления в своем аргументе и копирует его в результат так, как он есть.

Поле зрения и поле памяти

Если не применены специальные операции блокирования вычислений, первый аргумент списка интерпретируется как функция, аргументами которой являются оставшиеся элементы списка. Это позволяет программу также задавать списком.

Рассмотрим подробнее структуру данных языка lisp. Она является двухуровневой. На верхнем уровне имеется структура списков. На нижнем находится структура информации, сопоставленной атому. Она изображена на рис. 8.1. Оба этих уровня рекурсивно ссылаются друг на друга, например, атрибуты атома являются списками.

Типы данных (в смысле программирования) в lisp есть, но они определяются динамически. В частности, если действительное число придано атому как значение, то тип атома становится float.

Модель вычислений LISP

Для lisp (как и для любого другого функционального языка) не обязательно2 говорить, где и как размещаются структуры данных (списки).

Их стоит рассматривать как сугубо математические объекты со сложной структурой, которая всегда точно указывает на текущие вычислительные элементы:

До выполнения шага вычисления - это список, включающий имя функции и ее аргументы.

Во время выполнения шага вычисления - это те фрагменты списочной структуры поля зрения, которые доступны для использования вычисляемой функцией (в частности, среди них есть список, связанный с именем функции, который определяет ее вычислительный процесс).

Общую структуру данных и программы функционального языка можно рассматривать как связный нагруженный граф, динамически изменяющийся в ходе вычислений, у которого имеются активные вершины, т. е. функции, вычисляемые в данный момент, потенциально активные вершины, соответствующие функциям, которым назначено вычисление или продолжение вычисления (отложенного, приостановленного и т. п.), и пассивные вершины, участие которых в вычислениях в данный момент не запланировано. Дуги графа отмечают связи по управлению или по данным между функциями. Такой граф далее называется абстрактным графом функциональных вычислений.

Конкретизировать такой граф и стратегию отработки активных вершин можно разными способами. При этом могут появляться разные ипостаси функционального программирования.

Для абстрактного вычислителя граф функциональных зависимостей естественно считать полем памяти функциональной программы, в котором выделено поле зрения: активные (либо активные и потенциально активные) вершины-функции со своими аргументами.

Прагматические добавления и динамическое порождение программ

Разберем возможности языка lisp в комплексе.

Выразительные средства конкретно-синтаксического представления общей структуры данных и программ языка lisp крайне скупы. Но это представление позволяет практически однозначно связать синтаксис и реализационную структуру.

Диктат линейности укоренился настолько глубоко, что даже в тех случаях, когда он мог бы быть преодолен безболезненно, языковая система чаще всего все равно строится как линейная. Это касается не только функционального стиля5.

Объекты и LISP

Стандартная надстройка над Common Lisp, имитирующая объектно-ориентированный стиль, это модуль CLOS - Common Lisp Object System. Сама по себе объектность не дает никакого выигрыша по сравнению с языком lisp, поскольку возможности динамического вычисления функций в lisp даже шире. Видимо, именно поэтому в CLOS имеются две интересных модификации, делающие его не совсем похожим на стандартное ООП.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ: (REFERENCES)

1. Agrawal, G. K., Heragu, S. S., A Survey of Automated Material Handling Systems in 300- mm Semiconductor Fabs, IEEE Transactions Semiconductor Manufacturing. mm Semiconductor Fabs, IEEE Transactions on Semiconductor Manufacturing, 19(1), 112-120, (2006)
2. Atherton, L., Atherton, R. W., Wafer Fabrication: Factory Performance and Analysis. Kluwer Academic Publishers, Boston, Dordrecht, London, (1995)
3. Balasubramanian, H., Mönch, L., Fowler, J. W., M. Pfund, Genetic Algorithm Based Genetic Algorithm Based Scheduling of Parallel Batch Machines with Incompatible Job Families to Minimize Total, 42(8), 1621-1638, (2004)
4. Barua, A., Narasimhan, R., Upasani, A., Uzsoy, R., Implementing Global Factory Schedules in Face Stoch Implementing Global Factory Schedules in the Face of Stochastic Disruptions, International Journal of Production Research, 43(4), 94. Research, 43(4), 94-109, (2005)
5. Битран Г.Р., Тирупати Д., Разработка и внедрение системы составления расписания для завода по производству пластин, Международный журнал производственных исследований, 43(4), 94-10, (2005)
6. Bixby, R., Burda, R., Miller, D., Short-Interval Detailed Production Scheduling in 300mm semiconductor manufacturing using Mixed Mixed Mixed Scheduling. Advanced Semiconductor Manufacturing Conference, 148-154, (2006)
7. Brucker, P., Knust, S. Complex Scheduling, Springer, Berlin, Heidelberg, (2006)
8. Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M.Y., Potts, C.N., Tautenhahn, T., van de Velde, S., Scheduling a Batching Machine, Journal of Scheduling, 1, 31-54, (1998)
9. Бюро, М., Дозере-Перес, С., Югма, К., Вермариен, Л., Мария, Ж.-Б., Моделирование Результаты и формализм для глобально-локального планирования в производстве полупроводников, Труды Зимней конференции по моделированию 2007, 1768-1773, (2007)
10. Абдуллаева О. С., Исманова К. Д., Мирзаев Ж. И. Организация учебной деятельности во время лекционных, практических, лабораторных занятий //Молодой ученый. – 2014. – №. 19. – С. 487-490.