УДК 004.414.2(045)

Добрина О. А.
Национальный авиационный университет, Киев

ПРОЕКТИРОВАНИЕ СИСТЕМ МЕТОДАМИ ДЕКЛАРАТИВНОГО ПРОГРАММИРОВАНИЯ

В статье предоставлен обзор современных подходов программирования и расставлен акцент на декларативном подходе для проектирования систем, как наиболее практичного для специфического класса, так и альтернативного, более компактного и упорядоченного для общего класса систем. На конкретном примере методами декларативного программирования была проведена разработка системы расписания. Приведенный анализ показал преимущества выбранного подхода именно для этой системы.

На протяжении многих тысячелетий человечество занимается накоплением, обработкой и передачей знаний. Для этих целей непрерывно изобретаются новые средства и совершенствуются старыед. Большую роль в технологии обработки знаний сыграло появление компьютеров.

В октябре 1981 года Японское министерство международной торговли и промышленности объявило о создании исследовательской организации — Института по разработке методов создания компьютеров нового поколения (Institute for New Generation Computer Technology Research Center). Целью данного проекта было создание систем обработки информации, базирующихся на знаниях. Предполагалось, что эти системы будут обеспечивать простоту управления за счет возможности общения с пользователями при помощи естественного языка – человек сможет использовать ЭВМ пятого поколения так же легко, как любые бытовые электроприборы типа телевизора, магнитофона и пылесоса. Японцы надеялись, что им удастся не подстраивать мышление человека под принципы функционирования компьютеров, а приблизить работу компьютера к тому, как мыслит человек, отойдя при этом от фон неймановской архитектуры компьютеров. В 1991 году предполагалось создать первый прототип компьютеров пятого поколения, что так и не удалось осуществить.

Так как все человечество стремится перейти от информационного общества к обществу знаний, оно должно найти такие решения, которые докажут, что оно поистине к этому готово. Именно поиски новых технологий или усовершенствования старых для обработки колоссальных объемов информации и опыта, собранного человечеством, является актуальным вопросом в настоящее время.

Объем накопленных знаний и опыта человечества колоссальный. Но так как люди не всегда умели хранить и правильно оформлять все это, много важной информации было утеряно. Поэтому для общества будущего, которое формируется сейчас, возникает ряд вопросов: что такое знание? как им оперировать? как его хранить?

Существующие методы решения проблемы

С появлением ЭВМ работа со значительными объемами информации стала возможной. Используя такой инструментарий как языки программирования, многие процессы по обработке информации, в каком виде она ни поступала, стали автоматизированными. Появилось много подходов относительно самого процесса, среди которых можно выделить два основых: императивный и декларативный.

Традиционно под программой понимают последовательность операторов (команд, выполняемых компьютером). Этот стиль программирования принято называть императивным. Программируя в императивном стиле, программист должен объяснить компьютеру, как нужно решать задачу.

Противоположный ему стиль программирования – так называемый декларативный стиль, в котором программа представляет собой совокупность утверждений, описывающих фрагмент предметной области или сложившуюся ситуацию. Программируя в декларативном стиле, программист должен описать, что нужно решать. Данный подход возникает в 60-х годах, который до сих пор успешно конкурирует с императивным.

На начальном этапе развития декларативным языкам программирования было сложно конкурировать с императивными в силу объективных трудностей эффективной реализации трансляторов. Программы работали медленнее, однако они

могли решать более абстрактные задачи с меньшими трудозатратами.

В 70-х годах возникла еще одна ветвь языков декларативного программирования, связанная с проектами в области искусственного интеллекта, а именно языки логического программирования.

Согласно логическому подходу к программированию, программа представляет собой совокупность правил или логических высказываний. Кроме того, в программе допустимы логические причинно-следственные связи, в частности, на основе операции импликации.

Таким образом, языки логического программирования базируются на классической логике и применимы для систем логического вывода, в частности, для так называемых экспертных систем. На языках логического программирования естественно формализуется логика поведения, и они применимы для описаний правил принятия решений, например, в системах, ориентированных на поддержку бизнеса.

В 1965 году в работе "A machine oriented logic based on the resolution principle", опубликованной в 12 номере журнала "Journal of the ACM", Дж Робинсон представил метод автоматического поиска доказательства теорем в исчислении предикатов первого порядка, получивший название «принцип резолюции». На самом деле, идея данного метода была предложена Эрбраном в 1931 году, когда еще не было компьютеров (Herbrand, "Une methode de demonstration", These, Paris, 1931). Робинсон модифицировал этот метод так, что он стал пригоден для автоматического, компьютерного использования, и, кроме того, разработал эффективный алгоритм унификации, составляющий базис его метода.

В 1973 году «группа искусственного интеллекта» во главе с Аланом Колмероэ создала в Марсельском университете программу, предназначенную для доказательства теорем. Программа доказательства теорем получила название Prolog (от Programmation en Logique). Она и послужила прообразом Пролога. Программа была написана на Фортране и работала довольно медленно.

В 1976 г. Ковальский вместе с его коллегой Маартеном ван Эмденом предложил два подхода к прочтению текстов логических программ: процедурный и декларативный.

В 1977 году в Эдинбурге Уоррен и Перейра создали очень эффективный компилятор логического языка Пролог для ЭВМ DEC–10, который послужил прототипом для многих последующих реализаций компиляторов декларативных языков. Что интересно, компилятор был написан на самом Прологе.

В 1980 году Кларк и Маккейб в Великобритании разработали версию декларативного языка для персональных ЭВМ.

На сегодня существует довольно много реализаций Пролога. Наиболее известные из них следующие: BinProlog, AMZI-Prolog, Arity Prolog, CProlog, Micro Prolog, Prolog-2, Quintus Prolog, Strawberry Prolog, SWI Prolog, UNSW Prolog и т. д. В странах СНГ были разработаны такие версии Пролога как Пролог-Д (Сергей Григорьев), Акторный Пролог (Алексей Морозов), а также Флэнг (А Манцивода, Вячеслав Петухин).

Особенности

Императивные языки основаны на фон неймановской модели вычислений компьютера. Решая задачу, императивный программист вначале создает модель в некоторой формальной системе, а затем переписывает решение на императивный язык программирования в терминах компьютера. Но, во-первых, для человека рассуждать в терминах компьютера довольно неестественно. Во-вторых, последний этап этой деятельности (переписывание решения на язык программирования) по сути дела не имеет отношения к решению исходной задачи. Очень часто императивные программисты даже разделяют работу в соответствии с двумя описанными выше этапами. Одни люди, постановщики задач, придумывают решение задачи, а другие, кодировщики, переводят это решение на язык программирования.

В основе декларативных языков лежит формализованная человеческая логика. Человек лишь описывает решаемую задачу, а поиском решения занимается императивная система программирования. В итоге получаем значительно большую скорость разработки приложений, значительно меньший размер исходного кода, легкость записи знаний на декларативных языках, более понятные, по сравнению с императивными языками, программы.

Отличительной особенностью данного подхода является то, что любая программа, написанная на таком языке, может интерпретироваться как функция с одним или несколькими аргументами. Такой подход дает возможность прозрачного моделирования текста программ математическими средствами, а значит, весьма интересен с теоретической точки зрения. Сложные

программы при таком подходе строятся посредством агрегирования функций. При этом текст программы представляет собой функцию, некоторые аргументы которой можно также рассматривать как функции. Таким образом, повторное использование кода сводится к вызову ранее описанной функции, структура которой, в отличие от процедуры императивного языка, математически прозрачна.

Более того, типы отдельных функций, используемых в функциональных языках, могут быть переменными. Таким образом, обеспечивается возможность обработки разнородных данных (например, упорядочение элементов списка по возрастанию для целых чисел, отдельных символов и строк) или полиморфизм.

При программировании на декларативном языке усилия программиста должны быть направлены на описание логической модели фрагмента предметной области решаемой задачи в терминах объектов предметной области, их свойств и отношений между собой, а не деталей программной реализации. Но следует учесть, что декларативные языки требуют иного стиля мышления, отказа от стереотипов императивного программирования.

Достоинства и недостатки

Декларативный подход существенно проще и прозрачнее формализуется математическими средствами. Следовательно, программы проще проверять на наличие ошибок (тестировать), а также на соответствие заданной технической спецификации (верифицировать).

Высокая степень абстракции также является преимуществом данного подхода. Фактически, программист оперирует не набором инструкций, а абстрактными понятиями, которые могут быть достаточно обобщенными.

Еще одним важным преимуществом реализации языков функционального программирования является автоматизированное динамическое распределение памяти компьютера для хранения данных.Таким образом, при создании программ на функциональных языках программист сосредотачивается на области исследований (предметной области) и в меньшей степени заботится о рутинных операциях (обеспечении правильного с точки зрения компьютера представления данных, «сборке мусора» и т.д.).

Поскольку функция является естественным формализмом для языков функционального программирования, реализация различных аспектов программирования, связанных с функциями, существенно упрощается. В частности, становится прозрачным написание рекурсивных функций, т.е. функций, вызывающих самих себя в качестве аргумента. Кроме того, естественной становится и реализация обработки рекурсивных структур данных (например, списков – базовых элементов, скажем, для языков семейства LISP, деревьев и др.).

Важным преимуществом такого подхода является достаточно высокий уровень машинной независимости, а также возможность откатов – возвращения к предыдущей подцели при отрицательном результате анализа одного из вариантов в процессе поиска решения (скажем, очередного хода при игре в шахматы), что избавляет от необходимости поиска решения путем полного перебора вариантов и увеличивает эффективность реализации.

Естественно, языки функционального программирования не лишены недостатков. Часто к ним относят нелинейную структуру программы и относительно невысокую эффективность реализации. Однако первый недостаток достаточно субъективен, а второй успешно преодолен современными реализациями, в частности, рядом последних трансляторов языка SML, включая и компилятор для среды Microsoft .NET.

Еще одним недостатком практического характера является сложность эффективной реализации для принятия решений в реальном времени, скажем, для систем жизнеобеспечения. Нелинейность структуры программы является особенностью декларативного подхода и, строго говоря, представляет собой оригинальную особенность, а не объективный недостаток.

Области применения и современные примеры использования

Основные области применения декларативного подхода:

- быстрая разработка прототипов прикладных программ;
- автоматический перевод с одного языка на другой;
- создание естественно-языковых интерфейсов для существующих систем;
- символьные вычисления для решения уравнений, дифференцирования и интегрирования;
- проектирования динамических реляционных баз данных;
- экспертные системы и оболочки экспертных систем;
- автоматизированное управление производственными процессами;

- автоматическое доказательство теорем;
- полуавтоматическое составление расписаний;
- системы автоматизированного проектирования;
- базируется на знаниях программное обеспечение;

Области, для которых Пролог не предназначен: большой объем арифметических вычислений (обработка аудио, видео и т.д.); написания драйверов.

В качестве примера применения декларативного подхода к созданию ПО, рассмотрим программу MYCIN, которая создана для диагностирования больного и назначение правильного курса лечения. Она включает в себя около 600 правил и реализована на языке Lisp. Такая система дает допустимую терапию до 70 %, что лучше, чем у некоторых экспертов по инфекционным заболеваниям.

Группа проектировщиков занималась реализацией системы, автоматизирующей работу диспечера для составления расписания ВУЗа. Такая задача принадлежит области теории расписаний, которая возникла буквально на несколько лет раньше, чем декларативный подход. Реализация данной задачи, а именно составления расписания для ВУЗа достаточно просто описывается используя именно этот подход. Так как декларативные языки имеют прекрасное взаимодействие с базами данных (Prolog также считается реляционным языком, и система создается именно на этом логиечском языке программирования), то в систему с фиксированным количеством правил поступают данные для составления расписания (аудитории, нагрузки преподавателей и студентов) в виде динамической базы данных. Сам процесс генерации расписания проходит в два этапа. Так как нам абсолютно не важны все варианты получившегося расписания (так как число возможных вариантов превышает 2^{400}, то на просчет абсолютно всех вариантов у нас уйдет неоправданное количество времени и ресурсов). На первом этапе создается черновой вариант расписания, то есть имеющиеся предметы «разбрасываются» без учета окон. На втором этапе предлагается редактирование расписания в одном из возможных режимов – в ручном, когда диспетчер сам устраняет окна; автоматическом, когда всю работу по устранению окон и оптимизацию расписания перекладывают на систему; полуавтоматическом, когда система дает возможные варианты устранения окон, а диспетчер сам принимает решение.

На наш вгляд, именно декларативный подход к созданию программ и языка декларативного программирования является перспективным направлением, его развитие и вклад в него времени и усилий принесет принципиально новое программное обеспечение, что позволит работать со знаниями как таковыми.

Список литературы

1. *Братко И.* Алгоритмы искуственного интеллекта на языке. М.: Пролог,1990. – 560 с.

2. *Харрисон Филд.* Функциональное программирование. М.: Мир, 1993. – 640 с.

3. *Шумова Л.В., Носова Т.Н.* Декларативный подход в функциональном программировании. Магнитогорск: МГТУ, 1999. – 65 с.

Научный руководитель – Добрина Е. В., ст. преподаватель