Андрей Викторович Столяров

Библиотека InteLib — инструмент мультипарадигмального программирования

Настоящий документ представляет собой тезисы доклада, сделанного на II конференции разработчиков свободных программ «На Протве» 26 июля 2005 года. Просьба при цитировании ссылаться на версию, опубликованную в сборнике тезисов докладов конференции с использованием следующей библиографической информации:

> Андрей Столяров. **Библиотека InteLib — инструмент мультипарадигмального программирования.** // Тезисы докладов II II конференции разработчиков свободных программ «На Протве», г. Обнинск, июль 2005 г.

Для пользователей LaTeX приводится библиографическая информация в формате `bibtex`:

```
@INCOLLECTION{stolyarov:intelib2005,
        AUTHOR={{Андрей Столяров}},
        TITLE="Библиотека InteLib~--- инструмент
               мультипарадигмального программирования",
        BOOKTITLE = "Тезисы докладов II конференции
               разработчиков свободных программ {{"<На Протве">}}",
        MONTH = "июль",
        YEAR = "2005",
        PAGES = "56--62",
        ADDRESS = "Обнинск"
}
```

Библиотека InteLib — инструмент мультипарадигмального программирования

Андрей Столяров
МГУ им. Ломоносова, ВМиК

Аннотация

Доклад посвящен практическому решению проблемы интеграции выразительных механизмов языков программирования принципиально различной природы в рамках одного проекта. Дается краткая классификация существующих подходов и указываются их недостатки. В качестве решения проблемы предлагается подход, получивший название метода непосредственной интеграции, и описывается библиотека InteLib, позволяющая, не выходя за рамки языка C++, записывать выражения, семантически эквивалентные и синтаксически близкие конструкциям языка Lisp (а в перспективе – и других языков, таких как Рефал, Prolog, Datalog и др.)

Все многообразие существующих в настоящее время языков программирования можно при желании классифицировать по признаку господствующих традиций осмысления программы, принятых в сообществе программистов, пишущих на данном языке. Так, традиционные императивные языки стимулируют осмысление программы как набора приказов по изменению тех или иных данных; объектно-ориентированные языки позволяют представлять программу как набор абстрактных объектов, обменивающихся сообщениями; при работе на языках логического программирования программа представляется как система фактов и аксиом, а исполнение программы состоит в попытке опровержения заданного утверждения; наконец, функциональные языки программирования описывают программу как набор функций, вычисляющих значение по заданным аргументам, а выполнение программы представляется вычислением значения некоторого выражения.

Менее фундаментальные особенности языков программирования также накладывают определенный отпечаток на мышление программиста; прекрасный пример этого приводит Тимоти Бадд в книге [1].

Решение вопроса о «наиболее правильном» или «наиболее удобном» образе мышления существенно зависит от решаемой задачи. Так, на языке Lisp удобно обрабатывать формульные и другие слабо структурированные данные, но до крайности неудобно строить диалоговые системы на основе событийно-ориентированных оконных интерфейсов, а для языка Java дела обстоят ровно противоположным образом.

Типичной является ситуация, когда в силу тех или иных причин для реализации крупного проекта выбирается тот или иной язык программирования (чаще всего выбор падает на императивный или императивно-объектный язык), при этом в проекте возникают небольшие подзадачи, которые было бы гораздо удобнее решать на совершенно другом языке.

```
(defun isomorphic (tree1 tree2)
 (cond ((atom tree1) (atom tree2))
       ((atom tree2) NIL)
       (t (and (isomorphic (car tree1)
                           (car tree2))
               (isomorphic (cdr tree1)
                           (cdr tree2))))))
```

Рис. 1: Код функции ISOMORPHIC на языке Лисп

Традиционные средства интеграции разнородных парадигм программирования можно условно разделить на несколько основных подходов:

1. создание нового языка программирования (возможно, как расширения одного из существующих)
2. встраиваемые интерпретаторы
3. расширяемые интерпретаторы
4. трансляция из одного языка в другой
5. пакеты взаимосвязанных программ
6. сборка объектных модулей, полученных из разных систем программирования
7. реализация частных возможностей

Эти подходы подробно анализируются в работе [2]. К сожалению, каждый из них имеет определенные недостатки, существенно ограничивающие области их применения.

Являющаяся основным предметом рассмотрения данного доклада библиотека InteLib [4] реализует качественно иной подход к решению проблемы интеграции разнородных языковых выразительных средств.

Благодаря поддержке в языке C++ концепции абстрактных типов данных и возможности переопределения символов стандартных операций для объектов пользовательских типов язык C++ может рассматриваться как язык *моделирования алгебр*. С другой стороны, семантика языка Lisp может рассматриваться как *алгебра S-выражений*, одной из операций которой является *вычисление* S-выражения.

Чтобы представить конструкции языка Lisp средствами C++, достаточно описать класс, способный хранить (инкапсулировать) S-выражение любого поддерживаемого типа и дать набор операций для построения точечных пар и списков из атомарных S-выражений. Используя свойство полиморфности объектов, можно построить иерархию классов с общим предком для хранения различных типов S-выражений, что позволит вводить новые типы S-выражений.

В результате получаем возможность записи в языке C++ выражений, семантически эквивалентных и синтаксически близких конструкциям языка Lisp (или другого моделируемого языка).

Сказанное можно проиллюстрировать на примере функции, исходный код которой в синтаксисе языка Lisp приведен на рис. 1.

```
LSymbol ISOMORPHIC("ISOMORPHIC");
void LispInit_isomorphic() {
  static LSymbol TREE1("TREE1");
  static LSymbol TREE2("TREE2");
  (L|DEFUN, ISOMORPHIC, (L|TREE1, TREE2),
    (L|COND,
      (L|(L|ATOM, TREE1), (L|ATOM, TREE2)),
         (L|(L|ATOM, TREE2), NIL),
         (L|T, (L|AND,
           (L|ISOMORPHIC, (L|CAR, TREE1),
                          (L|CAR, TREE2)),
           (L|ISOMORPHIC, (L|CDR, TREE1),
                          (L|CDR, TREE2)))))).Evaluate();
}
```

Рис. 2: Код функции ISOMORPHIC на языке Си++

Следует особо подчеркнуть, что код, приведенный на рис. 2, является корректным кодом на языке C++. Трансляция этого кода производится обычным компилятором C++; никакого предварительного преобразования кода не требуется.

Библиотека состоит из двух слоёв. На первом из них вводятся S-выражения как структуры данных, на втором на базе введенных S-выражений строятся вычислитли, моделирующие семантику соответствующих языков программирования (в настоящее время - языков Lisp и Refal). Первый слой позволяет, помимо прочего, использовать концепцию S-выражений для построения универсальных гетерогенных контейнеров [3].

Список литературы

[1] T. A. Budd. *Multy-Paradigm Programming in LEDA*. Addison-Wesley, Reading, Massachusets, 1995.

[2] А. Столяров. Интеграция разнородных языковых механизмов в рамках одного языка программирования. *Диссертация на соискание степени канд. физ.-мат. наук*, Москва, 2002.

[3] Stolyarov, A. V. A framework of heterogenous dynamic data structures for object-oriented environment: the S-expression model In *Knowledge-Based Software Engineering. Proceedings of the 6th JCKBSE*, vol. 108 of *Frontiers in Artificial Intelligence and Applications*, pages 75–82, Protvino, Russia, August 2004. IOS Press.

[4] Официальный сайт проекта InteLib. http://www.intelib.org