Internet-учебный курс для обучения языку Лисп

Дацун Н.Н., Хован А.П.
Донецкий Государственный Технический Университет, Донецк, Украина
e-mail: datsun@pmi.dgtu.donetsk.ua

Abstract

Цель: создание учебного Internet-курса для обучения функциональному программированию студентов специальности “Программное обеспечение”. *Задачи:* разработка структуры и создание *дистанционного Автоматизированного Учебного Курса* (АУК) программированию на языке ЛИСП с системой тестирования и *интерпретатора языка COMMON LISP* для встраивания в Internet-курс. *Методы:* в АУК реализованы требования Б.Скинера, при разработке интерпретатора ЛИСПа выполнено определение подмножества этого языка в терминах формальных грамматик, выделено множество встроенных функций, опеределены структура и функции инструментальных средств для работы со списком символов языка. *Результаты:* разработан АУК средствами HTML и VBScript и интерпретатор TeahLisp в виде элемента управления ActiveX средствами C++.

1. Введение

Методы функционального программирования традиционно используются для решения задач искусственного интеллекта (ИИ): создание систем символьных вычислений, экспертных систем, обработка естественного языка и эвристические алгоритмы. Если раньше иссследования в этих областях проводились в рамках университетских проектов, то сегодня технологии ИИ перестали быть чисто академическими и успешно применяются при создании коммерческих продуктов. Язык функционального программирования Лисп вышел за рамки языка ИИ и активно применяется для создания прикладных программ (система символьных вычислений Mathematica и САПР AutoCAD). О постоянном интересе к этому языку свидетельствует наличие нескольких фирм, разрабатывающих трансляторы Лиспа, и тот факт, что вместе с трансляторами поставляются инструментальные средства для создания Интернет-приложений, интеграции с другими языками программирования (ЯП), доступа к базам данных (БД). Изменились и сами трансляторы языка – вместо традиционного интерпретатора каждая из предлагаемых систем разработки программ на Лиспе представляет собой интегрированную среду программирования и включает Лисп-ориентированный редактор, компилятор, профайлер, библиотеки для создания оконных интерфейсов и пр. В табл. 1 приводятся сведения о современных трансляторах Лиспа.

Table 1. Характеристики трансляторов Лиспа

Название	**Редактор**			**Операционная система**	**Интеграция с другими ЯП**	
Разработчик **Источник**	**Отладчик**				**Средства доступа к БД**	
	Компилятор				**Другие возможности**	
Allegro CL	+	+	+	Windows	Java, C++	-
Franz Inc. [1]					Поддержка HTML и XML	
MCL / Digitool [2]	+	+	+	MacOS	-	-
					Оконный интерфейс CLIM	
LispWorks / Harlequin [3]	+	+	+	Windows, Linux, FreeBSD, SCO Unix	Prolog	ODBC, SQL
					Интерфейс CLIM, встроенный интерпретатор Пролога	
CMUCL / Университет Карнеги-Мелона [4]	+	+	+	FreeBSD, Linux, SunOS, Irix	-	-
					Поддержка оконной системы X11, профайлер	

Активное использование функционального программирования говорит о том, что профессиональный программист должен владеть методами и средствами этого стиля программирования, что делает актуальной задачу обучения языку Лисп. Коммерческие системы, предназначенные для обучения функциональному программированию, отсутствуют. Систем, разработанных образовательными учреждениями, немного, что свидетельствует о “штучной” (или скорее “элитарной”) потребности в подготовке специалистов по этому разделу программирования. Существующие системы значительно отличаются друг от друга. Система [5] предназначена для освоения методов рекурсивного программирования на Лиспе. Стратегия учебного курса *Learning Lisp* (Association of Lisp Users) [6] такова: концепции Лиспа являются достаточно простыми и легко осваиваются, поэтому основное внимание уделяется изучению примеров программ на Лиспе (изучение языка производится путем разбора текстов программ). *LISP TUTOR* (университет Tulane) [7] можно считать дальнейшим развитием раннее известной одноименной системы [8, 9]. Разработчики LISP TUTOR напротив, считают изучение Лиспа сложным для начинающих программистов, особенно не знакомых с функциональным стилем программирования. LISP TUTOR – это интерактивный курс обучения Лиспу на базе Internet. Выбор тем и порядок их изучения определяется пользователем,

причем, обучаемый может проверить степень усвоения текущей темы перед переходом к новой. В конце курса обучаемому предлагается пройти тест для общей оценки полученных знаний.. Для нас интерес представляют сценарии взаимодействия LISP TUTOR с обучаемым. Реализованы три варианта:

- при изучении отдельной темы система предоставляет теоретический материал и несколько примеров, просит обучаемого запустить интерпретатор Лиспа и выполнить несколько упражнений и предполагает, что обучаемый действительно сделал это;
- обучающая система сама запускает интерпретатор, а затем ожидает, когда обучаемый закончит практическое задание;
- система получает написанный студентом код и самостоятельно взаимодействует с интерпретатором Лиспа для проверки результатов.

На сегодняшний день не существует систем дистанционного обучения языку Лисп, которые имели бы в своем составе транслятор языка в качестве “решателя задач” [10]. Такая интеграция могла бы повысить эффективность обучающей системы, так как обучаемый мог бы немедленно попытаться применить полученную порцию знаний на практике. Наиболее близко к этой идее подошли разработчики системы LISP TUTOR, однако и в этом случае обучающая система взаимодействует со внешним интерпретатором Лиспа. Поэтому актуальной является задача разработки дистанционного АУК функциональному программированию с интерпретатором языка COMMON LISP, который должен быть реализован в виде элемента управления ActiveX для встраивания в Internet-курс.

2. Структура дистанционного АУК языку Лисп

По Б.Скинеру [10] АУК должен удовлетворять таким требованиям: учебный материал должен быть разбит на порции, которые предъявляются последовательно и линейно по нарастанию степени сложности; один и тот же вопрос должен быть рассмотрен неоднократно под разными углами зрения; при проверке степени усвоения учебного материала необходимо использовать стратегии подкрепления для закрепления рефлекса успеха.

Современные Internet- и гипертекстовые технологии провоцируют обучаемого изучать материал не в линейной последовательности, а произвольно, активно используя глоссарии, словари, справочники или библиотеки примеров, а также гипертекстовые ссылки. Но и в дистанционном АУК требования Б.Скинера следует выдерживать. При разработке АУК языка Лисп были выделены темы “Общие сведения о языке” и “Средства языка”. В свою очередь тема 2 разбита на такие уроки: “S-выражения, атомы, списки”, “Работа со списками”, “Предикаты”, “Функции и функционалы”, “Управляющие структуры”. Каждый урок содержит определение понятий и примеры использования рассмотренной конструкции языка. Обучаемый имеет возможность вызвать интерпретатор языка COMMON LISP при работе с АУК для непосредственного выполнения примеров.

Реализация АУК выполнена средствами HTML и VBScript.

3. Интерпретатор TeachLisp языка COMMON LISP

Разработка транслятора языка COMMON LISP требует решения следующих задач: определение лексики и синтаксиса языка; разработка лексического, синтаксического и семантического анализаторов; выбор структур данных для внутреннего представления (моделирования) лисповской памяти; реализация алгоритма интерпретации конструкций языка; взаимодействие с АУК.

3.1. Формальное определение подмножества Лисп в терминах грамматик

Синтаксис Лиспа предельно прост: программа является *S-выражением,* которое есть либо *атом,* либо *список*, содержащий один или несколько атомов или списков внутри круглых скобок (точечная пара трактуется как общий случай списка). Значительно облегчает задачу построения транслятора и то, что Лисп является *нетипизированным* языком.

Для формального определения *атома* Лиспа достаточно грамматики типа 3 по Хомскому. Однако регулярные выражения не позволяют задавать шаблоны скобок произвольной длины. Поэтому формальное определение *списка* в Лиспе задается контекстно-свободной грамматикой (типа 2):

STRATOM → letter | letter REST
REST → letter | digit | letter REST | digit REST
INTATOM → sign INT | INT
INT → digit | digit INT
FLOATATOM → INTATOM dot INT | INTATOM dot EXP INTATOM
EXP → letter_e | letter_E
LIST → open_p SEXPR close_p | nil
SEXPR → LIST | ATOM | DOTPAIR
ATOM → STRATOM | INTATOM | FLOATATOM
DOTPAIR → open_p SEXPR dot SEXPR close_p
letter = ‘a’..’z’
digit = 0..9
sign = + | -
dot = ‘.’
letter_e = ‘e’
letter_E = ‘E’
open_p = ‘(‘
close_p = ‘)’
nil = ‘NIL’

3.2. Разработка лексического, синтаксического и семантического анализаторов интерпретатора

Первая стадия трансляции программы на Лиспе - лексический анализ - проста из-за простоты лексем и минимального количества

символов-разделителей (пробел, круглые скобки и символ ‘). Поэтому *сканер* Лиспа реализуется детерминированным конечным автоматом, который соответствует языку типа 3. Собранная на этой фазе последовательность лексем подается на грамматический разбор - вторую фазу трансляции.

Синтаксический анализ программы на Лиспе необходимо чередовать с семантическим. Сначала синтаксический анализатор (*парсер*) идентифицирует последовательность лексем, формируя синтаксическую единицу (S-выражение, применение функции и т.п.). Затем для обработки последней вызывается семантический анализатор. Для связи синтаксического и семантического анализаторов используется стек. Синтаксический анализатор помещает в стек различные элементы найденной синтаксической единицы, а затем они выбираются и обрабатываются семантическим анализатором.

Семантический анализ является центральной фазой трансляции. В общем случае семантический анализатор может породить выполняемый объектный код. Но в реализуемом интерпретаторе Лиспа выходом этой стадии трансляции служит внутренняя форма выполняемой программы в виде списка Лисп-ячеек. Семантический анализатор разделен на ряд более мелких семантических анализаторов, каждый из которых предназначается для одного конкретного типа программной конструкции. Разработаны отдельные анализаторы для обработки арифметических выражений, инструкций типа SETQ и блокированных (невычисляемых S-выражений). Соответствующий семантический анализатор вызывается синтаксическим анализатором, как только последний распознает синтаксическую единицу, требующую обработки.

Семантические анализаторы взаимодействуют между собой посредством информации, хранящейся в *таблице символов* и *списке ассоциаций.* Например, семантический анализатор, обрабатывающий инструкцию SETQ, вносит в *таблицу символов* атом, который связывается со значением (если он там еще не существовал), а в *список ассоциаций* – пару “атом”-“значение”. Позже какой-либо семантический анализатор, обрабатывающий данный атом, может использовать информацию из списка ассоциаций.

3.3. Структуры данных интерпретатора TeachLisp

Внутренними структурам данных интерпретатора TeachLisp являются Лисп-ячейки, таблица символов и список ассоциаций.

Внутреннее представление Лисп-программы. Лисп-программа из текстовой формы транслируется во внутреннее представление в виде ячеек памяти (*Лисп-ячейка*, или списочная ячейка). Лисп-ячейка представляет собой структуру, которая может хранить строку, число, функцию, указатели на другие ячейки или служебную информацию интерпретатора. Лисп-ячейка состоит из следующих полей данных: (см. табл.2):

Table 2. Поля Лисп-ячейки

Имя	Описание
Type	поле, хранящее тип ячейки
Flag	флажок, указывающий область видимости ячейки
Car	поле CAR списочной ячейки

Cdr	поле CDR списочной ячейки
Str	строка символов
Int	целое число
Float	вещественное число
Plist	указатель на список свойств

Поле *Type* указывает какую информацию хранит ячейка и может принимать значения: STRATOM (строковый атом), INTATOM (целое число), FLOATATOM (вещественное число), CONS (указатели на другие ячейки), DOTPAIR (точечная пара), ERRMSG (сообщение об ошибке;), SCOPEMARKER (ограничитель области фактических параметров функции). Поле *Flag* показывает, доступна ли ячейка только внутри функции, выполняемой в данный момент (т.е. ячейка хранит переданный функции фактический параметр), или всем другим функциям. Поля *Car* и *Cdr* содержат указатели на другие Лисп-ячейки. Поля *Str*, *Int* и *Float* хранят соответственно строковое, целочисленное и вещественное значение. *PList* содержит указатель на список свойств символа. Среди свойств симола может быть строка, определяющая его внешний вид; тело функции, ассоциированной с этим символом и т.д.

Таблица символов хранит все встретившиеся или созданные программой атомы. В процессе трансляции программы во внутреннее представление каждый встретившийся в программе атом ищется в таблице символов (если атом уже существует, то извлекается указатель на него, в противном случае создается новый атом, который добавляется в таблицу символов).

Список ассоциаций (А-список) представляет собой среду ссылок. Это список Лиспа, состоящий из пар, каждый элемент которой является указателем на слово, представляющее атом (поле *Car),* и его текущую *ассоциацию-значение* (поле *Cdr).* Обработка ссылок подчинена правилу последней ассоциации: поиск по А-списку от начала (самые последние ассоциации) к концу (наиболее старые ассоциации) до тех пор, пока не будет найдена соответствующая ассоциация. Модификация А-списка во время выполнения программы происходит: при вызове функции (связывание формальных параметров и значений фактических параметров с добавленем этих пар в начало А-списка); при завершении вычисления функции (ассоциации, добавленные в А-список, исключаются из него); при непосредственной модификации последней ассоциации атома с помощью функции SETQ.

3.4. Функции Лиспа, реализованные в интерпретаторе

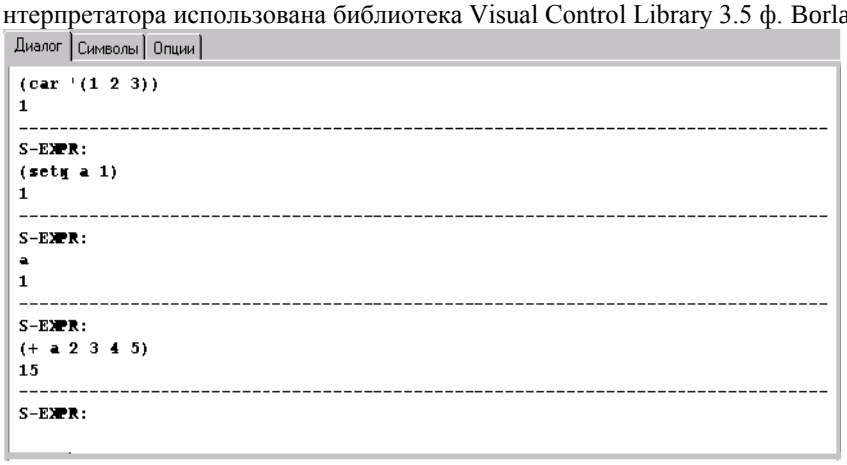
Итерпретатор поддерживает следующие функции языка COMMON LISP: функции для работы со списками (*car, cdr, nth, cons, list, append, list-length, assoc*); функции управления вычислением (*eval, quote*); применяющие и отображающие функционалы (*apply, funcall, mapcar, maplist*); функции определения символов *(setq, defun*); функции для работы со свойствами символов (*putprop, remprop, get, symbol-plist, symbol-value*); функции управления вычислительным процессом (*if, cond, load*); арифметические функции (+, -, *, /); логические функции (*and, or, not*); функции сравнения чисел *(*>, <, = , /=, >=, <=*);* предикаты проверки: (*atom, null, eq, equalp, numberp*). Перечисленные функции образуют первую группу

подпрограмм – это встроенные функции Липа (*SUBR*), для каждой из которых существует функция-реализатор. Второй тип функций, разрешенных в интерпретаторе – функции, определяемые пользователем (*EXPR*).

3.5. TeachLisp как элемент управления ActiveX

Интерпретатор TeachLisp должен обладать возможностью объединяться с другими программными комплексами, в частности в дистанционным АУК, реализованным средствами HTML и VBScript. Для этого необходима его реализация с использованием COM-технологии. Для наших целей наиболее подходящим способом будет реализация интерпретатора в форме *элемента управления ActiveX* (реализация в форме сервера автоматизации также является приемлемой, но в этом случае интерпретатор должен будет только предоставлять доступ к своим функциям и не обязан включать в себя какой-либо интерфейс пользователя). Элемент управления, содержащий интерпретатор Лиспа, должен поддерживать несколько стандартных интерфейсов – IUnknown, IDispatch, IoleControl; не требуется реализация никаких более сложных интерфейсов. Пользователь работает с интерпретатором напрямую, используя его собственный интерфейс пользователя (см. рис. 1). В этом случае информация от пользователя попадает непосредственно к интерпретатору, минуя программу-контейнер. Следовательно, нет необходимости создавать отдельный интерфейс для обмена информацией между программой-контейнером и элементом управления. При работе пользователя со списком символов пользователь должен видеть пары “имя символа (атом)” – “значение символа (ассоциация)”, а также список свойств выбранного символа. Кроме того, в режиме списка символов (см. рис.1) можно выполнять редактирование свойств символа.

Реализация интерпретатора TeachLisp выполнена в среде MS Visual C++. TeachLisp содержит модули *Parser* (анализаторы), *AList* (управление памятью виртуальной Лисп-машины), *Helpers* (вспомогательне функции), *Realizers* (программный код функций-реализаторов), *EnvIface* (взаимодействие интерпретатора с элементом управления ActiveX). При реализации интерфейса нтерпретатора использована библиотека Visual Control Library 3.5 ф. Borland.



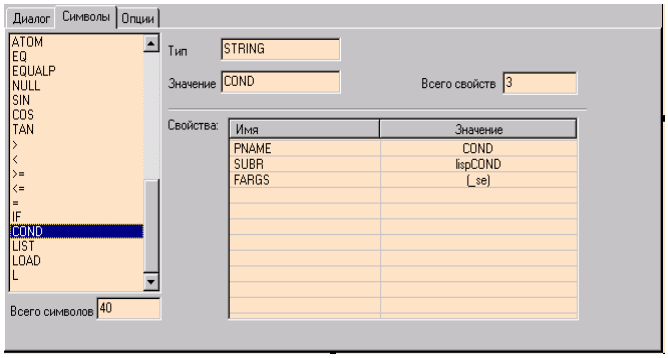


Fig. 1. Работа интерпретатора в режимах “Диалог” и “Символы”

4. Заключение

Internet-курс для обучения языку COMMON LISP с интерпретатором TeachLISP предназначен для использования при дистанционном обучении студентов специальности “Программное обеспечение автоматизированных систем”, а также для самостоятельной работы студентов очной формы обучения этой специальности.

Основные направления дальнейших исследований:

- комплексное тестирование интерпретатора при выполнении контрольных и лабораторных работ студентами;
- расширение подмножества встроенных функций COMMON LISP типа SUBP, реализованных в интерпретаторе TeachLisp;
- развитие системы тестирования (переход от однопользовательского режима работы к сетевому варианту) и совершенствование стратегии подкрепления;
- переход от АУК к адаптивной обучающей системе.

Литература

1. http://www.franz.com
2. http://www.digitool.com
3. http://services.harlequin.com/lisp/lww.html
4. http://www.cons.org/cmucl/
5. Watanabe T., Nishiguchi N., Sugie N. A tutoring method for composing LISP recursive programs. – Proc. EW-ED’94. – Crimea, 1994. - Part.1. - P.12.
6. http://www.elwoodcorp.com/alu/
7. http://www.eecs.tulane.edu/webcourses/lisp/

8. Мельников И.А., Монкус В.В., Тамм Б.Г. Обзор и анализ зарубежных компьютерных обучающих систем в области программирования// Прикладная информатика. – М.: ФиС, 1984. - Вып. 15. - с.131-153.
9. Андерсон Дж.Р., Рейзер Б.Дж. Учитель Лиспа/ Реальность и прогнозы искусственного интеллекта. – М.: Мир, 1987. – с.24-27.
10. Петрушин В. А.. Экспертно-обучающие системы – К.: Наукова думка, 1992. – 196с.
11. Пратт Т. Языки программирования: разработка и реализация. – М.: Мир, 1979. - 574 с.