Министерство образования и науки Российской Федерации

ФГБОУ ВПО «Тульский государственный педагогический университет
им. Л. Н. Толстого»

А. Р. ЕСАЯН, В. Н. ЧУБАРИКОВ,
Н. М. ДОБРОВОЛЬСКИЙ, А. В. ЯКУШИН

ПРОГРАММИРОВАНИЕ В *MAXIMA*

Учебное пособие

Допущено УМО
по классическому университетскому образованию
в качестве учебного пособия
для студентов высших учебных заведений,
обучающихся по направлениям подготовки
высшего профессионального образования
010100 «Математика»,
010200 «Математика и компьютерные науки»

Тула
Издательство ТГПУ им. Л. Н. Толстого
2012

ŋqdqɏeɎ bɆ dədɆbɎɜə d ɕɔɑbɔɏ Ɉqddɏqɏè

УДК 681.3.06(075)
ББК 32.973
Е81

Рецензенты:
доктор физико-математических наук, профессор *В. И. Иванов*
(кафедра информационных технологий
Тульского государственного университета);
доктор педагогических наук, профессор *А. С. Симонов*
(кафедра алгебры, математического анализа и геометрии
ТГПУ им. Л. Н. Толстого)

Есаян, А. Р.
Е81 Программирование в *Maxima*: Учеб. пособие / А. Р. Есаян, В. Н. Чубариков, Н. М. Добровольский, А. В. Якушин.– Тула: Изд-во Тул. гос. пед. ун-та им. Л. Н. Толстого, 2012.– 351 с.

ISBN 978-5-87954-646-0

В пособии описывается структура простого и удобного языка программирования интерпретирующего типа системы *Maxima* (с оболочкой *wxMaxima*). Рассказывается о правилах и приемах составления программ и подпрограмм, особенностях программирования с использованием прямой и косвенной рекурсии. Приводятся программные реализации различных алгоритмов генерирования перестановок и основанных на них алгоритмов решения задач: «о назначении», «о рюкзаке», «о коммивояжере», «о количестве разупорядочений» и т. д. Рассказывается также о решении в *Maxima* разнообразных задач, традиционно относящихся к курсам алгебры и математического анализа, в том числе решение алгебраических, рекуррентных и дифференциальных уравнений и систем таких уравнений; вычисление пределов, производных, конечных сумм, сумм рядов, конечных произведений, определенных и неопределенных интегралов.

Пособие предназначено студентам и аспирантам университетов, технических и педагогических вузов, обучающимся по специальностям, связанным с информатикой, математикой, физикой и экономикой. Значительная часть материала доступна школьникам старших классов. Студенты педагогических вузов, обучающиеся по основной или дополнительной специальности «Информатика», могут использовать пособие по целому спектру дисциплин, среди которых «Программирование», «Компьютерная алгебра», «Компьютерное моделирование», «Численные методы», «Системы и алгоритмы компьютерной обработки данных» и т. д.

УДК 681.3.06(075)
ББК 32.973

ISBN 978-5-87954-646-0

Предисловие

В предлагаемом учебном пособии речь идет о программировании в системе числовых и символьных вычислений *Maxima*. Описание опирается на последнюю версию системы *Maxima* 5.25.1 с оболочкой *wxMaxima* 11.0.8.0 (декабрь 2011 г). Материал пособия основан на набросках лекций и семинарских занятий по специальным и факультативным курсам, прочитанным авторами студентам факультета математики, физики и информатики Тульского государственного педагогического университета им. Л. Н. Толстого в последние годы. Изложение сопровождается рассмотрением большого количества примеров. Их в пособии более 200 – от простых и чисто иллюстративных до содержательных и разумно сложных.

В пособии "*Maxima*. Данные и графика" [9], уже затрагивались некоторые темы, непосредственно относящиеся к разделу программирования. И среди них – определение пользовательских функций. В данном пособии эта тема расширяется и углубляется, и затем описываются основные структуры простого и удобного встроенного языка программирования интерпретирующего типа, правила и приемы составления программ и подпрограмм, особенности программирования с использованием прямой и косвенной рекурсии. Приводятся программные реализации различных алгоритмов генерирования перестановок и основанных на них алгоритмов решения задач: "о назначении", "о рюкзаке", "о коммивояжере", "о количестве разупорядочений" и т. д. Во второй части пособия описывается решение в *Maxima* разнообразных задач, традиционно относящихся к курсам алгебры и математического анализа, в том числе, решение алгебраических, рекуррентных и дифференциальных уравнений и систем таких уравнений; вычисление пределов, производных, конечных сумм, сумм рядов, конечных произведений, определенных и неопределенных интегралов. В приложении рассказывается о контекстах и трансляции *Maxima*-выражений в *TeX*-выражения.

Пособие ориентировано на студентов, аспирантов, учителей, преподавателей вузов и, вообще, всех тех, кто интересуется компьютерными методами решения задач. Большая часть материала пособия доступна ученикам старших классов профильной школы.

Пособие создано в рамках выполнения исследований по гранту РФФИ 11-01-00571-А и издано при финансовой поддержке этого гранта.

Авторы признательны всем тем людям, которые ознакомились с различными частями рукописи и внесли полезные предложения по их улучшению. Особенно хотелось бы поблагодарить коллег по работе: Шулюпова В. А., Лапицкую Л. П., Ванькову В. С., Мартынюк Ю. М., Сундукову Т. О., Даниленко С. В. С авторами пособия можно связаться по электронной почте:

esayanalbert@mail. ru *chubaric@mechmat.msu.su*
dobrovol@tspu.tula.ru *yakushin@tspu.tula.ru*

Введение

Свободно распространяемая система символьных и числовых вычислений *Maxima* является развитием коммерческого продукта *Macsyma*, созданного по проекту *MACSYMA*. Аббревиатура *MACSYMA* получена из начальных букв слов *MAC's SYmbolic Manipulation*. Заметим, что слово *MAC* само является аббревиатурой, правда расшифровываемой двумя разными способами – *Man and Computer* (человек и компьютер) или *Machine Aided Cognition* (познание с помощью компьютера). Нам представляется, что по отношению к системам символьной математики, каковой и является *Macsyma*, более подходит второе толкование акронима *MAC*.

С 1968 по 1982 год в Массачусетском технологическом институте в рамках проекта *Project MAC symbolic manipulation*, финансируемого Министерством энергетики США, разрабатывалась система под названием *Macsyma*. С 1982 по 2001 год вариант этой системы под названием *DOE Macsyma* поддерживался в Техасском университете профессором Уильямом Шелтером (*William F. Shelter*). Благодаря своей открытости *Macsyma* произвела в свое время переворот в компьютерной алгебре и оказала большое влияние на развитие многих других систем символьной математики. Исходный код *DOE Macsyma* был опубликован в 1998 г. под общедоступной лицензией *General Public Licence* (*GPL*) и с 2000 г. развивается под именем *Maxima* как свободно распространяемая система.

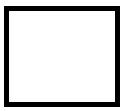
По набору возможностей *Maxima* близка к таким коммерческим продуктам, как *Mathematica* и *Maple*. Написана она на языке *Common Lisp*, имеет небольшой размер, чрезвычайно просто устанавливается на компьютер и необычайно быстро загружается. Обладая высочайшей степенью переносимости, *Maxima* может работать под управлением всех основных современных операционных систем и практически на всех компьютерах.

wxMaxima – комплекс программ, обеспечивающих графический пользовательский интерфейс системы *Maxima* (*Graphical User Interface – GUI*). Он разработан под руководством Андрея Водопивека (*Andrej Vodopivec*), активно им поддерживается, основан на работе с документами и обладает хорошим представлением формул на выводе. *WxMaxima* является кроссплатформенной оболочкой, функционирующей под управлением операционных систем *Windows*, *X*11 и *Mac OS X*. Она создана для платформы *Windows* на основе библиотеки программ с открытым исходным кодом *wxWidgets* и поставляется вместе с *Maxima*. Именно эту оболочку мы и будем иметь в виду при описании и работы с системой *Maxima*. Последняя версия системы *Maxima* 5.25.1 с оболочкой *wxMaxima* 11.0.8.0 вышла в декабре 2011 г. Скачать ее можно по адресу http://andrejv.github.com/wxmaxima/index.html. Много интересной и полезной информации о *Maxima* расположено на ее родном сайте – http://wxmaxima.sourceforge.net/ и русскоязычной локализация этого сайта – http://maxima.sourceforge.net/ru/.

1. Пользовательские функции

Нам предстоит познакомиться с различными разновидностями пользовательских функций системы *Maxima*. Пока же перечислим их возможные типы: функции с простой правой частью; функции с составной правой частью; функции с переменным числом аргументов; λ-функции (*lambda*-функции); массивовые (запоминающие) функции; блоки; функции, определенные по define; локальные функции; макросы. Наиболее важный из них тип – это блоки. Рассмотрены также средства и технология автозагрузки определений функций и значений переменных из иных документов, а также средства для получения информации о функциях. Часть представленного материала можно найти в [9. с. 84].

1.1. Функции с простой и составной правой частью



Пользовательские функции, их аргументы и переменные

Пользовательские функции записываются в одной из форм:

$$\mathrm{p}(x) := body1(x), \quad \mathrm{q}(x, y) := body2(x, y), \ \ldots\ , \tag{1}$$

где: p, q, … – имена функций; x, y, … – аргументы функций; *body*1, *body*2, … – тела функций, являющиеся простыми или составными выражениями. Функцию с n аргументами называют функцией от n переменных. Переменные, находящиеся в документе вне функций, называют глобальными. Переменные, объявленные внутри функций, называют локальными. Различные конкретизации записей (1) будут даны в следующих пунктах. При работе с функциями следует руководствоваться такими соображениями:

- аргументами функции, называемыми также формальными параметрами, могут быть любые переменные;
- аргументы могут изменяться в теле функции. После завершения вычислений они становятся неопределенными;
- если имя аргумента совпадает с именем некоторой определенной глобальной переменной, то на вычисление функции значение этой переменной никакого влияния не оказывает и после завершения вычислений по функции сохраняется неизмененным;
- тело функции может содержать глобальные переменные, отличные от аргументов, причем в теле функции можно использовать значения таких переменных и изменять их. Кроме того, в теле функции могут создаваться новые глобальные переменные. В любом случае после завершения вычислений по функции все глобальные переменные будут иметь последнее полученное значение;

- в теле функции (см. блоки) могут создаваться локальные переменные. При этом они обязательно должны быть объявлены. После завершения вычислений эти переменные становятся неопределенными;
- для запуска механизма вычисления функции к ней следует обратиться. Например, обращение к функции q может быть таким: q(*a*, *b*), где *a* и *b* – некоторые выражения. Аргументы в обращении к функции называют фактическими параметрами или фактическими значениями;
- вычисление функции в общем случае проводится так. Формальные параметры функции замещаются на соответствующие им фактические значения и вычисляется тело функции. В качестве результата вычислений возвращается значение последнего вычисленного выражения.

Отметим также, что в *Maxima* используется концепция "динамической области видимости переменных", в соответствии с которой конкретный идентификатор виден не только внутри блока, где он декларирован, но также и в любой функции, вызванной из этого блока. Такой подход, используемый в языках программирования *Lisp*, *Basic* и т. д., обладает большей гибкостью, чем альтернативная схема – "лексическая область видимости", но в то же время служит источником сложных для выявления ошибок. Лексическая область видимости характерна для большинства компилируемых языков программирования высокого уровня.

В дальнейшем мы будем рассматривать выражения двух типов – простые и составные. Простые выражения в *Maxima* строятся по определенным правилам из простых и индексированных имен, числовых и логических констант, различного типа чисел, обращений к встроенным и пользовательским функциям, арифметических и иных операторов, круглых скобок. Понятно, что слабым и уязвимым местом в приведенной выше расшифровке понятия "простое выражение" являются слова "формируется по определенным правилам". Однако мы откажемся от дополнительных разъяснений по этому поводу, ограничившись лишь приведением примеров простых выражений: *x*+*y*^2 sqrt(*x*)+log(*x*^2+1); *x*+*y*^3; *w*1+sin(*w*2)^*w*3, *y* = *x*^3+1, *a* < *b* и т. д. Из последовательности простых выражений *ex*1, *ex*2, …, *exn* можно создать составное выражение, которое записывается в форме (*ex*1, *ex*2, …, *exn*).

A. g(*x*1, *x*2, …, *xm*) := *ex*

Функции с простой правой частью

Здесь: g – имя функции, *x*1, *x*2, …, *xm* – переменные, называемые аргументами функции, или, по-другому, формальными параметрами, *ex* – простое выражение, зависящее от *x*1, *x*2, …, *xm* и, возможно, от каких-либо глобальных переменных. Обращения к функции *A* происходят в виде g(*b*1, *b*2, …, *bm*), где *b*1, *b*2, …, *bm* – некоторые выражения, называемые фактическими параметрами, или фактическими значениями. Выполнение *A* при обращении g(*b*1, *b*2, …, *bm*) реализуется так. Формальные параметры принимают значения соответствующих им фактических параметров, и вычисляется значение простого выражения *ex*. Полученное значение и является результатом вычисления *g*(*b*1, *b*2, …, *bm*), то есть результатом вычисления функции g в точке (*b*1, *b*2,

…, *bm*). Примеры определения функций с простой правой частью: f(*x*, *y*) := *x*+*y*^2, g(x) := sqrt(*x*)+log(*x*^2+1), h(*w*1, *w*3) := *w*1+sin(*w*2)^*w*3, tt(a, b) := *a* < *b*.

A. g(*x*1, *x*2, …, *xm*) := (*ex*1, *ex*2, …, *exn*)

Функции с составной правой частью

Здесь: g – имя функции; *x*1, *x*2, …, *xm* – аргументы функции; *ex*1, *ex*2, …, *exn* – последовательность простых выражений, зависящих от *x*1, *x*2, …, *xm* и, возможно, от каких-либо глобальных переменных. Обращения к функции *A* происходят в виде g(*b*1, *b*2, …, *bm*), где *b*1, *b*2, …, *bm* – некоторые выражения. Выполнение *A* при обращении g(*b*1, *b*2, …, *bm*) реализуется так. Формальные параметры принимают значения соответствующих им фактических параметров, и последовательно вычисляются значения выражений *exk* (*k* = 1, 2, …, *n*). Значение последнего выражения, то есть *exn*, и является результатом вычисления g(*b*1, *b*2, …, *bm*).

Примеры 1. Функции с простой и составной правой частью.

•	%i1	S(r) := %pi*r^2 /* площадь круга. *A* */;
	%o1	S(r) := $\pi \cdot r^2$
•	%i2	S(1);
	%o2	π
•	%i3	SS(a, b) := %pi*a*b /* площадь эллипса. *B* */;
	%o3	SS(a, b) := $\pi \cdot a \cdot b$
•	%i4	SS(3, 5)
	%o4	$15 \cdot \pi$
•	%i5	/* Формула Герона. *C* */ G(a, b, c) := (p : (a+b+c)/2, sqrt(p*(p-a)*(p-b)*(p-c)));
	%o5	G(a, b, c) := $\left(p : \frac{a+b+c}{2},\ \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)} \right)$
•	%i6	G(5, 6, 7)
	%o6	$6^{3/2}$
•	%i7	G(5, 6, 7), float;rrr
	%o7	14.69693845669907
•	%i8	float(G(5, 6, 7));
	%o8	14.69693845669907
•	%i9	/* количество цифр в числе n!. *D* */
	%o9	f(n) := (n!, string(%%), slength(%%))$

•	%i10	[7!, f(7), slength(string(7!))];
	%o10	[5040, 4, 4]

1.2. Функции с переменным числом аргументов

A. g([*x*]) := *ex*
B. g(*x*1, *x*2, …, *xm*, [*x*]) := *ex*
C. g(*x*1, *x*2, …, *xm*, [*x*]) := (*ex*1, *ex*2, …, *exn*)

Функции с переменным числом аргументов

Определелить функцию с переменным числом аргументов можно способами *A*, *B* или *C*, где: g – имя функции; *x*1, *x*2, …, *xm* – обычные переменные-аргументы; [*x*] – список из одной переменной *x*, являющейся общим именем переменного числа аргументов; *ex*, *ex*1, *ex*2, …, *exn* – выражения, зависящие от *x*1, *x*2, …, *xm*, *x* и, возможно, от каких-либо глобальных переменных, в том числе и от переменных, созданных в этих выражениях. Опишем действие функций *A*-*C* лишь для случая, когда значения выражений *ex*, *ex*1, *ex*2, …, *exn* не являются структурами типа списки, множества, матрицы или массивы. В примерах приведены функции с переменным числом аргументов, когда *ex* является списком и множеством.

Функция *A*. Обращение к *A* может содержать любое количество фактических значений, то есть иметь вид g(*a*1, *a*2, …, *ak*). Тогда по *A* будет возвращен список [h(*a*1), h(*a*2), …, h(*ak*)] из *k* элементов, получаемых соответственно обращениями к обычной функции h(*x*) := *ex* (правая часть из *A*).

Функция *B*. Обращение к *B* должно содержать *m* фактических значений для аргументов *x*1, *x*2, …, *xm* и любое количество дополнительных значений для аргумента *x*, то есть иметь вид g(*b*1, *b*2, …, *bm*, *a*1, *a*2, …, *ak*). Тогда по *B* будет возвращен список [h(*b*1, *b*2, …, *bm*, *a*1), …, h(*b*1, *b*2, …, *bm*, *ak*)] из *k* элементов, получаемых соответственно обращениями к обычной функции h(*x*1, *x*2, …, *xm*, *x*) := *ex*.

Функция *C* выполняется как *B*, но здесь правая часть функции – составное выражение со всеми вытекающими отсюда особенностями вычислений.

Примеры 2. Функции с переменным числом аргументов.

•	%i1	g([x]) := sin(x); g(1, 2, 3, 4, 5)	/* *A* */;
	%o1	g([x]) := sin(x)	
	%o2	[sin(1), sin(2), sin(3), sin(4), sin(5)]	
•	%i3	map(sin, [1, 2, 3, 4, 5]);	
	%o3	[sin(1), sin(2), sin(3), sin(4), sin(5)]	
•	%i4	h(a, b, [c]) := (a+b)*c$ h(1, 2, 3, 4, 5)	/* *B* */;

	%o5	[9, 12, 15]
•	%i6	w(c) := (1+2)*c$ map(w, [3, 4, 5]);
	%o7	[9, 12, 15]
•	%i8	hh(a, b, [c]) := (r : a+b, r*c)$ hh(1, 2, 3, 4, 5) /* *C* */;
	%o9	[9, 12, 15]
•	%i10	q(a, b, [x]) := [a, b, x]$ q(i, j, z1, z2, z3, z4) /* *D* */;
	%o11	[i, j, [z1, z2, z3, z4]]
•	%i12	q(a, b, [x]) := {a, b, x}$ q(i, j, z1, z2, z3, z4) /* *E* */;
	%o13	{i, j, [z1, z2, z3, z4]}
•	%i14	q(a, b, [x]) := {a, b, x^2}$ q(i, j, z1, z2, z3, z4);
	%o15	{i, j, [$z1^2$, $z2^2$, $z3^2$, $z4^2$]}

1.3. λ-функции

A. lambda([*x*1, *x*2, …, *xm*], *ex*1, *ex*2, …, *exn*)
B. lambda([[*x*]], *ex*1, *ex*2, …, *exn*)
C. lambda([*x*1, *x*2, …, *xm*, [*x*]], *ex*1, *ex*2, …, *exn*)

λ-функции (*lambda*-функции)

λ-функции называют также анонимными функциями, неименованными функциями или λ-выражениями. Предложения *A*-*C* демонстрируют возможный синтаксис λ-функций, где: *ex*1, *ex*2, …, *exn* – произвольные выражения; [*x*] – список из одной переменной; *x*1, *x*2, …, *xm* – аргументы. Ввод *A*-*C* приводит к индикации соответствующей λ-функции. Как и для обычной функции, любое из предложений *A*-*C* можно присвоить переменной. Но в данном случае такая переменная фактически становится именем соответствующей λ-функции. Последующие вычисления λ-функции возможны уже по этому обретенному имени и соответствующим фактическим значениям аргументов. Правда такое использование λ-функций не представляет большого интереса. Более часто их напрямую используют в блоках (см. ниже), в других λ-функциях, а также в качестве аргументов при обращениях к функциям, если эти аргументы должны быть именами функций. Вычисление λ-функции состоит в последовательном вычислении выражений *ex*1, *ex*2, …, *exn*. При этом ссылаться на последнее полученное значение можно переменной %%. Возвращается по λ-функции значение выражения *exn*. λ-функции могут содержать функции throw и catch, но не функцию return.

Функция *A*. При вычислении функции *A* последовательно вычисляются выражения *ex*1, *ex*2, …, *exn* при конкретных значениях аргументов *x*1, *x*2, …, *xm*, рассматриваемых как локальные переменные. В вычислении *A* могут

участвовать и глобальные переменные. Вычислить A в конкретной точке можно, например, так: lambda([$x1$, $x2$, …, xm], $ex1$, $ex2$, …, exn)($b1$, $b2$, …, bm). Возвращается по A значение последнего вычисленного выражения, то есть exn.

Функция *B*. λ-функция может иметь переменное число аргументов. Этот факт задается списком [x], где x – общее имя аргументов. Здесь выражения $ex1$, $ex2$, …, exn зависят от x и, возможно, от каких-либо глобальных переменных. Пусть, например, функция B имеет имя g и к ней произведено обращение g($a1$, $a2$, …, ak). Тогда по B будет возвращен список из k элементов, получаемых соответственно функциями lambda([x], $ex1$, $ex2$, …, exn)(as) (s = 1, 2, …, k).

Функция *C* имеет фиксированное количество аргументов $x1$, $x2$, …, xm (как A), а также еще и переменное количество аргументов, задаваемых списком [x], где x – общее имя аргументов (как B). Здесь выражения $ex1$, $ex2$, …, exn зависят от $x1$, $x2$, …, xm, x и, возможно, от каких-либо глобальных переменных. Пусть, например, функция C имеет имя h и к ней произведено обращение h($b1$, $b2$, …, bm, $a1$, $a2$, …, ak). Тогда по C будет возвращен список из k элементов, получаемых соответственно функциями lambda([$b1$, $b2$, …, bm, x], $ex1$, $ex2$, …, exn)(as) (s = 1, 2, …, k).

Примеры 3. Вычисления с λ-функциями.

•	%i1	t : lambda([x, y], x^2+y^2)	/* *A* */;
	%o1	lambda([x, y], x^2+y^2)	
•	%i2	[t(3, 4), t(a, b), t(1.2, 1.3)];	
	%o2	[25, b^2+a^2, 3.13]	
•	%i3	lambda([x, y], x^2+y^2)(3, 4)	
	%o3	25	
•	%i4	b : lambda([i], i^3)	/* *B* */$
		[b(5), b(-5), b(1.1)];	
	%o5	[125, -125, 1.331]	
•	%i6	lambda([i], i^3)(5);	
	%o6	125	
•	%i7	map(lambda([i], i^2), [c, d, e, f, g, h]);	
	%o7	[c^2, d^2, e^2, f^2, g^2, h^2]	
•	%i8	lambda([x, y, z], x : x+y+z, x^2)(1, 2, 3)	/* *C* */;
	%o8	36	
•	%i9	lambda([[w]], w : w+1, w^2)(1, 2, 3)	/* *D* */;
	%o9	[4, 9, 16]	