ГОСУДАРСТВЕННЫЙ НАУЧНЫЙ ЦЕНТР РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНСТИТУТ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ

ИФВЭ 99–56
ОМВТ

В.Н. Ларин

ЯЗЫК HECAS-2a

КРАТКОЕ ОПИСАНИЕ

Протвино 1999

УДК 618.3.06 М–24

Аннотация

Ларин В.Н. Язык HECAS-2a. Краткое описание.: Препринт ИФВЭ 99–56. – Протвино, 1999. – 13 с., библиогр.: 3.

В кратком изложении представлены основные элементы входного языка новой версии системы компьютерной алгебры HECAS-2a.

Abstract

Larin V.N. HECAS-2a Language. Brief Description.: IHEP Preprint 99–56. – Protvino, 1999. – p. 13, refs.: 3.

The main elements of input language of the new HECAS-2a computer algebra system version are briefly described.

E-mail: larin@mx.ihep.su

Тщетно хотят аналитики скрыть это от самих себя: они не доказывают, а комбинируют, составляют ...

Э. Галуа

Первое сообщение о системе компьютерной алгебры (СКА) HECAS относится к 1983 году [1]. Подробное описание языка системы дано в работе [2]. С тех пор система и ее язык претерпели существенные изменения. О некоторых из них было сделано презентативное (и последнее) сообщение [3]. Тем не менее работа по сопровождению системы продолжалась, в том числе в направлении повышения ее быстродействия и расширения предоставляемых услуг. Время показало, что наиболее перспективным направлением является последнее. Конкурировать в первом направлении с быстро развивающейся компьютерной техникой оказалось ˋˋнеблагодарным'' занятием. Поэтому в дальнейшем работа в основном велась именно по второму направлению. Однако в силу разных причин сколь-нибудь приемлемое описание новой версии системы и ее входного языка до сих пор не опубликовано. Настоящая работа устраняет отчасти указанный недостаток, хотя и не претендует на полноту[1].

Порядок изложения материала выбран с учетом того, что с ним, возможно, будет знакомиться пользователь, которому язык системы не знаком. В первом разделе коротко обсуждаются вопросы ˋˋобщения'' с системой — ее вызов, работа с файлами и т.д. В следующем разделе в сжатой (на сколько возможно) форме с максимальным использованием новаций излагается концепция входного языка системы. Далее обсуждается важный ˋˋинструмент'' любой СКА — аппарат подстановок пользователя. В последнем разделе коротко говорится о тех новациях, которым не нашлось места в предшествующих разделах.

[1]Толчком к появлению этой публикации явилась постановка системы на DEC-Alpha, выполненная по просьбам пользователей.

Как начать работу с системой

Проще всего ввести в командный файл `LOGIN.COM` определение глобальной переменной, назвав ее, например, так

```
HECAS == "@USER$LAMP:[LARIN.SYM.POLAR]PHECAS"
```

Теперь (после исполнения командной процедуры `LOGIN.COM`), чтобы начать работу, достаточно ввести команду `HECAS P1 P2`, где `P1, P2` — необязательные параметры, в качестве которых могут выступать имена *входного* и *выходного* файлов пользователя или знак вопроса "?" — для *справки*. Ниже перечислены варианты вызова системы.

`HECAS ?`	–	Выводит список "заголовков" рубрик справочного файла.
`HECAS ? заголовок`	–	Выводит краткую информацию, содержащуюся в рубрике под указанным заголовком.
`HECAS`	–	Ввод/вывод данных осуществляется с/на терминал.
`HECAS in_f`	–	Ввод данных из файла с указанным именем, вывод — на терминал.
`HECAS - out_f`	–	Ввод с терминала, вывод в файл с указанным именем.
`HECAS in_f out_f`	–	Ввод/вывод осуществляется из/в файл с указанными именами.

Во всех случаях, кроме обращения к справочному файлу, HECAS-процессу приписывается рабочий файл-протокол с именем `LP2`, в который по умолчанию дублируются входные строки и результат вычислений, если только пользователь не откажется от этого с помощью соответствующих команд языка системы (см. ниже, а также работу [2]).

Если на вход системы подается входной файл, ее работа прекращается, когда инструкции, содержащиеся в файле, исчерпаны. Если работа ведется в интерактивном режиме, то закончить сеанс можно двумя путями, во-первых, вводом (после очередного приглашения системы) четырех символов `****`, начиная с первой позиции, и, во-вторых, стандартной командой прерывания `CTRL/Y`.

Концепция входного языка

Язык HECAS предназначен для выполнения аналитических (символьных) преобразований математических выражений. Основными его элементами являются: *операция присваивания, команды, имена* системных и определенных пользователем *объектов* и математические *выражения*, построенные из последних с помощью *"стандартных" знаков* математических операций с использованием *чисел*. Рассмотрим коротко каждый из них.

Специальные символы

К первой группе специальных символов относятся знаки алгебраических операций: *сложение* (`+`), *вычитание* (`-`), *умножение* (`*` или `пробел`), *деление* (`/`), *степень* (`^`), два вида *скобок* (`( ) [ ]`).

Вторую группу символов составляют знаки, несущие различную синтаксическую нагрузку. К ним относятся (здесь "/" — разделитель символов):

Разделители:	/./,/;/	(см. примеры)
Признаки:	/$/:/?/	(команды, комментария и имени dummy-объекта соответственно)
Системные имена:	/@/!/#/	(σ-, γ-матрицы и мнимой единицы (i))
Присваивание:	/=/	(см. *"Операция присваивания"*)

Все перечисленные символы **запрещено** использовать в именах объектов пользователя, где наравне с буквами допускается использование только *двух* символов: /%/'/.

Имена объектов

Отличительная особенность языков компьютерной алгебры — возможность использования "абстрактных" символьных объектов (из допустимых в данной системе классов), с которыми СКА может выполнять определенные манипуляции без учета их явного вида или обращения к конкретным элементам этих объектов. На уровне входного языка объекты представляются именами.

В языке HECAS имя объекта — это цепочка от одной до восьми литер (букв, цифр, разрешенных специальных символов). Она не может начинаться с числа. Как уже отмечалось, ряд имен зарезервирован для системных (как правило, "стандартных" математических) объектов, и пользователь не может их переопределять. Характерным для имен системных объектов является использование только прописных букв. Полный список имен "новых" системных объектов приведен в приложении[2] Определение имен объектов пользователя демонстрируется в подразделе *"Команды"* .

Будем называть *простым* объект, представленный только собственным именем; во всех других случаях (даже если за именем следует только показатель степени) объект трактуется как *составной* или *сложный*.

Система поддерживает десять объектных классов, четыре из которых системные. К ним относятся (в скобках указаны системные имена) единичный полностью антисимметричный *тензор Леви-Чивиты* — ε (`EPS`), σ-, γ- и λ-*матрицы* (`@, !, LAM` соответственно). Остальные шесть классов хотя и включают системные объекты, предоставлены пользователю для определения его собственных объектов с помощью *команд-описателей* (см. ниже). Это — *переменные, векторы/индексы, тензоры, функции* и *матрицы* (одномерные и квадратные). В следующем подразделе описано определение объектов класса *полином*.

Отсутствие среди перечисленных классов объектов комплексного типа в определенной мере компенсируется наличием системного объекта с именем "`#`" (класс "переменные"), обладающего свойствами мнимой единицы. С помощью этого объекта пользователь может конструировать собственные комплексные выражения.

Операция присваивания

Эта операция является основной конструкцией языка. В общем виде ее можно представить так:

$$M = E,$$

где E — выражение, которое нужно преобразовать, а M — простой или составной объект, или *моном*, т.е. произведение объектов. (**Внимание:** M не может быть суммой; M и

[2]Полный список "старых" системных имен приведен в работе [2].

“=” должны располагаться на одной строке.) Символ “;” является признаком окончания *фразы* языка.

Операция присваивания инициирует обработку выражения E, и по ее завершении (т.е. когда к выражению нельзя применить ни одного из встроенных или заданных пользователем преобразований) оно присваивается ˜объекту˝ M.

Если M — имя простого, не определенного с помощью *команд-описателей* объекта, то это означает, что вводится новый объект класса ˜полином˝ с указанным именем, которому присваивается обработанное выражение. После этого вхождение данного объекта в другое выражение, подаваемое на обработку, вызывает его замену на присвоенное ему значение. Повторное использование имени полинома в левой части операции присваивания вызывает уничтожение ранее присвоенного ему выражения, а имя полинома теперь будет указывать на новое выражение. Заметим, что по умолчанию выражение, которое присваивается полиному, *автоматически* выводится как в рабочий файл (`LP2`), так и на терминал или в выходной файл пользователя.

Во всех других случаях операция присваивания трактуется системой как задание *подстановки* пользователя (см. раздел ˜Подстановки˝).

Числа

В системах компьютерной алгебры, как и в обычной алгебре, используются не только абстрактные (символьные) объекты, но и числа. Это — численные коэффициенты в алгебраических выражениях и показатели степени.

Язык системы HECAS поддерживает три типа численных данных: *целый* (`13, -1092`), *вещественный* (`3.14159, -10.E-5`) и характерный для СКА *дробно-рациональный* (`-3/7, 59/11`). Диапазон значений *численных коэффициентов*, в том числе каждой из компонент рациональной дроби, ограничен внутренним представлением в формате двойной точности. *Показатель степени* может быть целым числом n в диапазоне $-500<n<500$ (кроме показателя степени при выражении или подвыражении, где $n\leq 100$) или числом в любом из выше перечисленных представлений, но заключенным в круглые скобки.

Продемонстрируем теперь простейшие примеры работы программы.

Пример №1. Арифметика

Вход:	**Выход:**
`r1 = 11-1/3*4;`	`r1 = 29/3;`
`r2 = 2^7 2^3/7;`	`r2 = 1024/7;`
`r3 = (3/2)^2+2^(3/2);`	`r3 = 9/4+2^(3/2);`
`r4 = 1.55+1-1/2;`	`r4 = 2.05;`

Последний пример демонстрирует использование вещественного числа вместе с двумя другими типами чисел, результатом *всегда* является вещественное число.

Команды

Команда — это предписание системе выполнить определенные действия или установить определенный режим работы. Исключением являются команды-описатели, с помощью которых декларируются объекты пользователя. Команды должны начинаться с символа `$` и имеют формат

```
$NAME parameter(s);
```

где NAME — имя команды, parameter(s) - один или несколько обязательных и/или необязательных параметров (численных и/или символьных). Команда, как и любая фраза языка, должна завершаться символом ";".

С помощью команд пользователь может декларировать имена собственных объектов, управлять обработкой выражений и выводом результатов обработки, получать информацию о текущем состоянии системы (например, таблицу имен системных и/или собственных объектов). В соответствии с этим их можно разбить на четыре группы:

- Команды – описатели.
- Команды – операции.
- Команды управления обработкой выражений.
- Команды управления выводом информации.

Ниже дается полная сводка команд-описателей с простейшими примерами использования декларируемых объектов. (Значок ♠ в комментарии указывает на отсутствие соответствующего описателя в предыдущей версии [2].)

Пример №2. Описание объектов

Вход:	**Выход:**	**Комментарий:**
`$VAR x,y; r=(x-y)^2;`	`r=-2x y+x^2+y^2`	Скалярные переменные.
`$IND 8 a,b; r=a.b b.a;` `$IND i,j,k; r=i.k j.i;`	`r=8` `r=j.k`	Индексы размерности 8 (по умолчанию dim=4).

`$MOM 3 s,t; r=s.s/2;`
`$MOM p,q; r=p.i q.i`
`+p.j p.j;`
	`r=1/2*Ms^2`
	
	`r=p.q+Mp^2`
		Векторы размерности 3
		(dim=4; `Ms, Mp` – модули соответствующих векторов, определяемые автоматически).

`$TENS 1 T,S; r=S.i T.i;`	`r=T.I4A S.I4A`	Несимметричные тензоры (`I4A` – системный "немой" индекс dim=4).
`$ATENS AT; r=AT.j.i;`	`r=-AT.i.j`	Антисимметричные тензоры.
`$STENS 3 S';r=S'.k.j.i;`	`r=S'.i.j.k`	Симметричные тензоры.
`$PLR Rho;  $PLR 4 H4;`	См. в [2]	Поляризационные тензоры.
`$TENFUN 1+2 Tf;` `r=Tf.a.(x,y) a.b;`	`r=T.b.(x,y)`	Тензорные функции♠ (m+n, где m – число индексов, n – аргументов, $1 \le m+2 \cdot n \le 15$); для всех типов тензоров по умолчанию rank=2, максимальный rank=15.
`$FUN 2 F; r=F(x^2,y);` `$FUN g; r=g^3(y);`	`r=F(x^2,y)` `r=g^3(y)`	Функции пользователя. Число аргументов $n \le 15$; по умолчанию $n = 1$.

`$MATR 3 M; M.1.3=1-x^2;`
`M.2.2=y+1;`
`$VECT V; V.1=x^2;`
`V.2=x+p.q;`
	См. в [2]	Квадратные матрицы 3×3
		(по умолчанию dim=2×2).
		Одномерные матрицы
		(по умолчанию dim=2).

Два последних описателя задают матричные объекты, элементами которых являются полиномы (начальные значения равны нулю). Здесь же приведены примеры задания ненулевых значений элементам матричных объектов. Размерность таких объектов ограничивается размером *таблицы имен*, рассчитанной на 700 объектов пользователя, а каждому элементу матрицы в ней отводится одна ячейка.

Примеры работы с матрицами из-за их громоздкости здесь не рассматриваются (см. работу [2]). Обратим лишь внимание на три отличия от предыдущей версии системы, связанные: *а)* с обращением к элементам матриц (см. Пример №2); *б)* с изменением результата вычисления обратной матрицы (ранее вычислялась *присоединенная* матрица — $M^{aug} = \det(M) \cdot M^{-1}$) и *в)* с изменением управления обработкой матриц (команда `$MULMATR;` см. ниже). А также напомним, что система поддерживает следующие операции с матрицами: *сложение* и *умножение*, умножение матрицы на число, переменную или полином, вычисление *следа* (`SPUR(M)`), *детерминанта* (`DET(M)`) и *обратной* матрицы (`M^-1`), *транспонирование* (`TRPS(M)`) и *эрмитово сопряжение* (`HETR(M)`). Если результатом выполнения операции с матрицами является матричный объект `m`, для его вывода необходимо использовать команду `$OUT m;`.

Далее рассматриваются команды трех последних групп. При этом перечисляются все ˝новые˝ команды (помечены значком ♠), а также некоторые часто используемые и модернизированные ˝старые˝ команды[3].

Пример №3. Команды – операции

Формат команды	**Комментарий**
`$OUT r,V,M;`	Выводит выражение, присвоенное полиному `r`, и элементы матриц `V, M`.
`$VLIST;` `$VLIST 1;`	Выводит таблицу имен *только* объектов пользователя. Выводит таблицу имен объектов, начиная с 1-го (с 1 по 140 – системные объекты); параметр может быть другим целым числом, начиная с которого выводится таблица.
`$CLEAR x,p,M;`	Удаляет все подстановки пользователя (см. ниже), содержащие в левой части объекты, перечисленные в списке параметров. Для матриц (например, `M`) действие команды сводится к занулению элементов♠.
`$DELETE q,y,F,V;`	Удаляет имена, указанные в списке параметров, из таблицы. При этом для них автоматически выполняется команда `$CLEAR`. Удаление матрицы сопровождается уничтожением всех ее элементов♠.

`$PSIZE r;`
 	
`$PSIZE;`
	Выдает информацию о полиноме с именем `r`: размер памяти (в словах) и количество членов; так, для первого полинома из *˝Примера №2˝* формат сообщения: `: SIZE : 143 r - 22 WORDS, 3 MEMBERS`, где 143 — номер полинома в таблице имен.
	Выдает информацию о всех полиномах, определенных на момент ввода команды, и суммарный объем памяти♠.

[3]Полный список последних см. в [2].

Эти команды объединяет то, что предписываемая ими операция выполняется сразу после их ввода и только один раз.

Команды двух следующих групп устанавливают режимы (моды) работы системы, которые сохраняются вплоть до их отмены теми же командами, как правило, с нулевым параметром или эквивалентным ему параметром `OFF`. Рассмотрим примеры действия команд управления обработкой выражений, где только 1-я команда — модернизированная ˝старая˝, остальные — ˝новые˝. (Ввиду многочисленности команд этой группы, не вошедшие в примеры команды перечислены в приложении.) Далее везде *подразумеваемые* значения параметров, которые можно не указывать, заключены в *квадратные* скобки, а *исходные* значения — в *круглые*.

Команды управления обработкой выражений **I**

Команда	n	**Комментарий**
`$MULMATR n;`	(0)	Операции с матрицами **не** выполняются.
`$MULMATR;`	[1]	Выполняются, кроме вычисления системной функции `DET0(M)` ($= \det(M)$), возникающей при вычислении `M^-1`.
	2	То же, что и $n = 1$, но `DET0(M)` вычисляется (˝новая˝ мода).
`$UNISUB n;`	0	Режим многотабличной обработки подстановок (в ряде случаев увеличивает скорость их выполнения, но ограничивает действие двух следующих команд).
`$UNISUB;`	([1])	Режим работы с одной *общей* таблицей подстановок; в многотабличной моде она используется для ˝сложных˝ подстановок типа ˝моном˝ (действие двух следующих команд распространяется только на эту таблицу).
`$LASTSUB n;`	0	Каждая вновь заданная подстановка помещается в *конец* общей таблицы (алгоритм реализации подстановок просматривает таблицы с ˝головы˝).
`$LASTSUB;`	([1])	Каждая вновь заданная подстановка помещается в *начало* общей таблицы.
`$ADDSUB n;`	0	Режим *переопределения* ранее заданных подстановок (старые значения уничтожаются).
`$ADDSUB;`	([1])	Режим *дополнения*, при котором старые значения для вновь определяемых подстановок сохраняются в таблице.
`$LEVSUB;`	([0])	Режим выполнения подстановок только на ˝нулевом˝ уровне обрабатываемых выражений.
`$LEVSUB n;`	>0	Устанавливает ˝глубину˝ просмотра выражений для реализации подстановок в *подвыражениях* (см. Пример №5).

Далее приведены команды той же группы со ˝сложным˝ параметром.

Команды управления обработкой выражений **II**

Команда	**Комментарий**
`$DIMREG (exp);`	Режим *переопределения* размерности 4-индексов (*"размерная регуляризация"*); так например, если ввести последовательность: `$VAR e; $IND i; $DIMREG (4-2*e); r=i.i;` результатом будет `r=4-2*e`.
`$DIMREG OFF;`	Режим *возврата* 4-индексам размерности, равной **4**.
`$DEPEND x,y,z;`	Устанавливает зависимость переменных `x, y, z` от "аргумента" дифференцирования по вектору (см. Пример №8). По умолчанию все переменные независимы.
`$DEPEND 0 x,z;`	Возвращает переменным `x,z` статус *независимых*.

Рассмотрим примеры команд последней группы. Они управляют выводом входной и выходной информации на терминал и/или в файл. (Как и прежде, новые команды помечены значком ♠.)

Команды управления выводом информации

Команда	n	**Комментарий**
`$DENOUT n;`	0	Режим вывода выражений и подвыражений согласно правилу: *один моном на одну строку* (новая мода).
`$DENOUT;`	([1])	То же *только* для выражений; число мономов подвыражений в строке лимитируется их длиной и длиной строки.
	2	*Плотный вывод:* выражения и их подвыражения развертываются построчно.
`$SUBOUT n;`	0	При выводе отключает подстановку подвыражений (представляются своими именами).
`$SUBOUT;`	([1])	Реализует подстановку подвыражений при выводе.
`$SPECOUT n;`	(0/H)	Реализует "стандартный" вывод в HECAS-формате (допускается *символьный* параметр)♠.
`$SPECOUT;`	[1/M]	Реализует вывод на языке системы `MACSYMA`.
	2/R	Реализует вывод на языке системы `REDUCE`.
	3/F	Реализует вывод на языке программирования `FORTRAN`[4].
`$INPLOG n;`	0	Отключает дублирование входных строк в рабочий файл-*протокол* с именем `LP2.`♠
`$INPLOG;`	([1])	Восстанавливает дублирование входных строк в `LP2.`

Подстановки

Операция *подстановки* является важным инструментом, с помощью которого пользователь может задавать собственные преобразования выражений. Ввиду важности этого вопроса, который детально обсуждался в работе [2], коротко рассмотрим его и здесь.

В настоящей версии системы реализовано два класса подстановок, которые будем называть *"глобальными"* и *"локальными"* (последний класс реализован только в этой версии системы).

Глобальные подстановки

Это подстановки, которые задаются только операцией присваивания (LHS=RHS), где LHS — один или произведение (*не сумма!*) объектов, то, что будет заменяться в обрабатываемых выражениях на RHS — произвольное выражение. В прежней версии системы этому равенству должна была предшествовать команда $LET. Сейчас это необязательно. "Глобальный" характер таких подстановок проявляется в том, что они выполняются до тех пор, пока не последует их явная отмена либо вводом *только* левой части (LHS;), либо командой $CLEAR.

Напомним также, что язык системы позволяет выполнять так называемые *обобщенные* подстановки, используя специальные объекты двух классов с именами: ?X, ?Y,... — для переменных и ?A, ?B,... — для индексов. С их помощью можно реализовать семантически сложную конструкцию языка: *"для произвольных* X *и* A *выполнить подстановку* ...X...A... = ..."*.*

Ниже приведены простые примеры HECAS-"программ" с подстановками пользователя при исходном положении ключей — команд управления подстановками (см. выше).

Пример №4. Глобальные подстановки

Вход:	**Выход:**
`$VAR x,y,s1,s2;`	
`$IND i,j; $MOM k,p,q;`	
`:  Задание подстановок`	
`x^2 = s2;  r1 = x + x^2 y + x^3;`	`r1=x+x s2+y s2;`
`y k.p = s1; r2 = x k.i y^2 p.i+`	
`y k.j x^2 q.j;`	`r2=x y s1+y s2 k.q;`
`: Программа вычисления факториала`	
`$FUN F; F(?X)=?X F(?X-1), F(1)=1;`	
`r3=F(15);`	`r3=1307674368000;`
`r3'=F(20);`	`r3'=2.432902E+18;`

В последнем примере (r3') осуществляется автоматический переход к представлению численного коэффициента в вещественном виде, так как значение целого числа вышло за пределы формата двойной точности. Точность, с которой это число выводится, можно увеличить до двойной с помощью команды $FORMAT n; (n=1 или 4)[5].

Локальные подстановки

Возможны случаи, когда подстановку необходимо выполнить только в некоторой части выражения, не затрагивая других его частей. Для реализации таких подстановок в настоящей версии системы введена специальная *функция* с именем SUB, которая имеет формат

SUB(l1=r1,l2=r2,...,subexpres),

[5]См. работу [2], с.43.

где последний параметр есть подвыражение, к которому нужно применить подстановки, перечисленные в качестве предшествующих параметров.

Заметим, что локальные подстановки имеют более высокий приоритет по сравнению с глобальными, т.е. выполняются в первую очередь. В следующем примере демонстрируется одновременное использование подстановок обоих классов (в нем также использованы стандартные функции `SIN` и `LN`).

Пример №5. Локальные подстановки

Вход:	**Выход:**

`$VAR x,y,s1;  $FUN F,G;`
`x y=s1;r1=x^2 y^2+y SUB(x=F(y),y=G(x),x y) x;`
`r2=G(x)F(y)+SUB(F(y)=SIN(y),G(?X)=LN(?X),r1);`
 	
`$LEVSUB 1;  :: Применение команды $LEVSUB:`
`r2=G(x)F(y)+SUB(F(y)=SIN(y),G(?X)=LN(?X),r1);`
	`r1=s1 F(y)G(F(y))+s1^2;`
	`r2=s1^2+F(y)G(x)`
	`  +s1 SIN(y)LN(F(y));`
	
	`r2=s1^2+F(y)G(x)`
	`  +s1 SIN(y)LN(SIN(y));`

Важное замечание: при описании подстановки (в отличие от определения полинома или матрицы) один и тот же объект не может присутствовать в обеих частях операции присваивания. В простейших случаях (если, например, при вычислении `r1` из *Примера №5* задать подстановку `x=F(x)`) система обнаружит это, выдаст сообщение

```
HCREAD:  VARIABLE  "x"  CANNOT BE IN RIGHT-HAND-SIDE
```

и проигнорирует соответствующую подстановку. В более сложных случаях (при задании цепочки подстановок: `x=y, y=z, z=x` или, например, в программе вычисления факториала из *Примера №4* не будет указан "ограничитель" `F(1)=1;`) система не может обнаружить "зацикливания" при определении подстановки, и оно проявит себя только при обработке выражения, в котором такая подстановка реализуется, как *переполнение памяти*, наступающее практически мгновенно.

Другие новации языка

1. Более *"жестким"* стал синтаксис языка. Так, если в прежней версии системы можно было ввести `COS^3 x` или `TENS I.J`, теперь такая запись считается ошибкой. Верной является запись этих выражений в виде `COS^3(x)` и `TENS.I.J` соответственно (см. также примеры).

2. Введена "обобщенная" *показательно-степенная* функция `P^Q`, где `P` и `Q` — произвольные выражения. В общем случае они должны быть заключены в круглые скобки. Исключением являются целые числа, имена переменных и полиномов (без численных коэффициентов, а у `Q` — и без показателя степени). Следующий пример демонстрирует вычисление смешанной производной от такой функции ($\frac{\partial}{\partial x}\frac{\partial}{\partial y}(\ln x)^{tg\,y}$).

Пример №6. Вычисление производной

Вход:	`$VAR x,y;  r = D2'(x,y,(LN(x))^(TG(y));`
Выход:	`r = x^-1 LN(LN(x)) COS^-2(y) TG(y) (LN(x))^(TG(y)-1)` `  + x^-1 COS^-2(y) (LN(x))^(TG(y)-1)`

3. Для вычисления следов от произведений γ- и λ-матриц введены специальные функции с именами `GSP` и `LSP` соответственно, которые заменили громоздкую конструкцию с использованием так называемых ˝фиктивных спинорных скобок˝. В следующей таблице показана запись выражений с этими матрицами в старой и новой версиях системы. Причем в последней допускаются оба варианта.

Пример №7. Выражения с γ- и λ-матрицами

Матрицы	**HECAS-1**	**HECAS-2a**
γ	`U2.?A !i !j U1.?A`	`GSP(!i !j)`
λ	`VC2.?A LAM.a LAM.b VC1.?A`	`LSP(LAM.a LAM.b)`

4. Введена возможность вычислять производные по вектору. Для этого используется ˝стандартная˝ функция дифференцирования (`D'`), первым ˝аргументом˝ которой должна быть компонента данного вектора, например `D'(P.i,expres)`, где `expres` — произвольное выражение или имя ранее определенного полинома.

Пример №8. Вычисление производной по вектору

Вход:	**Выход:**
`$VAR a,b,c; $MOM k,p,q; $IND i; $TENS T;` `r1 = D'(p.i,(a p.k+b T.q.p+c q.k));`	`r1=a k.i+b T.q.i;`

Здесь уместно напомнить, что в системе HECAS ˝свертка˝ тензора с векторами, например $T_{ij}p^iq^j$ или на языке системы `T.i.j p.i q.j`, приводится к ˝каноническому˝ виду `T.p.q` (в математической нотации это могло бы выглядеть как T_{pq}).

Заключение

Настоящая работа не может рассматриваться как законченное описание языка системы — многое осталось ˝за скобками˝. Именно поэтому, при изложении материала приходилось неоднократно ссылаться на устаревшее, но достаточно полное описание языка [2]. Будет или нет опубликовано подробное описание языка новой версии системы, в большой степени зависит от пользователей и их реакции на эту работу. Тем не менее автор надеется, что предлагаемое *кратжое* описание языка будет полезно не только тем, кто знаком с системой, но и начинающим пользователям.

В заключение считаю своим приятным долгом выразить искреннюю благодарность моему соавтору по разработке системы С.Н.Грудцину (с которым, к сожалению, жизнь развела нас уже на несколько лет), а также моим коллегам С.Б.Луговскому, А.Д.Рябову, С.С.Садилову и В.Б.Мотякову за консультации в процессе адаптации системы на DEC-Alpha.

Список литературы

[1] Грудцин С.Н., Ларин В.Н. – В сб.: Материалы Третьей всесоюзной конференции *"Диалог человек-ЭВМ" (Протвино, 1983).* – Серпухов, 1984, с.201.

[2] Грудцин С.Н., Ларин В.Н. – *Препринт ИФВЭ 87-26.* Серпухов, 1987.

[3] Grudtsin S.N., Larin V.N. – In: *Proc. of the IV International Conference on Computer Algebra in Phisical Research (Dubna, 1990)*, pp. 53–57. World Sci. Pub., Singapore, 1991.

Рукопись поступила 1 декабряря 1999 г.

Приложение

Имена новых системных объектов

Имя	**Класс**	**Наименование объекта**	**Примеры**
`G1,...,G9`	тенз.	Ковариантные производные	`G3.a.b.c[expr]`
`0,...,26`	инд.	Числовые индексы	`p.0, EPS.0.1.2`
`DET0`	матр.	Детерминант матрицы ($det(A)$), для которой вычисляется обратная (A^{-1})	`B.1.1=expr1/DET0(A)` `B.1.2=expr2/DET0(A)` ...
`GSP`	функ.	Оператор вычисления следа от произведения γ-матриц	`GSP(!i !j ...)`
`LSP`	—	То же для λ-матриц	`LSP(LAM.a ...)`
`SUB`	нет	Оператор локальной подстановки	`r=x^2 SUB(x=y,x+y)`
`METRIC`	тенз.	Метрический тензор	`METRIC.0.1`

Команды управления обработкой выражений **I**

Команда	n	**Комментарий**
`$BIANCI n;`	(0)	Алгоритмы применения тождеств Риччи и Бианки к тензорам Римана и Риччи (`RI4, RI2` соответственно) отменены.
`$BIANCI;`	[1]	˝Включает˝ алгоритм, использующий тождество Риччи: $R_{hijk} + R_{hkij} + R_{hjki} = 0$.
	2	То же, плюс алгоритм, использующий тождества Бианки: $\nabla_h R_{ijkl} + \nabla_i R_{jhkl} + \nabla_j R_{hikl} = 0$; $\nabla_i R_{ijkl} = \nabla_k R_{jl} - \nabla_l R_{jk}$; $\nabla_i R_{ij} = \frac{1}{2}\nabla_j R$, где R_{ij}, R — тензор Риччи и скалярная кривизна (`RI0`) соответственно.
`$GRCOMM n;`	(0)	Ковариантные производные (∇_i; системное имя `G1'.i`) не коммутируют.
	1	Коммутируют $[\nabla_i, \nabla_j] = 0$ (в системе: `G2'.i.j=G2'.j.i`).
`$GRCOMM;`	[2]	Коммутатор не равен нулю: $[\nabla_i, \nabla_j]\Phi_{abc\ldots} = -(\Phi_{xbc\ldots}R^x_{aij} + \Phi_{axc\ldots}R^x_{bij} + \ldots)$, где $\Phi_{abc\ldots}$ — произвольное выражение с 4-индексами, R^x_{aij} — тензор Римана.
`$INDALG;`	([1])	˝Стандартный˝ алгоритм обработки ˝немых˝ индексов.
`$INDALG n;`	2	Алгоритм, эффективный при работе с тензорами Римана.
`$ERSTOP n;`	(0)	По ошибке работа системы не прекращается.
	1	Ошибка игнорируется до ";", система продолжает работу.
`$ERSTOP;`	[2]	По ошибке работа системы прекращается.
`$REDUCE n;`	(0)	Алгоритм реализации редукционного соотношения для γ-матриц: $\gamma_\mu\gamma_\nu\gamma_\lambda = i\gamma_\rho\gamma_5\varepsilon_{\mu\nu\lambda\rho} + g_{\nu\lambda}\gamma_\mu + g_{\mu\nu}\gamma_\lambda - g_{\mu\lambda}\gamma_\nu$ *отключен.*
	[8]	Задействует указанный алгоритм, начиная с n γ-матриц в одной ˝линии˝; оптимальная установка $n = 8$ может быть изменена этой же командой с другим значением параметра.

Команды управления обработкой выражений **II**

Команда	**Комментарий**
`$METRIC;`	Режим замены ˝скалярных произведений˝ числовых индексов системным *метрическим тензором* с именем `METRIC` (например, `0.1` представляется как `METRIC.0.1`).
`$METRIC Mt;`	То же, что и в предыдущем случае, но замена осуществляется на заранее описанный тензор (например, `$TENS Mt;`).
`$METRIC OFF;`	Замена отменена (исходная мода).
`$COMMUT n x,T,F;`	Объявляет перечисленные объекты принадлежащими к n-му некоммутирующему классу (это могут быть переменные, функции и тензоры). Объявленные таким образом объекты не коммутируют только между собой.
`$COMMUT 0 y,g;`	Определяет ˝новый˝ некоммутирующий класс, отличный от всех ранее определенных.
`$COMMUT x,F,g;`	Возвращает объектам статус коммутирующих.

В.Н.Ларин.
Язык HECAS-2a. Краткое описание.

Оригинал-макет подготовлен с помощью системы LaTeX.
Редактор Н.В.Ежела. Технический редактор Н.В.Орлова.

Подписано к печати 3.12.99. Формат 60 × 84/8. Офсетная печать.
Печ.л. 1,6. Уч.-изд.л. 1,3. Тираж 130. Заказ 190. Индекс 3649.
ЛР №020498 17.04.97.

ГНЦ РФ Институт физики высоких энергий
142284, Протвино Московской обл.

Индекс 3649

П Р Е П Р И Н Т 99–56, И Ф В Э, 1999