2 Основы работы в Scilab

2.1 Текстовые комментарии

Текстовый комментарий в Scilab это строка, начинающаяся с символов //. Использовать текстовые комментарии можно как в рабочей области, так и в тексте файла-сценария. Строка после символов // не воспринимается как команда и нажатие клавиши Enter приводит к активизации следующей командной строки:

```
-->//6+8
-->
```

Листинг 2.1

2.2 Элементарные математические выражения

Для выполнения простейших *арифметических операций* в Scilab применяют следующие операторы:+ сложение, − вычитание, * умножение, / деление слева направо, \ деление справа налево, ^ возведение в степень.

Вычислить значение арифметического выражения можно, если ввести его в командную строку и нажать клавишу ENTER. В рабочей области появится результат:

```
--> 2.35*(1.8-0.25)+1.34^2/3.12
ans =
4.2180
```

Листинг 2.2

Если вычисляемое выражение *слишком длинное*, то перед нажатием клавиши ENTER следует набрать три или более точек. Это будет означать продолжение командной строки:

```
--> 1+2+3+4+5+6....
+7+8+9+10+....
+11+12+13+14+15
ans =
   120
```

Листинг 2.3

Если символ точки с запятой «;» указан в конце выражения, то результат вычислений не выводится, а активизируется следующая командная строка:

```
--> 1+2;
--> 1+2
ans =
3
```

Листинг 2.4

2.3 Переменные в Scilab

В рабочей области Scilab можно определять *переменные*, а затем использовать их в

выражениях. Любая переменная до использования в формулах и выражениях должна быть определена. Для *определения переменной* необходимо набрать имя переменной, символ «=» и значение переменной. Здесь знак равенства это *оператор присваивания*, действие которого не отличается от аналогичных операторов языков программирования. То есть, если в общем виде оператор присваивания записать как

```
имя переменной = значение выражения
```

то в переменную, имя которой указано слева, будет записано значение выражения, указанного справа.

Имя переменной не должно совпадать с именами встроенных процедур, функций и встроенных переменных системы и может содержать до 24 символов. Система различает большие и малые буквы в именах переменных. То есть ABC, abc, Abc, aBc это имена разных переменных. Выражение в правой части оператора присваивания может быть числом, арифметическим выражением, строкой символов или символьным выражением. Если речь идет о символьной или строковой переменной, то выражение в правой части оператора присваивания следует брать в одинарные кавычки.

Если символ «;» в конце выражения отсутствует, то в качестве результата выводится имя переменной и ее значение. Наличие символа «;» передает управление следующей командной строке. Это позволяет использовать имена переменных для записи промежуточных результатов в память компьютера:

```
-->//-----------------------------------------------
-->//Присваивание значений переменным a и b
--> a=2.3
a =
2.3000
--> b=-34.7
b =
-34.7000
-->//Присваивание значений переменным x и y,
-->//вычисление значения переменной  z
--> x=1;y=2; z=(x+y)-a/b
z =
3.0663
-->//Сообщение об ошибке - переменная c не определена
--> c+3/2
??? Undefined function or variable 'c'.
-->//-----------------------------------------------
-->//Определение символьной переменной
--> c='a'
c =
a
-->//Определение строковой переменной
--> h='мама мыла раму'
h =мама мыла раму
```

Листинг 2.5

Для очистки значения переменной можно применить команду

```
clear имя переменной;
```

которая отменяет определения всех переменных данной сессии. Далее приведены примеры применения этой команды:

```
-->//Определение переменных x и y
```

```
--> x=3; y=-1;
-->//Отмена определения переменной x
--> clear x
-->//Переменная x не определена
--> x
??? Undefined function or variable 'x'.
-->//Переменная y определена
--> y
y =
    -1
-->//Определение переменных a и b
-->a=1;b=2;
-->//Отмена определения переменных a и b
-->clear;
-->//Переменные a и b не определены
-->a
  !--error 4
undefined variable : a
-->b
  !--error 4
undefined variable : b
```

Листинг 2.6

2.4 Системные переменные Scilab

Если команда не содержи знака присваивания, то по умолчанию вычисленное значение присваивается специальной *системной переменной* `ans`. Причем полученное значение можно использовать в последующих вычислениях, но важно помнить, что значение `ans` изменяется после каждого вызова команды без оператора присваивания:

```
--> 25.7-3.14
ans =
22.5600
--> //Значение системной переменной равно 22.5600
--> 2*ans
ans =
45.1200
--> //Значение системной переменной увеличено вдвое
--> x=ans^0.3
x =
    3.1355
--> ans
ans =   45.1200
--> //После использования в выражении значение
--> //системной переменной не изменилось и равно 45.1200
```

Листинг 2.7

Результат последней операции без знака присваивания хранится в переменной `ans`. Другие *системные переменные* в Scilab начинаются с символа %:

`%i` мнимая единица ($\sqrt{-1}$);

%pi число π=3.141592653589793;
%e число e=2.7182818;
%inf машинный символ бесконечности (∞);
%NaN неопределенный результат (0/0,∞/∞ и т.п.);
%eps условный ноль %eps=2.22Œ -16.

Все перечисленные переменные можно использовать в математических выражениях:

```
-->a=5.4;b=0.1;
-->F=cos(%pi/3)+(a-b)*%e^2
F  =     39.661997
```

Листинг 2.8

Далее показан пример неверного обращения к системной переменной:

```
-->sin(pi/2)
      !--error 4
undefined variable : pi
```

Листинг 2.9

2.5 Ввод вещественного числа и представление результатов вычислений

Числовые результаты могут быть представлены с плавающей (например, 3.2Œ -6, 6.42E+2), или с фиксированной (например, 4.12, 6.05, 17.5489) точкой. Числа в формате с плавающей точкой представлены в экспоненциальной форме mE±p, где m мантисса (целое или дробное число с десятичной точкой), p порядок (целое число). Для того, чтобы перевести число в экспоненциальной форме к обычному представлению с фиксированной точкой, необходимо мантиссу умножить на десять в степени порядок.

Например,

-6.42E+2 = -6.42.10² = -642 3.2E-6 = 3.2.10-6 =0.0000032

При вводе вещественных чисел для отделения дробной части используется точка. Примеры ввода и вывода вещественных чисел:

```
-->0.123
 ans  =   0.123
-->-6.42e+2
 ans  =   - 642.
-->3.2e-6
 ans  =   0.0000032
```

Листинг 2.10.

Рассмотрим пример вывода значения системной переменной π и некоторой переменной q, определенной пользователем:

```
-->%pi
 %pi  =
    3.1415927
-->q=0123.4567890123456
 q  =
    123.45679
```

Листинг 2.11

Не трудно заметить, что Scilab в качестве результата выводит только восемь значащих цифр. Это формат вывода вещественного числа по умолчанию. Для того, чтобы контролировать количество выводимых на печать разрядов применяют команду `printf` с заданным форматом, который соответствует правилам принятым для этой команды в языке C:

```
-->printf("%1.12f",%pi)
3.141592653590
-->printf("%1.15f",%pi)
3.141592653589793
-->printf("%1.2f",q)
123.46
-->printf("%1.10f",q)
123.4567890123
-->//По умолчанию 6 знаков после запятой
-->printf("%f",q)
123.456789
```

Листинг 2.12

2.6 Функции в Scilab

Все *функции*, используемые в Scilab, можно разделить на два класса:

- *встроенные*;
- *определенные пользователем.*

В общем виде *обращение к функции* в Scilab имеет вид:

```
имя_переменной = имя_функции(переменная1 [, переменная2, …])
```

где

`имя_переменной` переменная , в которую будут записаны результаты работы функции; этот параметр может отсутствовать, тогда значение, вычисленное функцией будет присвоено системной переменной `ans`;

`имя_функции` имя встроенной функции или ранее созданной пользователем ;

`переменная1, переменная2, …` список аргументов функции .

2.6.1 Элементарные математические функции

Пакет Scilab снабжен достаточным количеством всевозможных встроенных функций, знакомство с которыми будет происходить в следующих разделах. Здесь приведем, только элементарные математические функции, используемые чаще всего (табл. 2.1).

Таблица 2.1. Элементарные математические функции

Функция	**Описание функции**
Тригонометрические	
`sin(x)`	синус числа *x*
`cos(x)`	косинус числа *x*
`tan(x)`	тангенс числа *x*
`cotg(x)`	котангенс числа *x*
`asin(x)`	арксинус числа *x*
`acos(x)`	арккосинус числа *x*
`atan(x)`	арктангенс числа *x*
Экспоненциальные	
`exp(x)`	Экспонента числа *x*

Функция	**Описание функции**
`log(x)`	Натуральный логарифм числа *x*
Другие	
`sqrt(x)`	корень квадратный из числа *x*
`abs(x)`	модуль числа *x*
`log10(x)`	десятичный логарифм от числа *x*
`log2(x)`	логарифм по основанию два от числа *x*

Пример вычисления значения выражения $z=\sqrt{\left|\sin(\frac{x}{y})\right|}\cdot e^{x^y}$:

```
-->x=1.2;y=0.3;
-->z=sqrt(abs(sin(x/y)))*exp(x^y)
 z  =
    2.5015073
```

Листинг 2.13

2.6.2 Функции, определенные пользователем

В первой главе мы уже упоминали о *файлах-сценариях* и даже создавали небольшую программу, которая решала конкретное квадратное уравнение. Но в эту программу невозможно было передать входные параметры, то есть это был обычный список команд, воспринимаемый системой как единый оператор.

Функция, как правило, предназначена для неоднократного использования, она имеет входные параметры и не выполняется без их предварительного задания. Рассмотрим несколько способов *создания функций* в Scilab.

Первый способ это применение оператора deff, который в общем виде можно записать так:

```
deff('[имя1,...,имяN]=имя_функции(переменная_1,...,переменная_M)',
'имя1=выражение1;...;имяN=выражениеN')
```

где `имя1,...,имяN` список выходных параметров, то есть переменных, которым будет присвоен конечный результат вычислений (параметров может быть от 1 до N), `имя_функции` имя с которым эта функция будет вызываться, `переменная_1,...,переменная_M` входные параметры (параметров может быть от 1 до M).

Далее приведен самый простой способ применения оператора `deff`. Здесь показано как создать и применить функцию для вычисления выражения $z=\sqrt{\left|\sin(\frac{x}{y})\right|}\cdot e^{x^y}$ (значение этого выражения уже было вычислено в листинге 2.13):

```
-->deff('z=fun1(x,y)','z=sqrt(abs(sin(x/y)))*exp(x^y)');
-->x=1.2;y=0.3;z=fun1(x,y)
 z  = 2.5015073
```

Листинг 2.14

Рассмотрим пример создания и применения функции, вычисляющей площадь треугольника со сторонами a, b и c по формуле Герона $S=\sqrt{(p-a)\cdot(p-b)\cdot(p-c)}$, где $p=\frac{(a+b+c)}{2}$:

```
-->deff('S=G(a,b,c)','p=(a+b+c)/2;S=sqrt((p-a)*(p-b)*(p-c))');
-->G(2,3,3)
 ans  = 1.4142136
```

Листинг 2.15

В следующем листинге приведен пример создания и применения функции, с помощью которой можно найти корни квадратного уравнения вида $ax^2+dx+c=0$ по формулам $D=b^2-4ac;\ x_{1,2}=\frac{-b\pm\sqrt{D}}{2a};$:

```
-->deff('[x1,x2]=korni(a,b,c)','d=b^2-4*a*c;
                        x1=(-b+sqrt(d))/2/a;x2=(-b-sqrt(d))/2/a');
-->[x1,x2]=korni(-2,-3,5)
 x2  = 1.
 x1  = 2.5
```

Листинг 2.16

Второй способ создания функции это применение конструкции вида:

```
function[имя1,...,имяN]=имя_функции(переменная_1,...,переменная_M)
    тело функции
endfunction
```

где `имя1,...,имяN` список выходных параметров, то есть переменных, которым будет присвоен конечный результат вычислений (параметров может быть от 1 до N), `имя_функции` имя с которым эта функция будет вызываться, `переменная_1,...,переменная_M` входные параметры (параметров может быть от 1 до M).

Все имена переменных внутри функции, а так же имена из списка входных и выходных параметров воспринимаются системой как *локальные*, то есть эти переменные считаются определенными только внутри функции.

Вообще говоря, функции в Scilab играют роль подпрограмм. Поэтому целесообразно набирать их тексты в редакторе и сохранять в виде отдельных файлов. Причем имя файла должно обязательно совпадать с именем функции. Расширение файлам-функциям обычно присваивают sci или sce.

Обращение к функции осуществляется так же, как и к любой другой встроенной функции системы, то есть из командной строки. Однако функции, хранящиеся в отдельных файлах должны быть предварительно загружены в систему, например при помощи оператора `exec(иям_файла)` или командой главного меню **File** **Exec..**, что в общем, одно и то же.

Решим несколько задач.

ЗАДАЧА 2.1.

Решить кубическое уравнение.

Кубическое уравнение

$$ax^3+bx^2+cx+d=0 \qquad (2.1)$$

после деления на *a* принимает канонический вид:

$$x^3+rx^2+sx+t=0\ , \qquad (2.2)$$

где

$$r=\frac{b}{a}\ ,\quad s=\frac{c}{a}\ ,\quad t=\frac{d}{a}\ .$$

В уравнении (2.2) сделаем замену

$$x = y - \frac{r}{3}$$

и получим следующее приведенное уравнение:

$$y^3 + py + q = 0 \quad , \tag{2.3}$$

где

$$p = \frac{(3s - r^2)}{3} \quad , \quad q = \frac{2r^3}{27} - \frac{rs}{3} + t \quad .$$

Число действительных корней приведенного уравнения (2.3) зависит от знака дискриминанта $D = (\frac{p}{3})^3 + (\frac{q}{2})^3$ (табл. 2.2).

Таблица 2.2. Количество корней кубического уравнения

Дискриминант	**Количество действительных корней**	**Количество комплексных корней**
D≥0	1	2
D<0	3	-

Корни приведенного уравнения могут быть рассчитаны по формулам Кардано:

$$y_1 = u + v \, , \; y_2 = \frac{-(u+v)}{2} + \frac{(u-v)}{2} i\sqrt{3} \, , \; y_3 = \frac{-(u+v)}{2} - \frac{(u-v)}{2} i\sqrt{3} \tag{2.4}$$

Здесь

$$u = \sqrt[3]{\frac{-q}{2} + \sqrt{(D)}} \, , \; v = \sqrt[3]{\frac{-q}{2} - \sqrt{(D)}} \quad .$$

Далее представлен список команд, реализующий описанный выше способ решения кубического уравнения:

```
function [x1,x2,x3]=cub(a,b,c,d)
r=b/a;
s=c/a;
t=d/a;
p=(3*s-r^2)/3;
q=2*r^3/27-r*s/3+t;
D=(p/3)^3+(q/2)^2;
u=(-q/2+sqrt(D))^(1/3);
v=(-q/2-sqrt(D))^(1/3);
y1=u+v;
y2=-(u+v)/2+(u-v)/2*%i*sqrt(3);
y3=-(u+v)/2-(u-v)/2*%i*sqrt(3);
x1=y1-r/3;
x2=y2-r/3;
x3=y3-r/3;
endfunction
//Вызов функции и вывод результатов ее работы:
-->exec('C:\Scilab\scilab-4.1.1\cub.sce');
-->disp('exec done');
```

```
Warning :redefining function: cub
 exec done
-->[x1,x2,x3]=cub(3,-20,-3,4)
 x3  =
    0.3880206
 x2  =
  - 0.5064407
 x1  =
    6.7850868
```

Листинг 2.17