# Средства Scilab для создания и описания sci-файлов


Лекция 2

В SciLab существует два типа **sci**-файлов: **файл-сценарии** и **sci**-функции.

***Файл-сценарий*** **-** это последовательность команд и функции Scilab (**без входных и выходных параметров**), которые оперируют данными из **Рабочей области**, причем результаты выполнения ***сценария*** доступны **Рабочей области** и могут быть использованы для дальнейших вычислений.

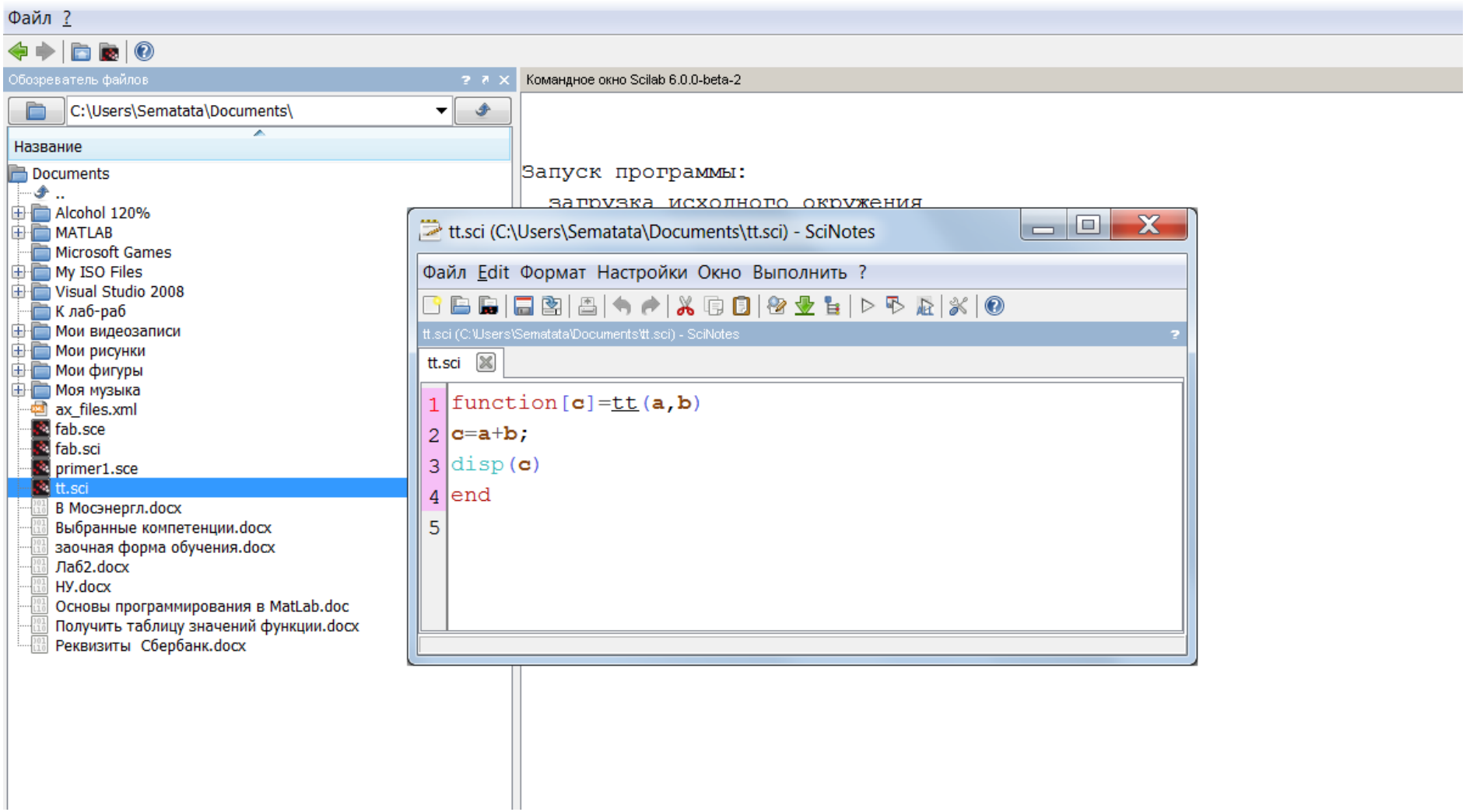
***Sci-функции*** – это функции Scilab, аналогичные функциям языков программирования.

Для создания нового **sci**-файла нужно путем активизации инструмента **SciNotes** открыть Редактор (1-я кнопка панели инструментов)**.**

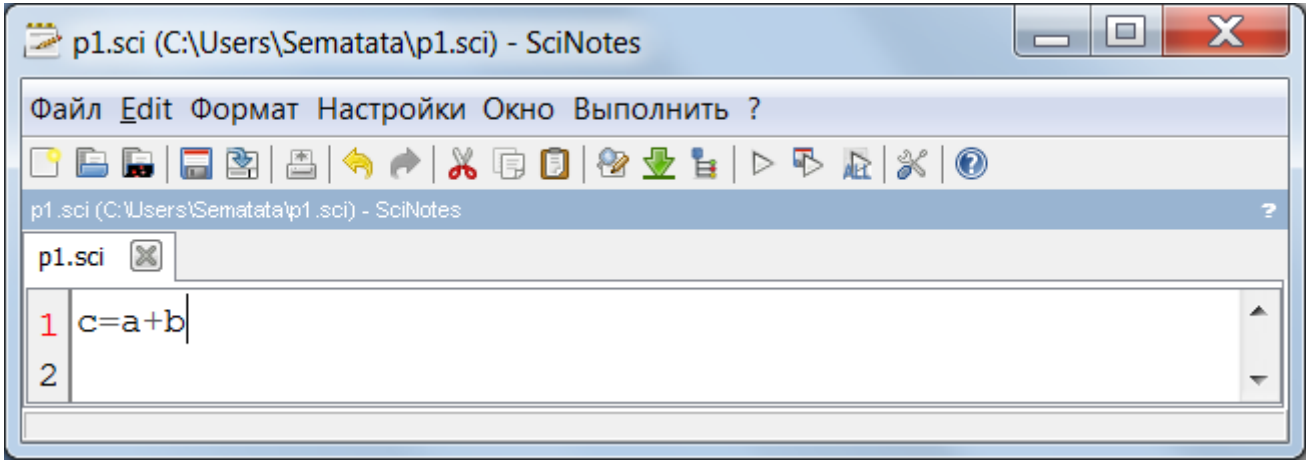
Для редактирования существующего **sci**-файла используется инструмент **Открыть** или двойной щелчок мышки по его имени.

Для сохранения созданного **sci-**файла в меню Файл выполнить команду **Сохранить как** и задав файлу имя нажать кнопку **Save.** Имя файла с расширением **.sci** появится в окне **Обозревателя файлов**

# Создание новых Script-файла (окно редактора)



# Выполнение sci-файла




Запуск файла **p1.sci**, находящегося в текущем каталоге, из командного окна осуществляется командой exec(‘имя файла’).

→ **exec('p1.sci'); //вывод содержимого sci-файла гасится ;**

Командное окно Scilab 6.0.0-beta-2

```
--> exec('p1.sci');  //вывод содержимого sci-файла гасится точкой с запятой
 c  =
    1.
-->  exec('p1.sci')
--> c=a+b
 c  =
    1.
```

# Особенности sci-функций

- начинаются с заголовка описания **sci**- функции;
- могут иметь входные и выходные параметры;
- все переменные, описанные в теле функции являются **локальными**, т.е. действуют только в пределах тела функции;
- **sci**- функции являются самостоятельными программными единицам, которые общаются с другими модулями посредством **имени** с **входными** и **выходными** параметрами.

# Структура sci-функции

Общая структура **sci-**функции с **n** входными и **m** выходными параметрами имеет вид:

**Function[***var1, …, var**m**,…***]** **=***f_name (список входных параметров)*

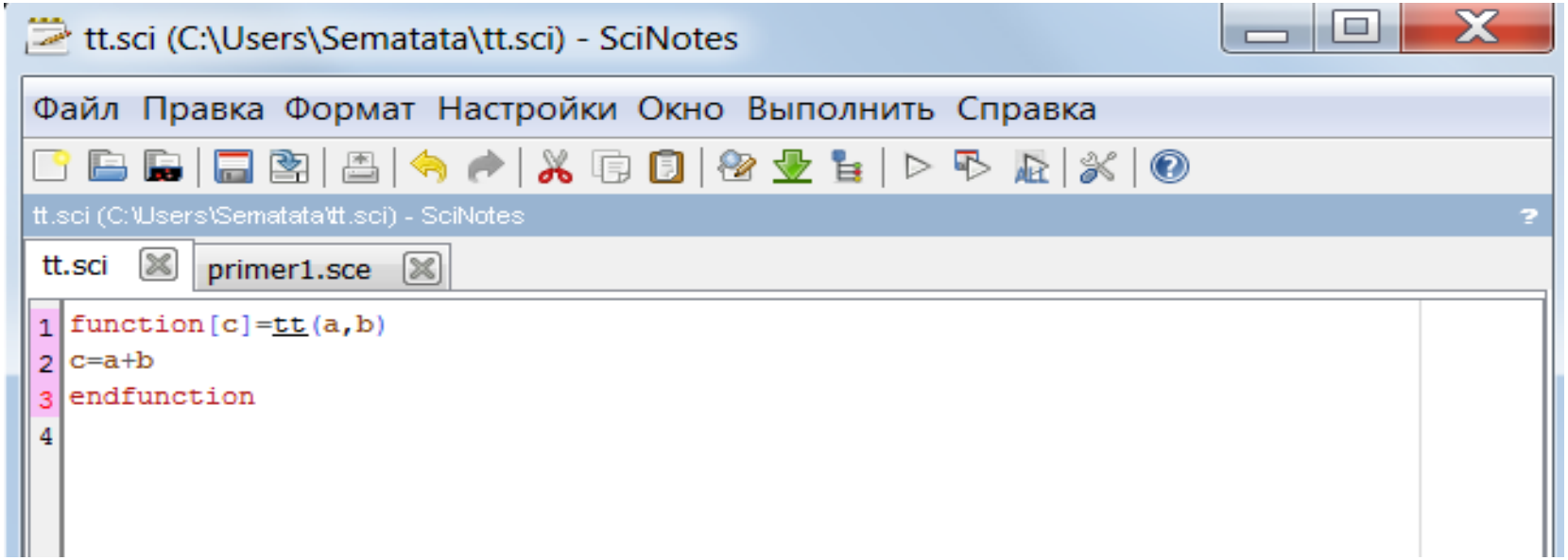
*Var1 = выражение*

*…*

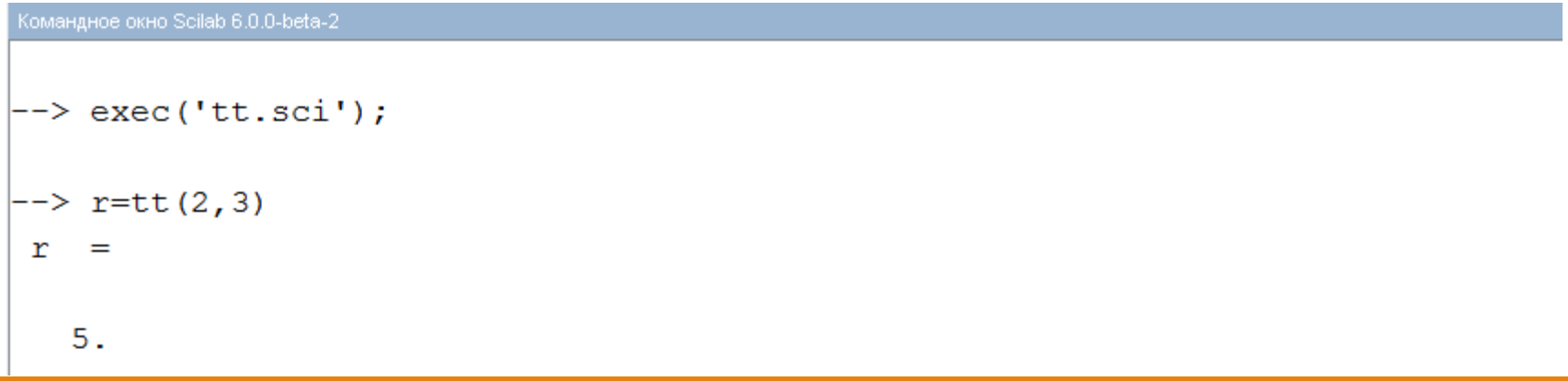
*Var**m** = выражение*

**Endfunction**

# Создание sci-функции




# Вызов sci-функции и выполнение



# Создание sci-функции с несколькими выходными параметрами

```
1 function [x1,x2]=quadeq(a,b,c)
2 //Нахождение корней квадратного уравнения
3 D=b^2-4*a*c;
4 x1=(-b+sqrt(D))/(2*a);
5 x2=(-b-sqrt(D))/(2*a);
6 endfunction
```

Командное окно Scilab 6.0.0-beta-2

```
--> exec('C:\Users\Sematata\quadeg.sci');

--> [r1,r2]=quadeg(1,3,2)
 r2  =

  -2.

 r1  =

  -1.
```

# Алгоритмические операторы Scilab

# Примеры операторов присваивания

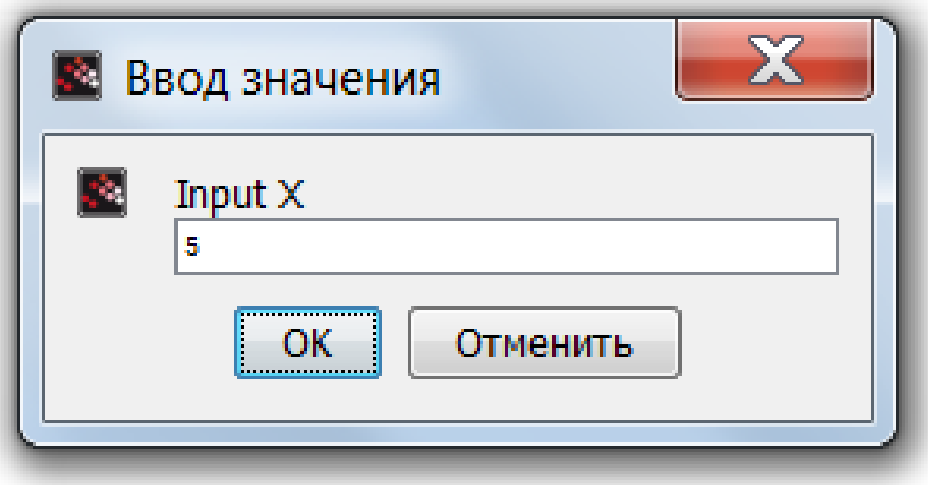
Командное окно Scilab 6.0.0-beta-2

```
--> a=2;

--> x=a^2-20;

--> x
 x  =

  -16.


--> A=[2 -4 7];

--> A
 A  =

   2.   -4.    7.
```

# Ввод данных с клавиатуры

Командное окно Scilab 6.0.0-beta-2

```
--> x=x_dialog('Input X','5')
```



# Пример усеченного разветвления

Командное окно Scilab 6.0.0-beta-2

```
--> a=4;

--> if a>0 r=sqrt(a); end

--> r
 r   =

    2.
```

# Пример стандартного разветвления

```
--> a=4;

--> if a>0 x=sqrt(a); else disp ('Подкоренное выражение <0'); end

--> x
 x  =

    2.


--> a=-4;

--> if a>0
  > x=sqrt(a)
  > else
  > disp ('Подкоренное выражение <0');
  > end

 Подкоренное выражение <0
```

$$t = \begin{cases} \max\{x, y\}, если\ xy < 0; \\ \max\{x^2, \sin(y), \cos(x)\}; если\ xy > 2; \\ x / y; \quad в\ противном\ случае. \end{cases}$$

```
function [t]=raz(x, y)
if x*y<0 then
      t=x;
      if y>t
        t=y;
      end
elseif    x*y>2
      t=x^2;
      if sin(y)>t
         t=sin(y);
      end
      if cos(x)>t
         t=cos(x);
      end
else
      t=x/y;
end
endfunction
```

# *Оператор множественного выбора switch*

**switch** *Выражение*

**case** *Зачение_1*

*Список_инструкций_1*

**case** *Значение_2*

*Список_инструкций_2*

**…**

**case** *Значение_N*

*Список_инструкций_N*

**otherwise**

*Список_инструкций_N+1*

**end**

$$t = \begin{cases} y = x, если \quad n = 1; \\ y = x(10 - x), если \quad n = 2; \\ y = xSin(nx), если \quad n = 3,4,5; \\ y = 1/(1 + x^2), в противном случае. \end{cases}$$

```
function [y]=multifunc(x, n)
 select n
    case 1 then y=x;
    case 2 then y=x*(10-x);
    case {3,4,5} then y=x*sin(n*x);
 else
   y=1/(1+x^2);
 end
endfunction
```

# Оператор регулярного цикла for...end

**for** **var = s:d:e**
*Инструкция1*
*….*
*ИнструкцияN*
**end**

где **s -** начальное значение переменной цикла **var**, **d -** приращение этой переменной и **e -** конечное значение управляющей переменной, при превышении которого цикл завершается. Возможна и запись в виде **s:e** (в этом случае **d=l**).

# Пример1: Найти сумму одномерного массива X

```
function [s]=summa(x,n)
//Сумма элементов массива x
s=0;
for i=1:n s=s+x(i);
end
endfunction
```

# Пример2: Найти сумму двумерного  массива a(3,3)

```
function [s]=summa_a(a)
//Сумма элементов двумерного массива a(3,3)
s=0;
for i=1:3
    for j=1:3
        s=s+a(i,j);
    end
end
endfunction
```

# Оператор итеративного цикла – while…end

**while** *ЛогическоеВыражение*

*Инструкции*

**End**

```
1  function zicl(r)
2  //Вычисление длины окружности радиусов r
3  while r>=0;
4      r=input('Введите радиус окружности r=');
5      if r>0 disp('  Длина окружности l='; disp(2*%pi*r),end
6  end
7  endfunction
```