ОСНОВИ ПРОГРАМУВАННЯ В ПАКЕТІ SCILAB

ПРОГРАМУВАННЯ В SCILAB

Робота в Scilab може здійснюватись не лише в режимі командної стрічки, але і в так званому програмному режимі.

Для зручності написання програм та скриптів (функцій) в Scilab є вбудований редактор - SciNotes.

Він дозволяє редагувати тексти функцій, виконувати їх в режимі відлагодження, містить функцію автодоповнення коду, а також засоби безпосередньої передачі тексту програми в середовище Scilab на виконання.

Відкрити редактор можна двома способами:

- набрати в консолі Scilab команду scinotes
- вибрати в головному вікні послідовно пункти меню Програми → SciNotes.

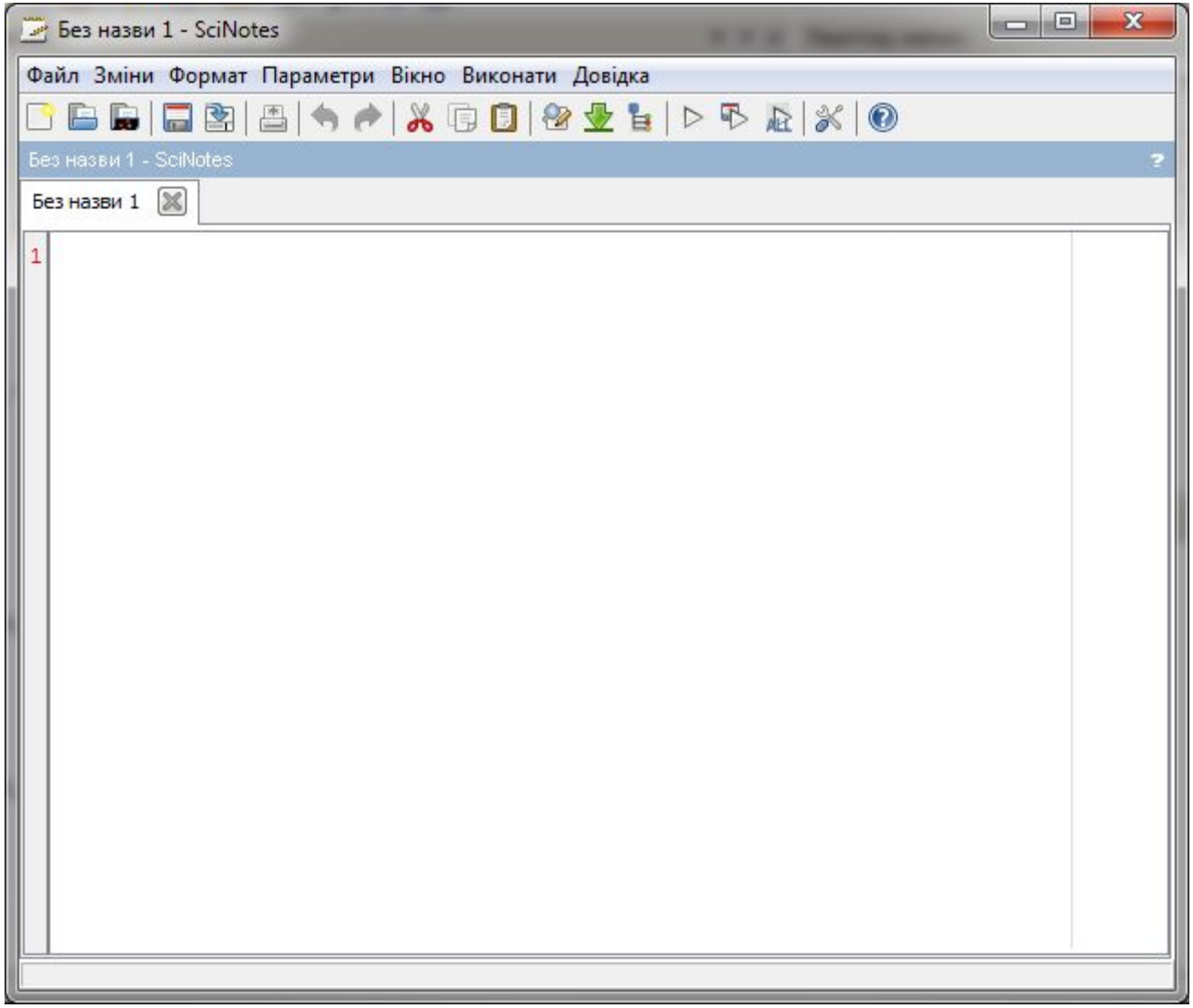

Рис 1. Вікно редактора SciNotes

Коротко розглянемо пункти меню:

- Файл (File) - тут знаходяться стандартні команди для роботи з файлами: відкрити (Open), закрити (Close file), зберегти (Save), також налаштування сторінки, перегляду друку і друк (Print) і т. д.
- Зміни (Edit) - містить стандартні для пункту меню Зміни операції: копіювати (Copy), вставити (Paste), вирізати (Cut), позначити все (Select All), функції пошуку по тексту і т. д.

- Формат (Format) – тут знаходяться команди для форматування тексту програми
- Параметри (Option) - тут знаходиться досить багато пунктів, які дозволяють налаштувати зовнішній вигляд і поведінку редактора від типу шрифту до гарячих клавіш.
- Вікно (Window) - команди управління робочим вікном. Дозволяють розбити вікно на частини по вертикалі і горизонталі, а також впорядкувати розміщення частин вікна.

- Виконати (Execute) - містить пункти, що дозволяють передати вміст редактора в середовище Scilab на виконання або виконати лише виділену частину.
- Довідка (Help) –містить довідку зі SciNotes.

Для створення програми в Scilab (програму в Scilab інколи називають сценарієм) необхідно:

- Викликати програму SciNotes з меню;
- У вікні редактора SciNotes набрати текст програми;
- Зберегти текст програми за допомогою команди Файл – Зберегти (File-Save) у вигляді файла з розширенням sce, наприклад, file.sce;

- Після цього програму можна викликати набравши в командній стрічці наприклад, exec file.sce. Інші способи виклику програми –використати команду меню Файл – Виконати (File- Exec) або знаходячись у вікні SciNotes виконати команду Виконати - …з виведенням (Ctrl+L).

Програмна мова достатньо зручна, оскільки вона дозволяє зберегти розроблений обчислювальний алгоритм у вигляді файла і повторювати його при інших вихідних даних у інших сесіях. Крім звертань до функції та операторів присвоєння в програмних файлах можуть використовуватись оператори мови програмування Scilab (мову програмування Scilab будемо називати Sci-мовою.)

ОСНОВНІ ОПЕРАТОРИ SCI-МОВИ.

Вивчення Sci-мови розпочнемо з вивчення функції **вводу – виводу.**

Для організації простого вводу можна скористатися простими функціями

x=input(’title’) або

x=x_dialog(’title’, ’string’);

Функція Input виводить в командній стрічці Scilab підказку title і очікує, поки користувач введе значення, яке у якості результату повертається у змінну x.

Функція x_dialog виводить на екран діалогове вікно з іменем title, після чого користувач може натиснути Ok і тоді string вертається у вигляді результату у змінну x, або ввести нове значення замість string , яке повертається у вигляді результату у змінну x.

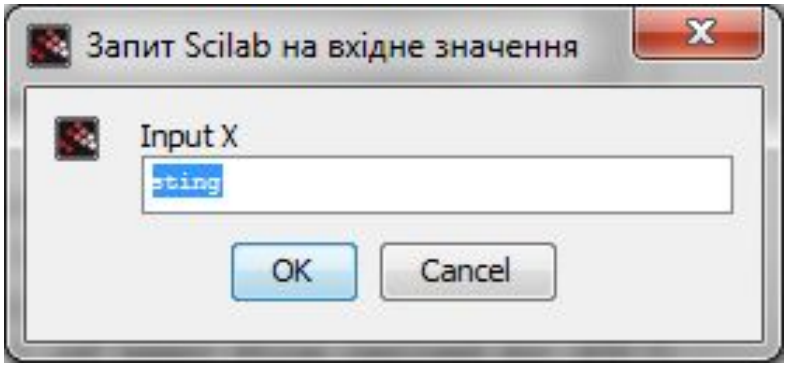

Рис. 2. Запит Scilab на вхідне значення.

Для виводу в текстовому режимі можна використати функцію disp наступної структури:

disp(b)

b – ім’я змінної або текст у лапках.

Оператор присвоєння

Оператор присвоєння має таку структуру:

a=b

a-ім’я змінної або елемента масиву,

b – значення або вираз. В результаті виконання оператора присвоєння змінній a присвоюється значення виразу **b**.

ОПЕРАТОРИ РОЗГАЛУЖЕННЯ ТА ЦИКЛУ

В Scilab є такі оператори розгалуження та циклу:

1) умовний оператор **if**
2) оператор вибору **select**
3) цикли **for** і **while**
4) інструкції **break** і **continue**

Умовний оператор if

Одним із основних операторів, який реалізує розгалуження у більшості мов програмування є умовний оператор **if**. Є звичайна та розширена форма оператора **if**. Звичайна форма має вигляд:

If *умова* ***then***

оператори 1

else *оператори 2*

end

Тут умова – логічний вираз, оператори 1, оператори 2 – оператори мови Scilab. Оператор if працює по такому алгоритму : якщо умова є істинною, то виконується оператор 1, якщо хибною, то оператор 2.

В Scilab для запису логічних виразів можуть використовуватись умовні оператори: &, and (логічне і), |, or (логічне або), ~, not (логічне заперечення), < (менше) , > (більше), == (дорівнює), ~=, <>(не дорівнює), >= (більше або дорівнює), <= (менше або дорівнює).

Найчастіше при розв’язуванні практичних задач недостатньо вибору однієї умови. У цьому випадку використовують розширену форму оператора **if** :

If *умова 1* ***then*** *Оператори 1*

Elseif *умова 2* ***then*** *Оператори 2*

…

Elseif *умова n* ***then*** *Оператори n*

Else *оператори*

End

У цьому випадку оператор **if** працює так:

якщо умова1 істина, то виконуються оператори 1, інакше перевіряється умова 2, якщо вона істинна, то виконуються оператори 2, інакше перевіряється умова 3 і т.д. Якщо ні одна із умов за розгалуженням **elseif** не виконується, то виконуються оператори за розгалуженням **else**.

Оператор **if** дозволяє виконати деякий блок інструкцій, коли умова є істинна. Умовою може бути змінна логічного типу або будь-який вираз, результатом обчислення якого є логічне значення. Блок завершується ключовим словом end.

Приклад. Обчислити значення функції y:

$$y = \begin{cases} x^2, & x < 0, \\ \cos x, & 0 \le x < 4, \\ \sqrt{x}, & x \ge 4. \end{cases}$$

```
x=input('input x=')
if x<0 then y=x^2
    elseif ((x>=0) & (x<4)) then y=cos(x)
    elseif x>=4 y=sqrt(x)
end
disp(y,x)
```

Задаючи різні значення x (-2, 0, 4) отримаємо такі результати:

input x=-2

- 2.

4.

input x=0

0.

1.

input x=4

4.

2.

Оператор select

Оператор select призначений для скороченого запису декількох послідовних перевірок змінної на рівність одному із ряду значень, які в іншому випадку необхідно було б записати у вигляді блоків elseif. В залежності від значення змінної оператор select виконує один з блоків case. Кількість таких блоків необмежена.

Нижче показано, як відобразити один із декількох можливих рядків відповідно до значення змінної i:

```
i = 2
select i
case 1 then disp ("Один")
case 2 then disp ("Два")
case 3 then disp ("Три")
else disp ("Інше значення")
end
```

Оператор циклу while

Оператор циклу while має такий вигляд:

while *умова*

Тіло циклу

end

Умова – це логічний вираз. Тіло циклу буде виконуватись циклічно до того часу, поки результат логічного виразу має істинне значення (true).

Оператор while призначений для повторення деякого блоку інструкцій до того часу, поки умова циклу є істинною. Перевірка умови виконується перед кожною (в тому числі першою) ітерацією. У певний момент умова повторення переходить в хибність (false) і цикл завершується.

Оскільки умова перевіряється перед виконанням тіла циклу, то тіло циклу може не виконуватись жодного разу, якщо умова буде хибність (false)

Наступний фрагмент програми демонструє застосування циклу while для сумування чисел від 1 до 10:

```
s = 0
i = 1
while ( i <= 10 )
    s = s + i
    i = i + 1
end
disp(s,’s=‘,i,’i=‘)
```

Значення змінних після закінчення виконання програми:

i =

11.

s=

55.

Зверніть увагу на порядок виводу функції **disp** : він є зворотнім, тобто останні елементи зі списку виводу виводяться першими

Оператор циклу for

Загальний вигляд оператора циклу **for** наступний:

for *<лічильник>=<вираз>*

<Тіло циклу>

end

У якості виразу може виступати все що завгодно. Якщо в якості виразу вказується вектор (матриця), то змінна-лічильник послідовно набуває всіх значень цього вектора (матриці).

Приклад:

```
a=[1,3,5,7]
for i=a
    disp(i)
end
```

В результаті отримаємо:

1.

3.

5.

7.

Якщо згадати, як в Scilab створюються масиви значень, то можна привести цю конструкцію до стандартного вигляду для циклу for у всіх мовах програмування:

for *x=xn:hx:xk*

Тіло циклу

end

x - лічильник – параметр циклу, *xn*- початкове значення параметра циклу, *xk* – кінцеве значення параметра циклу, *hx* – крок циклу.

Виконання циклу починається з присвоєння параметру стартового значення (x = xn). Потім іде перевірка , чи параметр не є більшим за кінцеве значення (x>xk). Якщо x>xk, то цикл вважається завершеним, і управління передається наступному за тілом циклу оператору. Якщо x<xk, то виконуються оператори тіла циклу. Далі параметр циклу збільшує своє значення на hx (x = x + hx). Після чого знову проводиться перевірка значення параметра циклу, і алгоритм повторюється.

Якщо крок циклу дорівнює 1, то hx можна опустити, і в цьому випадку оператор for буде мати такий вигляд:

for *x=xn:xk*

Тіло циклу

end

Наступний фрагмент програми виводить значення і від 1 до 5.

```
for i = 1 : 5
disp (i)
End
```

В консолі буде виведений результат:

1.
2.
3.
4.
5.

Використовуючи більш загальну форму запису оператора **for**, можна вивести тільки непарні числа в інтервалі від 1 до 5.

Для цього, очевидно, крок буде дорівнювати значенню 2:

```
for i = 1 : 2 : 5
disp (i)
end
```

В консолі буде виведено результат:

1.

3.

5.

Оператор **for** можна також використовувати для перебору значень лічильника в порядку зменшення. Наступний фрагмент програми виводить числа від 5 до 1 в порядку зменшення:

```
for i = 5 : -1 : 1
disp (i)
end
```

В консоль буде виведено результат:

5.
4.
3.
2.
1.

Цикл **for** є універсальним і дозволяє використовувати значення різних типів, у тому числі матриць і списків. Елементами матриць можуть бути дійсні та цілі числа, рядки і поліноми. В наступному прикладі цикл **for** використовується для елементів вектора-рядка, який складається з дійсних значень (1.5; e; pi).

```
v = [1.5  %e  %pi ];
for x = v
disp (x)
end
```

В консолі буде результат:

1.5

2.7182818

3.1415927

Інструкції break і continue

Інструкція break дозволяє перервати виконання циклу. Зазвичай вона застосовується для виходу з циклу при досягненні деякої визначеної умови. Наступний фрагмент програми демонструє використання інструкції break для обчислення суми чисел від 1 до 10. При досягненні змінною і значення, більшого за 10, цикл завершується:

```
s = 0
i = 1
while ( %t )
   if ( i > 10 ) then
      break
   end
   s = s + i
   i = i + 1
end
disp(I,'i=',s,'s=')
```

В консолі отримаємо результат:

s=

55.

i=

11.

Інструкція **continue** дозволяє негайно перейти до виконання наступної ітерації, пропустивши команди, які ідуть після continue в тілі циклу. В такому випадку інтерпретатор Scilab переходить до заголовка циклу, перевіряє умову продовження, і, якщо вона істинна, виконує наступну ітерацію.

Наступний приклад демонструє обчислення суми s = 1+3+5+7+9 =25. Використана функція цілочисельного ділення pmodulo(i,2) повертає 0 при умові, що i - парне число. В даному випадку скрипт нарощує значення i та використовує інструкцію continue для переходу до наступної ітерації:

```
s = 0
i = 0
while ( i < 10 )
    i = i + 1
    if ( modulo ( i , 2 ) == 0 ) then
        continue
    end
    s = s + i
end
disp(I,’i=‘,s,’s=‘)
```

В консолі отримаємо результат:

s=

25.

i=

10.

Дякую за увагу!