Тема 6. Функції. Процедури. Пакети. Бібліотеки 2D графіки

Функції

Функція (відображення, оператор, перетворення) - математичне поняття, що відображає зв'язок між елементами множин.

Функція - це правило, за яким кожному елементу однієї множини (область визначення) ставиться у відповідність деякий елемент іншої множини (область значень).

Часто під терміном «функція» розуміють числову функцію - коли одні числа ставляться у відповідність іншим.

Термін «функція» був введений Лейбніцом (1692).

Розрізняють функції:

- в залежності змінних - одної змінної, багатьох змінних і т.д.;
- формі представлення – таблична, аналітична;
- вигляду – алгебраїчна, тригонометрична тощо.

Функції в комп’ютерних системах поділяються на чотири типи:

- вбудовані в ядро системи;
- функції користувача;
- бібліотечні функції із пакетів або ж із бібліотек розширення;
- функції, які задані у вигляді програмного модуля.

Maple має великий набір стандартних математичних функцій, як елементарних, так і спеціальних.

Функція	Синтаксис Maple		Функція	Синтаксис Maple
ex	*exp(x)*		$\sqrt{x}$	*sqrt(x)*
ln(x)	*ln(x) або log(x)*		$\|x\|$	*abs(x)*
$\log_{10}(x)$	*log10(x)*		*sgn(x)*	*signum(x)*
$\log_a(x)$	*log[a](x)*		*n!*	*n! абож factorial(n)*

Визначення функцій

В Maple існує декілька способів визначення функцій:

- присвоювання змінній деякого виразу, наприклад:

F0:=cos(t)+1;

- за допомогою функціонального оператора ім'я функції:=список параметрів -> вираз; наприклад:

F1:=(x, y)→x(y^2);

- за допомогою команди *unapply(вираз, параметри)*, яка перетворює вираз в функціональний оператор, наприклад:

F2:=unapply(F0,t);

- за допомогою оператора *define(оператор, властивість1, властивість2, ...)*.

В першому випадку *F0* не є повноцінною функцією користувача, оскільки в ньому використовуються тільки глобальні змінні.

Maple не зрозуміє звертання до *F0* як до функції у вигляді *F0(0.1)*. Потрібно задати наступним чином:

t=0.1: F0:

що не завжди зручно.

У другому та третьому способі, змінні вказані у списку формальних параметрів, є локальними. Для цих способів звернення до функцій відбувається: *ім'я(a, b, ...)*, де *a, b, ...* - конкретні значення змінних.

Недоцільно привласненням := створювати функцію користувача наступним чином:

Fn(x):=(x^3:

Хоча Maple2016 дозволяє це зробити. *Fn(2)=8.*

Процедури-функції користувача

Процедура (функція) - сукупність оголошень і операторів для виконання деякої окремої логічно-оформленої задачі.

Процедури є важливим елементом структурного програмування.

Процедура на *Maple*-мові виглядає наступним чином:

name:=proc (формальні параметри)
вираз від формальних параметрів
end proc;

З використанням функції пов'язані поняття: **оголошення, виклик** і **визначення**.

Оголошення специфікує **назву** функції, **тип** яке повертається функцією, **атрибути** її формальних параметрів.

Формальні параметри функції - змінні, які приймають значення при виклику функції в послідовності їх опису списку параметрів.

Ідентифікатори формальних параметрів не повинні співпадати з ідентифікаторами змінних, які оголошуються в середині тіла функції (оператор *local*).

За допомогою оператора :: після формального параметра можна задати його тип. Наприклад, в оголошенні *proc(n::Integer)* змінна *n* є

цілою. Невідповідність фактичних параметрів типу заданих змінних веде до повідомлення про помилку і до відмови виконання процедури.

Для присвоєння результату роботи функції різних значень і виходу з неї в будь-якому місці використовують оператор *return(val).*

Процедури можуть не повертати яких-небудь значень після свого виконання.

Тіло функції представляє собою складений оператор. В *Maple* змінні які оголошені в тілі функції є локальними.

Загальний вигляд процедури:

```
name:=proc (parameters)
   local val1, ..., valn;
   ;
   res2;
end;
```

Виклик функції передає управління і фактичні аргументи (якщо вони є) заданій функції. При виконанні виклику функції відбувається

присвоєння фактичних аргументів відповідним формальним аргументам. Передача управління передається на перший оператор тіла функції. Повертається значення за замовчуванням останнього оператора з тіла функції. Виклик функції здійснюється по імені.

Приклад, процедура з ім'ям *f*, що обчислює суму змінних *x* і *y*:

```
F:=proc (x, y)
   x+y;
end;
```

Фактичні аргументи можуть бути будь-якими значеннями.

Викликана функція працює з копіями фактичних аргументів, тому ніякі зміни значень формальних аргументів не будуть впливати на зміну фактичних аргументів.

Формування **бібліотеки** здійснюється **.lib* функцій.

```
save sqrt "D:\\ParticleSurf\\BaseGeom.lib";
```

Включення бібліотеки у програмний код.

```
read "D:\\ParticleSurf\\BaseGeom.lib";
```

Пакети розширення *Maple*

Частина функцій *Maple* можуть перебувати в стандартній бібліотеці і в пакетах розширення.

Пакет являє собою набір команд для рішення завдань, що відносяться до певних розділів математики. Наприклад в пакеті *stats* зібрані команди для статистичної обробки результатів і т.д.

Перед використанням функцій пакетів їх треба завантажити окремо або цілим пакетом.

Список пакетів розширення можна отримати використовуючи команду:

? index [package]

Отримати інформацію про пакет розширення і знайти список входять до нього функцій можна за допомогою команди:

? name_package

Для звернення до функцій того чи іншого пакета використовується його повне завантаження командою:

with(пакет)

Підключивши пакет користувач може викликати всі його команди, просто набираючи їх ім’я і необхідні для виконання параметри прямо в області введення.

Якщо необхідно кілька конкретних функцій пакета, то замість підключення всього пакету цілком можна підключити тільки ці необхідні функції:

with(пакет, f1, f2, ...)

або

with (пакет, [f1, f2, ...])

Також команду можна завантажити тільки на час її виконання, після чого вона буде вивантажена з пам’яті. Для цього слід вказати її повне ім’я, що складається з імені пакета та імені самої команди:

Ім’я пакета [ім’я команди] (...);

Ввод / вивод

Для інтерактивного вводу рядку є функція *readline()*.

Для збереження результатів роботи на диску необхідно виконати команду: *writeto (`name.ext`).*

Для зчитування числової інформації з файлу використовується команда: *readdata (name, options, posint)* - де *name* - ім'я файлу, *options* - тип змінних (*integer* / *float*), *posint* - лічильник чисел (тобто скільки зчитувати чисел з рядка).

Зчитана інформація представляється у вигляді змінної типу *list*. Наприклад, нехай в текстовому файлі *a.txt* в рядку знаходяться числа: 1 2 3 4 5 6 7 8 9.

Наведена послідовність команд зчитує дану інформацію з файлу в змінну *u*, після чого обчислюється сума даних чисел.

```
Data: = readdata (`a.txt`, integer, 9);
u: = [[1,2,3,4,5,6,7,8,9]]
Sum (data [1] [k], k = 1..9);
```

Бібліотеки 2D графіки

У ядро *Maple* вбудовано дві функції графіки - побудова двовимірних графіків (*2D*-типу) - *plot*() і побудова тривимірних графіків (*3D*-типу) - *plot3d*(). Графіки будуються у вигляді ряду точок, які з'єднуються відрізками прямих.

Опції оператора *plot*

Найпростішими формами завдання функції *plot*() є:

3. *plot(f, xmin..xmax)* - побудова графіка функції *f*, заданої тільки ім'ям;
4. *plot(f(x), x=xmin..xmax)* - побудова графіка функції *f(x)*.

Основні параметри команди *plot*:

- *title="text"*, де *text*-заголовок графіка.
- coords=polar - установка полярних координат (за умовчанням декартові). Інші варіанти: *bipolar*, *cardioid*, *cassinian*, *elliptic*, *hyperbolic*, *invcassinian*, *invelliptic*, *logarithmic*, *logcosh*, *maxwell*, *parabolic*, *polar*, *rose*, *tangent*.

- *axes* - установка типу координатних осей: *axes=NORMAL* - звичайні осі; *axes=BOXED* - графік у рамці зі шкалою; *axes=FRAME* - осі з центром в лівому нижньому кутку рисунку; *axes=NONE* - без осей.
- *scaling* - установка масштабу: *scaling=CONSTRAINED* - однаковий масштаб по осях; *scaling=UNCONSTRAINED* - графік масштабується за розмірами вікна.
- *style=LINE* - побудова лініями, *style=POINT* побудова точками.
- *numpoints=n* - число обчислюваних точок графіка (за умовчанням *n=50*).
- color - установка кольору лінії: наприклад, yellow - жовтий і т.д. Інші варіанти: *aquamarine black blue navy coral cyan brown gold green gray grey khaki magenta maroon orange pink plum red sienna tan turquoise violet wheat white yellow.*
- *thickness=n*, де *n=1,2,3 ...* - товщина лінії (за замовчуванням *n=1*).
- *linestyle=n* - тип лінії: суцільна, пунктирна і т.д. (за умовчанням *n=1* - суцільна).

- *symbol=s* - тип символу, яким позначають точки: *BOX, CROSS, CIRCLE, POINT*.
- *font=[f, style, size]* - установка типу шрифту для виведення тексту: *f* задає назву шрифтів: *TIMES, COURIER, HELVETICA, SYMBOL*; *style* задає стиль шрифту: *BOLD, ITALIC, UNDERLINE*; *size* - розмір шрифту в *pt*.
- *labels=[tx, ty]* - написи по осях координат: *tx* - по осі *Ox* і *ty* - по осі *Oy*.

Приклад побудови двох графіків одночасно: *plot([x, x^2], x = -1 .. 3, thickness = [1, 3], color = [red, green])*;

Для швидкого побудови графіка можна використати функцію *smartplot(f)*.

Більшість графічних функцій представлені пакетами *plots* і *plottools*, які викликаються за допомогою команди *with*.

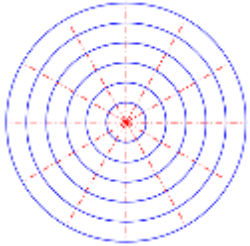
- Пакет *plots*

У пакет *plots* входять функції: *[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot, display, dualaxisplot, fieldplot, fieldplot3d, gradplot, gradplot3d, implicitplot, implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions, setoptions3d, shadebetween, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot]*.

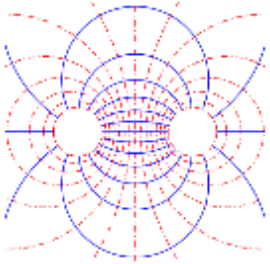
Криві 2-го порядку (коніки)

Існують системи координат на площині – декартова, полярна, еліптична та інші, які можна унаочнити за допомогою команди *coordplot(name)*:

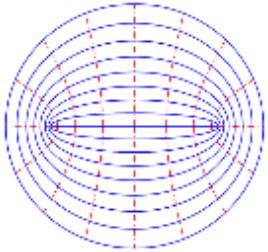
coordplot (polar)



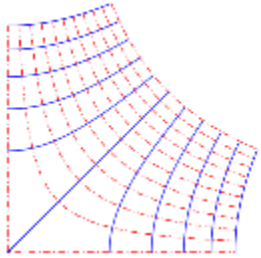
coordplot (bipolar)



coordplot (elliptic)



coordplot (hyperbolic)



Види рівнянь кривих:

- явне;
- неявне (канонічне);
- параметричне тощо.

Канонічне рівняння кривої 2-го порядку є квадратичним. Кількість точок перетину кривої 2-го порядку з довільною прямою рівняється 2.

Коло	Еліпс	Парабола	Гіпербола
• Неявне	• Неявне	• Явне	• Неявне

$x^2 + y^2 = a^2$ • Полярне $\rho = \varphi$	$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ • Полярне	$y = x^2$ • Полярне	$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$ • Полярне

Для побудови графіка в декартовій системі координат в

Для графічної побудови:

- неявних кривих використовується функція *implicitplot*.
- кривих в полярній системі координат – *polarplot*.
- параметричних кривих - команда *plot([f1(t), f2(t), t=tmin..tmax])*.

Окрім пакетів графіки *plots* та *plottools* сюди можна віднести пакети:

- *stats*[*statplots*] **– побудова графіки статистики;**
- *Student* –графічні оператори загальних розділів математики;
- *geometry* – побудова об’єктів геометрії Евкліда на площині.
- *geom3d* - побудова об’єктів геометрії Евкліда в просторі.

Анімація в площині

Пакет *plots* має функції для створення динамічних (анімованих) графіків. Перша з них служить для створення анімації графіків (їх повільну побудову), що представляють функцію однієї змінної *F(x)*: *animatecurve (F, r, ...);*

При виклику даної функції спочатку будується порожній шаблон графіка. Якщо активізувати шаблон мишею, то в рядку головного меню з'являється меню *Animation*, який містить команди управління анімацією - набуває вигляду панелі програвача кліпів. Аналогічне меню з'являється і в контекстному виклику:

- *Play* - запуск побудови графіка;
- *Next* - виконання наступного кроку анімації;
- *Backward* / *Forward* - перемикання напрямку анімації (назад / вперед);
- *Faster* - прискорення анімації;
- *Slower* - уповільнення анімації;

Continues / *Single cycle* - циклічність анімації.

Більші можливості анімації двовимірних графіків забезпечує функція *animate*, де: *x* задає межі зміни змінної *x*, *a t* - межі зміни додаткової змінної *t*.

animate (F, x, t)

Суть анімації при використанні функції *animate* полягає в побудові серії кадрів (як мультфільм), де кожен кадр пов'язаний зі значенням зміни в часі параметра *t*.

Для явного завдання кількості *N* кадрів анімації необхідно задати *frame=N*.