Почтовюк С.І.
НПУ імені М.П. Драгоманова

Початки програмування в середовищі MATLAB

При вивченні традиційних мов програмування, студенти затрачують чимало зусиль на знайомство з алфавітом та лексикою мови, формальних описів, виконання вправ. Але, при розв'язування конкретних задач, написані програми не завжди допомагають, або можуть виявитись складними у написанні.

При застосуванні математичних систем студентам одразу надається можливість виконувати більш змістовні приклади, спеціально не вивчаючи ази програмування. Також СКМ надає можливість використання великої кількості команд, функцій та вже готових алгоритмів для автоматизованого розв'язування широкого кола прикладних задач.

Питанням розробки, застосування та створення відповідної методичної підтримки засобів навчання на основі комп'ютерної техніки присвячені роботи таких науковців: М.І. Жалдака, Ю.В. Горошка [1], Ю.В. Триуса [5], С.А. Ракова [4], Г.О. Михаліна та інших.

У даній статті розглядаються деякі аспекти методики вивчення елементів програмування в середовищі MATLAB.

При вивченні основ програмування в середовищі MATLAB слід розглянути наступні теми: особливості програмування в середовищі, основні типи даних та прийоми роботи з ними, організацію розгалужених та циклічних обчислень. Розглянемо деякі приклади, які можна запропонувати при вивченні даних тем.

Поняття м-файлу

У систему MATLAB включена мова програмування високого рівня, за допомогою якої можна розв'язувати задачі векторної алгебри, лінійної алгебри і аналітичної геометрії за допомогою матричного запису, а також як автоматизувати, так і унаочнити процес дослідження впливу параметрів, що входять в рівняння, на положення і форму лінії і поверхні в координатному просторі.

Система MATLAB орієнтована на роботу в напівавтоматичному режимі, тобто необхідно в командному рядку ввести команду і передати її до ядра системи. Після перевірки синтаксису команда автоматично опрацьовується в ядрі і результат виводиться в командне і/або графічне вікно, після чого система переходить в режим очікування введення нової команди. Якщо необхідно переглянути результат виконання команди з іншими початковими даними, слід заново ввести в командний рядок всі необхідні команди, оскільки в системі MATLAB не передбачено можливості динамічного оновлення результату при зміні значення початкової змінної. Існує дві можливості автоматизації повторного введення серії команд. Перший спосіб полягає у використанні вікна Command History, в якому зберігаються введені раніше команди. Цей спосіб зручно використовувати для повторного виконання невеликої кількості команд. Проте в деяких випадках, залежно від поточних значень змінних, потрібно використовувати різні набори команд невідоме число разів. У таких випадках слід скористатися другим способом, заснованим на застосуванні *m*-файлів, де можуть міститися команди і управляючі структури мови MATLAB. Звертання до створеного *m*-файлу здійснюється за допомогою вказування (введення) його імені, а якщо необхідно – заданням вхідних і вихідних параметрів в командному рядку.

М-файли являють собою текстові файли, що збережені з розширенням m. Оскільки m-файл є текстовим файлом, то його можна створювати за допомогою будь-якого текстового редактора. До складу системи MATLAB входить редактор Editor/Debugger, за допомогою якого можна як створювати m-файли, так і виконувати їх налагодження, що приводить до скорочення часу створення робочого m-файлу. Відкрити редактор Editor/Debugger для створення нового m-файлу можна за допомогою команди M-file, розташованої в підменю New головного меню File, або за допомогою "натиснення" на кнопку New на панелі інструментів робочого столу системи MATLAB.

М-файли підрозділяються на два типи: сценарії (script) і функції (function).

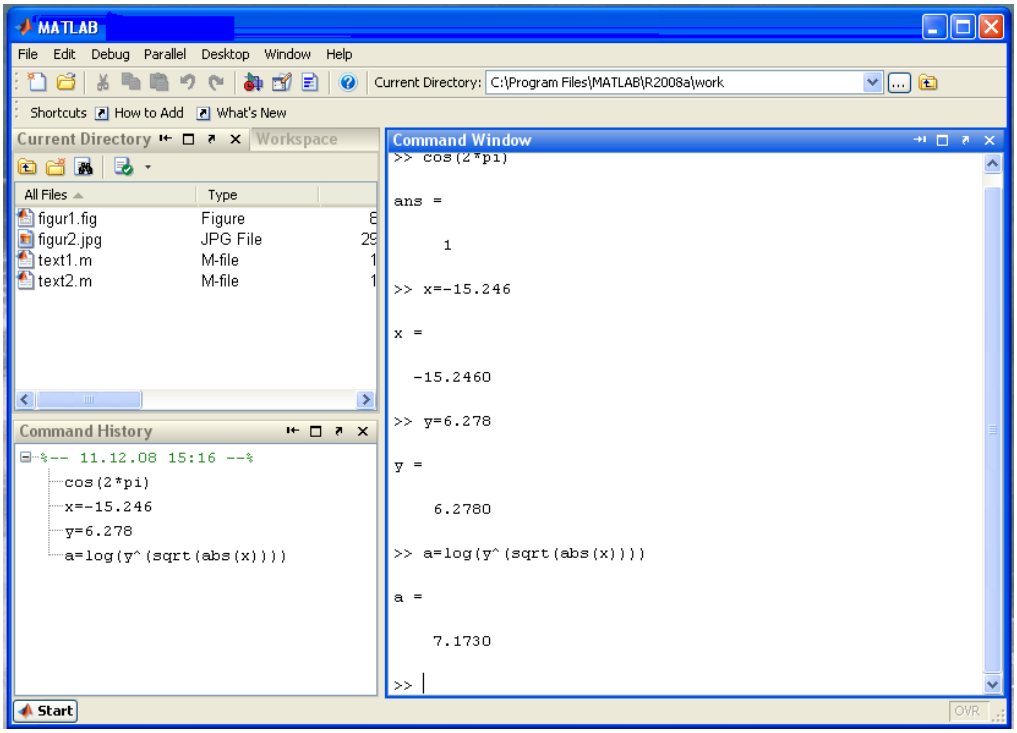


Рис. 1

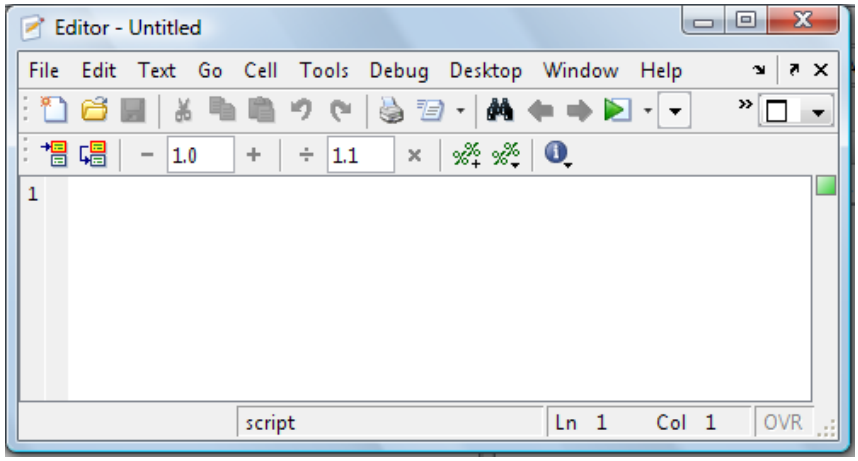


Рис. 2

Файли-сценарії

Сценарії, що є найпростішим типом m-файлів, містять послідовності команд, які задаються в командному рядку, і коментарі, що починаються із знаку %.

Особливостями сценаріїв є:

– виконання команд в режимі інтерпретації, тобто команди перетворюються у код програми і виконуються порядково;

– використовуючи тільки змінні, розташовані в робочому просторі MATLAB.

Отже, необхідно бути уважним при створенні змінних під час виконання сценарію, оскільки нові змінні, імена яких співпадають з іменами змінних, розташованих в робочому полі, замінять їх, що може призвести до втрати необхідних для подальшої роботи даних.

Головною особливістю описування програм мовою MATLAB є те, що типи змінних на початку програми не декларуються, досить надати змінній значення певного типу.

Приклад 1. Обчислити кути нахилу вектора $\bar{a}$ = *(-1;2;5)* до осей координат. Перевірити результат. Розв'язування оформити у вигляді сценарію з ім'ям `example_1`. Для цього необхідно виконати наступні дії:

В головному меню звернутися до команди File → New → M-file, в результаті відкриється вікно редактора m-файлів:

У цьому вікні необхідно ввести наступний набір команд:

```
A=[-1 2 5]
A=acos(A./sqrt(sum(A.*A)))*180/pi
%Перевірка результату
A=sum(cos(A./180*pi).^2)
```

Після завершення введення файл-сценарію необхідно зберегти його в поточному робочому каталозі MATLAB. Для цього у вікні редактора *m*-файлів потрібно звернутися до послуги (команди) File→Save as. В допоміжному вікні `Save file as` розкриється підкаталог `work` основного каталога MATLAB, який за замовчуванням визначений як поточний робочий каталог (`Current Directory`). У

полі Ім'я файлу слід ввести ім'я файл-програми `example_1` замість імені `Untitled`. У полі Тип файлу автоматично відображуватиметься необхідний тип файлу – M-files (* . m) який не слід змінювати.

Виконати створену файл-програму можна одним з наступних способів:

– у вікні редактора m-файлів обрати команду Debug → Run, або натискувати клавішу F5;

– ввести ім'я *m*-файла (без розширення), що містить файл-сценарій, в командний рядок і натиснути клавішу Enter.

Таким чином, при виклику файл-програми з командного рядка досить лише ввести її ім'я, і не потрібно задавати жодних вхідних даних і змінних – всі змінні створюються при виконанні програми або створені раніше і знаходяться в робочому просторі.

Результати виконання сценарію:

```
>> example_1
A =
    -1     2     5
A =
  100.5197   68.5833   24.0948
A =
     1
```

Файл-функції

Функції разом з сценаріями також містять команди, але є складнішим типом m-файлів в порівнянні з сценаріями. Відмінними особливостями функцій від сценаріїв є:

– можливість компіляції всієї функції у код програми з подальшим розміщенням його в пам'яті;

– наявність власного робочого простору, де зберігаються локальні змінні;

– наявність вхідних і вихідних параметрів.

Після виконання функції її робочий простір вилучається з пам'яті. Отже, значення всіх змінних, які були створені під час виконання функції і розташовувалися в її робочому просторі, вилучаються з пам'яті.

Приклад 2. Обчислити кути нахилу вектора $\bar{a} = (-1;2;5)$ до осей координат. Обчислення оформити у вигляді функції з ім'ям `angles`. Для цього необхідно звернутися до редактора *m*-файлів та ввести наступний набір команд:

```
function a=angles(A)
%Обчислення  кута нахилу вектора
A=acos(A./sqrt(sum(A.*A)))*180/pi
```

Тут перший рядок є заголовком функції. Він включає ім'я функції (в даному випадку `angles`), а також один вхідний (*A*) і один вихідний (*a*) параметри.

Наступний рядок з коментарем, який при обчисленні функції ігнорується. Це необов'язкова частина файл-функції. Проте якщо виникне потреба отримати довідку про функцію, коментар допоможе пригадати, яке призначення даної функції:

```
>> help angles
%Обчислення  кута нахилу вектора
```

Таким чином, коментар, введений після заголовка функції, інтерпретується як опис функції.

Наступний рядок – тіло функції – вираз, за яким обчислюється значення функції.

Після введення файл-функцію необхідно зберегти в поточному робочому каталозі, аналогічно до збереження файл-сценарію, але ім'я m-файлу, в якому зберігається файл-функція, обов'язково повинне співпадати з іменем функції.

Створену файл-функцію можна використовувати як в командному режимі (аналогічно до будь-якої вбудованої функції системи MATLAB), так і викликати з інших файл-програм або файл-функцій.

При виклику файл-функції потрібно вказати всі її вхідні і вихідні параметри.

Результати виконання функції `angles ()` :

```
>> A=[-1 2 5];
>> angles(A)
A =
  100.5197   68.5833   24.0948
>> angles ([-1 2 5])
A =
  100.5197   68.5833   24.0948
```

Файл-функції з кількома вхідними та вихідними аргументами

Приклад 3. Обчислити вираз $y(x)=x^3(a\sin^2(x)+b\cos^2(x))$. Обчислення оформити у вигляді функції з ім'ям `example3`. Для цього потрібно скласти файл-функцію з кількома вхідними аргументами:

```
function f =example3(x,a,b)
%Обчислення значення виразу
f=x.^3.*(a*sin(x).^2+b*cos(x).^2);
```

Результати виконання функції `example3 ()` :

```
>> example3(5,5,7)
ans =
  645.1161
```

Якщо функція має кілька вихідних параметрів (тобто повертається кілька значень), їх вказують в квадратних дужках через кому після слова `function`.

Приклад 4. Обчислити середнє значення елементів вектора та стандартне відхилення його елементів від середнього. Обчислення оформити у вигляді функції з іменем `example4`:

```
function [mean,stdev]=example4(x)
n=length(x);%Визначення довжини вектора
mean=sum(x)/n;%Обчислення середнього значення
stdev=sqrt(sum((x-mean).^2/n));%Обчислення стандартного відхилення
```

Результати виконання функції `example4` () для вектора *A*:

```
>> A=[2 4 6 8 9 7 5 3 1];
>> [mean,stdev]=example4(A)
m =
     5
s =
    2.5820
```

Інтерфейс редактора m-файлів

Викликати редактор *m*-файлів можна одним з наступних способів:

- за допомогою команди File→New→M-file;
- ввести в командний рядок вказівку edit.

У системі Windows запустити редактор *m*-файлів можна і без попереднього запуску системи MATLAB. Для цього потрібно звернутися до імені *m*-файлу в програмі Провідник Windows. При цьому редактор використовується як самостійний додаток, без можливості виконувати налагодження програми, запускати на виконання її окремі блоки, доступу до довідкової системи та до інструментів перевірки коду програми. Проте, всі можливості щодо створення і редагування *m*-файлів будуть доступні.

Якщо в редакторі відкрито кілька m-файлів, то він стає багатовіконним і включає кілька вкладинок, кожна з яких відповідає окремій програмі (рис. 3).

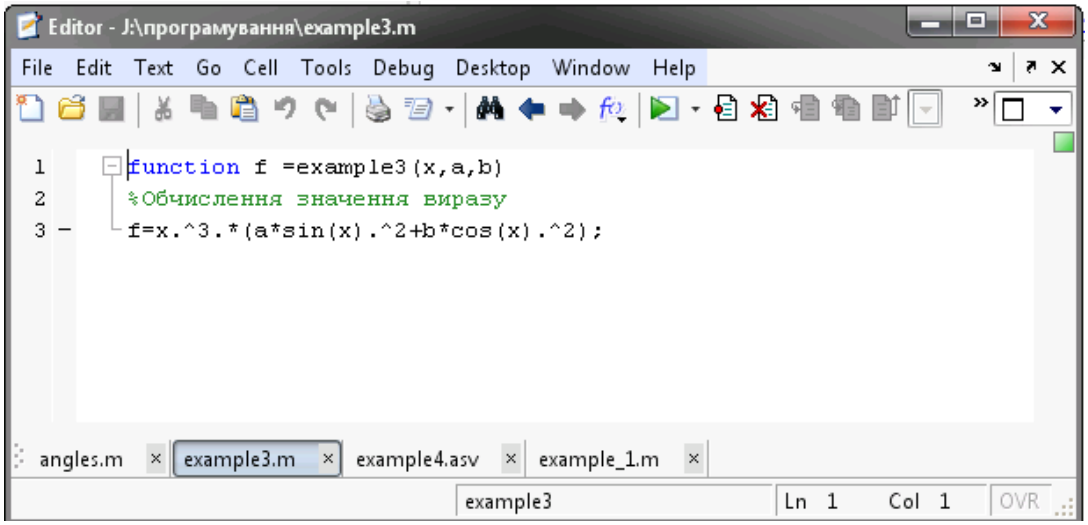


Рис. 3

Приклад 5. Створити в редакторі *m*-файлів файл-програму, результатом виконання якої є побудова в одному графічному вікні графіків трьох функцій: $y1 = \exp(x/2)$, $y2 = \cos 4x$, $y3 = \sin 4x$, що задані на проміжку [0, 4π]. Для цього необхідно звернутися до редактора m-файлів та ввести наступний набір команд:

```
x=[0:0.05:4*pi];
y1=exp(x/2);
subplot(3,1,1);
plot(x,y1);
y2=cos(4*x);
subplot(3,1,2);
plot(x,y2);
y3=sin(4*x);
subplot(3,1,3);
plot(x,y3);
```

Перейти до режиму фрагментації та поділити файл-програму на три фрагменти, кожний з яких буде включати команди для побудови графіків окремих функцій (y1, y2, y3) можна за допомогою звернення до команд Cell → Enable Cell Mode (рис. 4).

Типи даних

Більшість обчислювальних процесів в MATLAB виконуються над дійсними або комплексними числами, поданими у форматі double. Тип даних double є числовим типом, встановленим за замовчуванням.

Всі класи даних є матричними, підтримується робота з розрідженими матрицями, де перевагу складають нульові елементи. Для кожного класу даних встановлюються свої операції.

В MATLAB існує 15 фундаментальних типів даних (або класів об'єктів). Схема, де наведені основні типи даних і взаємозв'язки між ними, подана на рис. 5.

Арифметичні та логічні класи даних

Арифметичні класи даних (`NUMERIC`) і логічний клас (`logical`) є найбільш поширеними класами даних в системі MATLAB в порівнянні з іншими класами.

Арифметичні класи даних поділяються на цілі та дійсні.

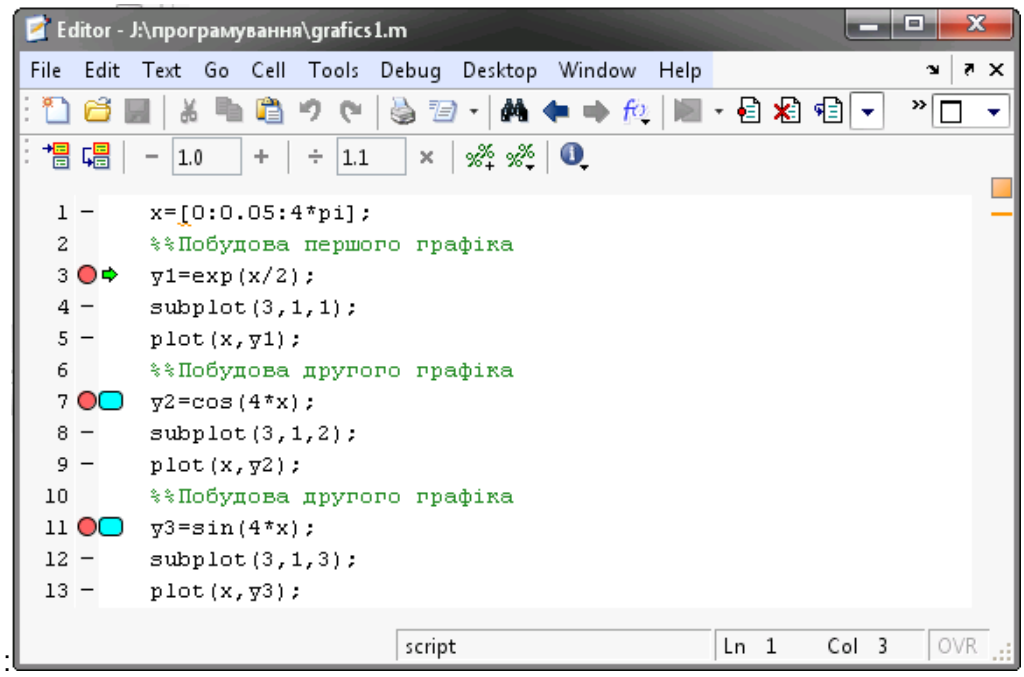


Рис. 4

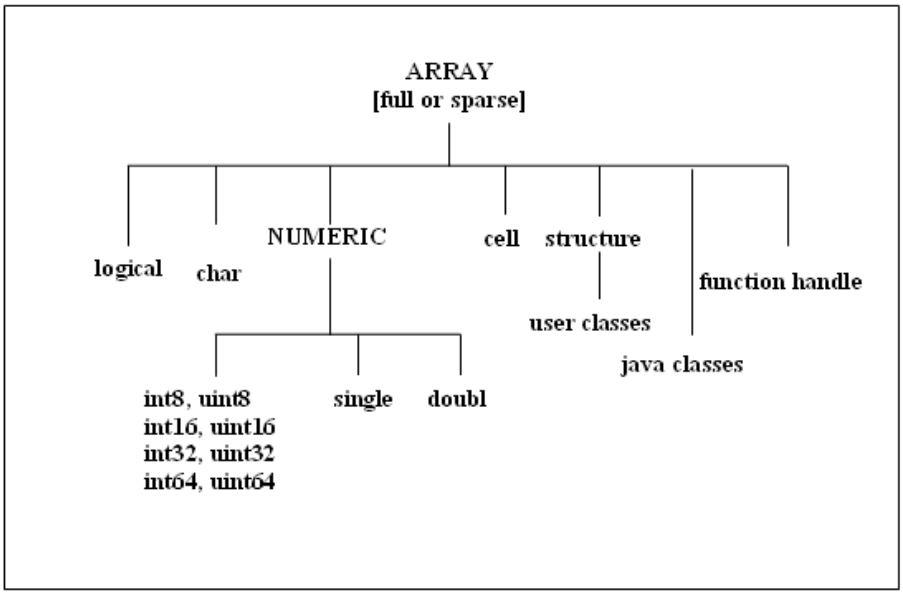


Рис. 5

Дані цілого класу можуть бути числами із знаком (класи `int8, int16, int32, int64`) або числами без знаку (`uint8, uint16, uint32, uint64`). Дані дійсного класу можуть бути числами одинарної (`single`) та подвійної (`double`) точності.

Дані логічного класу можуть набувати одного з двох значень: `true` або `false`. Значення логічного класу утворюються як результати операцій порівняння.

Приклад 6. Створити в редакторі m-файлів файл-функцію, за допомогою якої виконується побудова заштрихованої області у графічному вікні (рис. 6). Визначити, чи знаходиться точка $A_1(0,5;0,5)$ та $A_2(0,8;0,8)$ всередині цієї області.

```
function Aria(X,Y)
x=2-4*rand(1,1000000);% вісь x
y=2-4*rand(1,1000000);% вісь y
% Визначення вектора
L=((x.^2+y.^2)>=1)&(abs(x)<=1)&(abs(y)<=1);
plot(x(L),y(L),'.r')% Побудова області
grid on % Включення координатної сітки
```

```
    axis equal, axis([-1.1 1.1 -1.1 1.1])% Встановлення однакових
масштабів
    line([-2 2],[0 0],'color','black')% виведення осі Ox
    line([0 0],[-6 2],'color','black')% виведення осі Oy
    xlabel('x'), ylabel('y')
    z=(X.^2+Y.^2)>=1;
    z=and(z,and(abs(X)<=1,abs(Y)<=1))
```

В прикладі спочатку виконується формування двох масивів, які відповідають осям координат та містять 1000000 випадкових значень в діапазоні від -2 до 2, отриманих за допомогою функції `rand()`. Параметром кожного координатного вектора є вектор `L` , розмірність якого дорівнює розмірності масивів значень координат. Це приводить до того, що використовуючи функцію `plot ()` виконується відображення точки.

Результати виконання функції `Aria(X,Y)`:

```
>> Aria([0.5 0.8],[0.5 0.8])
z =
     0     1
```

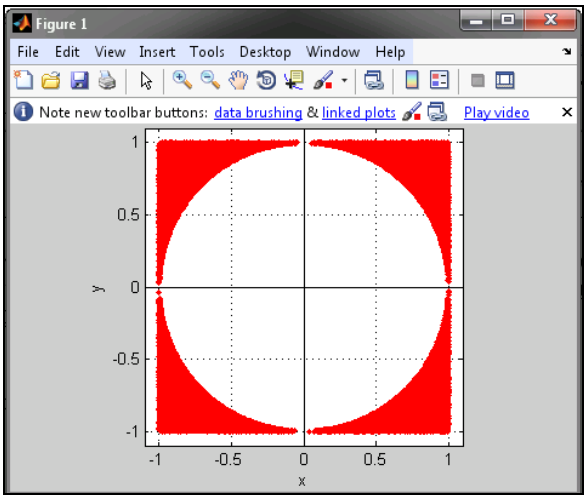


Рис. 6

Символьний клас даних

Разом з числовими і логічними даними в системі MATLAB можна працювати з даними, які є набором символів. Змінні, що набувають символьних значень, є змінними символьного класу (`char`). Для створення змінної символьного класу досить надати їй значення, що записано в апострофи.

Приклад 7. Створити символьну змінну student, в першому рядку якої розміщується прізвище студента, а в другому – його ім'я:

```
>> student=['Петренко ';'Олександр']
student =
Петренко
Олександр
>> first_name='Олександр'; last_name='Петренко ';
>> student =[last_name;first_name]
student =
Петренко
Олександр
>> name='Олександр';student =char('Петренко ',name)
student =
Петренко
Олександр
```

Приклад був розв'язаний трьома способами. Для вертикального об'єднання рядків використовувався набір символів [;]. В результаті змінна `student` є двовимірним масивом символів, що складається з двох рядків і дев'яти стовпців. При заданні значення змінної `student` в першому рядку для вирівнювання числа символів в рядках був добавлений один пропуск. В системі MATLAB передбачено використання при об'єднанні рядків в масиви символів імен інших змінних, які самі містять масиви символів.

В системі MATLAB можна об'єднувати рядки, що містять різну кількість символів. Ця операція реалізується за допомогою функції `char ()`, за якою автоматично вирівнюється довжини рядків додаванням, де необхідно, пропусків в кінці рядка.

Організація структури

В мові MATLAB підтримується робот із змінними, за допомогою яких репрезентують дані різних класів. Подання різних даних за допомогою однієї змінної в системі MATLAB реалізується за допомогою класу даних структура (`structure`).

У системі MATLAB можна використовувати два підходи до організації структури: матричний (рис. 7) і по елементний (рис. 8). Підходи розрізняються в розміщенні елементів структури. У першому

випадку дані про імена, прізвища, дати народжень і оцінки всіх студентів розміщуються в одному відповідному полі в матричному вигляді. При цьому перші елементи в кожному з полів – дані про першого студента, другі елементи – про другого студента і так далі. Цей підхід переважно застосовується у випадку, коли необхідно виконувати операції над всіма елементами структури. У другому випадку дані про студентів знаходяться окремо у відповідних полях стосовно кожного студента. Даний підхід є переважним, коли необхідно опрацьовувати дані окремих елементів структури.

Створити змінну класу структури можна двома способами: використовуючи роздільник [**.**]; за допомогою функції-конструктора `stract()`.

Приклад 8. Створити змінну `student`, яка складається з трьох полів: `first_name`, де знаходиться ім'я студента**,** `last_name` з його прізвищем і `year`, що містить дату його народження.

```
>> student =struct('last_name','Петренко',...
'first_name', 'Олександр','year',1993)
student =
     last_name: 'Петренко'
    first_name: 'Олександр'
          year: 1993
```

Масив комірок

Масивом комірок називається впорядкований набір даних різнорідних класів. Масиви комірок можуть мати довільний розмір. На рис. 9 показано двовимірний масив комірок *2*x*3*, який містить масив символьних даних *3*x*1*, масив арифметичних даних *2*x*3*, структура, масив символьних даних *5*x*7*, масив логічних даних *5*x*1* та масив комірок *2*x*3*. Масив комірок є відмінною рисою системи MATLAB. Їх використання дозволяє реалізувати всі переваги роботи з матричними даними. Масиви комірок слід використовувати при утворенні масиву рядків різної довжини, оскільки немає необхідності стежити за рівністю їх довжин. Масиви комірок широко використовуються при створенні функцій, оскільки це дозволяє працювати з довільним числом вхідних і вихідних параметрів функції.

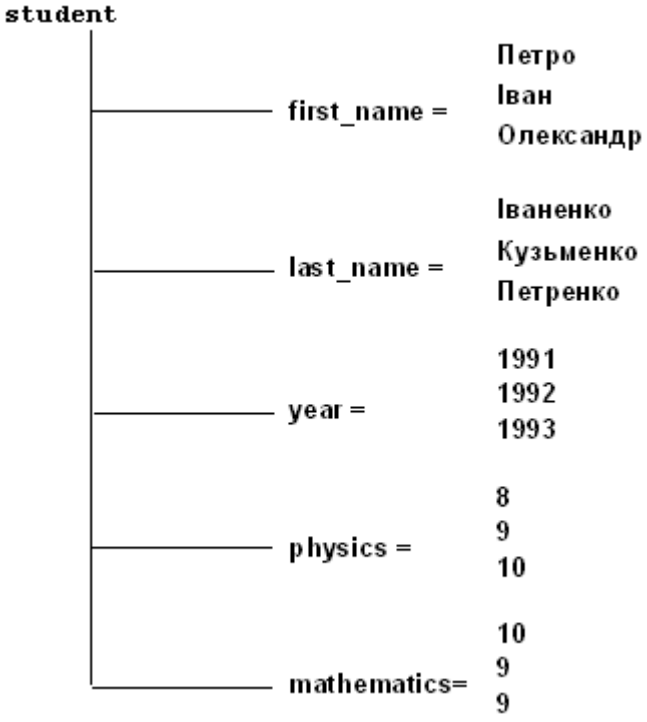


Рис. 7

Масив комірок можна створити двома способами: послідовно надавати значення коміркам; використовуючи функцію-конструктор `cell ()`.

Приклад 9. Використовуючи перший спосіб, створити масив комірок *A*, зображений на рис. 7. Шоста комірка містить порожній миссив.

```
>> A{1,1}=char('Франко','Іван','Якович');
>> A{1,2}=[27 8 1856;28 5 1916];
>> A{1,3}=struct('Ukranian_Writer',struct('Surname', ...
'Франко','Name','Іван','Patronymic','Франко',...
'Date',[27 8 1856;28 5 1916]));
>> A(2,1)={char('Борислав сміється', ...
'Захар Беркут','Катерина','Мойсей','Каменярі')};
>> A(2,2)={[true;true;false;true;true]};
>> A(2,3)={[]}
A =
    [3x6  char]    [2x3 double ]    [1x1 struct]
    [5x17 char]    [5x1 logical]              []
```

Фігурні дужки використовуються при встановлюванні відповідності комірок і даних. У трьох перших командах фігурні дужки знаходяться зліва від знака надання значення, а в трьох останніх − справа. У першому випадку використовується індексація змісту (content indexing), а в другому випадку − індексація комірок (cell indexing). Перше число в парі індексів позначає номер рядка, а друге − номер стовпця. Значення логічного масиву пов'язані з назвами творів, які знаходяться в другій комірці. Оскільки Іван Франко не є автором твору «Катерина», то відповідний елемент логічного масиву має значення `false`.

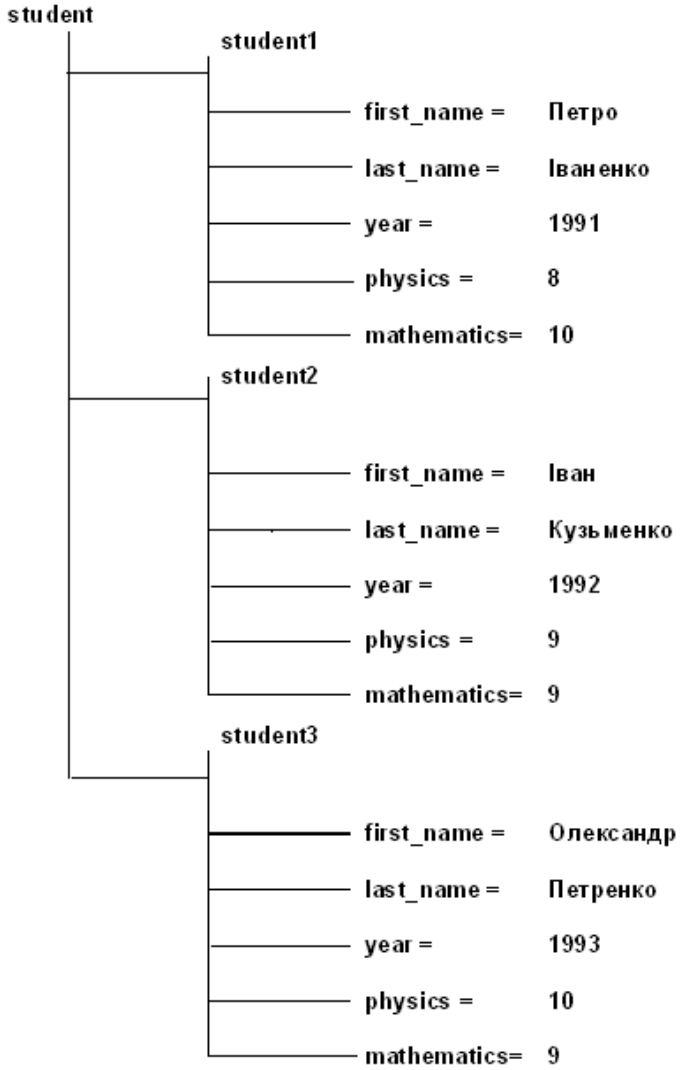


Рис. 8

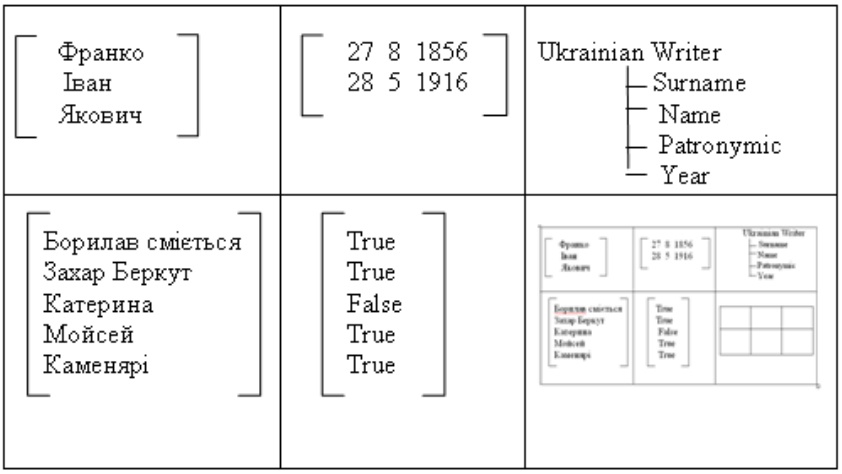


Рис. 9

Функція конструктор `cell ()` використовується для попереднього виділення пам'яті під порожній масив комірок вказаного розміру.

Доступ до даних масиву комірок здійснюється за допомогою індексації змісту, або за допомогою індексації комірок. Результатом виконання індексації змісту будуть дані, які містяться у відповідних комірках. Отже, клас результату відповідає класу даних, що містяться у відповідних комірках. При виконанні індексації комірок результатом буде масив комірок, розмірність яких відповідає розмірності набору індексів.

Звернутися до комірки можна і за допомогою одного індекса. Комірки нумеруються вздовж стовпців, які перебираються зліва направо.

Спеціальні конструкції мови MATLAB

У будь-якій мові програмування є спеціальні конструкції, за допомогою яких можна задавати порядок виконання команд в програмах. В мові MATLAB є три структури вибору: if end – структура з єдиним вибором, if else end – структура з подвійним вибором і switch – структура з множинним вибором; два типи структур повторення: while end – цикл з умовою і for end – цикл з параметром, а також оператори break, continue, return та структура try … catch. Всі ці оператори можна використовувати як у файл-програмах, так і у файл-фукціях.

Приклад 10. Задати масив цілих додатних чисел і визначити, скільки в ньому простих чисел. Якщо прості числа знайдені, то вивести їх кількість і самі ці числа. Розв'язок оформити у вигляді сценарію.

```
anser=input('Ввести масив додатних чисел:');
if any(isprime(anser(:))) % пошук простого числа в масиві
    S=sum(isprime(anser(:))); % обчислення кількості простих чисел
    string=['Ви ввели ' num2str(S) ' простих чисел'];
    disp(string)
      anser(isprime(anser(:))) % виведення рядка простих чисел
end
```

Результати виконання сценарію:

```
>> example_10
Ввести масив додатних чисел:[1 2 3 6 7 8 10 13 16 17]
Ви ввели 5 простих чисел
ans =
     2     3     7    13    17
```

Отже, в перших двадцяти натуральних числах міститься вісім простих чисел.

При обчисленні кількості простих чисел у введеному масиві виконуються наступні дії:

– початковий масив перетворюється в одновимірний масив (:);

– на місці простих чисел в перетвореному масиві записуються одиниці, а в решті місць – нулі (isprime ());

– підсумовування одиниць в масиві з логічними значеннями (sum ()).

При формуванні останнього текстового повідомлення використовувалася функція num2str (), при зверненні до якої перетворюється число в його символьне подання. Сформоване повідомлення надається як значення змінній `string`, яка стає символьного класу.

Функція disp () використовується для виведення текстового повідомлення в командне вікно. У даному прикладі функції disp () передається змінна `string`, значення якої виводиться в командний рядок.

За останньою інструкцією в тілі структури if end змінній ans ставиться у відповідність масив простих чисел, що знаходяться у введеному масиві, який і виводиться в командне вікно. Для компактного виведення використовувалася операція транспонування, за якою вектор-стовпець перетворюється у вектор-рядок.

Приклад 11. Задати в командному вікні два вектори і арифметичну операцію. За допомогою структури switch виконати відповідну арифметичну операцію і вивести результат в командне вікно. Якщо арифметична операція задана з помилкою, вивести повідомлення про помилку. Файл оформити у вигляді сценарію з ім'ям example_24.

```
a1=input('Ввести значення елементів першого вектора: ');
operation=input('Ввести знак арифметичної операції:','s');
a2=input('Ввести значення елементів другого вектора: ');
switch operation
    case '+'
        a3=a1+a2;
        rez=['Результат: ' num2str(a3)];
        disp(rez)
    case '-'
        a3=a1-a2;
        rez=['Результат: ' num2str(a3)];
        disp(rez)
    case '*'
        a3=a1.*a2;
        rez=['Результат: ' num2str(a3)];
        disp(rez)
    case '/'
        a3=a1./a2;
        rez=['Результат: ' num2str(a3)];
        disp(rez)
    otherwise
        disp('Введена невизначена арифметична операція')
end
```

Результати виконання сценарію:

```
>> example_11
Ввести значення елементів першого вектора: [2 5 12 8]
Ввести знак арифметичної операції: -
Ввести значення елементів другого вектора: [2 4 6 10]
Результат: 0  1  6 -2
```

При розв'язанні даного прикладу для введення числових значень та символу арифметичної операції за допомогою клавіатури використовувалася функції input (). За замовчуванням при зверненні цієї функції повертається значення арифметичного класу. Введення другого вхідного параметра у вигляді 's' задає символьний клас значення. У switch_виразі використовувалася змінна символьного класу operation, значення якої порівнювалося з символами арифметичних операцій, розташованими у case_виразів. Якщо значення operation дорівнює одному із заданих символів, виконується відповідна цьому символу група інструкцій. Інакше виводиться повідомлення про неправильно задану арифметичну операцію.

При реалізації алгоритмів, описаних мовою MATLAB, слід застосовувати векторний підхід, що дозволяє написати програму в компактному вигляді і підвищити швидкість її виконання. В більшості випадків при використанні векторного підходу можна виконувати математичні операції з масивами даних без структур повторення, які широко застосовуються в таких мовах програмування, як BASIC, Pascal, C++, FORTRAN та ін. Проте, при виконанні обчислень з їх подальшою візуалізацією іноді виникають ситуації, коли не можна обійтися без використання структур повторення.

У мові MATLAB є дві структури повторення, однією з яких є структура for end. Структуру повторення for end необхідно використовувати в тому випадку, якщо наперед відома кількість повторень або її можна обчислити.

Приклад 12. Розв'язати методом Гаусса наступні системи лінійних рівнянь:

$$\begin{cases} 3\cdot x + 2\cdot x - x = 4 \\ 2\cdot x - x + 3\cdot x = 9 \\ x - 2\cdot x + 2\cdot x = 3 \end{cases}, \quad \text{та} \quad \begin{cases} 3\cdot x + 2\cdot x - x = 9 \\ 2\cdot x - x + 3\cdot x = 3 \\ x - 2\cdot x + 2\cdot x = 4 \end{cases}$$

Змінну A використовувати для матриці коефіцієнтів при невідомих, змінну B – для матриці вільних членів, змінну AB – для розширеної матриці і X – для матриці невідомих. Метод Гаусса реалізувати у вигляді сценарію.

```
A=input('Ввести матрицю A: ');
B=input('Ввести матрицю B: ');
AB=[A B]; % об'єднання масивів
row_count=size(AB,1); % кількість рядків=кількості невідомих
col_count_A=size(A,2); % кількість стовпців в матриці A
col_count_B=size(B,2); % кількість стовпців в матриці B
row_ones=ones(1,col_count_B); % рядок одиниць
x=zeros(col_count_A, col_count_B); % резервування пам'яті для x
% прямий хід
for col=1:col_count_A-1 %
    [absmax,index]=max(abs(AB(col:end,col)));
    if index>1 % перестановка рядків
        AB([index+col-1 col],:)=AB([col index+col-1],:);
    end
    AB(col+1:end,col:end)=AB(col+1:end,col:end)-
AB(col+1:end,col)./AB(col,col))*AB(col,col:end);
end
% зворотний хід
for row=row_count:-1:1
    if row ==row_count
        x(row,:)=AB(row,col_count_A+1:end)./AB(row,row); % обчислення
останнього невідомого
    elseif row==row_count-1
        x(row,:)=(AB(row,col_count_A+1:end)-
AB(row,row+1).*x(row+1,:))./AB(row,row); % обчислення передостаннього
невідомого
    else
        x(row,:)=(AB(row,col_count_A+1:end)-
dot(AB(row,row+1:col_count_A)'*row_ones,x(row+1:end,:)))
./AB(row,row); % обчислення останніх невідомих
    end
end
x
```

Результати виконання сценарію:

```
>> example_12
Введіть матрицю A: [3 2 -1;2 -1 3;1 -2 2]
Введіть матрицю B: [[4;9;3] [9;3;4]];
x =
```

```
    1.0000      3.8462
   .2.0000     -2.4615
    3.0000     -2.3846
```

Перевіримо результат обчислення:

```
>> A*x
ans =
    4.0000      9.0000
    9.0000      3.0000
    3.          4.0000
```

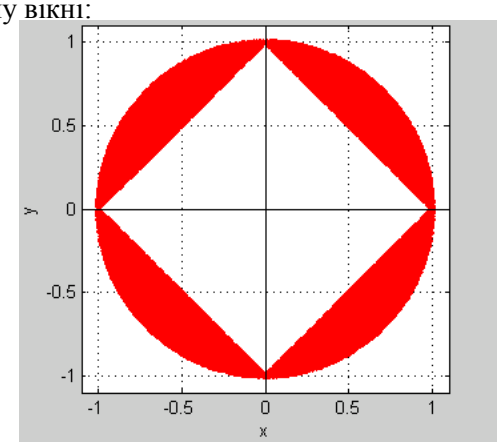
Структура повторення використовується в даному випадку разом з векторними операціями, оскільки під час виконання прямого і зворотного ходу потрібні «двовимірні обчислення». При використанні наведених в прикладі 12 інструкцій необхідно, щоб визначник матриці коефіцієнтів при невідомих не дорівнював нулю. Інакше результат буде неправильний.

Крім структури for end в мові MATLAB є ще одна структура повторення – while end. Її відмінною особливістю є те, що вказівки, розташовані в тілі структури повторення, починають виконуватися і повторно виконуються тільки в тому випадку, якщо твердження в умові є істинним.

Функції break () і continue () використовуються для зміни порядку виконання вказівок в тілі структур повторення for end і while end.

Після опрацювання наведених прикладів, студентам доцільно запропонувати наступні завдання для самостійної роботи:

1. Створити в редакторі m-файлів файл-функцію, за допомогою якої виконується побудова кола з центром в точці $A(x_0, y_0)$ та радіусом r ($x = x_0 + r * \cos(t)$, $y = y_0 + r * \sin(t)$).
2. Створити в редакторі *m*-файлів файл-функцію, за допомогою якої виконується обчислення кутів нахилу заданих векторів до координатних осей (вхідними аргументами є координати векторів *x*, *y*, *z*).
3. Створити в редакторі *m*-файлів файл-функцію, за допомогою якої виконується обчислення швидкості *V* та прискорення *a* у момент *t*, якщо закон руху заданий рівнянням: $S(t) = 2t^3 - 3t + 4$.
4. Створити в редакторі m-файлів файл-функцію, за допомогою якої виконується обчислення кількості електрики *dq*, що протікає за час *dt* з силою струму *I* за формулою: $dq = Idt$.
5. Створити в редакторі m-файлів файл-функцію, за допомогою якої виконується обчислення роботи, що здійснена силою *F*, напрямною під кутом α до напряму переміщення довжиною *S* за формулою: $A = F \cdot S \cdot \cos\alpha$.
6. Створити в редакторі *m*-файлів файл-функцію, за допомогою якої виконується побудова заштрихованої області у графічному вікні:



7. Створити дві змінні символьного класу `poelement_oper` та `matr_oper`, в яких розміщуються команди, за допомогою яких можна виконати поелементні та матричні операції в системі MATLAB. Вивести ці функції в командне вікно.
8. Використовуючи матричну організацію масиву структури, створити змінну `matematuk`, яка складається з трьох полів `surname`, `year_b` та `year_d` з прізвищем та роками життя видатних українських математиків: Остроградський М.В. (1801-1861), Левицький В.Й. (1872-1956), Шмідт О.Ю. (1891-1956), Кравчук М.П. (1892-1942), Глушков В.М. (1923-1982). Обчислити кількість років життя вчених.
9. Створити масив комірок, в якому: в першій комірці розташувати прізвище, ім'я та по-батькові письменника, в другій – дати життя, в третій – його твори, в четвертій – масив, що складається з логічних значень. В цьому масиві значення `true` вказує на те, що твір належить даному автору. (Михайло Михайлович Коцюбинський (17.09.1864 – 25.04.1913), „Тіні забутих предків", „Микола Джеря", „Сон", „Fata morgana", „Intermezzo"). Сформувати речення: Коцюбинський Михайло Михайлович є автором наступних творів: …
10. Використовуючи оператор циклу, створити в редакторі m-файлів файл-програму, за допомогою якої розв'язати наступну задачу: спортсмен за перший день пробіг 10 км. Кожного наступного дня він збільшував денну норму на 10% від норми попереднього дня. Який шлях пробіжить спортсмен за 7 днів?

11. Використовуючи структуру switch та class (), визначити клас змінної a та вивести відповідне повідомлення (наприклад, «Змінна логічного класу»). Розв'язування оформити у вигляді файл-програми, в якій використати функцію, за допомогою якої визначається клас вхідного аргументу та повертається результат у вигляді рядка.

12. Використовуючи оператор циклу, створити в редакторі m-файлів файл-програму, за допомогою якої розв'язати наступну задачу: Івана Олександровича Хлестакова запросили управляти департаментом. У перший день до нього прислали 1000 кур'єрів, а кожного наступного дня – вдвічі більше, ніж попереднього. Іван Олександрович погодився тоді, коли до нього прибуло відразу не менше 30 000 кур'єрів. На який день Хлестаков погодився управляти департаментом? Врахуйте, що він не вміє ні ділити, ні множити.

Висновки. Наведені приклади показують можливість та доцільність використання СКМ Matlab при розв'язування математичних та прикладних задач різної складності, при цьому вивільняється час для обмірковування алгоритмів розв'язування задач, постановки задач і побудови відповідних математичних моделей, досліджуються складніші моделі. Також знайомство з самою системи на заняттях з інформатики сприяє поступовому підвищенню рівня компетентності студентів, що дає можливість самостійно обрати ту чи іншу СКМ для розв'язування професійно-орієнтованих задач.

Література

1. Жалдак М.І. Математика з комп'ютером /Жалдак М.І., Горошко Ю.В., Вінніченко Є.Ф. – К.: РННЦ «Дініт», 2004. – 168 с.
2. Кривилев. А. В. Основы компьютерной математики с использованием системы MATLAB / Кривилев А.В. – М.: Лекс-Книга, 2005. – 496 с.
3. Почтовюк С.І. Удосконалення підготовки студентів технічного коледжу при вивченні дисциплін математичного циклу з застосуванням інформаційних технологій / Почтовюк С.І. // Збірник наукових праць № 3 (Педагогічні науки) Бердянського державного педагогічного університету. – 2009. – С. 146-151.
4. Раков С.А. Математична освіта: компетентнісний підхід з використанням ІКТ[Монографія] / Раков С.А. – Х.: Факт, 2005. – 360 с.
5. Триус Ю.В Комп'ютерно-орієнотоваі методичні системи навчання математики [Монографія] / Триус Ю.В. – Черкаси: Брама-Україна, 2005. – 400 с.