# Типы данных. Работа с данными

В MATLAB поддерживается несколько типов данных, которые являются классами.

Все классы MATLAB являются наследниками массива (*array*).

Наиболее используемыми типами данных являются типы *double, cell* и пользовательские классы.

Рассмотрим иерархию типов MATLAB

**Array**

**Сhar**

**Numeric**

**Сell**

**Structure**

**Java class**

**Logical**

**Function**\_**handle**

**int8 uint8**

**int16 uint16**

**int32 uint32**

**int64 uint64**

**single**

**double**

**User class**

**sparse**

* символьные данные
* числовые данные
* ячейки
* структуры
* java class
* логические массивы
* указатели на функции

Рассмотрим функции работы с классами

**class –** функция, позволяющая определить к какому классу (типу) принадлежит объект. Возвращает имя класса.

**isa ( var, class)** – возвращает 1 или 0, в зависимости от того, принадлежит ли объект к указанному в качестве второго параметра классу

**Пример:**

**a=[1 2 3]**

**b='4,5,6'**

**class(a)**

**ans = double**

**class(b)**

**ans = char**

**isa(a,'double')**

**ans = 1**

**isa(b,'char')**

**ans = 1**

**isa(a,'char')**

**ans = 0**

**Основные функции работы с символами**

|  |  |
| --- | --- |
| **blanks** | Сформировать строку пробелов |
| **cellstr** | Преобразовать массив символов в массив ячеек для cтрок |
| **char** | Сформировать массив символов |
| **deblank** | Удалить пробелы в конце строки |
| **double** | Преобразовать символы строки в числовые коды |

**Проверка строк**

|  |  |
| --- | --- |
| **ischar** | Истинно, если это массив символов (строка) |
| **iscellstr** | Истинно, если это массив ячеек для строк |
| **isletter** | Истинно, если это символ алфавита |
| **isspace** | Истинно, если это пробел |

**Операции над строками**

|  |  |
| --- | --- |
| **strcat** | Горизонтальное объединение строк |
| **strvcat** | Вертикальное объединение строк |
| **strcmp** | Сравнить строки |
| **strncmp** | Сравнить **n** символов строк |
| **findstr** | Найти заданную строку в составе другой строки |
| **strjust** | Выравнивание массива символов |
| **strmatch** | Найти все совпадения |
| **strrep** | Заменить одну строку другой |
| **strtok** | Найти часть строки, ограниченную разделителями |
| **upper** | Перевести все символы строки в верхний регистр |
| **lower** | Перевести все символы строки в нижний регистр |

**Преобразования строк**

|  |  |
| --- | --- |
| **num2str** | Преобразование числа в строку |
| **int2str** | Преобразование целого в строку |
| **mat2str** | Преобразование матрицы в строку |
| **str2mat** | Объединение строк в матрицу |
| **str2num** | Преобразование строки в арифметическое выражение и его вычисление |
| **sprintf** | Записать форматированные данные в виде строки |
| **sscanf** | Прочитать строку с учетом формата |

**Преобразования систем счисления**

|  |  |
| --- | --- |
| **hex2num** | Преобразовать шестнадцатеричное число в число удвоенной точности |
| **hex2dec** | Преобразовать шестнадцатеричное число в десятичное число |
| **dec2hex** | Преобразовать десятичное число в шестнадца­теричное число |
| **bin2dec** | Преобразовать двоичную строку в десятичное число |
| **dec2bin** | Преобразовать десятичное число в двоичную строку |
| **base2dec** | Преобразовать B-строку в десятичное число |
| **dec2base** | Преобразовать десятичное число в B-строку |

Еще один часто используемый тип данных – «массив ячеек» **(*cell array*)**. Элементами такого массива могут быть: числа, строки, массивы, другие массивы ячеек и т.д.

Задается он при помощи фигурных скобок **{ }**:

**Пример**:

**a={'Символьная строка', 27, [5 7]; {1},{[ 2 5 3]},{'строка в ячейке'}}**

**a =**

**'Символьная строка' [ 27] [1x2 double]**

**{1x1 cell} {1x1 cell} {1x1 cell }**

% Чтение и запись отдельных элементов производится с использованием фигурных скобок:

**>> a{1,3}**

**ans =**

**5 7**

**>> a{1,3}=[10,12,27]**

**>> a{1,3}**

**ans =**

**10 12 27**

**a =**

**'Символьная строка' [ 27] [1x3 double]**

**{1x1 cell} {1x1 cell} {1x1 cell }**

**% Результаты обращения к элементам массива ячеек: () и {}**

**>> a(1,3)**

**ans =**

**[1x3 double]**

**>> a{1,3}**

**ans =**

**10 12 27**

Команды **size** и **length** в случае с массивами ячеек ведут себя также, как и с массивами чисел:

**size(a)**

**ans = 2 3**

**length(a)**

**ans = 3**

**a={1,2,3}**

**a = [1] [2] [3]**

**size(a)**

**ans = 1 3**

**length(a)**

**ans = 3**

**b = {4;5;6}**

**[4]**

**[5]**

[6]

**size(b)**

**ans = 3 1**

**length(b)**

**ans = 3**

**cellfun** – применяет функцию, указанную в первом аргументе, для каждого элемента массива ячеек, указанного во втором аргументе.

**Пример**:

% Вычисление значений синуса, косинуса и тангенса для аргумента pi/4

**fh\_array = cellfun(@str2func, {'sin' 'cos' 'tan'}, 'UniformOutput', false);**

**fh\_array{1}(pi/4)**

**ans = 0.7071**

**>> fh\_array{2}(pi/4)**

**ans = 0.7071**

**>> fh\_array{3}(pi/4)**

**ans = 1.0000**

**Взаимное преобразование массивов double и cell**

Используются команды:

**num2cell –** преобразование числового массива в массив ячеек

**cell2mat –** преобразование массива ячеек в числовой массив (обратная команде **num2cell**)

**A=[1 2 3; 4 5 6]**

**A =**

**1 2 3**

**4 5 6**

Вызов команды **num2cell** с одним аргументом

**num2cell(A)**

**ans =**

**[1] [2] [3]**

**[4] [5] [6]**

Результат – массив ячеек.

Вызов команды **num2cell** с двумя аргументами

При вызове с двумя аргументами первый содержит исходный массив, а второй – номер измерения, по которому матрицу предстоит «разрезать» на полосы:

num2cell(A,1) «режет» матрицу по вертикали (по столбцам)

**n1=num2cell(A,1)**

**n1 =**

**[2x1 double] [2x1 double] [2x1 double]**

**n1{2}**

**ans =**

**2**

**5**

% num2cell(a,2) «режет» матрицу по горизонтали (по строкам)

**n2=num2cell(A,2)**

**n2 =**

**[1x3 double]**

**[1x3 double]**

**n2{1}**

**ans =**

**1 2 3**

Действие команды **cell2mat** обратно действию **num2cell**. Вызов осуществляется только с одним входным аргументом:

**cell2mat(num2cell(A))**

**ans =**

**1 2 3**

**4 5 6**

Результат – числовая матрица A

Команда **mat2cell** –  разбиение матрицы на массив ячеек. На входе функция принимает три аргумента – исходный массив и два массива, содержащих размеры вертикальных и горизонтальных блоков.

**Пример:**

Рассмотрим матрицу размерность 5x7.

Разобьем матрицу на блоки линиями: по вертикали (горизонтальные линии: 2 строки, 1 строка и 2 строки), по горизонтали (вертикальные линии: 2 столбца, 3 столбца, 2 столбца).

В соответствии с этим получим два массива: [2 1 2] и [2 3 2].

**X=[1 2 3 4 5 6 7;6 9 10 11 12 13 14;15 16 17 18 19 20 21; 22 23 24 25 26 27 28;...**

**29 30 31 32 33 34 35]**

**X =**

**1 2 3 4 5 6 7**

**6 9 10 11 12 13 14**

**15 16 17 18 19 20 21**

**22 23 24 25 26 27 28**

**29 30 31 32 33 34 35**

**Y=mat2cell(X,[2 1 2],[2 3 2])**

**Y =**

**[2x2 double] [2x3 double] [2x2 double]**

**[1x2 double] [1x3 double] [1x2 double]**

**[2x2 double] [2x3 double] [2x2 double]**

**Y{3,2}**

**ans =**

**24 25 26**

**31 32 33**

**Y{3}**

**ans =**

**22 23**

**29 30**

**Y{3}(2)**

**Использование структур и пользовательских классов**

Все пользовательские классы являются наследниками класса структуры **(struct)**. Структура в пакете MATLAB создается функцией **struct()**, которая в качестве входных параметров принимает набор пар – имени и значения поля: **struct (имя1, значение1, имя2, значение2, …..)**.

Обращение к полям структуры осуществляется при помощи оператора «точка».

Значением поля структуры может быть объект любого класса, в том числе массив чисел или массив ячеек.

**Пример**:

% Создание структуры с помощью функции struct()

**s=struct('name', 'Белошапко Алексей Геннадьевич','group','112')**

% создание массива структур: простым добавлением. Введем 2-й элемент массива структур

**s(2)=struct('name','Бубнов Владислав Владимирович','group','112')**

% доступ к полю элемента массива структур

**s(2).name**

% добавление поля к структуре

**s(2).rating=[5 4 5 4 5 4]**

**s(2)**

**Внимание!**

При объявлении структуры, у которой одно из полей является массивом ячеек, используются одни «лишние» скобки. Если их не поставить, то будет создан массив структур. При этом функция **struct** работает следующим образом.

* Если лишь одно поле структуры в качестве значения имеет массив ячеек, то создается массив структур, в котором каждый из элементов принимает значения из массива ячеек, а остальные поля фиксированы.
* Если два или более полей заданы массивами ячеек, то эти массивы должны быть одинаковой длины и команда **struct** создает массив структур, в котором элементы поочередно принимают соответствующие значения.
* Если же количество ячеек разное, то выдается сообщение об ошибке

% структура с одним полем ячеек (вывод всей информации)

**d=struct('name','Белошапко Алексей Геннадьевич','subject',{{'Математический анализ' ' Алгебра' ' Геометрия'}},'ratings',[5 4 5])**

**d =**

**name: 'Белошапко Алексей Геннадьевич'**

**subject: {'Математический анализ' ' Алгебра' ' Геометрия'}**

**ratings: [5 4 5]**

% структура с массивом ячеек (вывод «по элементам массива ячеек»)

% одно поле «меняется», другие «закреплены»

**d=struct('name','Белошапко Алексей Геннадьевич','subject',{'Математический анализ' ' Алгебра' ' Геометрия'},'ratings',[5 4 5])**

**d(1)**

**d(2)**

**d(3)**

**d =**

**1x3 struct array with fields:**

**name**

**subject**

**ratings**

**ans =**

**name: 'Белошапко Алексей Геннадьевич'**

**subject: 'Математический анализ'**

**ratings: [5 4 5]**

**ans =**

**name: 'Белошапко Алексей Геннадьевич'**

**subject: ' Алгебра'**

**ratings: [5 4 5]**

**ans =**

**name: 'Белошапко Алексей Геннадьевич'**

**subject: ' Геометрия'**

**ratings: [5 4 5]**

% структура с массивами ячеек (элементы массивов ячеек должны быть согласованы)

% не ячейки закреплены, массивы ячеек согласованы

**d=struct('name','Белошапко Алексей Геннадьевич','subject',{'Математический анализ',...**

**'Алгебра','Геометрия'},'ratings',{5 4 5})**

**d(1)**

**d(2)**

**d(3)**

**d=struct('name','Белошапко Алексей Геннадьевич','subject',{'Математический анализ',...**

**'Алгебра','Геометрия'},'ratings',{5 4 5})**

**d(1)**

**d(2)**

**d(3)**

**d =**

**1x3 struct array with fields:**

**name**

**subject**

**ratings**

**ans =**

**name: 'Белошапко Алексей Геннадьевич'**

**subject: 'Математический анализ'**

**ratings: 5**

**ans =**

**name: 'Белошапко Алексей Геннадьевич'**

**subject: 'Алгебра'**

**ratings: 4**

**ans =**

**name: 'Белошапко Алексей Геннадьевич'**

**subject: 'Геометрия'**

**ratings: 5**

% заполнение «базы данных»

% согласованный вывод элементов массива ячеек

**bd=struct('Name', {'Белошапко А.Г.' 'Бубнов В.В.' 'Герм А.В.'},...**

**'Sum\_result',{83,79,90},'Rating',{2 3 1})**

**bd(1)**

**bd(2)**

**bd(3)**

**bd =**

**1x3 struct array with fields:**

**Name**

**Sum\_result**

**Rating**

**ans =**

**Name: 'Белошапко А.Г.'**

**Sum\_result: 83**

**Rating: 2**

**ans =**

**Name: 'Бубнов В.В.'**

**Sum\_result: 79**

**Rating: 3**

**ans =**

**Name: 'Герм А.В.'**

**Sum\_result: 90**

**Rating: 1**

% добавление в «базу данных»

**bd(4)=struct('Name','Журавлева Л.В.','Sum\_result','95','Rating',1)**

Для работы со структурами имеются следующие функции: **isfield, getfield, setfield**, **rmfield, fieldnames**.

1. Функция isfield(s,fldname) возвращает 1 если у структуры s есть поле fldname и 0 в противном случае.
2. Функция getfield (s,fldname) возвращает значение поля fldname структуры s, т.е. является синонимом для s.fldname.
3. Функция setfield (s,fldname,fldvalue) возвращает копию объекта s, у которой полю fldname присвоено значение fldvalue. Сам объект s при этом не изменяется, т.е. это неравносильно присваиванию s.fldname=fldvalue.
4. Функция rmfield(s,fldname) предназначена для удаления поля fldname у объекта s. Для того чтобы добавить новое поле, достаточно просто присвоить ему значение: s.new\_field\_name=some\_value.
5. Функция fieldnames(s) возвращает массив ячеек, содержащий имена полей структуры s.

Примеры использования этих команд.

**isfield(bd,'name')**

**isfield(bd,'Name')**

**getfield(bd,'Name')**

**getfield(bd(3),'Name')**

**new\_record=setfield(bd(3),'Name','Герм Александр Вадимович')**

**bd(3)**

**new\_record**

% добавление нового поля и присвоение в 1-записи структуры полю group значения 112

**bd(1).group='112'**

**fieldnames(bd)**

% удаление поля group из структуры bd

% обратить внимание, слева стоит та же структура (проверить что будет, если не стоит)

**bd=rmfield(bd,'group')**

**fieldnames(bd)**

% групповая операция копирования элементов поля структуры в массив

% обратить внимание на применение квадратных скобок

**r=[bd.Rating]**

**Структура структур**

В качестве содержимого некоторого поля структуры может выступать другая структура, то есть структуры могут быть вложенными.

**Пример**:

% создание структуры, одно из полей которой является структурой

**struct2=struct('T\_name','Иванов А.А.','O\_name','Введение в MATLAB','Students', ...**

**struct('Name','Журавлева Л.В.','Sum\_result','95','Rating','1'))**

**struct2**

**T\_name: 'Иванов А.А.'**

**O\_name: 'Введение в MATLAB'**

**Students: [1x1 struct]**

**struct2.Students**

**Name: 'Журавлева Л.В.'**

**Sum\_result: '95'**

**Rating: '1'**

% добавление во внутреннюю структуру дополнительного поля group

**struct2.Students.group='112'**

**struct2 =**

**T\_name: 'Иванов А.А.'**

**O\_name: 'Введение в MATLAB'**

**Students: [1x1 struct]**

**struct2.Students**

**ans =**

**Name: 'Журавлева Л.В.'**

**Sum\_result: '95'**

**Rating: '1'**

**group: '112'**

% копирование данных из структуры bd в структуру Students

**struct2.Students=bd**

**struct2 =**

**T\_name: 'Иванов А.А.'**

**O\_name: 'Введение в MATLAB'**

**Students: [1x4 struct]**

% обращение к третьей записи вложенной структуры

**struct2.Students(3)**

**ans =**

**Name: 'Герм А.В.'**

**Sum\_result: 90**

**Rating: 1**

**group: []**

% вывод на экран всех значений поля struct2.Students.Name

**for i = 1 : length(struct2.Students)**

**disp(struct2.Students(i).Name)**

**end**

**Белошапко А.Г.**

**Бубнов В.В.**

**Герм А.В.**

**Журавлева Л.В.**

% получение значения поля Rating из третьей записи внутренней структуры

**value = getfield(struct2.Students, {3}, 'Rating')**

**value =**

**1**

**Задание:** Организовать форматный вывод «базы данных».

Вывод массива ячеек: **celldisp (имя\_массива ячеек)**

**celldisp(cel\_ar)**

Вывод макета массива ячеек: **cellplot (имя\_массива ячеек)**. Массивы чисел и символьные строки выводятся в виде таблиц, структуры выводятся в виде квадратов.

**cellplot (cel\_ar)**

cel\_ar{1,1}.Name='Панькина Наталья Андреевна';

cel\_ar{1,1}.Group='101';

cel\_ar{1,1}.Sex='f';

cel\_ar{1,2}.Name='Нечипорук Антон Алексеевич';

cel\_ar{1,2}.Group='101';

cel\_ar{1,2}.Sex='m';

cel\_ar{2,1}='Академическая гимназия'

cel\_ar{2,2}='Академическая гимназия'

cel\_ar{3,1}=[1 4 7 9; 3 5 8 9]

cel\_ar{3,2}=[2 5 6 8; 1 2 4 7]

***celldisp(cel\_ar)***

***cellplot(cel\_ar)***

**Доступ к переменным из различных рабочих пространств**

Существует еще один способ доступа к переменным родительского рабочего пространства – при помощи функций **assignin** и **evalin**.

Синтаксис вызова первой из этих команд **assignin(ws,'name',v)**, где

**ws** – имя рабочего пространства,

**'name'** – имя присваиваемой переменной,

**v** – ее значения.

Команда **assignin** ***заносит значение*** **v** в переменную по имени **'name'** рабочего пространства **ws.**

Параметр ws может принимать два значения:

**'base'** – для обозначения корневого рабочего пространства;

**'caller'** – для обозначения рабочего пространства вызывающей функции.

Для того чтобы ***получить значение*** переменной (или ***вычислить выражение***) в другом пространстве, используется команда **evalin(ws,'expression').**

Рассмотрим пример использования пространства имен **'base'** из функции **k=assignin\_demo**:

**function y=assignin\_demo**

**assignin('base','z',25);**

**y=evalin('base','x\*5');**

Вызовем ее из командного окна:

**x=7;**

**z=10;**

**y=assignin\_demo**

**z**

Обращаясь к рабочему пространству **'caller'** можно получить доступ лишь к тем локальным переменным, которые упомянуты в списке входных или выходных аргументов.

**Пример**: % доступ к пространству **'caller'**.

% рассмотрим две функции **assignin\_demo1** и **assignin\_demo2**:

**function m=assignin\_demo1**

**disp('Переменная (массив) in assignin\_demo1 workspace:')**

**m=[1 2 ; 3 4] ;**

**assignin\_demo2;**

**disp('После вызова функции assignin\_demo2 переменная m имеет значения:')**

**disp(m)**

**function assignin\_demo2**

**m=evalin('caller','m');**

**disp('переменная m in caller(assignin\_demo1) workspace:')**

**m**

**disp(Изменение переменной m assignin\_demo1 workspace:')**

**m=[5 6; 7 8] ;**

**assignin('caller','m',m);**

Вызовем assignin\_demo1

**d=assignin\_demo1;**

**Обработка входных и выходных аргументов функций MATLAB**

Большинство функций MATLAB поддерживают различный синтаксис вызова данных (полиморфизм). Для обработки аргументов используется 4 функции:

**varargin** – массив входных аргументов,

**varargout** – массив выходных аргументов,

**nargin** – количество входных аргументов,

**nargout** – количество выходных аргументов.

Так как входные и выходные аргументы могут быть разного типа, то для их хранения используют массивы ячеек.

Таким образом, первый входной аргумент будет адресоваться как **varargin{1},** второй – **varargin{2}** и т.д.

Рассмотрим функцию sum\_and\_difference, которая будет корректно работать для случая, когда передается **0, 1, 2 и >2** параметров, а также с различным представлением выходных параметров функции.

**function varargout=sum\_and\_difference2(varargin);**

**if nargin>=1**

**a=varargin{1};**

**else**

**a=0;**

**end**

**if nargin>=2**

**b=varargin{2};**

**else**

**b=0;**

**end**

**s=a+b;**

**d=a-b;**

**if nargout==2**

**varargout{1}=s;**

**varargout{2}=d;**

**else**

**varargout{1}=[s, d];**

**end**

**Теперь можно будет вызывать функцию с разными вариантами синтаксиса: [s,d]=sum\_and\_difference2(3,7)**

**s=sum\_and\_difference2(3,7)**

**s=sum\_and\_difference2(3)**

**[s,d]=sum\_and\_difference2(3)**

**sum\_and\_difference2**

**Задание**:

Написать программу-функцию polimorf(). У функции предусмотреть варианты с разными входными и выходными параметрами. Вызывается функция polimorf1(). Рассмотреть следующие варианты:

1. **Ведение базы данных**. Входные параметры: массив структур. Организовать форматный вывод данных. Выходных данных: нет.
2. **Решение системы линейных уравнений**. Входные параметры: матрица и вектор столбец. Выходной параметр: решение СЛАУ. Сделать форматный вывод на экран.
3. **Ведение базы данных – статистический анализ**. Найти студента с минимальным средним баллом, максимальным средним баллом, средний балл по всем студентам. Входные параметры: массив структур. Выходные параметры: ФИО студента с минимальным баллом и минимальный балл, ФИО студента с максимальным баллом и максимальный балл, средний балл по группе студентов.

***Данные для задачи:***

% Матрица системы **A** и вектор свободных членов **b**

A=[2 3 1;3 1 -1;2 -1 3];

b=[11;2;9];

% База данных

z=struct('Department','ПМ-ПУ','Group','103','Name','Петрова Наталья Андреевна','Marks',...

struct('Subject',{'Математический анализ','Алгебра','Геометрия'},'Rating',{4 5 5}));

z(2)=struct('Department','ПМ-ПУ','Group','103','Name','Николаев Антон Алексеевич','Marks',...

struct('Subject',{'Математический анализ','Алгебра','Геометрия'},'Rating',{4 5 4}));

z(3)=struct('Department','ПМ-ПУ','Group','103','Name','Иванов Иван Иванович','Marks',...

struct('Subject',{'Математический анализ','Алгебра','Геометрия'},'Rating',{3 4 3}));

***Формат вывода результатов выполнения программы:***

>> polimorf

Решение СЛАУ:

1.00 2.00 3.00

Проверка:

11.00 2.00 9.00

Вывод базы данных:

ПМ-ПУ 103 Петрова Наталья Андреевна

Математический анализ 4

Алгебра 5

Геометрия 5

ПМ-ПУ 103 Николаев Антон Алексеевич

Математический анализ 4

Алгебра 5

Геометрия 4

ПМ-ПУ 103 Иванов Иван Иванович

Математический анализ 3

Алгебра 4

Геометрия 3

Имя студента с min баллом: Иванов Иван Иванович. Балл: 10

Имя студента с max баллом: Петрова Наталья Андреевна. Балл: 14

Средний балл по группе: 12.333333