Tipi di Dato Strutturati: Stringhe e Strutture

Informatica, AA 2021/2022

Francesco Trovò

8 Ottobre 2021

https://trovo.faculty.polimi.it/

francesco1.trovo@polimi.it

Stringhe

Stringhe

Le stringhe sono array di caratteri. Per l'assegnamento delimito i caratteri tra apici singoli

```
msg = 'ciao';
```

```
>> whos msg
  Name      Size            Bytes  Class    Attributes

  msg       1x4                 8  char
```

Posso modificare gli elementi della stringa come faccio normalmente con i vettori

Manipolazione di stringhe

Esempi:

```
>> msg = 'ciao mamma';
>> msg = [msg , ' torno per cena']
msg =
ciao mamma torno per cena
```

Concatenazione di stringhe

```
>> msg(1) = 'C'
msg =
Ciao mamma torno per cena
```

Modificare i valori di una stringa

Stringhe: array di caratteri

Con **disp**, come per **fprintf**, è possibile inserire i valori di alcune variabili all'interno del testo visualizzato

Per le stringhe basta concatenare la variabile

```
nome = 'Giacomo'
disp(['ciao, ', nome])
ciao, Giacomo
```

Per le variabili numeriche occorre `num2str` che trasforma un valore numerico in sequenza di caratteri

```
nome = 'Giacomo'
anno = 1
disp(['ciao, ', nome, ' ho questo corso da anni: ', num2str(anno)])
```

Cosa fa?

```
for c = 'ciao'
      disp(c)
end
```

Cosa fa?

```
for c = 'ciao'
      disp(c)
end
```

Stampa a schermo i caratteri all'interno di 'ciao' uno per riga

Se volessi non andare a capo dopo ogni carattere userei:

```
for x = 'ciao'
    fprintf('%c', x);
end
```

Confronto tra Stringhe

Per confrontare due stringhe è possibile procedere:

- Confronto elemento per elemento
- Utilizzando la funzione **`strcmp(s1 , s2)`** che restituisce true o false (comoda anche la funzione **`strcmpi`** che è case-insensitive)
- Operazioni vettoriali (vedremo nella seconda parte)

Esercizio

Scrivere un programma che determina se due parole contengono le stesse vocali nello stesso ordine

Es: *mamma* e *anna* contengono le stesse vocali
cosa e *caso* non contengono le stesse vocali

Hint:

1. Estrarre, da ogni parola, un vettore contenente le vocali
2. Confrontare i due vettori delle vocali

Soluzione

```
% richiedere parole all'utente
p1 = input('inserire parola1: ', 's');
p2 = input('inserire parola2: ', 's');

v1 = [];
jj = 1;
% creo vettore v1 contenente le vocali di p1
for ii = 1 : length(p1)
    if (p1(ii) == 'a' || p1(ii) == 'e' || p1(ii) == 'o' || p1(ii) == 'i' ||
p1(ii) == 'u')
       v1(jj) = p1(ii); jj = jj + 1; % oppure v1 = [v1 , p1(ii)];
    end
end
disp(['le vocali di ', p1, ' sono ', v1])

%% controlla se v1 coincide con v2 (NEXT SLIDE)
```

Soluzione

```
%% controlla se v1 coincide con v2

if(length(v1) == length(v2))
    flag = true;
    ii = 1;
    while(ii < length(v1))
        if(v1(ii)  ~= v2(ii))
            flag = false;
        end
        ii = ii + 1;
    end
end

if flag
    fprintf('\n %s e %s hanno le stesse vocali\n', p1, p2);
else
    fprintf('\n %s e %s NON hanno le stesse vocali\n', p1, p2);
end
```

Esercizio

Scrivere un programma che determina se una parola è palindoma

Matrici: Array Bidimensionali

Le Matrici

Le matrici sono array 2-D

Hanno quindi due indici:

- L'indice di riga (il primo)
- L'indice di colonna (il secondo)

Esempio:

```
A = 16     2     3    13
     5    11    10     8
     9     7     6    12
```

Le Matrici

Le matrici sono array 2-D

Hanno quindi due indici:

- L'indice di riga (il primo)
- L'indice di colonna (il secondo)

Esempio:

```
A = 16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

Elemento alla riga 2 colonna 3

```
>> A(2, 3)

ans = 10
```

Creazione di Matrici

Le matrici vengono definite **affiancando vettori di dimensioni compatibili**

- Usiamo sempre gli operatori **,** (spazio) e **;** (vai a capo)
- L'operazione di **trasposizione** inverte le righe e le colonne della matrice

Es:

```
>> a = [1 , 2 ; 3 ,4 ]
a =
     1     2
     3     4
```

```
a' =
     1     3
     2     4
```

Le dimensioni di una matrice

Per scorrere una matrice è spesso necessario conoscere le sue dimensioni

Il comando size(A, dim) restituisce il numero di elementi di A lungo la dimensione dim

```
A = 16     2     3    13
     5    11    10     8
     9     7     6    12
righe = size(A, 1);
colonne = size(A, 2);
```

Le Matrici: operatore CAT

La concatenazione dei vettori avviene mediante operatore **CAT** che richiede **dimensioni consistenti** dei vettori

```
>> a = [1 : 3]
a =
     1     2     3

>>  b = [4; 5; 6]
b =
     4
     5
     6
```

```
>> A = [a; b]
Error using vertcat
CAT arguments dimensions are
not consistent.
>> A =[a, b]
Error using horzcat
CAT arguments dimensions are
not consistent.
>> A =[a; b']
A =
     1     2     3
     4     5     6
```

Accedere agli Elementi di una Matrice

Per accedere agli elementi di una matrice occorre specificare un valore per ogni indice

`nomeMatrice(indice1, indice2)`

Seleziona il valore alla riga **`indice1`** colonna **`indice2`** nella variabile **`nomeMatrice`**

Es

```
>> A = [1 : 3; 4 : 6; 7: 9 ]
A =
     1     2     3
     4     5     6
     7     8     9
```

```
>> A(2, 3)
ans =
     6
>> A(3,5)
Index exceeds
matrix dimensions.
```

Esempio

Visualizzare a schermo le prime 10 tabelline (utilizzando solo operazioni matriciali)

Stampare solo la parte triangolare bassa delle tabelline

Stampare solo la parte triangolare alta delle tabelline

Memorizzazione degli Array

Gli array vengono salvati linearmente in memoria.

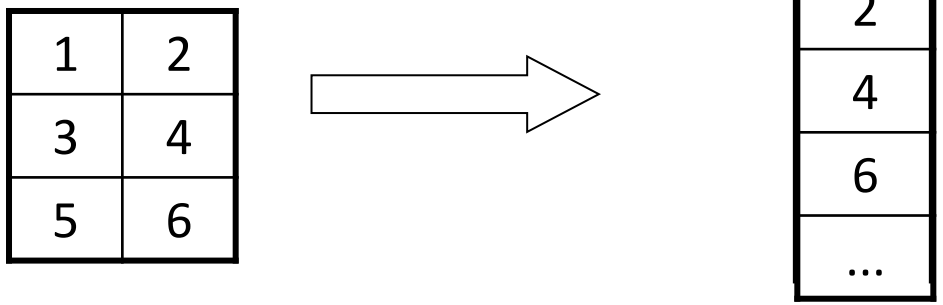
In particolare le matrici sono memorizzate

- per colonna: colonna 1, poi colonna 2, 3, etc.
- ogni colonna memorizzata per indici di riga crescenti

Array memorizzati in forma lineare nella RAM variando

- più velocemente i primi indici
- più lentamente quelli successivi

NB: opposto a quanto avviene in C



Array: forma *linearizzata*

Si può accedere a un array a più dimensioni come se ne avesse una sola

Usando un unico indice si segue l'ordine della memorizzazione

```
>> a = [1 2 3; 4 5 6; 7 8 9; 10 11 12]
a =
     1     2     3
     4     5     6
     7     8     9
    10    11    12
>> a(3, 2)
ans =
     8
>> a(10)
ans =
     6
```

TODO: Esercizio su Matrici

Si scriva un programma che prende in ingresso una matrice quadrata e controlla che sia simmetrica.

Una matrice è simmetrica se per ogni elemento vale la seguente proprietà: L'elemento alla riga i, colonna j coincide con l'elemento alla riga j colonna i

Es di matrice simmetrica

1	**12**	**1**
12	**0**	**3**
1	**3**	**23**

Strutture in Matlab

Struct vs Array

Gli **array** permettono di aggregare variabili **omogenee** in una sequenza

Le `struct` permettono di aggregare variabili **eterogenee** in una sola variabile

- Le `struct` è una sorta di "contenitore" per variabili disomogenee di tipi più semplici.
- Le variabili aggregate nella struct sono dette **campi** della struct

Esempio: variabile per contenere anagrafica di impiegati

- *nome, cognome, codice fiscale, indirizzo, numero di telefono, stipendio, data di assunzione etc.*
- *Non posso metterli in un array, sono variabili diverse, è molto sconveniente metterle in variabili separate, specialmente se ho diversi impiegati*

Creazione di una struct

Creazione di una struttura :

Utilizzando la funzione struct()

```
studente = struct('nome', 'Giovanni', 'eta', 24)
```

Assegnamento dei valori ai campi (e contestuale definizione dei campi)

```
studente.nome = 'Giovanni';
studente.eta = 24;
```

Accedere ai campi di una `struct`

Per accedere ai campi si usa l'operatore ***dot***.
Sintassi:

```
nomeStruct.nomeCampo;
```

Quindi, **`nomeStruct.nomeCampo`** diventa, a tutti gli effetti, **una «normale» variabile** del tipo di **`nomeCampo`**.

- **Ai campi** di una struttura applicabili tutte le **operazioni caratteristiche** del tipo di appartenenza
- In questo senso, il *dot* è l'omologo di **`(indice)`** per gli array

Creazione di una struttura campo per campo

Esempio: creo una struttura studente

```
studente.nome = 'Giovanni Rossi';
studente.indirizzo = 'Via Roma 23';
studente.citta = 'Cosenza';
studente.eta = 25;
```

Accesso ai campi come nel C con l'operatore .

```
nomeStruttura.nomeCampo
```

Es

```
disp([studente.nome, ' (' , studente.citta ,') ha ' ,
num2str(studente.eta) , ' anni'])
```

Creazione di una struttura campo per campo

Esempio: la struttura studente

```
studente.nome = 'Giovanni Rossi';
studente.indirizzo = 'Via Roma 23';
studente.citta = 'Cosenza';
studente.media = 25;
```

É possibile far diventare **`studente`** un array di strutture, accodando un altro elemento in **`studente(2)`**.

```
studente(2).nome = 'Giulia Gatti';
studente(2).media = 30;
```

Tutte le strutture dell'array devono avere gli stessi campi (**l'array deve essere omogeneo, la struttura non necessariamente**).

É possibile assegnare solo alcuni campi a **`studente(2)`**: i campi non assegnati rimangono vuoti.

Aggiunta di campi

Aggiunta di un campo

```
%facciamo riferiemento alla definizione di studente delle slide precedenti
studente(2).esami = [20 25 30];
```

Il campo esami viene aggiunto a tutte le strutture che fanno parte di studente

- Avrà un valore iniziale per studente(2). Sarà vuoto per tutti gli altri elementi dell’array

Creazione di una struttura mediante la funzione struct

Consente di preallocare una struttura o un array di strutture

```
S = struct('campo1',val1,'campo2',val2, …)
```

Es:
```
rilieviAltimetrici =
struct('latitudine',30,'longitudine',60,
'altitudine', 1920)
```

Creazione di una struttura mediante la funzione struct

Consente di preallocare una struttura o un array di strutture

```
S = struct('campo1',val1,'campo2',val2, …)
```

Es: `rilieviAltimetrici = struct('latitudine',30,'longitudine',60, 'altitudine', 1920)`

Esempio array di strutture:

```
s(5) = struct('x',10,'y',3);
```

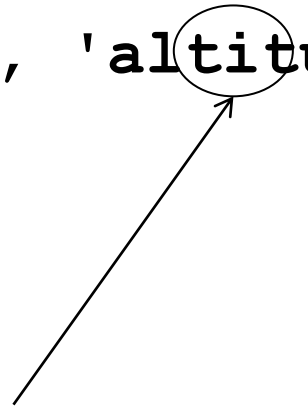
- s è un array 1x5 in cui ogni elemento ha attributi x e y
- solo il quinto elemento di s viene inizializzato con i valori x=10 e y=3
- gli altri elementi vengono inizializzato con il valore di default: [] (array vuoto)

Creazione di una struttura mediante la funzione struct

Consente di preallocare una struttura o un array di strutture

```
S = struct('campo1',val1,'campo2',val2, …)
```

Es: `rilieviAltimetrici(1000) = struct('latitudine',30,'longitudine',[], 'altitudine', 1920)`



Array vuoto. Attenzione: se si Inserisce un valore (es. 20), questo viene assunto dal campo longitudine dell'elemento 1000, ma non dallo stesso campo degli altri elementi dell'array

Array di strutture innestati

Un campo di una struttura può essere di qualsiasi tipo

E` quindi possibile avere un campo che è, di nuovo, una struttura o un array di strutture

Esempio

```
studente(1).corso(1).nome='InformaticaB';
studente(1).corso(1).docente='Von Neumann';
studente(1).corso(2).nome='Matematica';
studente(1).corso(2).docente='Eulero';
```

corso è un array di strutture

```
>> studente
studente =
    corso: [1x2 struct]
```

Esercizio

Si sviluppi uno script matlab che acquisisce da tastiera i dati relativi ad un numero arbitrario di rilievi altimetrici e che quindi stampa a video l'altitudine media di tutti i rilievi che si trovano nell'intervallo

- latitudine [30, 60]
- longitudine [10, 100]

Soluzione

```
% s = struct('altezza',[],'latitudine',[], 'longitudine',[])
n = input(['quanti rilievi? ']);
 % acquisizione dei rilievi
for ii = 1 : n
    s(ii).altezza = input(['altezza rilievo nr ', num2str(ii), ' ']);
    s(ii).latitudine= input(['latitudine rilievo nr ', num2str(ii), ' ']);
    s(ii).longitudine= input(['longitudine rilievo nr ', num2str(ii), ' ']);
end
% creo dei vettori con i valori dei campi
LAT = [s.latitudine];
LON = [s.longitudine];
ALT = [s.altezza];
% operazioni logiche per definire il sottovettore da estrarre da altezza
latOK = (LAT > 30) & (LAT <60);
lonOK = (LON> 10) & (LON<100);
posOK =latOK & lonOK;

% estrazione sottovettore e calcolo media
mean(ALT(posOK));
```

Array Multidimensionali

È possibile specificare una terza (quarta, quinta…) dimensione lungo la quale indicizzare un array.

Ad esempio le immagini a colori sono definite con tre piani colore (RGB), quindi

- un'immagine a colori 10 Mpixels, aspect ratio (3:4) richiede una matrice 3D di 2736 x 3648 x 3 elementi
- 10 sec di video full HD (1080 x 768) a 24fps richiede una matrice 4D di 1080 x 768 x 3 x (10 x 24) elementi