# FONDAMENTI DI INFORMATICA

## TERZA LEZIONE

Prof. Alfredo Accattatis

( accattatis@ing.uniroma2.it )

Tutor:

Prof. Vittorio Emanuele Calafiore

( vcalafio@gmail.com )

# ISCRIZIONE

- DELPHI 166
- TEAMS 186

- Grazie bis

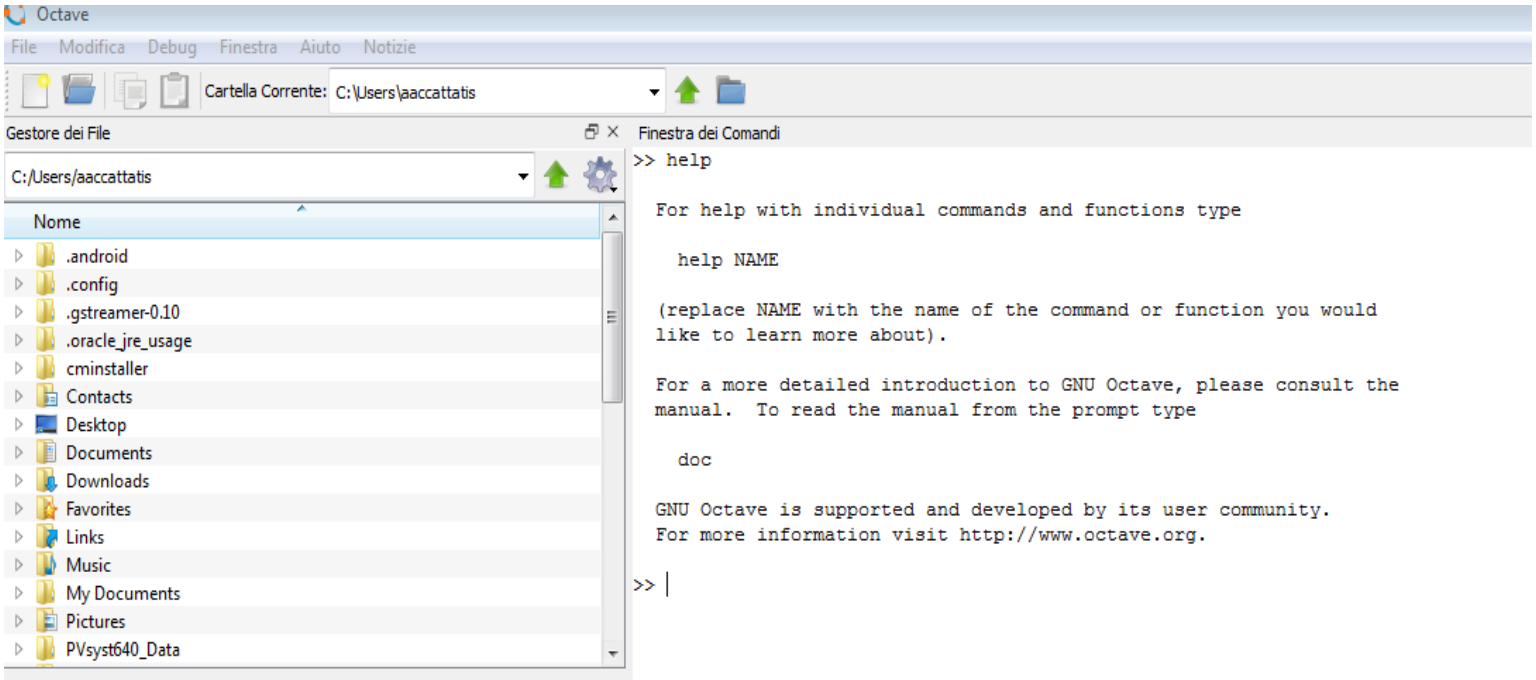
# Usiamo Matlab: primi rudimenti della scrittura dei programmi

Si parlerà di…

- Variabili e assegnazioni di valori
- Caratteri e codifica
- Tipi di dati numerici
  - Interi
  - Reali: Floating point (numeri in virgola mobile)
  - Cenni al tipo “carattere” e “vettore”
- Espressioni numeriche

# Usare Matlab/Octave

- Installazione
- Lancio Matlab oppure Octave



# Usare Matlab/Octave 2

- Linguaggio di programmazione
- Calcolatrice scientifica
- Ambiente grafico

# Variabili ed assegnazioni

- **Assegnazione di valore**

**`nomevariabile = espressione`**

"=" è l'operatore di assegnazione; non significa "uguale".

La sintassi è simile all'algebra convenzionale ma ha un significato differente!

```
z = x + y
z = 4*x - y
```

***Algebra***: implica relazione tra x e y per mezzo di addizioni e semplificazioni:

```
2y = 3x
```

***Programmazione***: cambio di mentalità…devo cambiare il modo di pensare in merito a quello che voglio memorizzare nella variabile z

**z = z + 1** ….. Che significa?

# Espressioni (informatica)

- In un linguaggio di programmazione un'**espressione** è un costrutto che combina uno o più valori (ad esempio costanti o variabili), utilizzando operatori e funzioni.
- Le operazioni e le funzioni sono interpretate (*valutate*) secondo particolari regole di precedenza e di associazione per un particolare linguaggio di programmazione, producendo un valore.
- Esempio: **`z = 4*x - y + SIN(y) + Cost`**
- **`4*x - y + SIN(y) + C`** è **l'espressione**
- **`*,-,+`** = **operatori** (tra uno o più operandi)
- **`x,y,z`** sono **variabili**
- **`SIN`** = **funzione**
- **`Cost`** = valore **costante**

# Nome della variabile

- Deve sempre essere alla sinistra e l'espressione sulla destra
- Puo essere una combinazione di lettere maiuscole e minuscole, numeri ed i caratteri speciali "_" e "$". Lo spazio non è permesso.
- DEVE iniziare con una lettera dell'alfabeto

```
myfirstvariable
myFirstVar
my_1st_var
1stvarofmine
```


- MATLAB è "case sensitive": le variabili `mynum`, `MYNUM` e `Mynum` sono differenti!
- Le "Reserved words" o "Keywords" non possono essere usate come nomi di variabili

# Tipi di dati

- Un linguaggio di programmazione tratta i dati memorizzati in una variabile come appartenenti ad una determinata categoria (es. Interi, Reali, Caratteri etc.)
- Non “tipizzati” (e.g. linguaggio macchina)
- “Tipizzati”, e.g. C, Fortran
  - Nome e Tipo devono essere specificati
  - “Tipizzazione debole” o dinamica
  - “Tipizzazione forte” o statica

# Tipi di dati

- Tipo: nome che indica l'insieme di valori che una variabile (oppure il risultato di un'espressione) può assumere e le operazioni che si possono compiere con essi. In generale in Matlab è definita Classe.
- Esempio: numeri interi (1,2,3…) => (+, -, *….)
- Esempio: numeri reali (3.1416, 2.7….) => (+,:, *…)
- Esempio: carattere ('a', 'b' etc.) => (concatenazione…)

# Tipi di dati

- a Metri;
- b Secondi;
- c,d Metri x Secondi;
- a+b, b+a,c+a **KO**
- c=a*b + d **OK**

# NASAs metric confusion caused Mars orbiter loss

September 30, 1999



CNN NASA lost a 125 million Mars orbiter because one engineering team used metric units while another used English units for a key spacecraft operation, according to a review finding released Thursday.

For that reason, information failed to transfer between the Mars Climate Orbiter spacecraft team at Lockheed Martin in Colorado and the mission navigation team in California. Lockheed Martin built the spacecraft.

- **SPAZIO: SONDA MARTE, UN ERRORE DI CONVERSIONE METRI-MIGLIA**
- **INDIFENDIBILE" L'USO DEL SISTEMA EMPIRICO NEGLI USA**
- Pasadena, 1 ott. (Adnkronos) - Un errore di conversione dal sistema metrico decimale a quello commerciale americano basato su pollici e piedi è stata la causa del fallimento della missione della Nasa su Marte del Climate Orbiter, la sonda costata 125 milioni di dollari che avrebbe dovuto misurare per un anno marziano le condizioni meteo del Pianeta rosso, e che invece ha cessato di inviare segnali a terra poco dopo essere entrata nell'orbita, dopo un viaggio di 286 giorni. L'errore non ha fatto altro che fomentare le polemiche contro l'uso "oramai indifendibile", come ha commentato Lorelle Young, Presidente dell'Associazione metrica americana, del sistema commerciale negli Stati Uniti.
- Gli ingegneri della Lockheed Martin, la capo commessa industriale della sonda, il colosso aerospaziale americano che lavora abitualmente per la Nasa, usa infatti il sistema basato su pollici, piedi e miglia mentre all'Agenzia spaziale sono stati introdotti da tempo, dal 1990, i calcoli con il sistema metrico decimale.
- L'errore di conversione emerso durante i lavori della commissione di revisione interna della Nasa a cui hanno partecipato dieci esperti di navigazione ha quindi impedito il corretto trasferimento di informazioni dai tecnici di Denver che hanno sviluppato la sonda, compreso il software che regola i motori dell'Orbiter, e quelli del Ket Propulsion Laboratory di Pasadena che la avevano commissionata (...omissis...)

# Comandi utili

- `namelengthmax`
- `who`
- `whos`
- `clear`
- `clear <variablename>`
- `clear <variablename1> <variablename2> …`

# Tipi di dati

- MATLAB = tipizzazione debole (ossia tipizzate o tipizzate "automaticamente")
- Tipizzazione automatica

```
>> a = [2 3]      % a è un vettore con due elementi
a =
     2     3
>> x=0.5*a
x =
   1.0000e+00   1.5000e+00
```

- Tipizzate (si effettua un typecast)

```
>> ai = uint8(a)  %Per informazioni aggiuntive sul tipo uint8
                  %digitare «help uint8» sulla console
                  %e/o consultare la documentazione online
ai =
    2    3
>> x = 0.5*ai
x =
    1    2
```

# Effetti della tipizzazione dinamica

letter è un carattere (character)

```
>> letter = 'A'

letter = A

>> letter = letter+1

letter =   66
```

letter è un double

| Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |

```
>> radius=49
radius =
    49
>> radius + 1
ans =
    50
>> radius='radius of a circle'
radius =
radius of a circle
>> radius + 1
ans =
  Columns 1 through 16
   115    98   101   106   118   116    33   112   103    33    98    33   100   106   115   100
  Columns 17 through 18
   109   102
```

# Caratteri e codifica

- Un carattere in MATLAB è rappresentato usando le virgolette singole
  - `'a','x','3'`
- I caratteri sono ordinati usando una codifica dei caratteri specifica
- La più comune e la codifica ASCII
- In MATLAB c'è una funzione per convertire un carattere nel suo equivalente numerico

```
>> numequiv = double('a')
  numequiv =  97
```

# La codifica ASCII

- ASCII sta per “American Standard Code for Information Interchange”.
- I Computer “comprendono” solo numeri, pertanto, un codice ASCII è la rappresentazione numerica di un carattere come p. es. 'a' oppure '@’.
- ASCII fu sviluppato tanto tempo fa ed attualmente i caratteri “non stampabili” sono usati raramente (lo scopo per cui furono creati è venuto meno…)
- Di seguito la tabella ASCII con la descrizione dei primi 32 caratteri non-stampabili. ASCII fu in realtà progettato per uso con dispositivi (teletypes) oramai non più esistenti. Se qualcuno vi dice che vuole il vostro CV in formato ASCII significa semplicemente che vuole il testo “puro” (in inglese plain-text) ossia un testo senza caratteri di formattazione come tabs, neretto o sottolineatura – il formato “grezzo” che ogni computer può comprendere. In tal modo è possibile importare il file in ogni applicazione senza problemi. Notepad.exe crea testo in ASCII puro, oppure in MS Word è possibile selezionare il salvataggio come “solo testo”.

# La tabella ASCII

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

# Tipo di default

- Di default, MATLAB memorizza tutti i valori numerici come “reale doppia precisione”
- Tipi di dati addizionali memorizzano testo, interi o valori a singola precisione, oppure una combinazione di dati correlati in una variabile singola (esempio vettore)

# Gli script

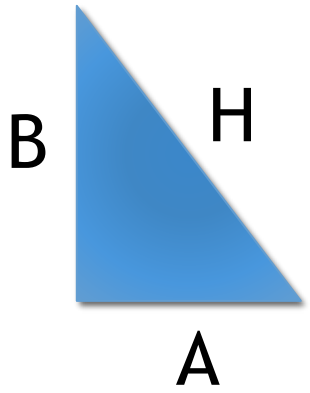
- Lo script è generalmente contenuto in un file
- Contiene una sequenza di comandi
- I comandi sono (per esempio) gli stessi che abbiamo visto sinora
- Esempio: assegnazione di valore a variabile
- Esempio: comando who
- Esempio: operazione di somma, sottrazione divisione
- Una volta creato lo script con estensione .m lo potremo usare come un comando predefinito di Matlab, chiamandolo da linea di comando

# Un primo problema

Creare uno script che faccia uso del teorema di Pitagora per il calcolo dell'ipotenusa di un triangolo rettangolo:

$$H^2 = A^2 + B^2$$

Dove A e B sono i cateti, ed H è l'ipotenusa.




```
clear
clc


A = 3; % primo cateto
B = 4; % secondo cateto
hypSq = A^2 + B^2; % quadrato dell'ipotenusa(algoritmo)
H = sqrt(hypSq) % …il risultato
```

# Gli script

- Creare uno script «vuoto» da Octave o Matlab in tempo reale

# Tipi interi in MATLAB (ed altri linguaggi)

| Class | Range of Values | Conversion Function |
|---|---|---|
| Signed 8-bit integer | $-2^7$ to $2^7-1$ | `int8` |
| Signed 16-bit integer | $-2^{15}$ to $2^{15}-1$ | `int16` |
| Signed 32-bit integer | $-2^{31}$ to $2^{31}-1$ | `int32` |
| Signed 64-bit integer | $-2^{63}$ to $2^{63}-1$ | `int64` |
| Unsigned 8-bit integer | 0 to $2^8-1$ | `uint8` |
| Unsigned 16-bit integer | 0 to $2^{16}-1$ | `uint16` |
| Unsigned 32-bit integer | 0 to $2^{32}-1$ | `uint32` |
| Unsigned 64-bit integer | 0 to $2^{64}-1$ | `uint64` |

- Per conoscere il range di un tipo intero usare le funzioni `intmin` e `intmax`, e.g. `intmin('int16')`
- MATLAB memorizza i dati numerici come double per default: per memorizzare dati come interi bisogna convertirli da double al tipo intero desiderato. Usare una delle funzioni di conversione che vedremo a breve.

# Tipo Floating point

- Tipo doppia precisione (o double) : IEEE Standard 754 : ogni valore memorizzato come double occupa 64 bit.

| Bits | Usage |
|---|---|
| 63 | Sign (0 = positive, 1 = negative) |
| 62 to 52 | Exponent, biased by 1023 |
| 51 to 0 | Fraction f of the number 1.f |

- Tipo singola precisione (o single) : IEEE Standard 754: ogni valore memorizzato come single occupa 32 bit.

| Bits | Usage |
|---|---|
| 31 | Sign (0 = positive, 1 = negative) |
| 30 to 23 | Exponent, biased by 127 |
| 22 to 0 | Fraction f of the number 1.f |

Numeri reali rappresentati da un single sono rappresentati con meno precisione rispetto ai numeri rappresentati da un tipo double.

# Calcolo combinatorio

- Perché intmin('int16') restituisce -32768?
- E perché intmax('int16') restituisce 32767?
- int16 è un intero a 16 bit
- Calcolo combinatorio (disposizioni con ripetizione)
- $D'(n,k) = n^k$
- Formula per calcolare il numero di diversi simboli rappresentabili data una sequenza di k caratteri ognuno dei quali può assumere n valori

# Calcolo combinatorio

- D'(2,16) = 2^16 = 65536 *differenti simboli*
- Un bit per il segno
- D'(2,15) = 2^15 = 32768 *differenti numeri negativi*
- *Uno per lo ZERO*
- *I restanti 32767 per i numeri positivi*

# Conversione Double → Integer

- Esempio: memorizzare 325 come un 16-bit intero con segno assegnato alla variabile X

```
>> x = int16(325);
```

- Se il numero da convertire in intero ha una parte decimale (parte frazionaria), MATLAB arrotonda all'intero più vicino.
- Se la parte frazionaria è esattamente 0.5, allora tra due interi «equidistanti» MATLAB sceglie quello maggiore (quello il cui valore assoluto e più elevato in ampiezza) :

```
>> x = 325.499;
>> int16(x)
ans = 325

>> x = x + .001;
>> int16(x)
ans = 326
```

# Double → Integer: schemi di arrotondamento alternativi

- Se avete bisogno di arrotondare un numero con criteri diversi da quelli standard (di default) MATLAB fornisce quattro funzioni:
  - `round`: verso l'intero più vicino
  - `fix`: arrotonda all'intero più vicino allo zero (= taglia la parte decimale)
  - `floor`: verso il più basso intero in direzione -infinito (numero intero immediatamente inferiore con segno )
  - `ceil`: verso il più alto intero in direzione +infinito (numero intero immediatamente superiore con segno)

```
>> x = 325.9;
>> int16(fix(x))
ans = 325
```

# Espressioni numeriche

- Espressioni possono essere create utilizzando valori, variabili, operatori, funzioni built-in, parentesi.

```
>> 2 * sin(1.4)
```

- Un operatore può essere
  - unario: un solo operando
  - binario: due operandi
- Operatori comuni sono:

| | |
|---|---|
| + | Addizione |
| - | Negazione , sottrazione |
| * | Moltiplicazione |
| / | Divisione a destra |
| \ | Divisione a sinistra (o inversa) |
| ^ | Elevamento a potenza |

# Espressioni numeriche

- Le espressioni aritmetiche vengono valutate in base alle regole di precedenza degli operatori.
- Le parentesi modificano la precedenza

```
>> 4 + 5 * 3        ans = 19
>> (4 + 5) * 3      ans = 27
```

- Ordine di precedenza dal più alto al più basso

| | |
|---|---|
| () | parentesi |
| ^ | elevamento a potenza |
| ~,- | negazione |
| *, /, \ | moltiplicazione e divisione |
| +, - | addizione e sottrazione |

- Esempi

```
>> -5^2          ans = -25
>>(-5)^2         ans = 25
>> 10-6/2        ans = 7
>> (10-6)/2      ans = 2
```

# Funzioni predefinite (built-in)

- Trigonometriche: `sin(x), cosh(x), atan(x), …`
- Esponenziali e logaritmiche: `exp(t), log(y), …`
- Generatori numeri/matrici pseudocasuali: `rand(N, z), …`
- … e molte altre

- Dettagli si ottengono usando il comando

```
>> help NOME_FUNZIONE
>> doc NOME_FUNZIONE
```

- Quando si “chiama” la funzione si passano anche gli argomenti. La funzione restituisce il valore del calcolo

```
>>sin(pi/2)          ans = 1
```

# Operazioni con numeri in virgola mobile

Gli operatori binari possono svolgere calcoli tra operandi di classi differenti. Vince il tipo meno «complesso»

| | Double | Single | Integer |
|---|---|---|---|
| **Double** | Double | Single | Integer |
| **Single** | Single | Single | Integer |
| **Integer** | Integer | Integer | Integer |

Con il type-casting è possibile convertire il tipo di una variabile.

```
>> d = pi
d = 3.141592653589793

>> s = single(d)
s = 3.1415927
```

# Operazioni con numeri in virgola mobile

- Come detto, esistono alcuni nomi di variabili che sono «riservati»

Attenzione!

- Questo non significa che non possiate sovrascriverne il valore
- Esempio

```
>> pi
ans =  3.1416
>> pi = 5
ans = 5
>> clear pi
>>pi
ans =  3.1416
```

# Espressioni relazionali

- Per verificare una data condizione si utilizza una espressione relazionale
- Operatori relazionali

| | |
|---|---|
| > | maggiore |
| < | minore |
| >= | maggiore o uguale |
| <= | minore o uguale |
| == | uguaglianza |
| ~= | disuguaglianza |

- Il risultato è il valore logico di classe *BOOLEAN* TRUE (1) oppure FALSE (0)

```
>> 'd'<'g'        ans = 1      (TRUE)
>> ~(2<3)         ans = 0      (FALSE)
```

# Algebra Booleana - ripasso

- Branca dell'algebra le cui variabili possono assumere solo due valori distinti
  - TRUE (vero) in MATLAB/Octave indicato da 1
  - FALSE (falso) in MATLAB/Octave indicato da 0

- Le operazioni di base sono quelle logiche:
  - Prodotto logico (congiunzione): AND in MATLAB/Octave **&&**
  - Somma logica (disgiunzione): OR in MATLAB/Octave **||**
  - Complementazione (negazione): NOT in MATLAB/Octave **~**

- Qualunque funzione logica può essere riportata ad una combinazione dei tre operatori di base.

# Algebra Booleana – ripasso

- Tabella della verità

| A | B | A && B | A \|\| B | ~A |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

```
&& AND
|| OR
~ NOT
```

```
0 FALSE
1 TRUE
```

# Operatori numerici, relazionali e logici

- Ordine di precedenza dal più alto al più basso

| | |
|---|---|
| () | parentesi |
| ^ | elevamento a potenza |
| -, ~ | negazione |
| *, /, \ | moltiplicazione, divisione |
| +, - | addizione, sottrazione |
| <, <=, >, >=, ==, ~= | relazionali |
| && | AND |
| \|\| | OR |
| **&** | **AND** |
| **\|** | **OR** |
| = | assegnazione |

- Esempio

```
>> 2 < 2 + 1 || 'x' == 'y'
ans = 1
```

# Organizzazione:

- *Vettori* e *matrici*: concetti basilari
- Creare un vettore in MATLAB
- Funzioni built-in di MATLAB per la creazione di vettori

# Array: cenni

Terminologia informatica: ***ARRAY*** è un insieme di dati omogenei organizzato in maniera che ciascun elemento (dato) è identificato univocamente.

*Es: contenitore con tanti cassetti*.

Scalari, vettori e matrici sono delle tipologie particolari di array:

Scalare: array di un unico elemento (1 X 1)

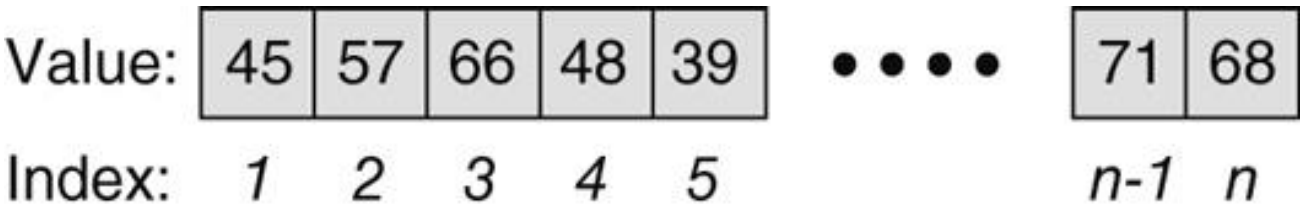
Vettore: array monodimensionale (m X 1 o 1 X n)

Matrice: array bidimensionale (m X n)

# Vettori

Nel vettore gli elementi sono disposti su una riga o su una colonna. Ciascun elemento di un vettore ha due attributi separati e distinti che lo rendono univoco:

- ***Valore***
- ***Posizione (indice numerico)***




Esempio: vettore riga - l'elemento di valore 39 è nella quinta posizione del vettore. Potrebbero esserci altri elementi di valore 39 ma nessun altro elemento potrà trovarsi in posizione 5. Con l'indice 5 si identifica un unico elemento del vettore.

# Matrici

Una matrice è una tabella di valori: gli elementi sono disposti in righe e colonne. Le dimensioni di una matrice sono *m* X *n*, dove *m* è il numero di righe, *n* è il numero di colonne.

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \mathrm{a}_{ij} \quad \mathrm{i} = 1, \ldots, \mathrm{m}; \quad \mathrm{j} = 1, \ldots, \mathrm{n}$$

# Operazioni sui vettori

Le operazioni basilari sui vettori sono:

- Creare un vettore
- Determinare la dimensione del vettore
- Estrarre dati dal vettore tramite gli indici
- Ridurre un vettore
- Calcoli matematici e logici sui vettori

- MATLAB = MATrix LABoratory

# Creare un vettore

- Inserimento diretto degli elementi

```
>> V_RIGA = [2, 5, 7, 1, 3]
>> V_RIGA = [2 5 7 1 3]
>> V_COLONNA = [32; 74; 50; 81; 2]
```

- Inserimento automatizzato
  - Range di valori
  - Valori equamente distribuiti
  - Valori casuali, **0**, **1**

# Creare un vettore: range di valori

$B = [v_1, v_2, \ldots, v_n] =$

$\{v_i \mid v_{min} <= v_i <= v_{max},\ v_i = v_{i-1} + K,\ v_1 = v_{min},\ i=2,\ldots,n\}$

Il vettore è definito da:

1. Valore iniziale del range $v_{min}$
2. Valore finale del range $v_{max}$
3. Passo di incremento $K$ (intero di valore costante)

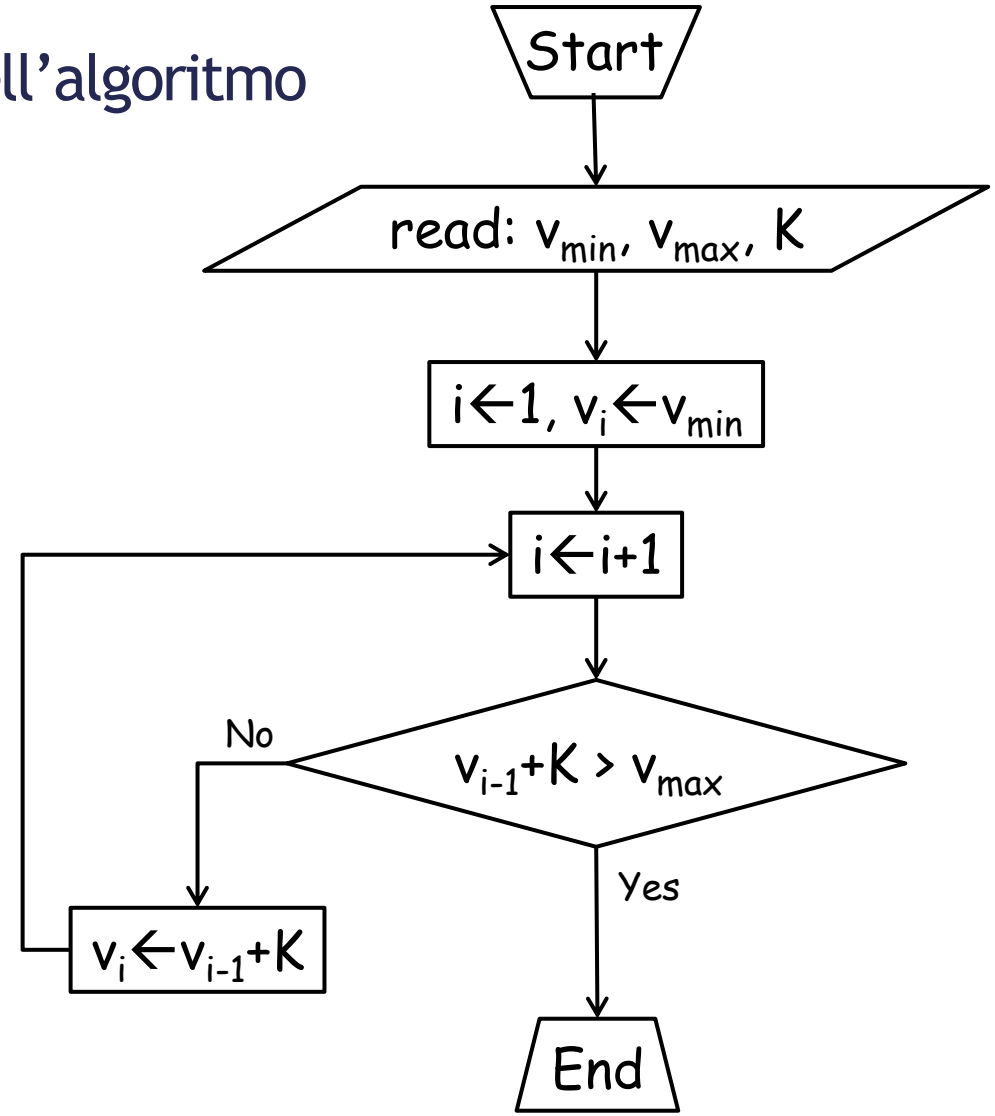
Esempio: K=3, $v_{min}$=1 and $v_{max}$=20

**B=[1,4,7,10,13,16,19]**

# Creare un vettore: range di valori (2)

Diagramma di flusso dell'algoritmo

$B = [v_1, v_2, \ldots, v_n] =$
$\{v_i \mid v_{min} <= v_i <= v_{max},$
$v_i = v_{i-1} + K,$
$v_1 = v_{min},\ i=2,\ldots,n\}$



# Creare un vettore: range di valori (3)

- MATLAB: si usa l'operatore due-punti (*colon*)

$B = [v_1, v_2, \ldots, v_n] =$

$\{v_i \mid v_{min} <= v_i <= v_{max},\ v_i = v_{i-1} + K,\ v_1 = v_{min},\ i=2,\ldots,n\}$

- Sintassi

$$B = v_{min} : K : v_{max}$$

Le parentesi quadre non sono necessarie

```
>> B = 1:3:20

   B = 1  4  7  10  13  16  19
```

# Creare un vettore: linspace

- Vettore di N elementi equamente distribuiti tra due estremi inclusi

$$C = [X_1, \ldots, X_N] \mid X_1 = Xmin, X_N = Xmax$$

Il vettore è definito da:

1. Valore del primo elemento Xmin
2. Valore dell'ultimo elemento Xmax
3. Il numero di elementi N (intero di valore costante)

- L'incremento è calcolato da (Xmax - Xmin) / (N - 1)

Esempio: N=11, $X_1$=0, $X_N$=20

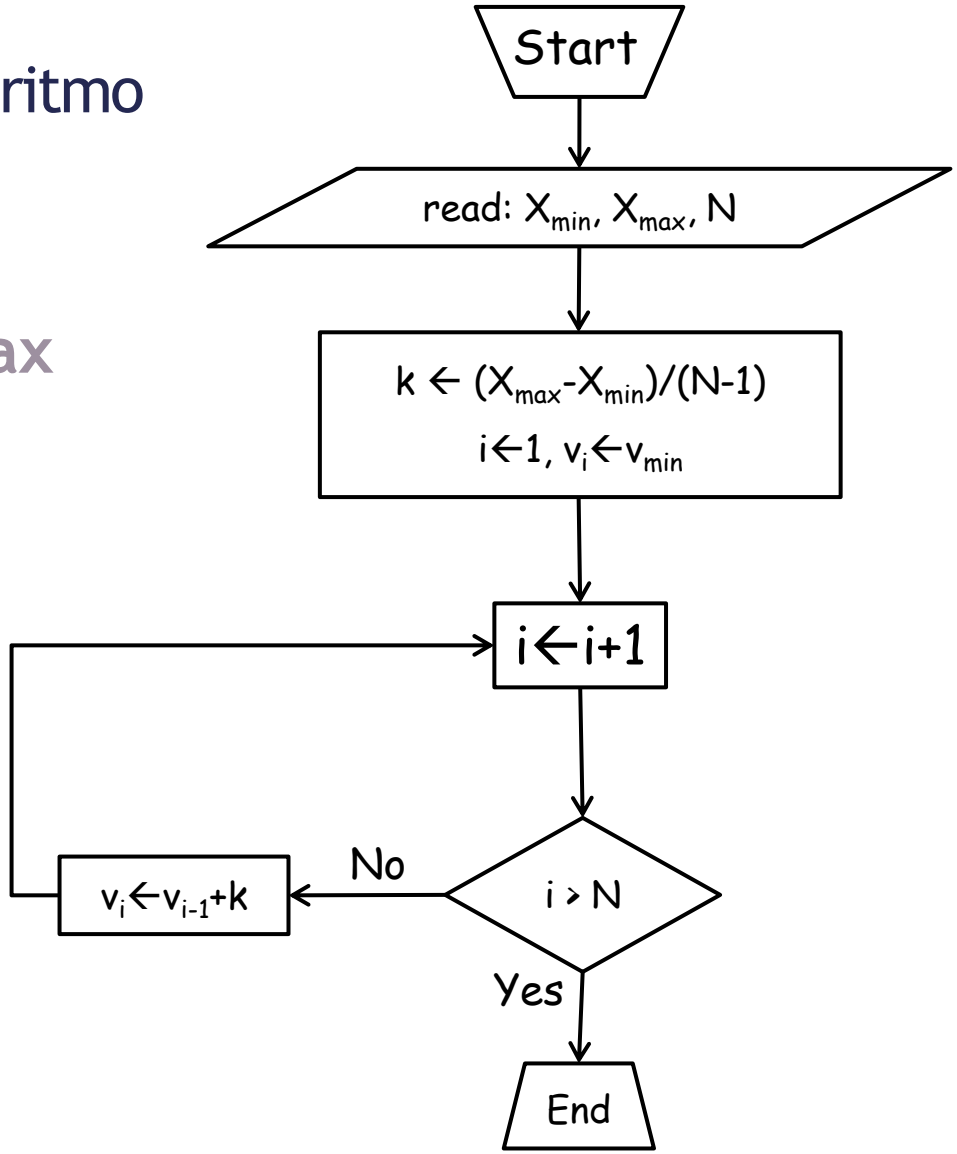
**C = [0 2 4 6 8 10 12 14 16 18 20]**

# Creare un vettore: linspace (2)

Diagramma di flusso dell'algoritmo

$C = [X_1, \ldots, X_N]$ |

$X_1 = Xmin$, $X_N = Xmax$



# Creare un vettore: linspace (3)

- MATLAB: su usa la funzione predefinita `linspace`:

  $C = [X_1, \ldots, X_N] \mid X_1 = Xmin, X_N = Xmax$

- Sintassi

```
C = linspace (Xmin, Xmax, N)
```

Esempio

```
>> C = linspace(0,20,11)
C = 0 2 4 6 8 10 12 14 16 18 20
```

NOTA: Se non si indica il numero N, MATLAB assume N=100

# Creare un vettore: funzioni built-in

- Vettore di N elementi di valore 0, 1, oppure casuale

B = [v1, v2, …, vN];
  vi=0 for i=1,..,N oppure
  vi=1 for i=1,..,N oppure
  vi=x dove x è un valore casuale (pseudorandomico)

- Il vettore è definito da:
    1. Funzione built-in: valore dell'elemento
    2. Il numero di elementi N

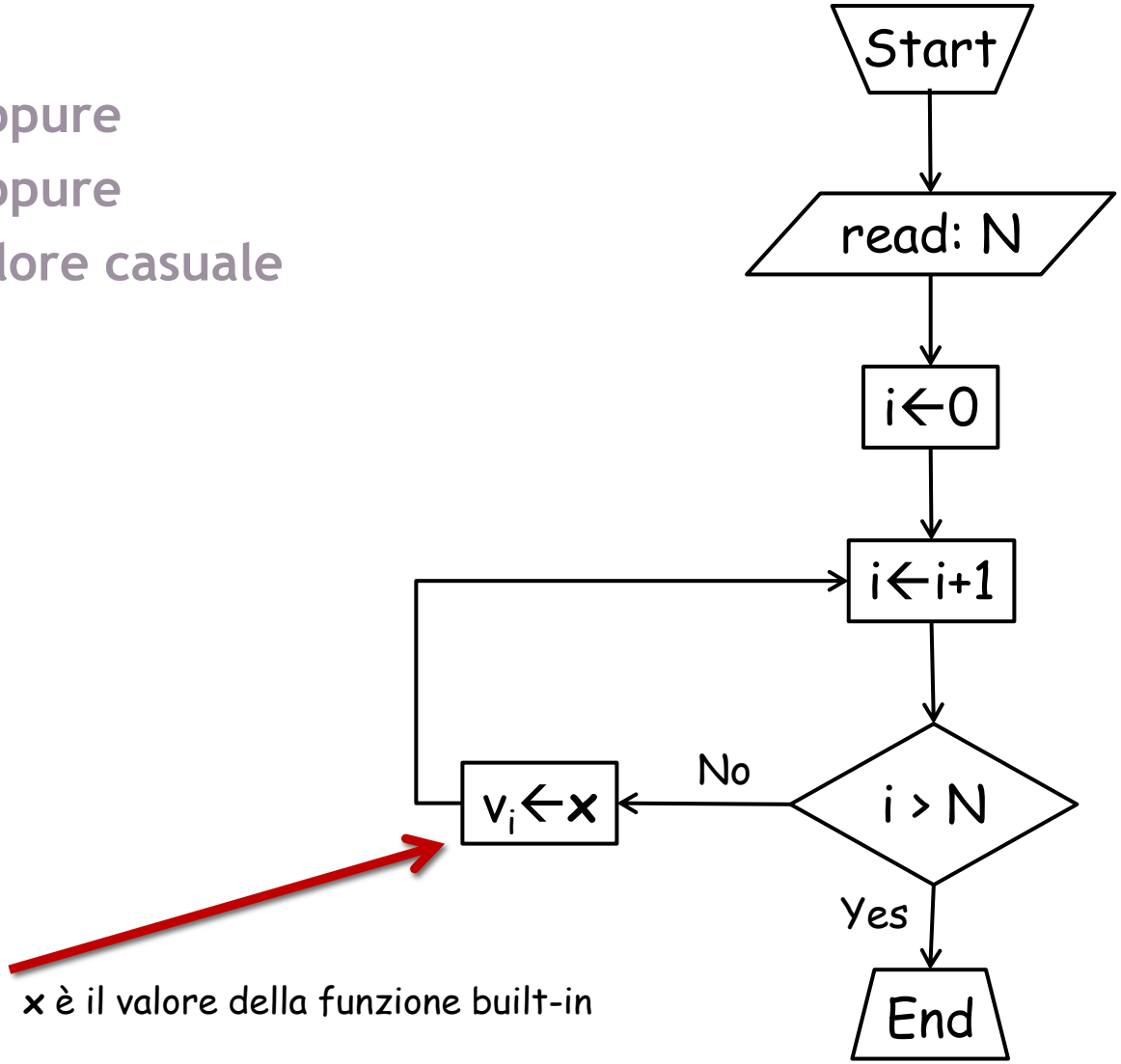
# Creare un vettore: funzioni built-in (2)

B = [v1, v2, …, vN];

vi=0 for i=1,..,N oppure

vi=1 for i=1,..,N oppure

vi=f dove f è un valore casuale



# Creare un vettore: funzioni built-in (3)

- MATLAB: si usano le funzioni predefinite

  `zeros(), ones(), rand()`

- Sintassi

```
A = zeros(1,N)
```

```
B = ones(1,N)
```

```
C = rand(1,N)
```

- Gli argomenti indicano il numero di righe e colonne

# Dimensioni (Size) di vettori e matrici

Problema: dato un vettore (matrice) vogliamo sapere il numero dei suoi elementi: le sue dimensioni.

MATLAB fornisce tre funzioni "built-in" per determinare le dimensioni dei vettori e delle matrici:

- **`size(A)`** quando applicato al vettore (matrice) A restituisce un vettore contenente due quantità: numero di righe e colonne
- **`length(A)`** restituisce il valore massimo tra le dimensioni (righe e colonne); se è un semplice vettore coincide con la sua lunghezza.
- **`numel(A)`** restituisce il numero degli elementi (per una matrice è il prodotto di righe e colonne)

# Esempio

```
>> A=rand(1,5)
A =
    0.4854    0.8003    0.1419    0.4218    0.9157
>> s=size(A)
s =
     1     5
>> whos s
  Name        Size                Bytes    Class     Attributes
  s           1x2                    16    double

>> b=rand(3,1)
b =
    0.7922
    0.9595
    0.6557

>> sb=size(b)
sb =
     3     1
```

# Esempio

- Verifichiamo che

” **length(A)** restituisce il valore massimo tra le dimensioni”

```
 >> confronto = length(A)==max(s)
confronto =
     1
>> whos confronto
  Name        Size    Bytes  Class     Attributes
  confronto   1x1     1      logical

>> length(A)
ans =
     5
```

# Accedere alle componenti (indici)

- Indica le modalità di accesso e modifica degli elementi di un vettore
- Gli elementi in un vettore sono numerati sequenzialmente, in MATLAB iniziano dal valore 1 (altri linguaggi da 0; es.: C++)
- Sintassi:
  - **`v(index)`** restituisce l'elemento (o gli elementi) alla(e) posizione(i) specificata(e) dall'*indice*
  - **`v(index)=value`** modifica gli elementi alla(e) posizione(i) specificata(e) dall'indice.
- Il vettore indice può contenere sia valori numerici che logici

```
>> A=1:3:12
A =
     1     4     7    10
>> A(1) %legge primo elemento
ans =
     1
>> B=[2,4]
B =
     2     4
>> A(B)
ans =
     4    10
```

B: vettore indice

# Un esempio di vettore indice (index vector)

```
>> A=randn(1,10) %crea A riempita con valori distribuiti normalmente
A =
  Columns 1 through 7
   8.8840e-01  -1.1471e+00  -1.0689e+00  -8.0950e-01  -2.9443e+00
1.4384e+00   3.2519e-01
  Columns 8 through 10
  -7.5493e-01   1.3703e+00  -1.7115e+00
>> A=round(A)
A =
     1    -1    -1    -1    -3     1     0    -1     1    -2
>> iv=2:2:10
iv =
     2     4     6     8    10
>> A(iv)
ans =
    -1    -1     1    -1    -2
>> A(2:2:10)
ans =
    -1    -1     1    -1    -2
```