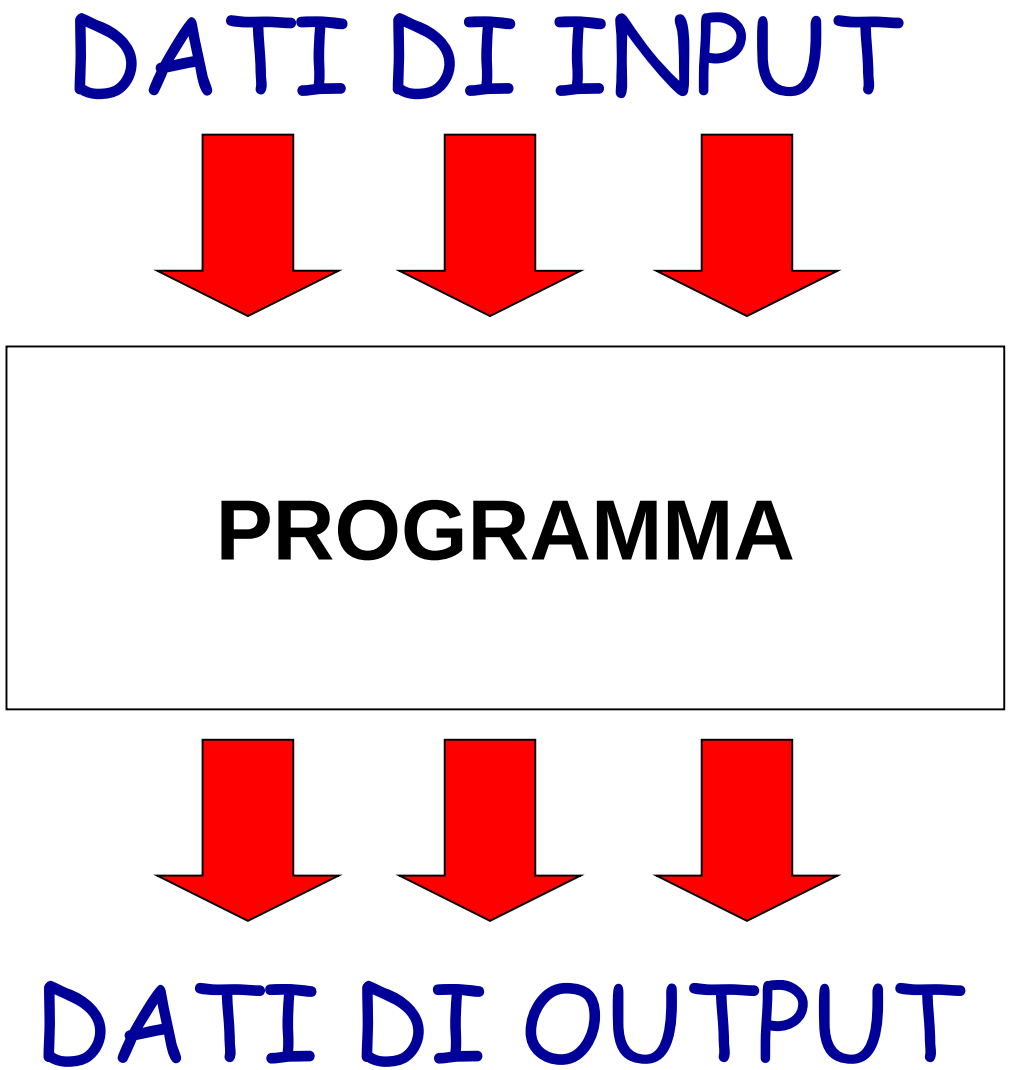
Prof. Davide Manca – Politecnico di Milano

Strumentazione e Controllo di Impianti Chimici

Tutorial

Introduzione a Matlab

PSE-Lab

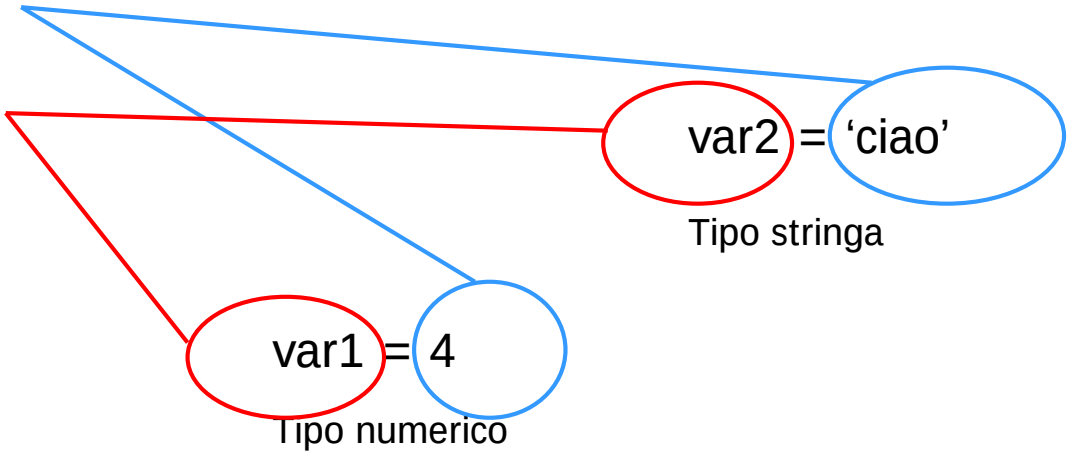

DATI DI INPUT
PROGRAMMA
DATI DI OUTPUT

Concetti fondamentali

- VARIABILE
- ISTRUZIONI FONDAMENTALI
- FUNZIONE

Variabili

- Le variabili sono caratterizzate da:
  - Il tipo (variabili numeriche, stringhe di caratteri,...)
  - Il valore
  - Il nome



Le variabili

- Matlab è **case-sensitive**: la variabile “pippo” è diversa dalla variabile “PiPpo”
- È bene usare la *camel notation*:

  laMiaNuovaVariabile

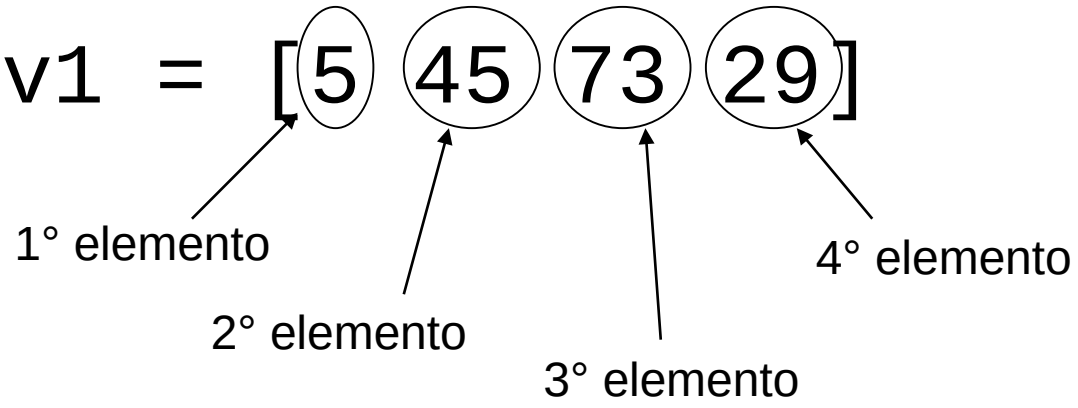
- È bene evitare variabili chiamate “l”, perché in Matlab la elle minuscola sembra un uno, il che rende difficile rileggere il codice e scovare gli errori

Vettori

- In Matlab un vettore si rappresenta come

```
v1 = [5 45 73 29]
```

- Gli elementi del vettore si contano a partire da 1:



Vettori

Per accedere all'i-esimo elemento del vettore `v1` si utilizza l'espressione:

```
v1(i)
```

Quindi per accedere al 3° elemento:

```
v1(3)
```

Vettori

- Creazione di un vettore con tutti gli elementi = 0

```
v1 = zeros(3)
      ⇒     v1  [0 0 0]
```

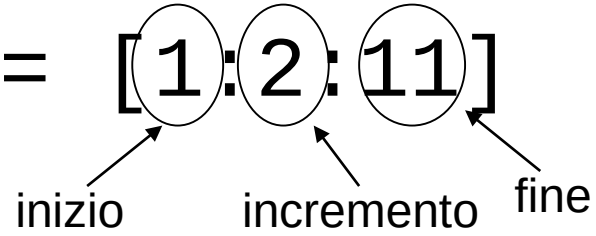
- Creazione di un vettore con tutti gli elementi = 1

```
v1 = ones(1,3)
      ⇒     v1  [1 1 1]
```

Vettori

- Vettore con elementi equispaziati

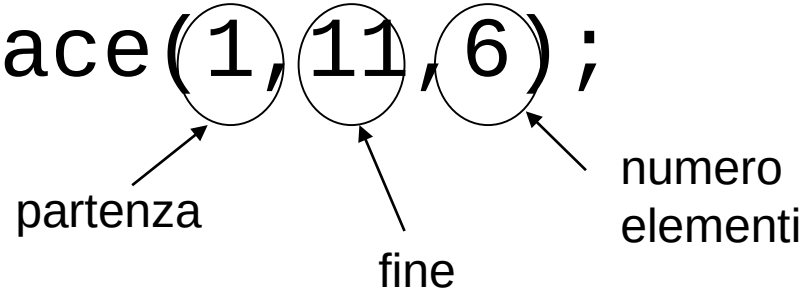
```
v1 = [1:2:11]
```



$\Rightarrow$ `[1 3 5 7 9 11]`

- Meglio usare l'istruzione

```
v1 = linspace(1,11,6);
```



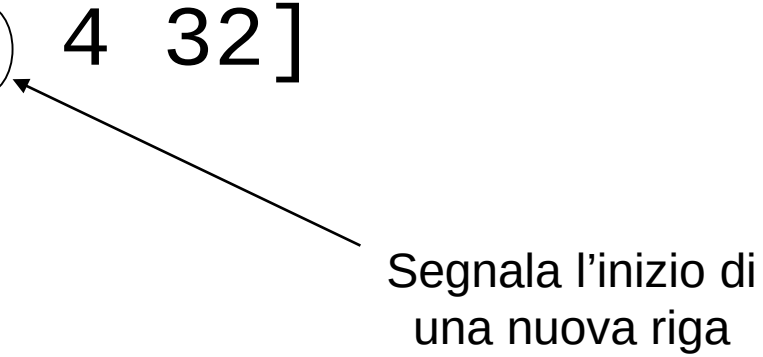
Matrici

- In Matlab una matrice si rappresenta come:

```
A = [17 15; 4 32]
```

Si ottiene la matrice:

```
A = [17 15
      4  32]
```



Matrici

Per accedere al j-esimo elemento della riga k:

```
A(k,j)
```

Ad esempio il 2° elemento della 1° riga è:

```
A(1,2)
```

E si ottiene

```
A = [17  15          ⇒ 15
      4  32]
```

Dimensioni

- Per conoscere le dimensioni di un vettore si utilizza il comando:

```
dim = length(v1)
```

- Per conoscere le dimensioni di una matrice si utilizza il comando:

```
[nRig,nCol] = size(A)
```

Copia di vettori e matrici

- Un vettore (o matrice) a può essere duplicato, copiato in un altro vettore (o matrice) b con l'istruzione: `b = a`

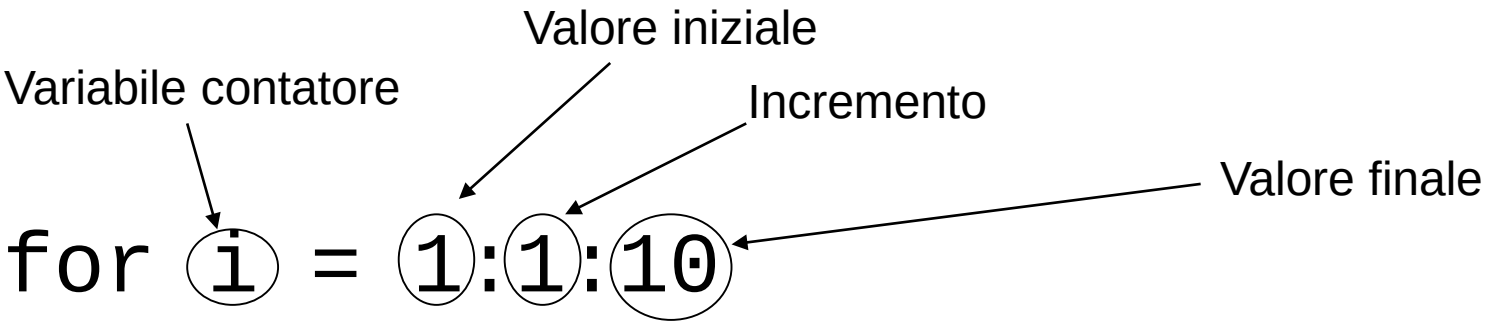
  Non serve, cioè, copiare tutti gli elementi uno ad uno

Costrutti

- Ciclo **FOR**
- Ciclo **WHILE**
- Costrutto **IF – ELSE – ELSE IF**

Ciclo FOR

- Il ciclo FOR è un'istruzione che permette di eseguire un certo numero di volte una serie di comandi.



Ciclo FOR - Esempio

- Sommare i numeri da 1 a 100:

```
somma = 0.;
for i = 1:100
    somma = somma + i;
end
```

Ciclo WHILE

- Il ciclo while è un ciclo che ripete le istruzioni al suo interno fino a che la condizione è vera

```
while(condizione)
    …… istruzioni
end
```

Ciclo WHILE - Esempio

- Sommare i numeri interi a partire da 1 fino a che la loro somma non sia maggiore o uguale a 325

```
somma = 0;
cont = 0;
while(somma < 325)
    cont = cont + 1;
    somma = somma + cont;
end
disp([‘Numero iter : ’,num2str(cont)]);
```

IF – ELSE – ELSE IF

- Le istruzioni vengono svolte solo se la condizione è vera

```
if(condizione)
    …… istruzioni
elseif(condizione)
    …… istruzioni
 elseif(condizione)
    …… istruzioni
……
else
    …… istruzioni
end
```

IF – ELSE – ELSE IF - Esempio

- Calcolare il valore del modulo di un numero

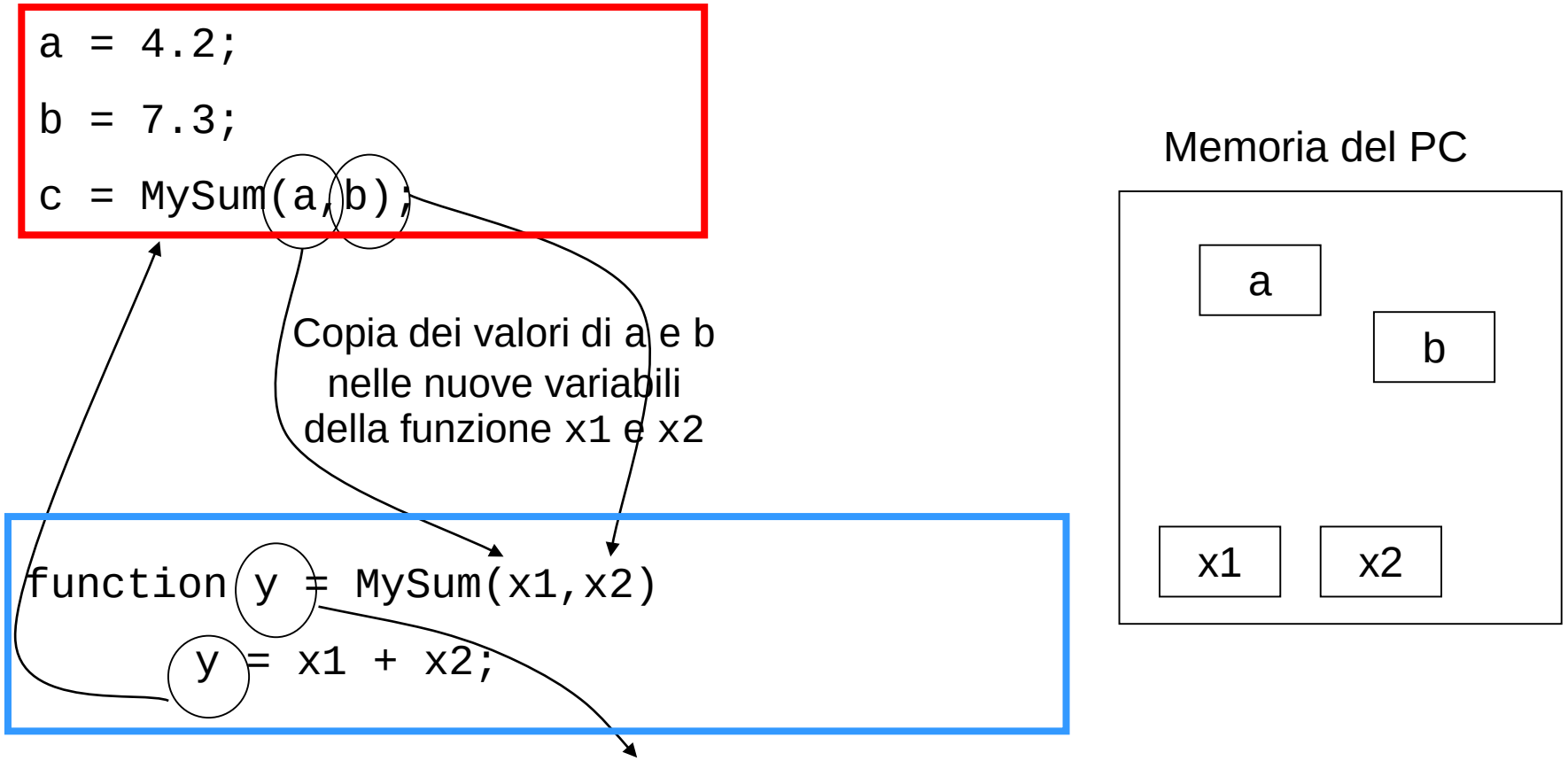
```
x = 45.;
if(x >= 0.)
   valoreAssoluto = x;
else
   valoreAssoluto = -x;
end
```

Le Funzioni

- Le funzioni sono blocchi di codice che svolgono un particolare compito.

Le Funzioni

Le funzioni sono blocchi di codice che svolgono un particolare compito.



In y viene salvato il risultato dell'operazione eseguita dalla funzione `MySum`. La variabile y viene restituita al codice che ha chiamato la funzione `MySum`

Note

- Quando Matlab entra nella funzione MySum, le variabili note sono SOLO quelle presenti nella funzione o passate alla funzione

Note

- Quando Matlab entra nella funzione MySum, le variabili note sono SOLO quelle presenti nella funzione o passate alla funzione

```
c = MySum(a,b);
```

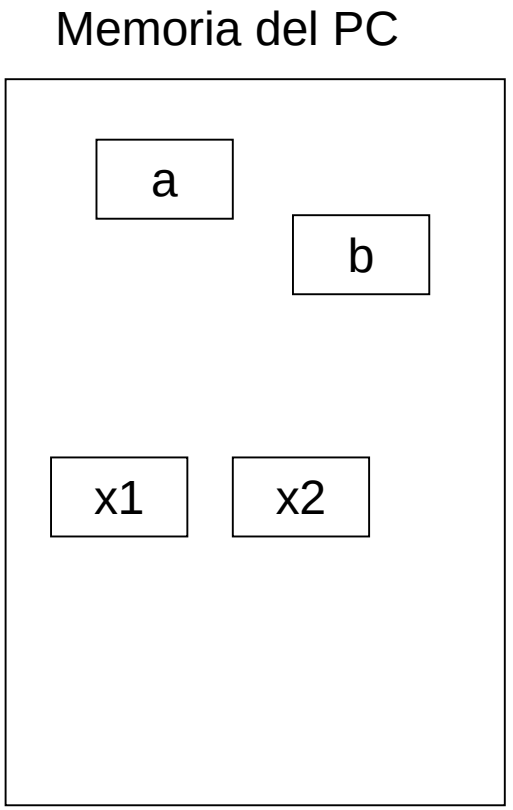
`a,b` = variabili passate alla funzione MySum

```
function y = MySum(x1,x2)
```

`x1,x2` = nome delle variabili passate a MySum e utilizzabili all'interno della funzione

`y` = valore restituito da MySum alla funzione che l'ha chiamata

- Se in MySum eseguiamo delle operazioni su x1 e x2 alterandone il valore, il valore di a e b rimane sempre lo stesso. Questo perché x1 e x2 non occupano lo stesso spazio di memoria di a e b



Le funzioni

- Le funzioni vanno salvate su file separati dal codice principale
- Il file deve avere lo stesso nome della funzione che è in esso contenuta
- il nome del file non può contenere spazi o caratteri di punteggiatura (!,?,;)

Funzioni

- Utilizzo delle funzioni:
  - ode45, ode15s
  - fzero, fsolve
  - interp
  - find

- Creazione di grafici

Risoluzione di equazioni differenziali

```
options = odeset('RelTol',1E-8,'AbsTol',1E-12);
[t,y] = ode15s(@(t,y)Sisdif(t,y,p),tSpan,ci,options);
```

- `y` = matrice delle variabili dipendenti. Ogni colonna corrisponde a una variabile
- `p` = elenco di parametri
- `tSpan` = vettore di tempi a cui si vuole conoscere la `y`
- `ci` = vettore delle condizioni iniziali
- `Sisdif` = nome della funzione che contiene il sistema differenziale (uguale al nome del file)

Risoluzione di equazioni differenziali

```
function dy = Sisdif(t,y,p)
  dy = zeros(nEq,1);

  dy(1) = …

  …

  dy(nEq) = …
```

Azzeramento

Per azzerare una sola equazione:

```
[x,fval,exitflag] = fzero(@(x)Func(x,p),x0);
```

`x` = soluzione

`x0` = valore di primo tentativo

`p` = parametri

`fval` = valore della funzione in x

`exitflag` = codice di errore (vedi prossima slide)

Azzeramento: exitflag

- 1 : Function converged to a solution x
- -1: Algorithm was terminated by the output function
- -3: NaN or Inf function value was encountered during search for an interval containing a sign change
- -4: Complex function value was encountered during search for an interval containing a sign change
- -5: fzero might have converged to a singular point

Azzeramento

- Per azzerare un sistema di equazioni:

```
[x,fval,exitflag] = ...
  fsolve(@(x)Func(x,p),x0);
```

`x` = vettore delle soluzioni

`p` = parametri

`fval` = vettore dei valori della funzione in `x`

`exitflag` = codice di errore (vedi prossima slide)

Azzeramento

Per fzero

```
function f = Funct(x,p)
  f = ...
```

Per fsolve

```
function f = Funct(x,p)
  f(1) = ...
  f(nEq) = ...
```

Interpolazione

Siano note le coppie:

```
(x , y)
```

e si voglia calcolare il valore dell'interpolante in `xi`:

```
yi = interp1(x,y,xi,method)
```

La variabile `method` può essere:

- `'linear'` : Linear interpolation (default)
- `'spline'` : Cubic spline interpolation

find

Sia dato il vettore

```
a = [14. 0.5 2. 29. 1.];
i = find(a > 3.);
```

restituisce gli indici `i` nel vettore `a` dove il valore soddisfa la condizione `a(i) > 3.`

In questo caso:

```
i = [1 4];
```

Creazione di grafici

```
nf = nf + 1;
figure(nf)
plot(x1,y1,'k-*',x2,y2,'r-. ', ...
     'LineWidth',3);
set(gca,'FontSize',18)
xlabel('x [m]')
ylabel('y [kg]')
legend('Mod 1', 'Mod 2',1)
text(xText,yText,'testo')
    saveas(figure(nf),‘C:\MiaFigura.emf')
```

Lettura dati da file

```
A =  load('D:\Progetto.txt');

x1 = A(:,1);

x2 = A(:,2);

...

xnVar = A(:,nVar);
```