Funzioni, parte 2

Informatica (ICA) AA 2020 / 2021

Giacomo Boracchi

3 Novembre 2020

giacomo.boracchi@polimi.it

Esercizio

Scrivere un programma che chiede all'utente di inserire un numero positivo n (nel caso in cui il numero non è positivo ripetere inserimento) e verifica se questo è perfetto

Se n non è perfetto dice se è abbondante o difettivo e richiede un secondo numero intero positivo m e controlla se n ed m sono amici. Si stampa a schermo il risultato di questo controllo.

Un numero è perfetto se corrisponde alla somma dei suoi divisori, escluso se stesso (es. 6 è perfetto 1 + 2 +3 = 6)
Un numero è abbondante se è > della somma dei suoi divisori (es 15 è abbondante 1 + 3 +5 < 15), altrimenti difettivo (es 12 è difettivo, 1+2+3+4+6 > 12)
Due numeri a,b sono amici (o amicabili) se la somma dei divisori di a è uguale a b e viceversa (es 220 e 284)

Implemento diverse funzioni che richiamo

```
function n = inserisciInteroPositivo()
% function n = inserisciInteroPositivo()
%
% richiede all'utente di inserire un intero positivo
% e lo restituisce

function somma = calcolaSommaDivisori(n)
%function somma = calcolaSommaDivisori(n)
%
% calcola la somma di tutti i divisori di n escluso n

function [res, abb] = controllaSePerfetto(n)
% function [res, abb] = controllaSePerfetto(n)
%
% res = true se n è perfetto (uguale alla somma dei suoi
divisori escluso se stesso)
%  se res = false e abb = true/false se è abbondante o
difettivo

function res = controllaSeAmici(a,b)
% function res = controllaSeAmici(a,b)
%
% res = 1 se a è amico di b, 0 altrimenti
```

```
function [res, abb] = controllaSePerfetto(n)
% function [res, abb] = controllaSePerfetto(n)
%
%  res = true se n è perfetto
%
%  se res = false e abb = true/false se è abbondante o difettivo

s = calcolaSommaDivisori(n); % assegno ad s ed evito 2 chiamate
abb = []; % è necessario inizializzarla per quando res ==false

if (n == s)
    res = true;
else
    res = false;
    if n > s
        abb = true;
    else
        abb = false;
    end
end
```

```
function res = controllaSeAmici(a,b)
%
% function res = controllaSeAmici(a,b)
%
% res = 1 se a è amico di b, 0 altrimenti

if b == calcolaSommaDivisori(a) && a ==
calcolaSommaDivisori(b)
    res = true;
else
    res = false;
end
```

```
function somma = calcolaSommaDivisori(n)
%
%function somma = calcolaSommaDivisori(n)
%
% calcola la somma di tutti i divisori di n escluso n

somma = 0;
for ii = 1 : n / 2 % inutile procedere oltre a n/2.
    if (mod(n, ii) == 0)
        somma = somma + ii;
    end
end
```

Script

```
n = inserisciInteroPositivo();
[perf, abbond] = controllaSePerfetto(n);
if(perf == true)
    disp([num2str(n), ' è perfetto']);
else
    disp([num2str(n), ' NON è perfetto']);
    if(abbond == true)
        disp([num2str(n), ' è abbondante']);
    else
        disp([num2str(n), ' è difettivo']);
    end
    m = inserisciInteroPositivo();
    amici = controllaSeAmici(n,m);
    if(amici)
        disp([num2str(n), ' e ', num2str(m), ' sono amici'])
    else
        disp([num2str(n), ' e ', num2str(m), ' NON sono amici'])
    end
end
```

Funzioni per Stringhe e Return

Funzioni per Stringhe

Esiste la funzione di **confronto**

```
TF = strcmp(str1 , str2)
```

- INPUT: **`str1`**, **`str2`** stringhe da confrontare
- OUTPUT: **`TF`** valore booleano o ,1 **(è diverso dal C)**
- Similmente **`strcmpi(str1, str2)`**non fa differenze tra maiuscole e minuscole

NB: in linea di principio è possibile confrontare le stringhe come due vettori, con l'operatore **==** . Questo però richiede che le **due stringhe abbiano le stesse dimensioni.** Altrimenti genera errori

- La funzione **`strcmp`** permette di confrontare anche stringhe di dimensione diverse (restituendo false).

Esempio

```
if('cane'== 'canguro')
 disp('uguali')
else
    disp('diverse')
End
>> Error using  ==
Matrix dimensions must agree.


if strcmp('cane','canguro')
    disp('uguali')
else
    disp('diverse')
end
>> diverse
```

Funzioni per Stringhe

Non occorre strlen (si usa length o size)

Non occorre strcpy (la copia tra stringhe è nativa in Matlab)

Esiste la funzione di **ricerca**

```
K = strfind(TEXT,PATTERN)
```

- INPUT: **PATTERN** stringa da ricercare in **TEXT**
- OUTPUT: **K** indice di tutte le occorrenze (vuoto se non ce ne sono)

Operazioni su Matrici

Sottoarray: Applicazione a Matrici

Si denota un sottoinsieme di un array usando vettori per valori degli indici

`nomeMatrice(vettore1,vettore2)`

restituisce una matrice che comprende gli elementi di **`nomeMatrice`** alle righe di indice in **`vettore1`** e alle colonne di indice in **`vettore2.`**

Sottoarray: Applicazione a Matrici

```
m =  9     8     7
     6     5     4
     3     2     1
     0    11    12
     0     0     0

>> m([1 4], [2 3])
```

Sottoarray: Applicazione a Matrici

```
m =  9     8     7
     6     5     4
     3     2     1
     0    11    12
     0     0     0
```

```
>> m([1 4], [2 3])
ans =   8     7
       11    12
```

tutti gli elementi sulle
righe 1 e 4 e sulle colonne 2 e 3

Sottoarray: Applicazione a Matrici

```
m =  9     8     7
     6     5     4
     3     2     1
     0    11    12
     0     0     0
```

```
>> m([1 4], [2 3])
ans =   8     7
       11    12
```

tutti gli elementi sulle
righe 1 e 4 e sulle colonne 2 e 3

```
>> m(1:2:5, 1:end)
```

Sottoarray: Applicazione a Matrici

```
m =  9     8     7
     6     5     4
     3     2     1
     0    11    12
     0     0     0
```

```
>> m([1 4], [2 3])
ans =   8     7
       11    12
```

tutti gli elementi sulle
righe 1 e 4 e sulle colonne 2 e 3

```
>> m(1:2:5, 1:end)
ans =   9     8     7
        3     2     1
        0     0     0
```

tutti gli elementi delle
righe 1, 3 e 5

Sottoarray: Applicazione a Matrici

```
m =  9     8     7
     6     5     4
     3     2     1
     0    11    12
     0     0     0

>> m(1:2:5, :)
ans =  9     8     7
       3     2     1
       0     0     0
```

notazione ‘:’ abbreviata per 1:end, cioè tutti i valori di quell’indice

Sottoarray: Applicazione a Matrici

```
    m = 9     8     7
        6     5     4
        3     2     1
        0    11    12
        0     0     0
>> m(1:2:5, :)
ans =  9     8     7
        3     2     1
        0     0     0

>> m(2:2:4, :) = [-1 -2 -3; -4 -5 -6]
m =    9     8     7
      -1    -2    -3
        3     2     1
       -4    -5    -6
        0     0     0
```

notazione ':' abbreviata per 1:end, cioè tutti i valori di quell'indice

uso della notazione dei sottoarray per individuare elementi oggetto di assegnamento

Assegnamenti con Scalari

È possibile associare a qualsiasi sotto array un valore scalare

`nomeVettore(vettoreIndici) = k`

Fa si che a tutti gli elementi di **`nomeVettore`** alle posizioni **`vettoreIndici`** venga assegnato il valore **`k`**

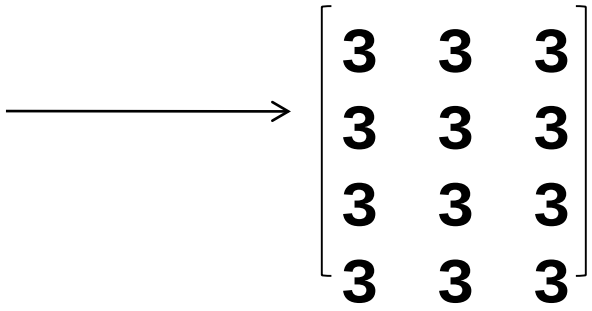
In questo modo è possibile inizializzare nuovi vettori.

```
» a = [1 : 10]
a =
     1     2     3     4     5     6     7     8     9    10

» a(1 : 3) = 0
a =
     0     0     0     4     5     6     7     8     9    10
```

Assegnamenti con Scalari

Esempio

```
m(1:4, 1:3) = 3
```

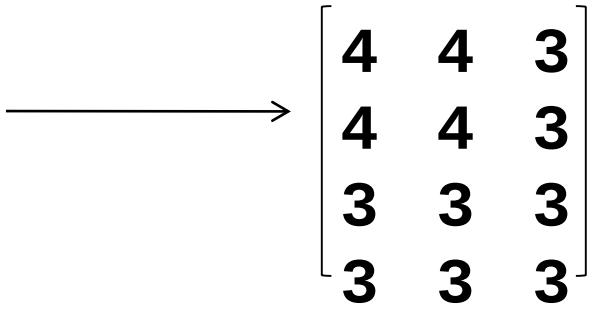


$$\begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$

Il modo con cui uno scalare viene assegnato a un array dipende dalla forma dell'array che viene specificata a sinistra dell'assegnamento

Esempio

```
m(1:2, 1:2) = 4
```



$$\begin{bmatrix} 4 & 4 & 3 \\ 4 & 4 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$

Esempio

```
% inizializzare una matrice 5x5 con tutti valori a zero

% modificare la colonna centrale in 1

% modificare la riga centrale in 3

% sommare 2 ai valori della colonna centrale

% porre a 2 gli elementi nel primo quadrante

% copiare nell'ultima riga la prima riga letta al
contrario
```

Esempio

```
% inizializzare una matrice 5x5 con tutti valori a zero
A(5,5) = 0;
% modificare la colonna centrale in 1
A(: , 3) = 1;
% modificare la riga centrale in 3
A(3 , : ) = 3;
% sommare 2 ai valori della colonna centrale
A(: , 3) = A(: , 3) + 2; % NB termini a dx e sx
dell'uguale hanno la stessa dimensione
% porre a 2 gli elementi nel primo quadrante
A(1 : 2 , 1 : 2) = 2;
% copiare nell'ultima riga la prima riga letta al
contrario
A(end , :) = A(1 , end : -1 : 1)
```

Assegnamenti con Scalari su matrici

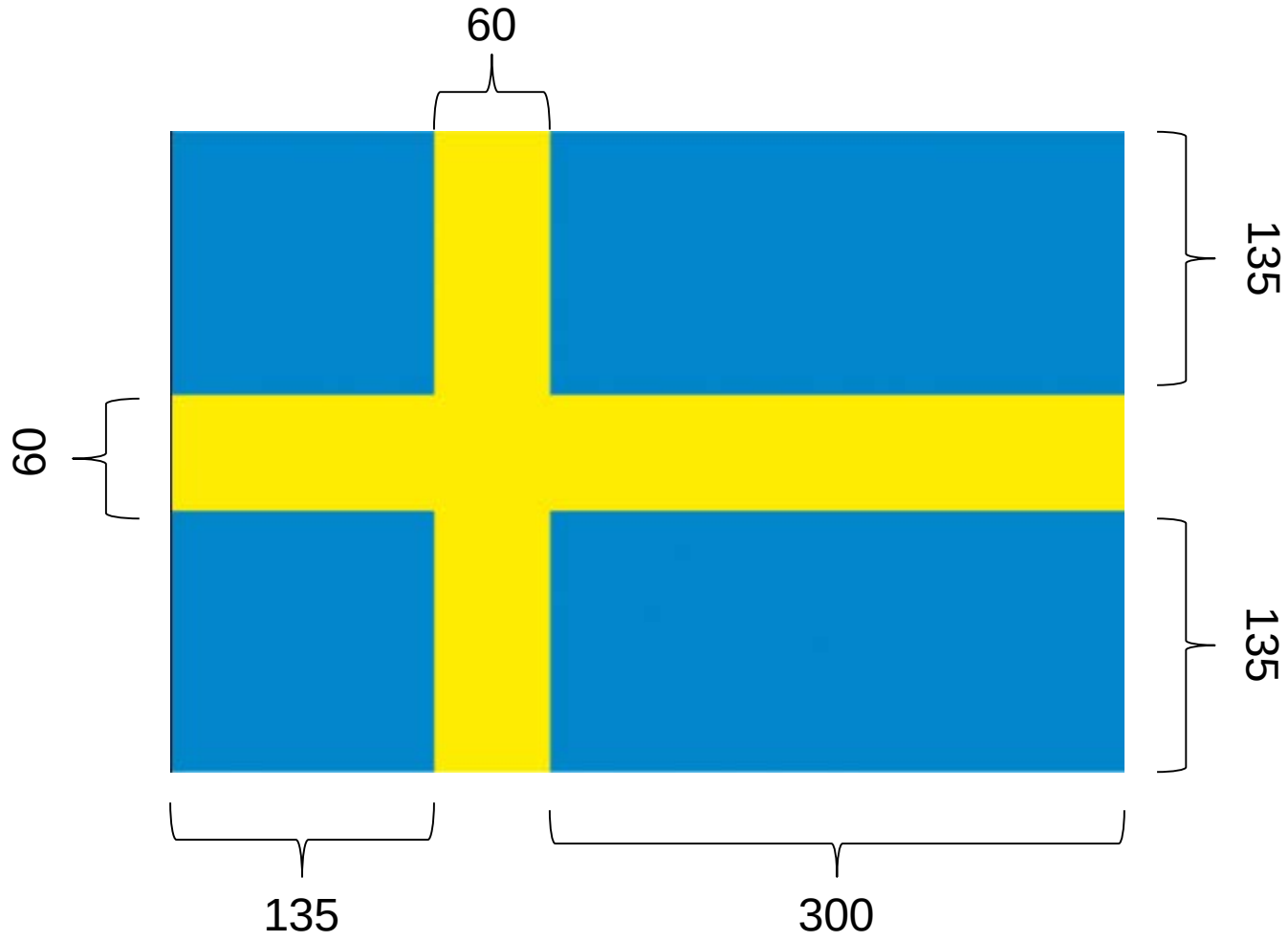
Il modo con cui uno scalare viene assegnato a un array dipende dalla forma dell'array che viene specificata a sinistra dell'assegnamento

Es.

```
>> clear m;
>> m(4, 3) = 3;
>> m(1:2, 1:2) = 4
ans =
     4     4     0
     4     4     0
     0     0     0
     0     0     3
```

Esercizio

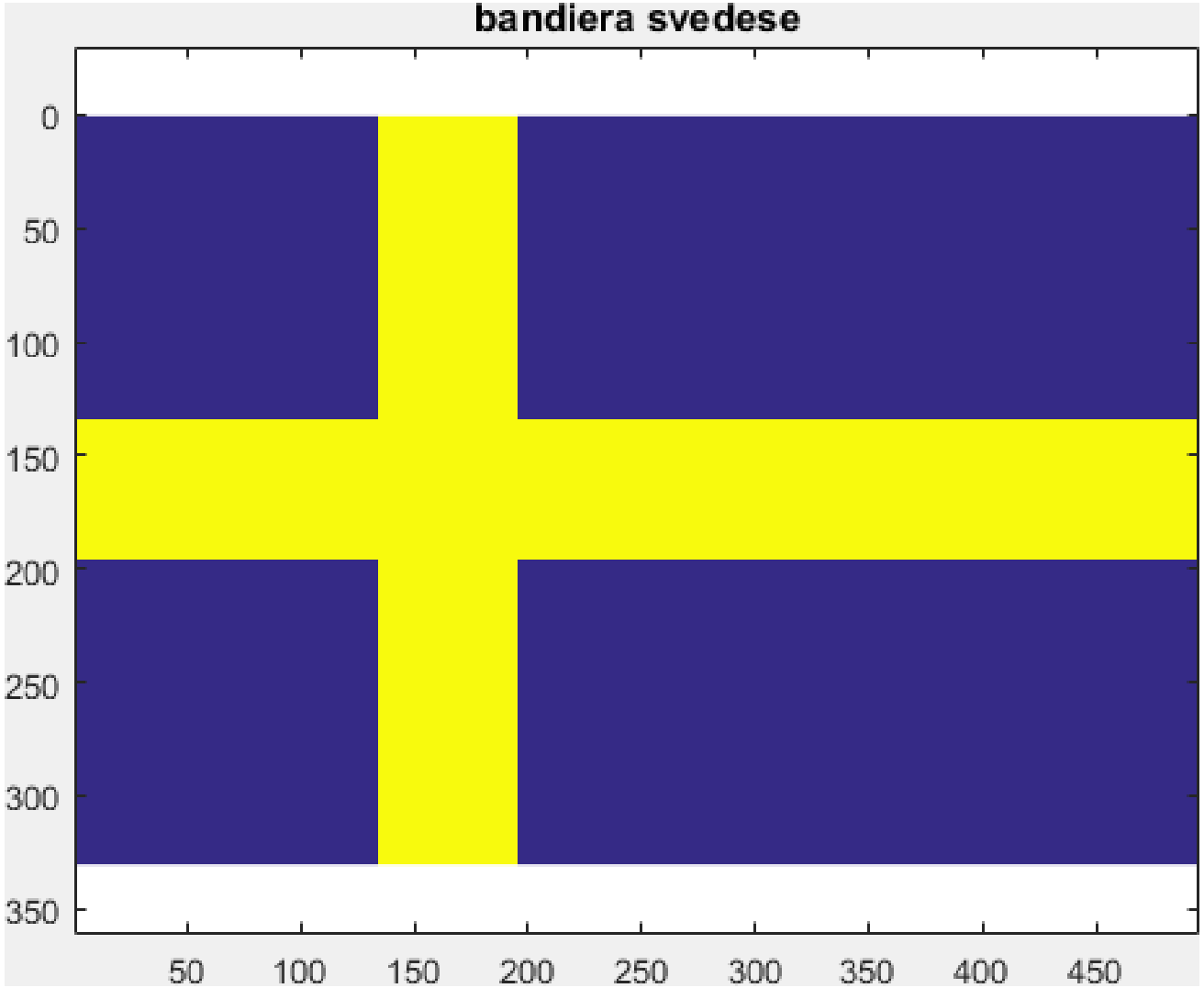
Disegnare la bandiera svedese



Bandiera Svedese

```
clear;
clc;
close all;

A(330, 495) = 0;
A(:, 135:195) = 1;
A(135:195, :) = 1;

figure();
imagesc(A);
title('Bandiera svedese');
axis equal;
```

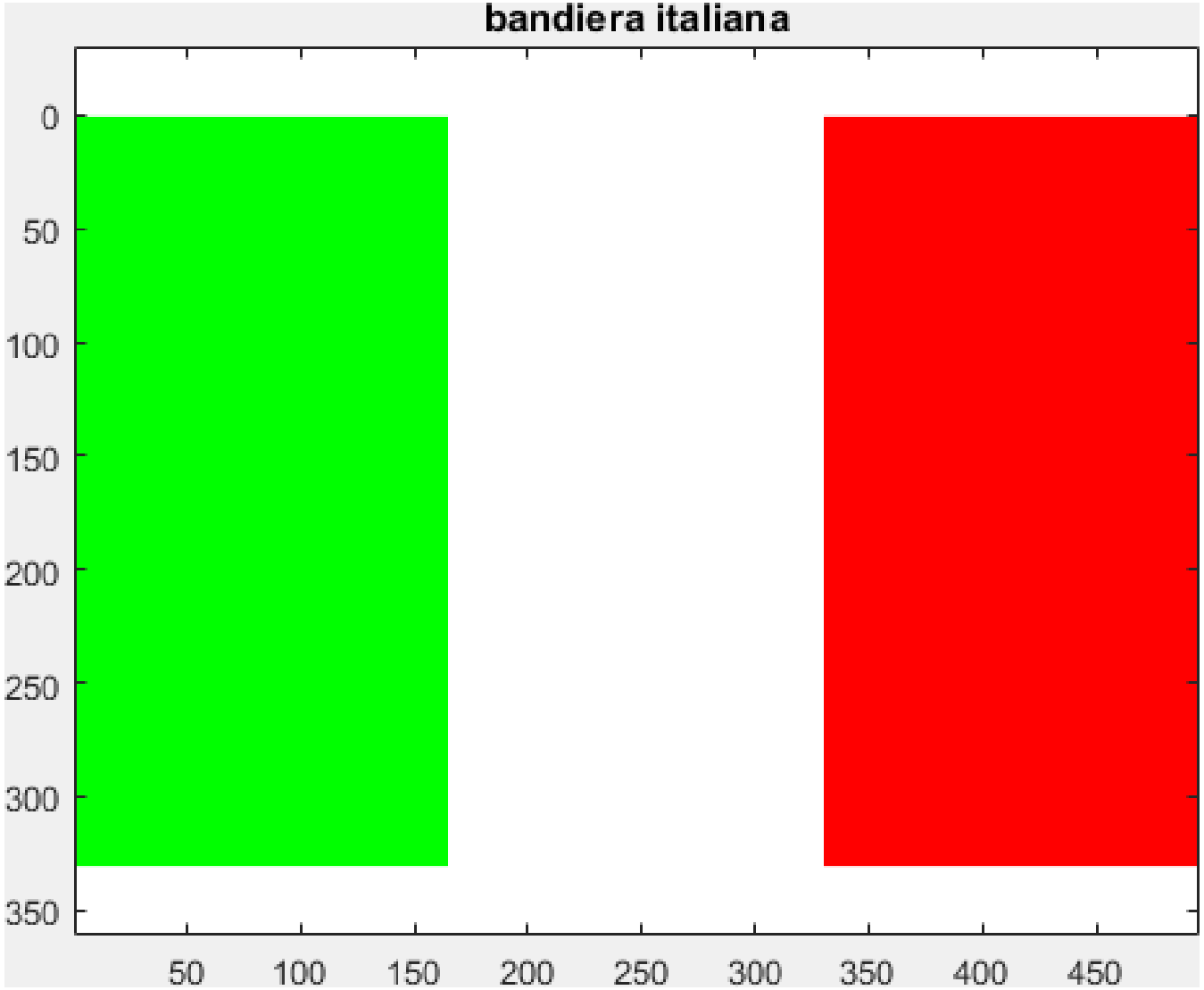
Ecco il Risultato (immagine 2D)



Bandiera Italiana (immagine 3D)

```
clear;
clc;
close all;

A(330, 495, 3) = 0;
A(:, 1 : 495/3, 2) = 1; % verde
A(:, 495/3 : (2*495)/3, :) = 1; % bianco
A(:, (2*495)/3 : end, 1) = 1; % rosso
A = uint8(255 * A);
figure(); imagesc(A);
title('Bandiera italiana');
axis equal;
```

Ecco il Risultato



Come creare le immagini a colori:

- E' possibile riprodurre i colori in due modi:
  - creando una matrice 3D B che ha N_righe x N_colonne x 3 dove il terzo piano indica il colore (B(:,:,1) è il rosso, B(:,:2) il verde, B(:,:,3) il blu) e visualizzando con imshow
  - creando una matrice 2D e modificando la colormap
  - **Cercate su internet i colori corretti in RGB e ricordate di convertire in uint8 prima di visualizzare la matrice**

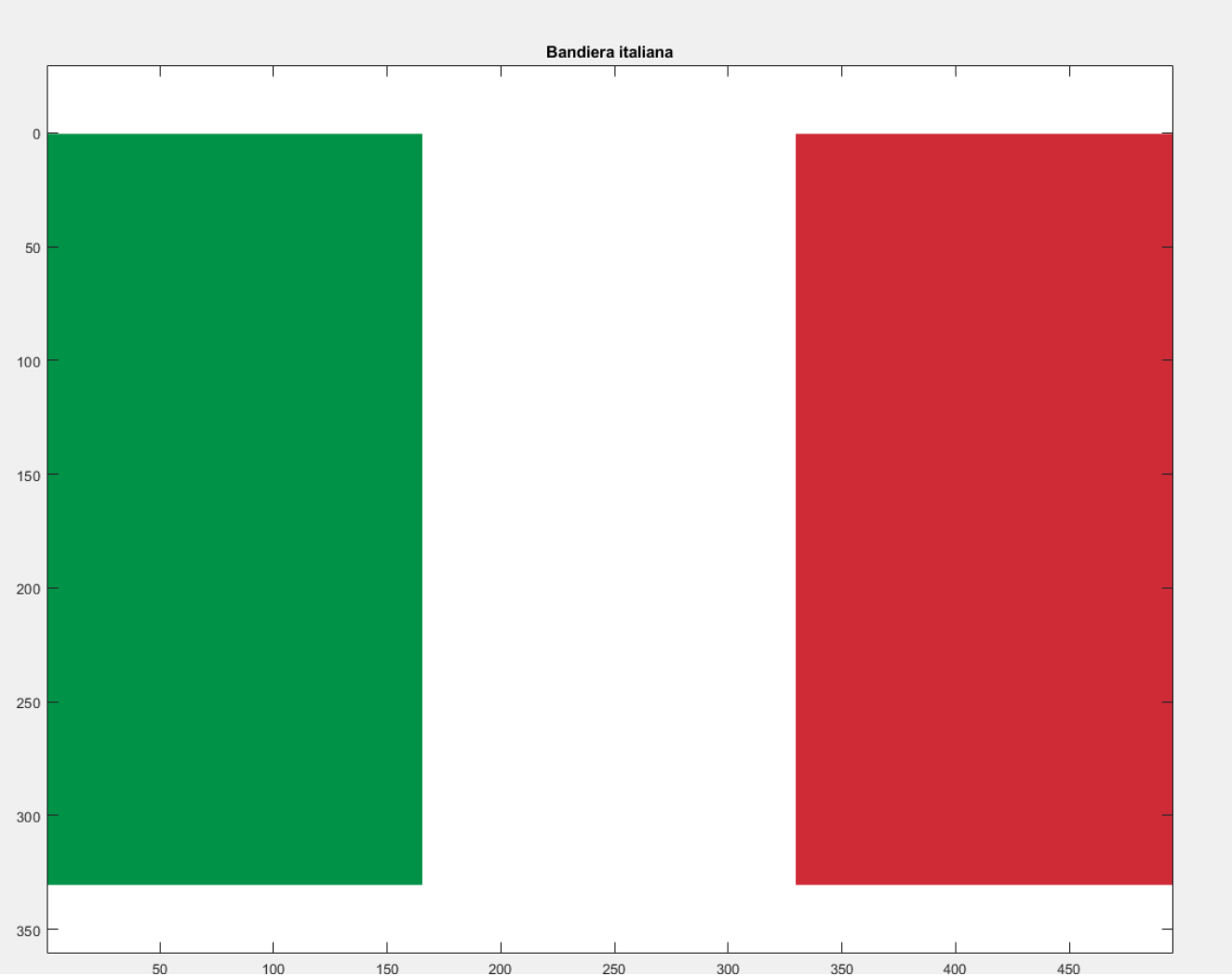
Colors in Palette

Color	Hex	RGB
	#f2f2f2	(242,242,242)
	#009246	(0,146,70)
	#ffffff	(255,255,255)
	#ce2b37	(206,43,55)
	#f2f2f2	(242,242,242)

Uso i colori corretti

```
A = ones(330, 495, 3) * 255 ;

A(:, 1 : 495/3, 1) = 0; % R: banda
verde

A(:, 1 : 495/3, 2) = 146; % G: banda
verde

A(:, 1 : 495/3, 3) = 70; % B: banda
verde


A(:, (2*495)/3 : end, 1) = 206; % rosso

A(:, (2*495)/3 : end, 2) = 43; % rosso

A(:, (2*495)/3 : end, 3) = 55; % rosso


A = uint8(A);

figure(); imagesc(A);
```

Bandiera italiana con i colori corretti



GRAN CONTEST DELLE BANDIERINE

Regole:

- La bandiera deve essere realizzata interamente con uno script MatLab, senza interazione con l'utente
- È necessario usare operazioni vettoriali, si possono usare cicli
- Verrà valutata sia la veridicità della bandiera, sia la struttura del codice utilizzato per realizzarla
- Potete presentare una sola bandiera a persona
- Inviate via mail a stefano1.marelli@polimi.it cc: giacomo.boracchi@polimi.it
  - un'immagine (png) della bandiera
  - lo script che la genera (opportunamente indentato e commentato)
- **Deadline: Domenica 22 Novembre 2020 (23.59 UTC+1)**

GRAN CONTEST DELLE BANDIERINE

Premi:

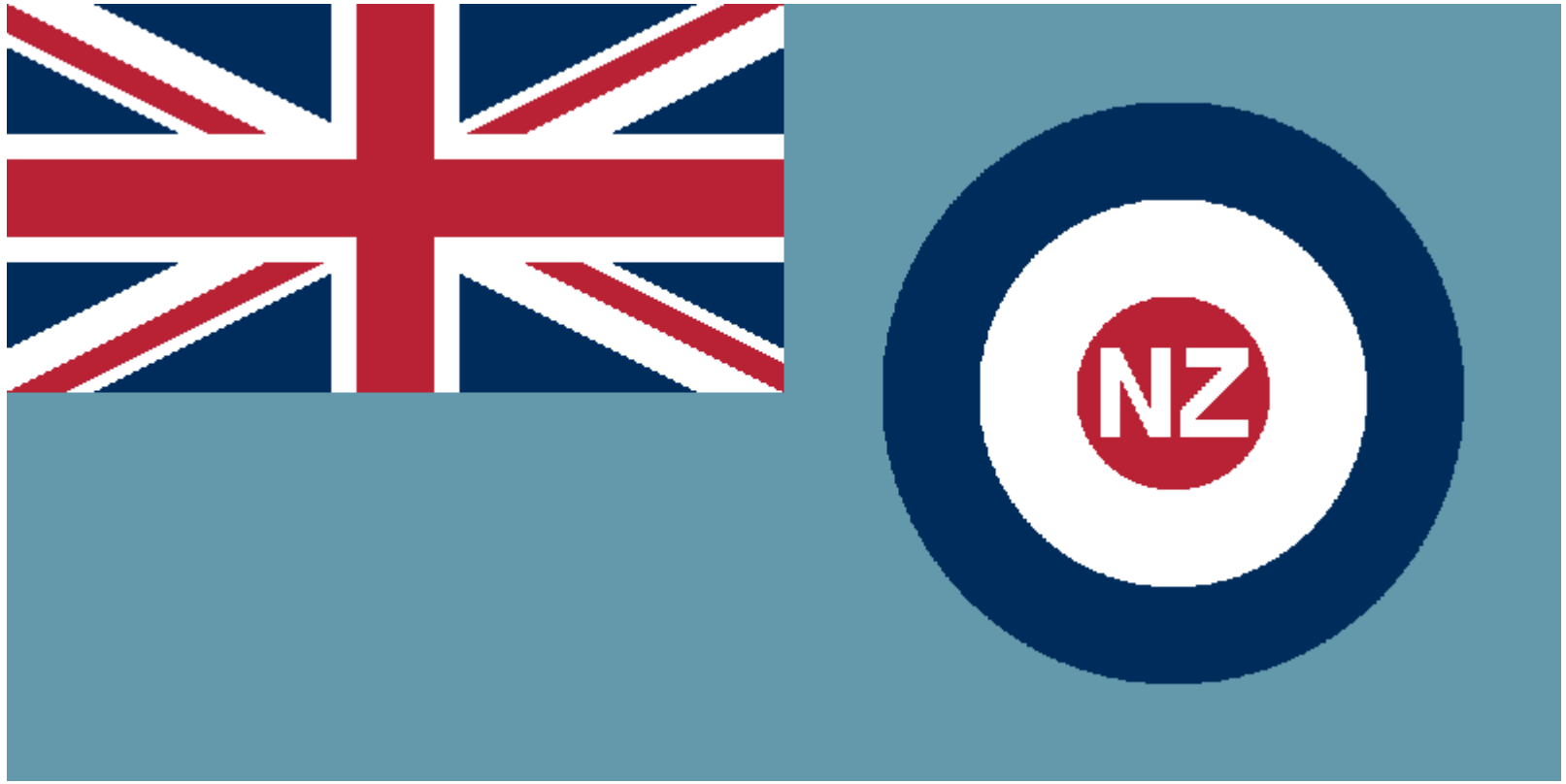
Tre punti alla bandierina/codice migliore e premi a scalare (a patto che lo studente dimostri di aver capito il codice scritto)

Deadline: Domenica 22 Novembre 2020 (23.59 UTC+1)

Esempi di bandiere realizzate dai vostri colleghi…

Bandiera della Catalogna

Esempi di bandiere realizzate dai vostri colleghi…



New Zealand Airforce