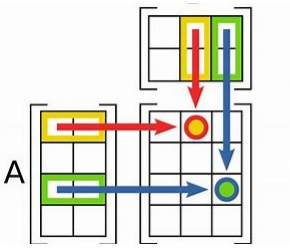
ITIS-LS “Francesco Giordani” Caserta

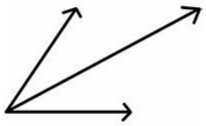
prof. Ennio Ranucci

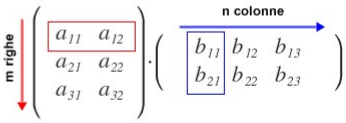
a.s. 2019-2020

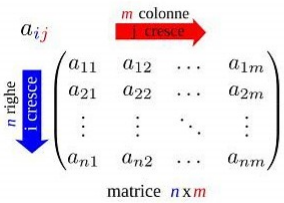
Vettori e Matrici

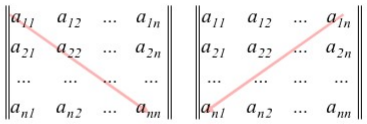
C++ Python Octave











Somma e differenza tra vettori

Siano $\vec{u} = (u_1, u_2, ..., u_n)$ e $\vec{v} = (v_1, v_2, ..., v_n)$ due vettori di $\mathbb{R}^n$.

1) La **somma tra vettori** è un'operazione interna

$$+ : \ \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$

$$(\vec{u}, \ \vec{v}) \mapsto \vec{u} + \vec{v}$$

che associa ai vettori $\vec{u}$ e $\vec{v}$ un nuovo vettore, detto *vettore somma*, le cui componenti sono date dalla somma delle componenti di $\vec{u}$ e di $\vec{v}$

$$\vec{u} + \vec{v} = (u_1 + v_1, \ u_2 + v_2, \ ..., \ u_n + v_n)$$

2) L'operazione di **differenza tra vettori** è una legge che associa ai vettori $\vec{u}$ e $\vec{v}$ il *vettore differenza* $\vec{u} - \vec{v}$ le cui componenti sono date dalla differenza delle rispettive componenti.

$$\vec{u} - \vec{v} = (u_1 - v_1, \ u_2 - v_2, \ ..., \ u_n - v_n)$$

Esempio di somma e di differenza tra vettori

Calcolare la somma e la differenza tra i vettori

$\vec{u} = (1, -5, 7, 4)$

$\vec{v} = (-2, 8, -12, 3)$

Svolgimento:

$\vec{u} + \vec{v} = (u_1 + v_1, \ u_2 + v_2, \ u_3 + v_3, \ u_4 + v_4) =$

$= (1 + (-2), \ -5 + 8, \ 7 + (-12), \ 4 + 3) = (-1, 3, -5, 7)$

$\vec{u} - \vec{v} = (u_1 - v_1, \ u_2 - v_2, \ u_3 - v_3, \ u_4 - v_4) =$

$= (1 - (-2), \ -5 - 8, \ 7 - (-12), \ 4 - 3) = (3, -13, 19, 1)$

ITI-LS "Francesco Giordani" Caserta
Anno scolastico: 2019/2020
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: es1
Versione: 1.0
Programmatore/i:
Sistema Operativo: Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: Somma di due vettori
Obiettivo del programma: Sommare due vettori

```
#include<iostream>
using namespace std;
int const dimFis=30;
int dimLog;
int vet1[dimFis]; int vet2[dimFis]; int vet3[dimFis];
int ottieniDim(int dimFisPar);
void caricaVet(int vetPar[], int dimLogPar);
void stampaVet (int vetPar[], int dimLogPar);
void sommaVettori(int vet1Par[], int vet2Par[], int vet3Par[], int dimLogPar);
int main()
{
   dimLog=ottieniDim(dimFis);
   caricaVet(vet1, dimLog);
   caricaVet(vet2, dimLog);
   stampaVet(vet1, dimLog);
   stampaVet(vet2, dimLog);
   sommaVettori(vet1, vet2, vet3, dimLog);
   stampaVet(vet3, dimLog);
   return 0;
}
int ottieniDim(int dimFisPar)
{
   int dim;
   do
   {
      cout<<"Inserisci il numero di elementi: ";
      cin>>dim;
   }
   while((dim<0)||(dim>dimFisPar));
   return dim;
}
```

```
void caricaVet(int vetPar[], int dimLogPar)
{
  for(int i=0; i<dimLogPar; i++)
    {
          cout<<"Inserisci il "<<i<<" elemento: ";
          cin>>vetPar[i];
    }
}
void stampaVet (int vetPar[], int dimLogPar)
{
  for(int i=0; i<dimLogPar; i++) cout<<vetPar[i]<<" ";
  cout<<endl;
}
void sommaVettori(int vet1Par[], int vet2Par[], int vet3Par[], int dimLogPar)
{
  for(int i=0; i<dimLogPar; i++)
    {
       vet3Par[i]=vet1Par[i]+vet2Par[i];
    }
}
```

Prodotto di un vettore per uno scalare

Siano $\lambda \in \mathbb{R}$ un numero reale, che d'ora in poi diremo *scalare*, e $\vec{v} = (v_1, v_2, ..., v_n)$ un vettore di $\mathbb{R}^n$.

Il **prodotto di uno scalare per un vettore** è un'operazione esterna

$$\mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$$

$$(\lambda, \vec{v}) \mapsto \lambda\vec{v}$$

che alla coppia $(\lambda, \vec{v})$ associa un nuovo vettore, le cui componenti si ottengono moltiplicando ogni componente di $\vec{v}$ per lo scalare λ.

$$\lambda\vec{v} = \lambda(v_1, v_2, ..., v_n) = (\lambda v_1, \lambda v_2, ..., \lambda v_n)$$

Esempio di prodotto di un vettore per uno scalare

Siano $\vec{v} = (-1, 3, 5, -9)$ e $\lambda = -2$. Calcolare il prodotto tra il vettore $\vec{v}$ e lo scalare λ.

Svolgimento:

$\lambda\vec{v} = -2(-1, 3, 5, -9) =$

$= (-2(-1),\ -2(3),\ -2(5),\ -2(-9)) = (2, -6, -10, 18)$

ITI-LS "Francesco Giordani" Caserta
Anno scolastico: 2019/2020
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: es2
Versione: 1.0
Programmatore/i:
Sistema Operativo: Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: Prodotto di un vettore per uno scalare
Obiettivo del programma: Scrivere il sottoprogramma (da aggiungere all'es1) che restituisce il prodotto di un vettore per uno scalare

```
void prodottoVettoreXScalare(int vet1Par[], int vet2Par[],int dimLogPar, int scalarePar)
{
   for(int i=0; i<dimLogPar; i++)
   {
      vet2Par[i]=vet1Par[i]*scalarePar;
   }
}
```

Prodotto scalare tra due vettori è una particolare operazione binaria che restituisce un numero.

ITI-LS "Francesco Giordani" Caserta
Anno scolastico: 2019/2020
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: es2.1
Versione: 1.0
Programmatore/i:
Sistema Operativo: Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: Prodotto scalare di due vettori
Obiettivo del programma: Scrivere il sottoprogramma (da aggiungere all'es1) che restituisce il prodotto scalare di due vettori

```
#include<iostream>
using namespace std;
int const dimFis=30;
int dimLog;
int vet1[dimFis]; int vet2[dimFis]; int vet3[dimFis];
int ottieniDim(int dimFisPar);
void caricaVet(int vetPar[], int dimLogPar);
void stampaVet (int vetPar[], int dimLogPar);
int prodottoScalareVettori(int vet1Par[], int vet2Par[], int dimLogPar);

int main()
{
   dimLog=ottieniDim(dimFis);
   caricaVet(vet1, dimLog);
   caricaVet(vet2, dimLog);
   stampaVet(vet1, dimLog);
   stampaVet(vet2, dimLog);
   cout<<prodottoScalareVettori(vet1,vet2,dimLog);
   return 0;
}

int ottieniDim(int dimFisPar)

{

   int dim;

   do

   {

      cout<<"Inserisci il numero di elementi: ";
```

```
        cin>>dim;
    }
    while((dim<0)||(dim>dimFisPar));
    return dim;
}
void caricaVet(int vetPar[], int dimLogPar)
{
  for(int i=0; i<dimLogPar; i++)
    {
            cout<<"Inserisci il "<<i<<" elemento: ";
            cin>>vetPar[i];
    }
}
void stampaVet (int vetPar[], int dimLogPar)
{
  for(int i=0; i<dimLogPar; i++) cout<<vetPar[i]<<" ";
  cout<<endl;
}
int prodottoScalareVettori(int vet1Par[], int vet2Par[], int dimLogPar)
 {
   int prodottoScalare=0;
   for(int i=0; i < dimLogPar; i++)
    {
      prodottoScalare += vet1Par[i]*vet2Par[i];
    }
   return prodottoScalare;
}
```

Prodotto di una matrice per uno scalare

Indichiamo con k uno scalare.
Vogliamo ora moltiplicare lo scalare k per la matrice A.
Il PRODOTTO prende il nome di PRODOTTO SCALARE e si indica con
k · A
Il prodotto scalare è la MATRICE che si ottiene MOLTIPLICANDO ogni ELEMENTO di A per k.
Ovvero: k · A = [k · aij]

Esempio:

$$\alpha \cdot A = 3 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 3 \cdot 1 & 3 \cdot 2 \\ 3 \cdot 3 & 3 \cdot 1 \end{pmatrix}$$
$$= \begin{pmatrix} 3 & 6 \\ 9 & 3 \end{pmatrix}$$

ITI-LS "Francesco Giordani" Caserta
Anno scolastico: 2019/2020
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: es3
Versione: 1.0
Programmatore/i:
Sistema Operativo: Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: Prodotto di una matrice per uno scalare
Obiettivo del programma: Scrivere il programma che restituisce il prodotto di matrice per uno scalare.

```
#include<iostream>
using namespace std;
int const dimFis=30;
int rig, col;
int mat1[dimFis][dimFis]; int mat2[dimFis][dimFis];
int ottieniDim(int dimFisPar);
void caricaMat(int matPar[][dimFis], int rigPar,int colPar);
void stampaMat(int matPar[][dimFis], int rigPar, int colPar);
void prodottoMatriceXScalare(int mat1Par[][dimFis],int mat2Par[][dimFis], int rigPar, int colPar,
int scalarePar);
int main()
{
   cout<<"inserimento numero righe"<<endl;
   rig=ottieniDim(dimFis);
   cout<<"inserimento numero colonne"<<endl;
```

```
    col=ottieniDim(dimFis);
    caricaMat(mat1, rig, col);
    cout<<"stampa matrice "<<endl;
    stampaMat(mat1, rig, col);
    prodottoMatriceXScalare(mat1,mat2,rig, col,3);
    cout<<"stampa prodotto di una matrice per uno scalare"<<endl;
    stampaMat(mat2, rig, col);
    return 0;
}
int ottieniDim(int dimFisPar)
{
    int dim;

    do

    {

        cout<<"Inserisci il numero di elementi: ";

        cin>>dim;

    }

    while((dim<0)||(dim>dimFisPar));

    return dim;

}

void caricaMat(int matPar[][dimFis], int rigPar, int colPar)

{

  for(int i=0; i<rigPar; i++)

     for(int j=0; j<colPar; j++)

    {

            cout<<"Inserisci l'elemento di posizione: "<<i<<","<<j<<" ";

            cin>>matPar[i][j];

    }

}

void stampaMat (int matPar[][dimFis], int rigPar, int colPar)

{

 for(int i=0; i<rigPar; i++)
```

```
  {
    for(int j=0; j<colPar; j++) cout<<matPar[i][j]<<"\t";
    cout<<endl;
  }
}
void prodottoMatriceXScalare(int mat1Par[dimFis][dimFis],int mat2Par[dimFis][dimFis], int rigPar, int colPar, int scalarePar)
{
    for(int i=0; i<rigPar; i++)
      for(int j=0; j<colPar; j++)  mat2Par[i][j]=mat1Par[i][j]*scalarePar;
}
```

la somma di due matrici

Siano $A = (a_{ij})$ e $B = (b_{ij})$ due matrici dello stesso tipo con m righe e n colonne. La somma tra matrici restituisce una nuova matrice $A + B$ i cui elementi si ottengono sommando gli elementi delle matrici A e B che occupano la stessa posizione.

Indicando con s_{ij} gli elementi della matrice somma $A + B$ abbiamo che

$$s_{ij} = a_{ij} + b_{ij},\ \text{con}\ i \in \{1, 2, ..., m\},\ j \in \{1, 2, ..., n\}$$

Esempi di somma tra matrici

1) Sommare le matrici

$$A = \begin{pmatrix} 2 & 5 & -3 \\ 1 & -2 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 7 & -5 & 2 \\ -9 & 4 & -1 \end{pmatrix}$$

Svolgimento: innanzitutto verifichiamo se è possibile svolgere l'operazione. Sia A che B hanno 2 righe e 3 colonne, quindi si possono sommare e gli elementi della matrice somma si ottengono dalla somma tra gli elementi di A e di B che occupano la stessa posizione.

$$A + B = \begin{pmatrix} 2 & 5 & -3 \\ 1 & -2 & 4 \end{pmatrix} + \begin{pmatrix} 7 & -5 & 2 \\ -9 & 4 & -1 \end{pmatrix} =$$

$$= \begin{pmatrix} 2+7 & 5+(-5) & -3+2 \\ 1+(-9) & -2+4 & 4+(-1) \end{pmatrix} = \begin{pmatrix} 9 & 0 & -1 \\ -8 & 2 & 3 \end{pmatrix}$$

ITI-LS "Francesco Giordani" Caserta
Anno scolastico: 2019/2020
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: es4
Versione: 1.0
Programmatore/i:
Sistema Operativo: Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: Somma di due matrici
Obiettivo del programma: Scrivere il sottoprogramma (da aggiungere all'es3) che restituisce la somma di due matrici

```
#include<iostream>

using namespace std;

int const dimFis=30;

int rig, col;

int mat1[dimFis][dimFis]; int mat2[dimFis][dimFis]; int mat3[dimFis][dimFis];

int ottieniDim(int dimFisPar);

void caricaMat(int matPar[][dimFis], int rigPar,int colPar);

void stampaMat(int matPar[][dimFis], int rigPar, int colPar);

void sommaMatrici(int mat1Par[][dimFis], int mat2Par[][dimFis], int mat3Par[][dimFis], int rigPar,
int colPar);

void prodottoMatriceXScalare(int mat1Par[][dimFis],int mat2Par[][dimFis], int rigPar, int colPar,
int scalarePar);

int main()

{

   cout<<"inserimento numero righe"<<endl;

   rig=ottieniDim(dimFis);

   cout<<"inserimento numero colonne"<<endl;

   col=ottieniDim(dimFis);

   caricaMat(mat1, rig, col);

   cout<<"caricamento seconda matrice"<<endl;

   caricaMat(mat2, rig, col);

   cout<<"stampa matrice 1"<<endl;

   stampaMat(mat1, rig, col);

   cout<<"stampa matrice 2"<<endl;

   stampaMat(mat2, rig, col);

   sommaMatrici(mat1, mat2, mat3, rig, col);

    cout<<"stampa matrice somma"<<endl;

   stampaMat(mat3, rig, col);
```

```
    prodottoMatriceXScalare(mat1,mat2,rig, col,3);
     cout<<"stampa prodotto matrice per uno scalare"<<endl;
    stampaMat(mat2, rig, col);
    return 0;
}
int ottieniDim(int dimFisPar)
{
    int dim;
    do
    {
        cout<<"Inserisci il numero di elementi: ";
        cin>>dim;
    }
    while((dim<0)||(dim>dimFisPar));
    return dim;
}
void caricaMat(int matPar[][dimFis], int rigPar, int colPar)
{
  for(int i=0; i<rigPar; i++)
     for(int j=0; j<colPar; j++)
    {
            cout<<"Inserisci l'elemento di posizione: "<<i<<","<<j<<" ";
            cin>>matPar[i][j];
    }
}
void stampaMat (int matPar[][dimFis], int rigPar, int colPar)
{
```

```
 for(int i=0; i<rigPar; i++)
  {
    for(int j=0; j<colPar; j++) cout<<matPar[i][j]<<"\t";
    cout<<endl;
  }
}

void sommaMatrici(int mat1Par[][dimFis], int mat2Par[dimFis][dimFis], int
mat3Par[dimFis][dimFis], int rigPar, int colPar)
{
  for(int i=0; i<rigPar; i++)
    for(int j=0; j<colPar; j++)  mat3Par[i][j]=mat1Par[i][j]+mat2Par[i][j];
}
void prodottoMatriceXScalare(int mat1Par[dimFis][dimFis],int mat2Par[dimFis][dimFis], int
rigPar, int colPar, int scalarePar)
{
    for(int i=0; i<rigPar; i++)
      for(int j=0; j<colPar; j++)  mat2Par[i][j]=mat1Par[i][j]*scalarePar;
}
```

Prodotto di due matrici (fonte: lezionidimatematica.net**)**

Supponiamo di avere:

- la **MATRICE** ***A*** di ordine ***(m* x *n)***;

e

- la **MATRICE** ***B*** di ordine ***(p* x *q)***;

Affinché il prodotto tra le due matrici **POSSA ESSERE ESEGUITO** è necessario che

$$n = p$$

CIOE' IL NUMERO DELLE COLONNE DELLA PRIMA MATRICE

DEVE ESSERE

UGUALE

AL NUMERO DELLE RIGHE DELLA SECONDA MATRICE.

Esempio:

A (3* x *4)	***B (4* x *5)***	si può eseguire il prodotto
A (2* x *5)	***B (5* x *3)***	si può eseguire il prodotto
A (3* x *4)	***B (3* x *4)***	**non** si può eseguire il prodotto

Vediamo un esempio concreto.

$$A = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & -7 & 8 & 0 \\ 6 & 9 & 1 & -3 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 0 \\ 3 & -6 \\ 5 & 2 \\ 7 & -8 \end{pmatrix}$$

Le due matrici possono essere moltiplicate perché la matrice ***A*** ha un numero di colonne (***4***) uguali al numero delle righe di ***B*** (***4***).

Una volta appurato che il prodotto tra le due matrici può essere eseguito chiamiamo ***C*** la matrice che si ottiene moltiplicando ***A*** per ***B***.

L'elemento

$$c_{11}$$

cioè l'elemento ***c*** che occupa la **prima riga** e la **prima colonna** della matrice ***C***

è il **PRODOTTO SCALARE** tra la **PRIMA RIGA di *A*** e la ***PRIMA COLONNA di B***. Ovvero:

$$c_{11} = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & -7 & 8 & 0 \\ 6 & 9 & 1 & -3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 3 & -6 \\ 5 & 2 \\ 7 & -8 \end{pmatrix}$$

$$c_{11} = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & -7 & 8 & 0 \\ 6 & 9 & 1 & -3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 3 & -6 \\ 5 & 2 \\ 7 & -8 \end{pmatrix}$$

$$c_{11} = 1 \cdot 1 + 2 \cdot 3 + 3 \cdot 5 + 5 \cdot 7 = 1 + 6 + 15 + 35 = 57.$$

L'elemento

c_{12}

cioè l'elemento ***c*** che occupa la **prima riga** e la **seconda colonna** della matrice ***C***

è il **PRODOTTO SCALARE** tra la **PRIMA RIGA di *A*** e la ***SECONDA COLONNA di B***. Ovvero:

$$c_{12} = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & -7 & 8 & 0 \\ 6 & 9 & 1 & -3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 3 & -6 \\ 5 & 2 \\ 7 & -8 \end{pmatrix}$$

$$c_{12} = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & -7 & 8 & 0 \\ 6 & 9 & 1 & -3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 3 & -6 \\ 5 & 2 \\ 7 & -8 \end{pmatrix}$$

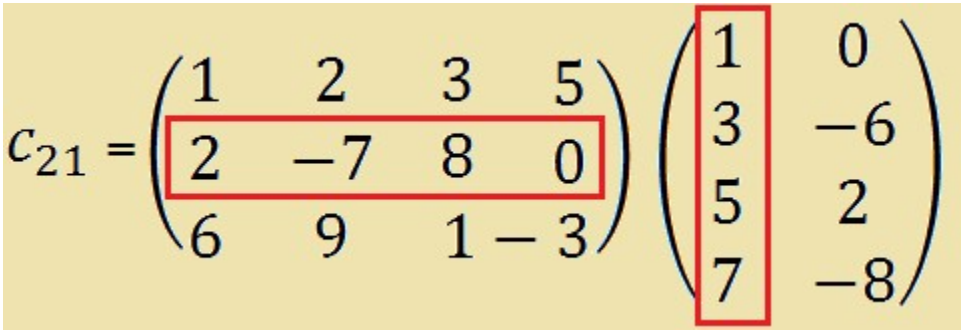
$c_{12} = 1 \cdot 0 + 2 \cdot (-6) + 3 \cdot 2 + 5 \cdot (-8) = 0 - 12 + 6 - 40 = -46$.

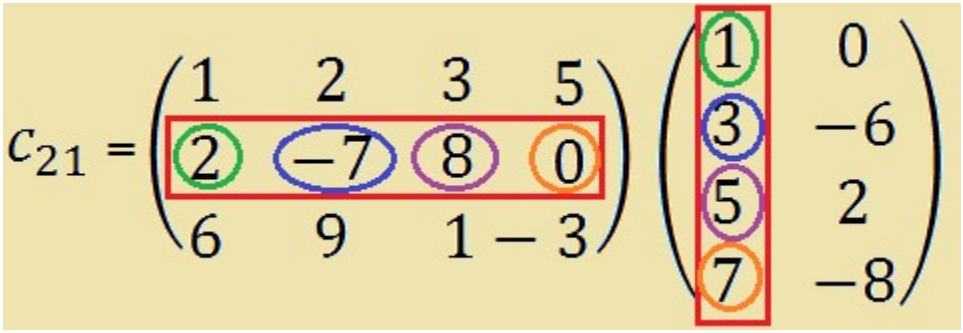
L'elemento

c_{21}

cioè l'elemento ***c*** che occupa la **seconda riga** e la **prima colonna** della matrice ***C***

è il **PRODOTTO SCALARE** tra la **SECONDA RIGA di *A*** e la ***PRIMA COLONNA di B***. Ovvero:





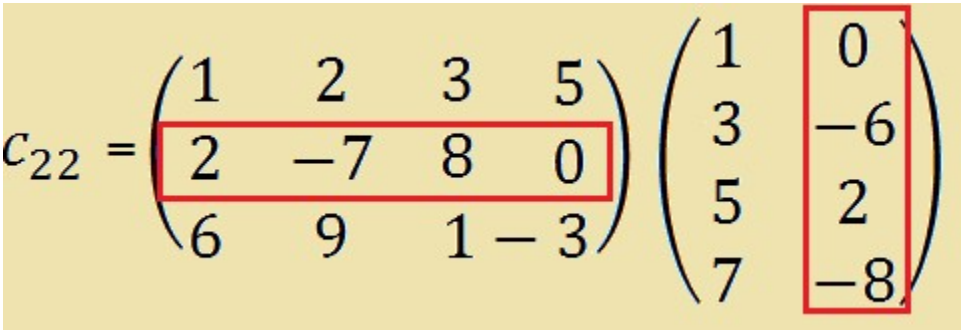
$c_{21} = 2 \cdot 1 + (-7) \cdot 3 + 8 \cdot 5 + 0 \cdot 7 = 2 - 21 + 40 + 0 = 21$.

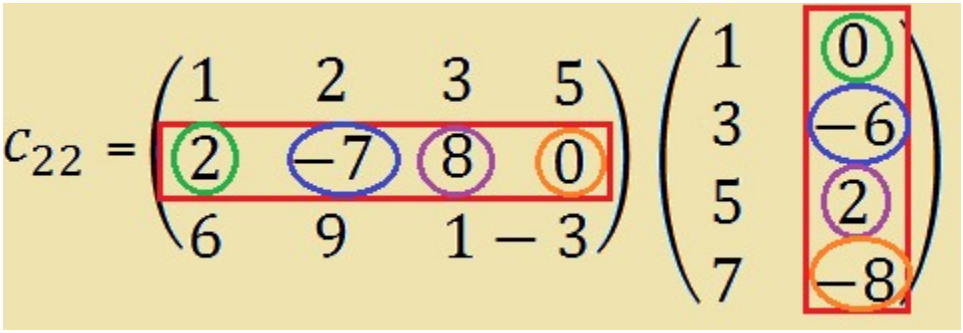
L'elemento

c_{22}

cioè l'elemento ***c*** che occupa la **seconda riga** e la **seconda colonna** della matrice ***C***

è il **PRODOTTO SCALARE** tra la **SECONDA RIGA di *A*** e la ***SECONDA COLONNA di B***.
Ovvero:





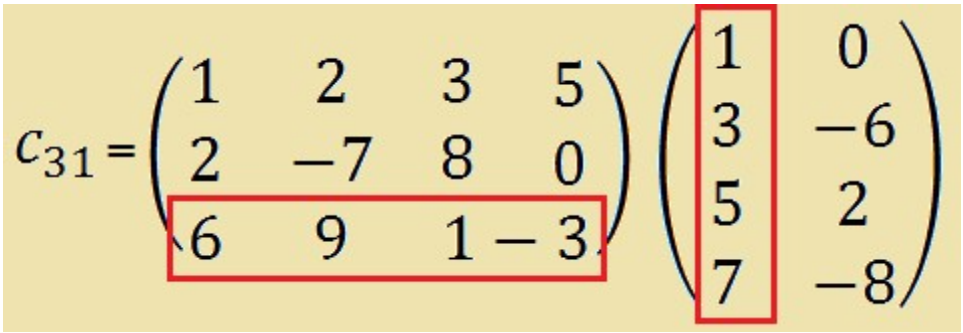
$c_{22} = 2 \cdot 0 + (-7) \cdot (-6) + 8 \cdot 2 + 0 \cdot (-8) = 0 + 42 + 16 + 0 = 58$.

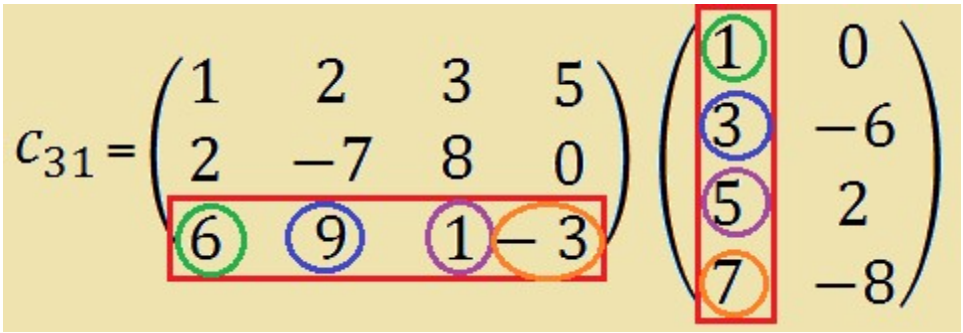
L'elemento

c_{31}

cioè l'elemento ***c*** che occupa la **terza riga** e la **prima colonna** della matrice ***C***

è il **PRODOTTO SCALARE** tra la **TERZA RIGA di *A*** e la ***PRIMA COLONNA di B***. Ovvero:





$c_{31} = 6 \cdot 1 + 9 \cdot 3 + 1 \cdot 5 + (-3) \cdot 7 = 6 + 27 + 5 - 21 = 17$.

L'elemento

$$c_{32}$$

cioè l'elemento ***c*** che occupa la **terza riga** e la **seconda colonna** della matrice ***C***

è il **PRODOTTO SCALARE** tra la **TERZA RIGA di *A*** e la ***SECONDA COLONNA di B***. Ovvero:

$$c_{32} = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & -7 & 8 & 0 \\ 6 & 9 & 1 & -3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 3 & -6 \\ 5 & 2 \\ 7 & -8 \end{pmatrix}$$

$$c_{32} = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & -7 & 8 & 0 \\ 6 & 9 & 1 & -3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 3 & -6 \\ 5 & 2 \\ 7 & -8 \end{pmatrix}$$

$$c_{32} = 6 \cdot 0 + 9 \cdot (-6) + 1 \cdot 2 + (-3) \cdot (-8) = 0 - 54 + 2 + 24 = -28.$$

Quindi, la nostra matrice ***C*** sarà:

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{pmatrix} = \begin{pmatrix} 57 & -46 \\ 21 & 58 \\ 17 & -28 \end{pmatrix}$$

Proprio per il modo come si procede al calcolo del **PRODOTTO TRA MATRICI** esso viene detto anche **PRODOTTO RIGA PER COLONNA**.

Sintetizzando possiamo dire che il generico elemento

c_{ij}

che si legge

c con i con j

è dato dalla **SOMMA dei PRODOTTI** degli elementi della **RIGA I-ESIMA** della matrice ***A*** per i corrispondenti elementi della **COLONNA J-ESIMA** della matrice ***B***.

Il tutto può essere scritto come segue:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kJ}$$

che si legge

c con i con j è uguale alla sommatoria per k che va da 1 ad n di a con i con k per b con k con j.

ITI-LS "Francesco Giordani" Caserta
Anno scolastico: 2019/2020
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: es5
Versione: 1.0
Programmatore/i:
Sistema Operativo: Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: Prodotto di due matrici
Obiettivo del programma: Scrivere il sottoprogramma (da aggiungere all'es1) che restituisce la matrice prodotto

```
#include<iostream>

using namespace std;

int const dimFis=30;

int rig1, col1,col2; //col1=rig2

int mat1[dimFis][dimFis]; int mat2[dimFis][dimFis]; int mat3[dimFis][dimFis];

int ottieniDim(int dimFisPar);

void caricaMat(int matPar[][dimFis], int rigPar,int colPar);

void stampaMat(int matPar[][dimFis], int rigPar, int colPar);

void prodottoScalareMatrici(int mat1Par[dimFis][dimFis],int mat2Par[dimFis][dimFis], int
mat3Par[dimFis][dimFis],int rig1Par, int col1Par, int col2Par );

int main()

{

   cout<<"inserimento numero righe matrice 1"<<endl;

   rig1=ottieniDim(dimFis);

   cout<<"inserimento numero colonne"<<endl;

   col1=ottieniDim(dimFis);

   caricaMat(mat1, rig1, col1);

   cout<<"caricamento seconda matrice"<<endl;

   cout<<"inserimento numero colonne(col1=rig2)"<<endl;

   col2=ottieniDim(dimFis);
```

```
    caricaMat(mat2, col1, col2);
    cout<<"stampa matrice 1"<<endl;
    stampaMat(mat1, rig1, col1);
    cout<<"stampa matrice 2"<<endl;
    stampaMat(mat2, col1, col2);
    prodottoScalareMatrici(mat1,mat2,mat3,rig1, col1,col2);
    cout<<"stampa matrice prodotto"<<endl;
    stampaMat(mat3, rig1, col2);
    return 0;
}
int ottieniDim(int dimFisPar)
{
    int dim;
    do
    {
        cout<<"Inserisci il numero di elementi: ";
        cin>>dim;
    }
    while((dim<0)||(dim>dimFisPar));
    return dim;
}
void caricaMat(int matPar[][dimFis], int rigPar, int colPar)
{
  for(int i=0; i<rigPar; i++)
     for(int j=0; j<colPar; j++)
    {
            cout<<"Inserisci l'elemento di posizione: "<<i<<","<<j<<" ";
```

```
            cin>>matPar[i][j];
    }
}
void stampaMat (int matPar[][dimFis], int rigPar, int colPar)
{
 for(int i=0; i<rigPar; i++)
  {
    for(int j=0; j<colPar; j++) cout<<matPar[i][j]<<"\t";
    cout<<endl;
  }
}


void prodottoScalareMatrici(int mat1Par[dimFis][dimFis],int mat2Par[dimFis][dimFis], int
mat3Par[dimFis][dimFis],int rig1Par, int col1Par, int col2Par)
{
    for(int i=0; i<rig1Par; i++)
      for(int j=0; j<col2Par; j++)
         {
           mat3Par[i][j] = 0;
           for(int k=0; k<col1Par; k++)     mat3Par[i][j] += mat1Par[i][k] * mat2Par[k][j];
         }
}
```

Trasposta di una matrice

Sia A una matrice MxN, si dice trasposta di A, e si indica con A^T, la matrice NxM ottenuta da A scambiando ordinatamente le righe con le colonne:

$$A = \begin{pmatrix} 2 & 3 & 1 \\ 4 & 7 & 5 \end{pmatrix} \qquad A^T = \begin{pmatrix} 2 & 4 \\ 3 & 7 \\ 1 & 5 \end{pmatrix}$$

Se la matrice è simmetrica allora $A^T = A$.

ITI-LS "Francesco Giordani" Caserta
Anno scolastico: 2019/2020
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: es6
Versione: 1.0
Programmatore/i:
Sistema Operativo: Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: Matrice trasposta
Obiettivo del programma: Data una matrice scrivere il sottoprogramma che restituisce la matrice trasposta

```
#include<iostream>

using namespace std;

int const dimFis=30;

int rig, col;

int mat1[dimFis][dimFis]; int matT[dimFis][dimFis];

int ottieniDim(int dimFisPar);

void caricaMat(int matPar[][dimFis], int rigPar,int colPar);

void stampaMat(int matPar[][dimFis], int rigPar, int colPar);

void matriceTrasposta(int mat1Par[dimFis][dimFis],int matTPar[dimFis][dimFis],int rig1Par, int colPar);

int main()

{

   cout<<"inserimento numero righe matrice 1"<<endl;
```

```
    rig=ottieniDim(dimFis);
    cout<<"inserimento numero colonne"<<endl;
    col=ottieniDim(dimFis);
    caricaMat(mat1, rig, col);
    cout<<"stampa matrice 1"<<endl;
    stampaMat(mat1, rig, col);
    matriceTrasposta(mat1,matT,rig,col);
    cout<<"stampa matrice trasposta"<<endl;
    stampaMat(matT, col, rig);
    return 0;
}
int ottieniDim(int dimFisPar)
{
    int dim;
    do
    {
        cout<<"Inserisci il numero di elementi: ";
        cin>>dim;
    }
    while((dim<0)||(dim>dimFisPar));
    return dim;
}
void caricaMat(int matPar[][dimFis], int rigPar, int colPar)
{
  for(int i=0; i<rigPar; i++)
     for(int j=0; j<colPar; j++)
    {
```

```
            cout<<"Inserisci l'elemento di posizione: "<<i<<","<<j<<" ";
            cin>>matPar[i][j];
    }
}
void stampaMat (int matPar[][dimFis], int rigPar, int colPar)
{
 for(int i=0; i<rigPar; i++)
  {
    for(int j=0; j<colPar; j++) cout<<matPar[i][j]<<"\t";
    cout<<endl;
  }
}

void matriceTrasposta(int mat1Par[dimFis][dimFis],int matTPar[dimFis][dimFis],int rig1Par, int
colPar)
{
    for(int i=0; i<rig1Par; i++)
      for(int j=0; j<colPar; j++)
         {
           matTPar[j][i] = mat1Par[i][j];
         }
}
```

Detta A^T la matrice trasposta di A, una matrice A è simmetrica quando: $A^T = A$
o equivalentemente quando i suoi elementi a_{ij} soddisfano:
$a_{ij} = a_{ji} \quad \forall i \quad \forall j$

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 0 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

OCTAVE/MATLAB

https://www.tutorialspoint.com/execute_matlab_online.php

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

A' % matrice trasposta

A*2 % prodotto di una matrice per uno scalare

zeros(N) % Matrice di zeri, NxN
zeros(M, N) % Matrice di zeri MxN
ones(N) % Matrice di uno, NxN
ones(M, N) % Matrice di uno MxN
eye(N) % Matrice identità, NxN
eye(M, N) % Matrice identità estesa, MxN
diag(V) % Matrice con il vettore V per diagonale

A=[1 2; 3 4]
B=[5 6; 7 8]
A*B % prodotto scalare di due matrici

rand(m,n) matrice di numeri random di ordine m per n
det(A) determinante della matrice A
size(A) numero di righe e colonne di A
inv(A) inversa di A

PYTHON

v = zeros(n) # crea un vettore "v" di "n" componenti di tipo ’int’ e di valore zero
v = rand(n) # creazione di un vettore random di lunghezza "n"
np.identity(n) #crazione matrice identità

M = rand(m,n) # creazione di una matrice random di lunghezza "mxn"

Somma di due vettori
```
import numpy as np
 v1=np.array([1,2,3])
v2=np.array([10,20,30])
v1+v2
```

Prodotto di due vettori
```
import numpy as np
 v1=np.array([1,2,3])
v2=np.array([10,20,30])
v1*v2
```

Prodotto scalare (usare la funzione np.dot)
```
np.dot(v1,v2)
```

Prodotto di un vettore per uno scalare
```
v1*3
```

Somma di due matrici
```
import numpy as np
m1=np.array([[1,2],[3,4],[5,6]])
m2= np.array([[7,8],[9,10],[11,12]])
m1+m2
```

Prodotto di due matrici
```
import numpy as np
m1=np.array([[1,2],[3,4],[5,6]])
m2= np.array([[7,8],[9,10],[11,12]])
m1*m2
```

Prodotto scalare (usare la funzione np.dot)
```
mat1=np.array([[1, 2, 3, 5],[2,-7, 8,0],[6,9,1,-3]])
mat2=np.array([[1, 0],[3,-6],[5,2], [7,-8]])
np.dot(mat1,mat2)
```

Prodotto di una matrice per uno scalare
```
m1*3
```