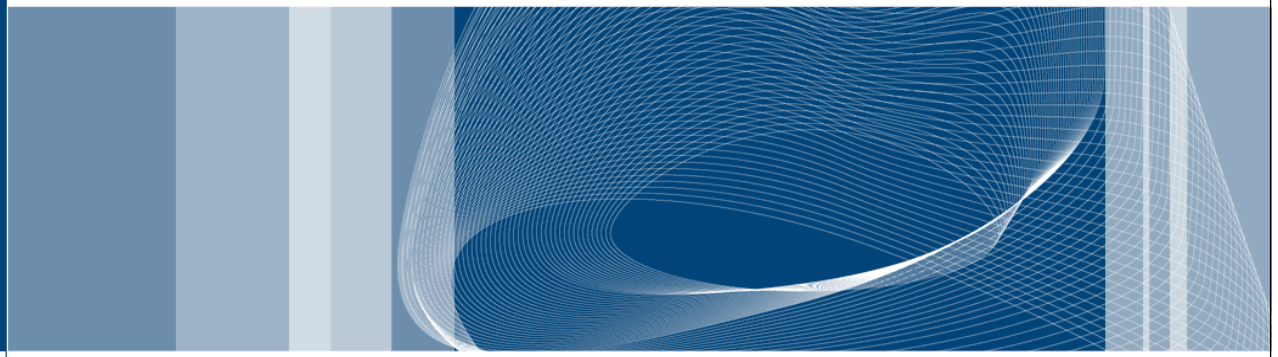
POLITECNICO DI MILANO



Fondamenti di Informatica 2013-2014

Vettori monodimensionali

Paola Mussida
Area Servizi ICT

Acquisire il nome dall'utente e stampare un messaggio di saluto personalizzato.

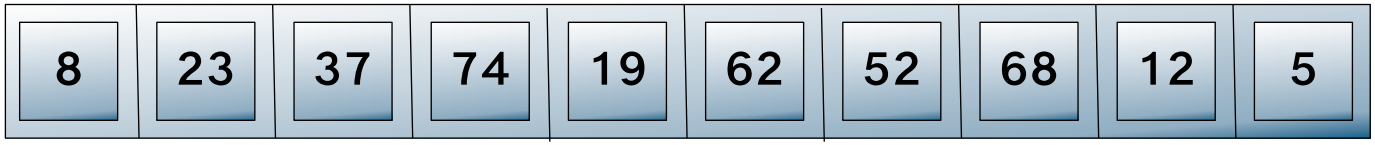
Esercizio V_0 - Nome

```c
#include <stdio.h>
int main()
{
    char nome [10];
    printf("\nCome ti chiami? ");
    gets(nome);  //acquisizione del nome

    printf("\nCiao %s!\n\n",nome);
}
```

Dichiarazione e inizializzazione di un array di 10 interi e ricerca della prima occorrenza di un valore chiesto all'utente.

Esercizio V_1 - Cerca

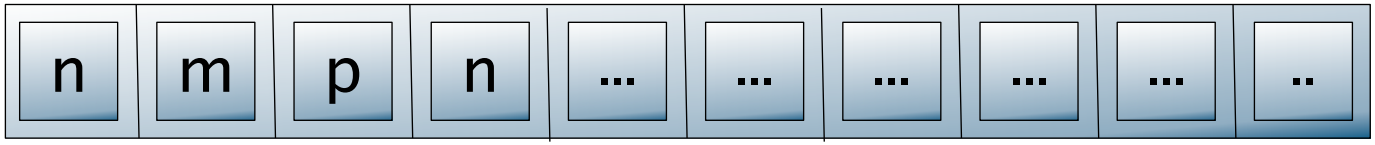
```
#include <stdio.h>
#define MAX 10
int numero, i;
int vettore[MAX];
```




```
void main()
{                                   /* inizializzazione array */
   vettore[0] = 8;
   vettore[1] = 23;
   vettore[2] = 37;
   vettore[3] = 74;
   vettore[4] = 19;
   vettore[5] = 62;
   vettore[6] = 52;
   vettore[7] = 68;
   vettore[8] = 12;
   vettore[9] = 5;
```

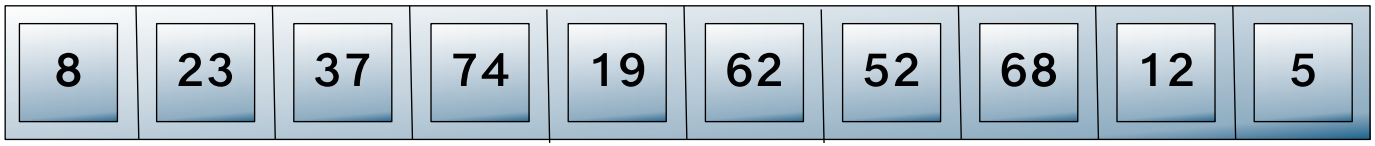
Dichiarazione non inizializzata


```
#include <stdio.h>
#define MAX 10
int numero, i;
int vettore[MAX];
```



... oppure

```
#include <stdio.h>
#define MAX 10

int numero, i;

int vettore[MAX]= {8,23,37,74,19,62,52,68,12,5};
/* inizializzazione array */
```




```
void main()
{
```

Esercizio V_1 - Cerca


```
/* numero da cercare */
printf("\nInserisci il numero da cercare: ");
scanf("%d", &numero);

/* ricerca */

 for(i=0;i<MAX;i++)
 {
     if (vettore[i]==numero)
     break;
 }


/* verifica esito della ricerca */
if (i==MAX)
  printf("\nIl numero richiesto non e' stato trovato!");
else
  printf("\nIl numero richiesto e' stato trovato nella
                                        posizione %d!", i);
}
```

for vs while

```
for(i=0;i<MAX;i++)
{
    if (vettore[i]==numero)
    break;
}
```

```
i=0;
while (i<MAX && vettore[i]!=numero)
{
    i++;
}
```

Inizializzazione casuale dell'array di numeri interi dall'1 al 100.

variante V_1

```
  //...

srand(time(NULL));

for(i=0;i<MAX;i++)
{
      vettore[i]=rand() % 100 + 1;
}

  //...
```

Esercizio V_2 - Copia

Copia invertita di un array in un secondo array. L'array iniziale ha una lunghezza pari a MAX_ARRAY ed è inserito dall'utente.

Esercizio V_2 - Copia

```c
#include <stdio.h>

#define MAX 10

int vettore[MAX], invertito[MAX], i;  //dichiarazione

void main()
{
      /* inserimento array originale */
      for(i=0;i<MAX;i++)
      {
         printf("\nInserisci %d-esimo numero:", i);
         scanf("%d",&vettore[i]);
      }
  //...
```

Esercizio V_2 - Copia

```
/* copia e inversione */

  for(i=0;i<MAX;i++)
  {
     invertito[(MAX-1)-i]=vettore[i];
  }


/* visualizzazione array */

  for(i=0;i<MAX;i++)
  {
     printf("\nOriginale %d, invertito %d",
                              vettore[i], invertito[i]);
  }
}
```

Esercizio V_3 - Inversione

Inversione dei valori di un array sul medesimo array. Il vettore di partenza è acquisito dall'utente. L'array ha una lunghezza pari a MAX_ARRAY. Per iniziare: la lunghezza dell'array è pari.

Esercizio V_3 - Inversione

```
#include <stdio.h>
#define MAX 10

int vettore[MAX], i, temp;                    //dichiarazione

void main()
{
      /* inserimento array originale */
      for(i=0;i<MAX;i++)
      {

         printf("\nInserisci %d-esimo numero:", i);
         scanf("%d",&vettore[i]);

      }

  //...
```

Esercizio V_3 - Inversione

```
/* copia e inversione */
 for(i=0;i<(MAX/2);i++)
 {
       temp=vettore[i];
       vettore[i]=vettore[(MAX-1) - i];
       vettore[(MAX-1) - i]=temp;
 }


/* visualizzazione array */
 for(i=0;i<MAX;i++)
 {
        printf("\n%d-esimo valore = %d",i, vettore[i]);
 }
}
```

Esercizio V_4 - Ripetizione

Verifica ripetizione di un elemento in un array. L'array è acquisito dall'utente.

Esercizio V_4 - Ripetizione

```
#include <stdio.h>
#define MAX 10

int i, j, vett[MAX];

main()
{
   /* inserimento dei numeri */
   printf("\nInserisci il vettore di Numeri :");
   /* Acquisizione */
   for (i=0;i<MAX;i++)
   {
      printf("\nInserisci il %d elemento: ",i);
      scanf("%d",&vett[i]);
   }
```

Esercizio V_4 - Ripetizione

```
/* verifica ripetizioni*/


for (i=0; i<(MAX-1); i++)
  for (j=i+1; j<MAX; j++)
    if (vett[i]==vett[j])
       printf("Gli elementi %d e %d coincidono\n",i,j);

}
```

Esercizio V_5 - Multipli di 5

Richiedere all'utente di riempire un array e copiare i valori multipli di 5 in un secondo vettore.

Esercizio V_5 - Multipli di 5

```
#include <stdio.h>
#define MAX 10
int v_in[MAX], v_out[MAX], i, n_out;

int main()
{
  printf("\nInserire i %d elementi dell'array\n",MAX);
    for(i=0;i<MAX;i=i+1)
      scanf("%d",&v_in[i]);
  n_out=0;
  for(i=0;i<MAX;i=i+1)
    if (v_in[i]%5==0 && v_in[i] != 0)
    {
        v_out[n_out]=v_in[i];
        n_out++;
    }

  printf("\nI valori multipli di 5 sono: ");
  for(i=0;i<n_out;i=i+1)
      printf("\t%d",v_out[i]);
}
```

Esercizio V_6 - Maggiore di stringhe

Utilizzando le funzioni della libreria "String.h", creare un programma che acquisisca un nome dall'utente e lo confronti con una stringa predefinita evidenziando quale sia il maggiore alfabeticamente. I due nomi devono esser visualizzati con l'indicazione della lunghezza.

Esercizio V_6 - Maggiore di stringhe


```
#include<stdio.h>
#include<string.h>

char name1[12], name2[12], mixed[25], nuova[12];
int main()
{

     /* copia */
     strcpy(name1,"Laura");

     /* lettura da tastiera */
     printf("Inserisci un nome: ");
     scanf("%s", name2);   //scanf("%s", &name2[0]);

     /* visualizzazione e lunghezza della stringa */
     printf("Il primo nome e' %s ed e' lungo %d caratteri\n", name1, s
     printf("Il secondo nome e' %s ed e' lungo %d caratteri\n", name2,
```

Esercizio V_6 - Maggiore di stringhe


```
  /* confronto */
  if (strcmp(name1,name2)==0) /* ritorna 0 se name1 == name2 */
        printf("I nomi coincidono\n");
  else
  {
      if(strcmp(name1,name2)>0)    /* ritorna 1 se name1 > name2 */
        strcpy(mixed,name1);

      else if (strcmp(name1,name2)<0)
        strcpy(mixed,name2);

      printf("Il nome alfabeticamente piu' grande e': %s\n",mixed);
  }
}
```

Gestione di un array di interi positivi. Implementare le funzionalità di:

- ✓ inserimento di un elemento nella prima cella libera;
- ✓ visualizzazione del contenuto dell'array;
- ✓ cancellazione della prima occorrenza di un elemento specifico inserito dall'utente.

Esercizio V_7 - Gestione


```
#include <stdio.h>
#define MAX 10
int vettore[MAX], i, operaz=-1, numero;
int main()
{
     for(i=0;i<MAX;i++)  /* inizializzazione array */
     {
          vettore[i]=-1;
     }
     while (operaz!=0)  /* elaborazione comando richiesto */
     {
                    /* richiesta operazione da eseguire */
                    printf("\nInserisci l'operazione (0 fine, 1 aggiungi,
                                                  2 elimina, 3 visualizza): ");
                    scanf("%d",&operaz);

                    /* elaborazione */
                    switch (operaz)
                    {
                                   /* ... */
                    }
     }
}
```

Esercizio V_7 - Gestione - inserimento


```
case 1:
      /* richiesta numero sul quale operare */
      printf("\nInserisci il numero positivo: ");
      scanf("%d",&numero);

      /* ricerca cella disponibile */
      i=0;
      while  (i<MAX && vettore[i] != -1)
      {
        i++;
      }

      if (i==MAX)            /* verifica inserimento */
        printf("\nNon ci sono celle disponibili!");
      else
        vettore[i]=numero;          /* inserimento */
  break;
```

Esercizio V_7 - Gestione - eliminazione


```
case 2:
  /* richiesta numero sul quale operare */
  printf("\nInserisci il numero positivo: ");
  scanf("%d",&numero);
   /* ricerca numero nell'array */
   i=0;
   while(i<MAX && vettore[i] != numero)
   {
       i++;
   }

   /* verifica eliminazione */
   if (i==MAX)
       printf("\nNumero non presente!");
   else
       /* sovrascrivo il valore "vuoto" */
       vettore[i]=-1;

  break;
```

Esercizio V_7 - Gestione - visualizzazione


```
case 3:
/* visualizzazione array */


     for(i=0;i<MAX;i++)
     {
       if (vettore[i]==-1)
         printf("\n elemento %d-esimo vuoto!", i);

       else
         printf("\n elemento %d-esimo = %d", i, vettore[i]);
     }




break;
```

```
case 0:
    printf("\nFine Programma!");
  break;


default:
    printf("\nOperazione non supportata!");
  break;
```

Osservazioni:

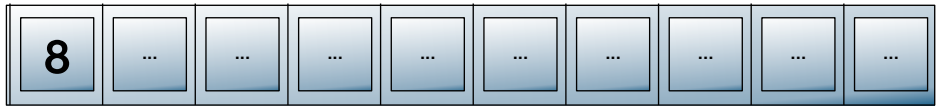
- ✓Inserimento nel primo posto disponibile; se volessimo conservare l'ordine degli inserimenti?
- ✓Eliminazione della prima occorrenza trovata; ma se ci fossero più occorrenze da eliminare?
- ✓Il valore della cella vuota è -1; come estendere l'esercizio a tutti gli interi?

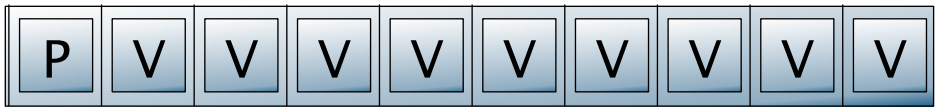
Eliminazione di tutte le occorrenze:


```
case 2:
  printf("\nInserisci il numero positivo: ");
  //richiesta numero sul quale operare
  scanf("%d",&numero);
  stato=0; /* ricerca numero nell'array */

      for(i=0;i<MAX;i++)
      {                          /* sovrascrivo il valore "vuoto" */
        if (vettore[i]==numero)
        {
          vettore[i]=-1;
          stato=1;
        }
      }

  if (stato==0) /* verifica eliminazione */
  {
      printf("\nNumero non presente!");
  }
break;
```

Variante dell'esercizio V_7 con vettore di appoggio per memorizzare lo stato della cella (piena/vuota).

Esercizio V_8 - Gestione con stato





```
#include <stdio.h>
#define MAX 10

typedef enum {VUOTA, PIENA}
disponibilita;

int i, operaz=-1, numero;

int vettore[MAX];
disponibilita stato[MAX];
```

Esercizio V_8 - Gestione con stato


```
int main()
{
        /* inizializzazione array */
     for(i=0;i<MAX;i++)
     {
       stato[i]=VUOTA;
     }

     while (operaz!=0)
     {
       /* richiesta operazione da eseguire */
       printf("\nInserisci l'operazione (0 fine, 1 aggiungi,
                                    2 elimina, 3  visualizza): ");
       scanf("%d",&operaz);
       switch (operaz) /* elaborazione */
       {
              /* ... */
       }
     }//end while
}//end main
```

Esercizio V_8 - Gestione con stato - inserimento

/* ... */

```
case 1:
        // richiesta numero sul quale operare
        printf("\nInserisci il numero positivo: ");
        scanf("%d",&numero);

        /* ricerca cella disponibile */
        i=0;
        while(i<MAX && stato[i] != VUOTA)
                i++;
        if (i==MAX)
                printf("\nNon ci sono celle disponibili!");
        else  /* inserimento */
        {
                vettore[i]=numero;
                stato[i]=PIENA;
        }

 break;
```

Esercizio V_8 - Gestione con stato - eliminazione

```
/* ... */

case 2:
        /* richiesta numero sul quale operare */
        printf("\nInserisci il numero positivo: ");
        scanf("%d",&numero);

        i=0;
        while(i<MAX && (stato[i]==VUOTA ||
              ((stato[i]==PIENA) && (vettore[i]!=numero)) ) )
           i++;

        /* verifica eliminazione */
        if (i==MAX)
                printf("\nNumero non presente!");
        else  /* eliminazione */
                stato[i]=VUOTA;


 break;
```

Esercizio V_8 - Gestione con stato - visualizzazione

/* ... */

```
case 3: /* visualizzazione array */




    for(i=0;i<MAX;i++)
    {
      if (stato[i]==PIENA)
          printf("\n vettore %d-esimo (PIENA) = %d", i, vettore[i]);
      else
          printf("\n vettore %d-esimo (VUOTA)", i);
    }



 break;
```

```
case 0:
    printf("\nFine Programma!");
 break;


default:
    printf("\nOperazione non supportata!");
 break;
```

Verifica del contenimento di una sequenza consecutiva in un array.
Sia la sequenza che l'array sono inseriti dall'utente.

Esercizio V_9 - Inclusione (≈ Tde II appello 08-09)


```c
#include <stdio.h>
#define MAX 20
#define L_SEQ 4
int i,j, ripet;
int vett[MAX], seq[MAX];
int main()
{
        /* inserimento array da scandire */
        printf("\nInserisci il vettore di numeri :");
        for (i=0;i<MAX;i++)       //acquisizione array da scandire
        {
                printf("\nInserisci il %d elemento: ",i);
                scanf("%d",&vett[i]);
        }

        /* inserimento sequenza da ricercare */
        printf("\nInserisci la sequenza:");
        for (i=0;i<L_SEQ;i++)
        {
                printf("\nInserisci il %d elemento: ",i);
                scanf("%d",&seq[i]);
        }
```

Esercizio V_9 - Inclusione (≈ Tde II appello 08-09)


```
    /* ricerca della sequenza ... */
    for (i=0; i<(MAX-L_SEQ+1); i++)
    {
          ripet=1;

          /* ... in ogni posizione */
          for(j=0; j<L_SEQ; j++)
                if (vett[i+j]!=seq[j])
                      ripet=0;

          /* risultato controllo */
          if (ripet==1)
          {
             printf("\nTrovata una ripetizione dalla posizione %d",i);
          }
    }

}
```

Acquisizione di un vettore di interi terminato dal numero '0' e calcolo della media di tutti i numeri contenuti inseriti.

Esercizio V_10 - Media

```c
#include <stdio.h>
#define MAX 10
float vettore[MAX], numero, somma=0.0;
int c, n=0;

int main()
{
        c=0;
        do{   /* inserimento numeri */
                printf("\nInserisci il numero: ");
                scanf("%f", &vettore[c]);
                c++;
        }while (c<MAX && vettore[c-1]!=0);

        while ( (vettore[n]!=0) && (n<MAX) ) /* calcolo media */
                somma = somma + vettore[n++];

        printf("La media e' %4.2f.\n", (somma / n) );
}
```

Esercizio V_11 - Media dei 3 adiacenti

Calcolo dei valori medi di tre elementi adiacenti in un array. L'array è acquisito dall'utente.

Esercizio V_11 - Media dei 3 adiacenti


```
#include <stdio.h>
#define MAX 10

float vett1[MAX],vett2[MAX];
int n;
main()
{
        for (n=0; n<MAX; n++)
        {
                printf("\nInserisci il %d valore : ",n);
                scanf("%f",&vett1[n]);
        }

        vett2[0]=(vett1[0]+vett1[1])/2.0;
        for(n=1; n<(MAX-1) ;n++)
                vett2[n]=(vett1[n-1]+vett1[n]+vett1[n+1])/3.0;
        vett2[MAX-1]=(vett1[MAX-2]+vett1[MAX-1])/2.0;

        for(n=0; n<MAX; n++)
                printf("\nmedia posizione %d : %6.2f",n,vett2[n]);
}
```

Acquisizione di un array e ordinamento per selezione.

Esercizio V_12 - Ordinamento


```c
#include <stdio.h>
#define MAX 20
int i,j,minval, scambio, vett[MAX];

int main()
{
        printf("\nInserisci il vettore di numeri :");
        for (i=0;i<MAX;i++)
        {
                printf("\nInserisci il %d elemento: ",i);
                scanf("%d",&vett[i]);
        }

        /* ordinamento */
           /* ... */

        for(i=0;i<MAX;i++)  /* vettore finale */
                printf("\nVett[%d] = %d",i,vett[i]);
}
```

Esercizio V_12 - Ordinamento


```
/* ordinamento */
for (i=0; i<(MAX-1); i++)
{
        /* ricerca del valore minimo */
        minval=i;
        for (j=i+1;j<MAX;j++)
                if (vett[j]<vett[minval])
                        minval=j;

        /* eventuale scambio tra il valore
            minimo e il primo */
        if (minval!=i)
        {
                scambio=vett[i];
                vett[i]=vett[minval];
                vett[minval]=scambio;
        }
}
```

Ordinamento con l'algoritmo BubbleSort.
L'array è costituito da interi casuali.

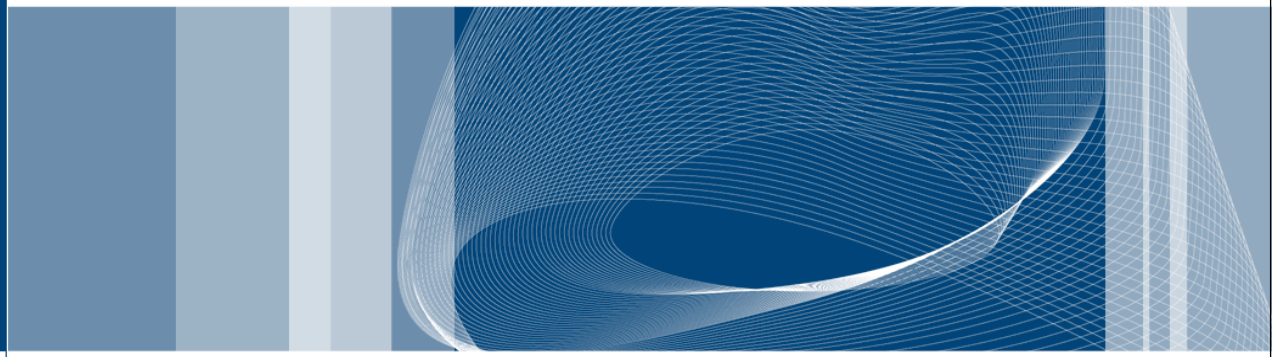
Esercizio V_13 - BubbleSort

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 15
int i, j, k=1, vett[MAX], scambio, tmp;

int main()
{
        srand(time(NULL));
        for (i=0; i<MAX ; i++)
                vett[i]= rand() % 100;

        printf("\n\n");
        for(j=0; j<MAX; j++)
                printf("%2d  ", vett[j]);

        printf("\n");  //favorisce la lettura del terminale
        printf(" *  ");
```

Esercizio V_13 - BubbleSort


```
        do {
                scambio=0;
                for(i=0; i< (MAX-k); i++)
                {
                        if (vett[i]>vett[i+1]) {
                                tmp=vett[i+1];
                                vett[i+1]=vett[i];
                                vett[i]=tmp;
                                scambio=1;
                        }
                        printf("\n\n");
                        for(j=0; j<MAX; j++)
                                printf("%2d  ", vett[j]);
                        printf("\n");
                        for(j=0; j<MAX; j++) {
                                if (j==i+1)
                                        printf(" *  ");
                                else if (j > (MAX-k))
                                        printf(" -  ");
                                else
                                        printf("    ");
                        }
                }
                k++;
        } while (scambio == 1);
}
```

POLITECNICO DI MILANO



Fondamenti di Informatica 2013-2014

Vettori bidimensionali

Paola Mussida
Area Servizi ICT

Scrivere un programma che acquisisca una matrice di interi e calcoli quale riga contiene i valori con la somma massima. Infine visualizzare somma e numero di riga.

Esercizio M_1 - Riga max

```c
#include <stdio.h>

#define MAX_ROW 5
#define MAX_COL 5

int i, j, nr, max, somma;

int mat[MAX_ROW][MAX_COL];
```

Esercizio M_1 - Riga max

```
int main()
{
    /* lettura matrice */
    for (i=0;i<MAX_ROW; i++)
        for(j=0;j<MAX_COL;j++)
        {
            printf("Input elemento
                                %d,%d: ",i,j);

            scanf("%d",&(mat[i][j]));
    }
```

Esercizio M_1 - Riga max


```
/* somma massima sulle righe */
for (i=0; i<MAX_ROW; i++)
{  /*calcolo della somma degli elementi sulla riga */
    somma=0;
    for(j=0;j<MAX_COL;j++)
          somma=somma+mat[i][j];
    if (i==0) {     /* confronto con somma massima */
          max=somma;
          nr=0;
    }
    else if (somma>max) {
          max=somma;
          nr=i;
    }
}
```

Esercizio M_1 - Riga max

```
/* risultato */
printf("\nSomma massima = %d
                        Riga = %d",max,nr);
}
```

Scrivere un programma che definisca 3 matrici bidimensionali globali A[r,p], B [p,c] e C[r,c] di numeri interi (definire anche le costanti 'r', 'p', 'c') e calcoli il prodotto tra le matrici A e B.

Il risultato e' memorizzato nella matrice C.

Si ricorda che ogni elemento di C è calcolato nel seguente modo:

(per i = 1.. R e j = 1 .. C)

$$C[i,j] = \Sigma (A[i,s] * B[s,j])$$

con s = [1 .. P]

Esercizio M_2 - Prodotto

```
#include <stdio.h>
#include <stdlib.h>

#define R 3
#define P 2
#define C 4

int x, y, z;


int mat_A[R][P], mat_B[P][C], mat_C[R][C];
```

```
int main() {

    srand(time(NULL));
    /* popola la prima matrice */
    printf("\n");
    for (x=0; x<R; x++)    {
        printf("\n");
        for (y=0; y<P; y++)    {

                mat_A[x][y] = rand() % 10;
                printf("\t%d",mat_A[x][y]);

        }
    }
```

```
/* popola la seconda matrice */

printf("\n");
for (x=0; x<P; x++) {
    printf("\n");
    for(y=0; y<C; y++) {
        mat_B[x][y] = rand() % 10;
        printf("\t%d",mat_B[x][y]);
    }
}
```

Esercizio M_2 - Prodotto


```
for (    x=0; x<R; x++    )
 for (     y=0; y<C; y++    ) {

    mat_C[x][y] = 0;
    for(z=0; z<P; z++)

      mat_C[x][y] = mat_C[x][y] +
                (mat_A[x][z] * mat_B[z][y]);


}
```

Esercizio M_2 - Prodotto

```
/* visualizza la terza matrice */


    printf("\n");
    for (x=0; x<R; x++) {
        printf("\n");
        for(y=0; y<C; y++)
            printf("\t%d",mat_C[x][y]);
    }

}
```

Memorizzare 10 parole inserite dall'utente in un vettore di stringhe. Restituire per ogni parola il numero di occorrenze della lettera 'a' (anche non consecutive).

La lunghezza massima per ogni parola è di 15 caratteri.

Esercizio M_3 - Conta a

```c
#include <stdio.h>
#include <string.h>

#define NUM 10


int i, j, cont;


char vocab[NUM][16];
```

Esercizio M_3 - Conta a

```
int main() {

    /* inserimento parole */
    for (i=0; i< NUM; i++)
    {
          printf("\nInserisci la %d parola: ",i);

          scanf("%s", vocab[i]);


    }
```

Esercizio M_3 - Conta a

```
    /* verifica contenuto */
    for (i=0; i< NUM; i++) {


            cont=0;
            for (j=0; j< strlen(vocab[i]); j++)
               if (vocab[i][j] == 'a' )
                     cont++;


            printf("\nLa parola %s contiene %d
                    lettere 'a'", vocab[i], cont);
    }
}
```

Memorizzare 10 parole inserite dall'utente in un vettore di stringhe, rifiutando le parole già inserite.
Lunghezza massima per ogni parola: 15 caratteri.

Esercizio M_4 - Dizionario

```c
#include <stdio.h>
#include <string.h>

#define NUM 10

typedef char parola[16];

parola vocab[NUM], nuova;

int i, j, cont;
```

Esercizio M_4 - Dizionario

```
int main() {  /* inserimento parole */
    for (i=0; i< NUM; i++) {

      do {
          printf("\nInserisci la %d parola: ", i+1);
          scanf("%s", nuova);
          /* verifica presenza nuova parola */
          j=0;
          while (j < i
                    && strcmp(vocab[j], nuova) != 0)
              j++;

      } while(j<i);

      strcpy(vocab[i], nuova);
    }
```

Esercizio M_4 - Dizionario

```
/* visualizzazione parole */


for (i=0; i< NUM; i++)
    printf("\n%d parola: %s", i+1, vocab[i]);




}
```

Implementare un programma che controlli se uno schema di "sudoku" sia correttamente risolto o meno.

Esercizio M_5 - Sudoku

```
int sodoku[9][9]= {
                    {1,2,3,4,5,6,7,8,9},
                    {4,5,6,7,8,9,1,2,3},
                    {7,8,9,1,2,3,4,5,6},
                    {2,3,4,5,6,7,8,9,1},
                    {5,6,7,8,9,1,2,3,4},
                    {8,9,1,2,3,4,5,6,7},
                    {3,4,5,6,7,8,9,1,2},
                    {6,7,8,9,1,2,3,4,5},
                    {9,1,2,3,4,5,6,7,8}
                  };

int r, c, x, y;
int n, trovato;
```

Esercizio M_5 - Sudoku

```
int main () {

/* visualizzazione */
for (r=1; r<10; r++) {

        for (c=1; c<10; c++)
                printf("%d
                        \t",   sodoku[r-1][c-1] );

        printf("\n");
}
```

Esercizio M_5 - Sudoku

```
/* verifica righe */
for (r=1; r<10; r++) {

   for (n=1; n<10; n++) {
      trovato = 0;
      c=1;
      do {

            if ( sodoku[r-1][c-1] == n)
               trovato=1;
            c++;

      } while (c < 10 && trovato==0)        ;
      if (trovato==0)
         printf("Sodoku non corretto! Non ho trovato il
                        numero %d nella riga %d\n", n, r);
   }
}
```

Esercizio M_5 - Sudoku


```
/* verifica colonne */
for (c=1; c<10; c++) {

    for (n=1; n<10; n++) {
        trovato = 0;
        r=1;

        do {
            if ( sodoku[r-1][c-1] == n)
                trovato=1;
                r++;
            } while(r < 10 && trovato==0);

        if (trovato==0)
            printf("Sodoku non corretto! Non ho trovato il
                    numero %d nella colonna %d\n", n, c);
    }
}
```

Esercizio M_5 - Sudoku


```
/* verifica quadrati interni */
for (x=0; x<3; x++)
   for (y=0; y<3; y++) {
      for (n=1; n<10; n++) {
         trovato = 0;
         r=x*3;
          do {
             c=y*3;
             do {
                if ( sodoku[r][c] == n)
                   trovato=1;
                c++;
             } while((c < (y+1)*3) && trovato==0);
             r++;
          } while((r < (x+1)*3) && trovato==0);
if (trovato==0)
   printf("Sodoku non corretto! Non ho trovato il numero
            %d nel quadrato %d-%d\n", n, x, y);
}}}
```