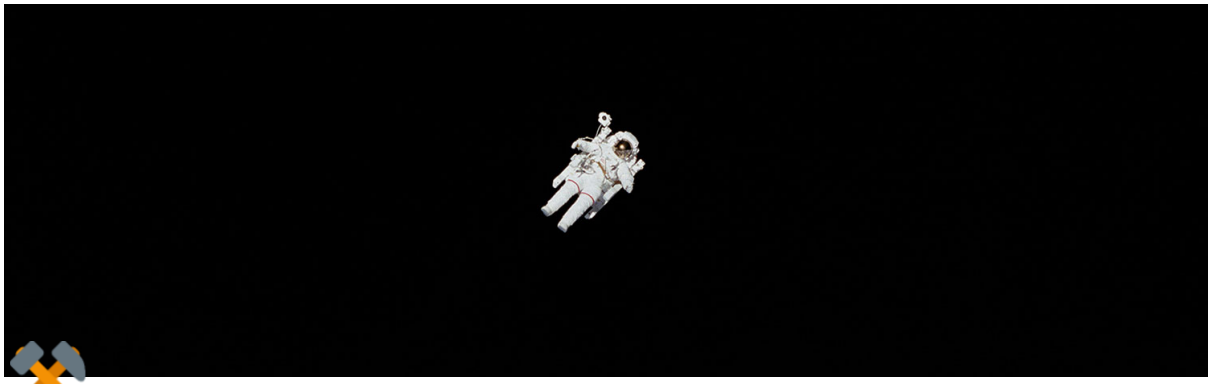


Esercitazione in C #3

Oct. 16

Esercizio 1

Scrivere un programma in C che dichiara una matrice di interi 10 x 10 e chiede un numero intero compreso tra 1 e 100 all'utente (controllare). Tale numero sarà "l'origine degli assi della matrice" (posizione in alto a sinistra). Il resto della matrice dovrà essere costruito realizzando la tabella pitagorica a partire da tale numero, dove ogni elemento è pari al prodotto dell'elemento corrispondente nella prima riga e prima colonna. Visualizzare la matrice ottenuta.

```
#include <stdio.h>
#define R 10 //numero righe
#define C 10 //colonna

int main()
{
  int mat[R][C];
  int origine, i, j;

  do
  {
    printf("Inserisci un numero tra 1 a 100\n");
    scanf("%d", &origine);

    if (origine < 1 || origine > 100)
      printf("Attenzione, inserisci un numero compreso tra 1 e 100");

  } while (origine < 1 || origine > 100);

  for (i = 0; i < C; i++) //la prima riga sarà composta dal numero inserito dall'utente e i 10 numeri successivi
    mat[0][i] = i + origine;

  for (i = 1; i < R; i++) //la prima colonna sarà composta dal numero inserito dall'utente e i 9 numeri successivi (il primo nu
    mat[i][0] = i + origine;

  for (i = 1; i < R; i++)
  {
    for (j = 1; j < C; j++)
    {
      mat[i][j] = mat[0][j] * mat [i][0];
      printf("%d ", mat[i][j]);
    }
    printf("\n");
  }
  return 0;
}
```

❗ **mat[i][i]** → *diagonale* di una matrice.

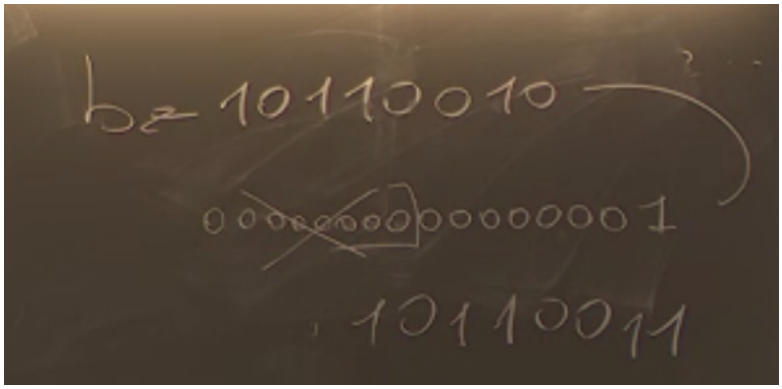
⚠ Somma di un INT e un CHAR

```
char b;
b = 'a';
```

```
b = b+1; //1 è su 16 bit, b è su 8 bit
```

La somma sarà fatta dal compilatore in questo modo:

- Vengono tagliati i primi 8 bit
- Si esegue la somma in complemento a 2 tra il codice ASCII di *b* (che contiene *a*) e il numero 1 su 8 bit (tagliato di 8 bit).
- I risultato sarà il carattere successivo alla **a,** quindi **b.**



Dopo 32 caratteri ASCII si hanno le lettere maiuscole.

Esercizio 2

Scrivere un programma in C che chiede all'utente di inserire due stringhe di al più 40 caratteri e (dopo che sono state interamente inserite) visualizza il numero di vocali uguali in posizione uguale. (si supponga che l'utente immetta stringhe solo con caratteri alfabetici e minuscoli).

```
#include <stdio.h>
#define DIM 40 + 1 //in modo da avere un array da 40 caratteri + il tappo \0
#define n_vocali 5 //grandezza array delle vocali

int main()
{
  char s1[DIM], s2[DIM]; //le due stringhe
  char vocali[n_vocali] = {'a', 'e', 'i', 'o', 'u'}; //Inserisco le vocali nell'array di caratteri
  int cont, i, j, L1, L2; //contatori

  printf("Immetti la prima stringa");
  scanf("%s", s1); //Non va messa la & vicino alla variabile perchè è un array
  printf("Immetti la seconda stringa");
  scanf("%s", s2); //Non va messa la & vicino alla variabile perchè è un array

  L1 = strlen(s1); //Restituisce la lunghezza dell'array al netto del carattere tappo \0
  L2 = strlen(s2);
  cont = 0; //Contatore del numero di vocali

  //METODO 1 - PIU' MACCHINOSO MA PIU' EFFICIENTE, MENO ISTRUZIONI
  /*for (i = 0; i < L1 && i < L2; i++)
    if(s1[i] == 'a') || (s1[i] == 'e') || (s1[i] == 'i') || (s1[i] == 'o') || (s1[i] == 'u')) && (s1[i] == s2[i])
      cont ++;*/

  //METODO 2 - PIU' SEMPLICE
  for (i = 0; i < L1 && i < L2; i++) //Ciclo per la stringa
    for(j = 0; j < n_vocali; j++) //Ciclo per scandire l'array delle vocali
      if (s1[i] == vocali[j] && s1[i] == s2[i])
        cont ++;

  printf ("Il numero di vocali uguali in pos è %d", cont);

  return 0;
}
```