FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

# FONDAMENTI DI PROGRAMMAZIONE

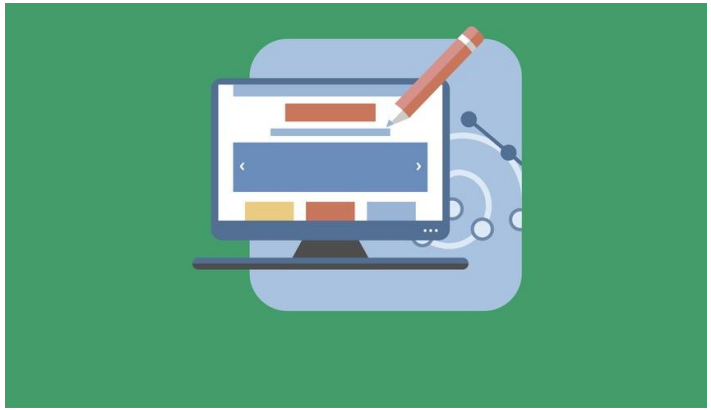
# C / C++

## Docente: Armando Valentino

in collaborazione con:
ISTITUTI TECNICI SUPERIORI
Ministero dell'Istruzione
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

# LINGUAGGIO C++

## ARRAY e MATRICI



Prof. Armando Valentino

POR Piemonte
FSE 2014-2020

# ARRAY

- L'Array è un insieme di elementi tutti dello stesso tipo.
- con l'array possiamo indicare tanti dati dello stesso tipo con un solo nome.
- Gli **elementi si distinguono uno dall'altro attraverso l'indice**

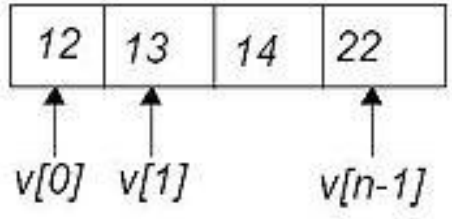
Definizione di un vettore:
**tipo nomeArray [dimensione];**

**Es:**
**int T[10];**
array di interi di 10 elementi

**Esempi:**
**int v[4];**
**char nome[20];**

# Array

Si può immaginare un array come una sorta di contenitore, le cui caselle sono dette **celle** o **elementi**.
Tutte le celle contengono valori dello stesso tipo.
Si parlerà perciò di "array di interi", "array di stringhe", "array di caratteri" e così via




Ciascuna delle celle dell'array è identificata da un valore di **indice**. L'indice è generalmente numerico e i **valori che gli indici possono assumere sono numeri interi *contigui* che partono da 0**

Gli elementi vengono individuati attraverso l'indice.
int v[10]; è un vettore di interi di 10 elementi.
V[0] individua il primo elemento
V[1] individua il secondo elemento
V[9] individua il decimo elemento

# ARRAY

- Un array è un elenco di elementi dello stesso tipo
- Sintassi:
  - *tipo id[dim];*
    *tipo id[dim]={lista_valori};*
    *tipo id[]={lista_valori};*
- Esempi:
  - *double a[25]; //array di 25 double*
    *const int c=2;*
    *char b[2*c]={'a','e','i','o'};*
    *char d[]={'a','e','i','o','u'};*
- Gli elementi di un array di dimensione N sono numerati da 0 a N-1
- Esempio: a[3]=7 assegna il valore 7 al quarto elemento

# Inserimento e prelievo di valori da un Array

- Per assegnare i valori all'array si deve far riferimento all'indice

```
int vet[10];            /* definisce la variabile di nome "vet" come array di 10 elementi interi */
```

```
vet[0] = 0;   /* assegna il valore "0" alla cella di indice 0 */
vet[1] = 1;
vet[2] = 1;
vet[3] = 2;
vet[4] = 3;
```

```
a = vet[0];   // preleva (o legge) il valore della cella di indice 0 e lo assegna alla variabile a
x= vettore[4];          // prende il valore della cella di indice 4 dell'array e lo assegna alla variabile x
```

**Le variabili a e x devono essere dello stesso tipo dei dati dell'array di cui si sta trattando**

L'indice del vettore può essere individuato con una variabile di tipo intero.
L'indice deve sempre far riferimento ad elemento valido, altrimenti si genera un errore n el programma
La dimensione dell'array una volta fissata non può essere modificata

# Inizializzazione di Array

- Gli array si possono inizializzare al momento della dichiarazione

```
int primo[] ={ 33, 55, 74, 22, 14};
Int secondo[6] ={ 33, 55, 74, 22, 14};
```

```
int terzo[5] = { 33, 55, 74, 22};          // l'ultimo elemento è nullo;
int quarto[3] = { 33, 55, 74, 22};          // errore di compilazione;
```

La stringa è un array di caratteri

```
char nome[ ]="testo della stringa";
```

// Conviene non specificare la dimensione perché viene calcolata automaticamente. La dimensione è il numero dei caratteri + 1 perché viene aggiunto il **carattere \0** alla fine che segnala la fine della stringa.

# Array di carattery

## Inizializzazione delle stringhe

Per inizializzare una stringa di caratteri si può usare la stessa notazione usata per gli array:

```
char mystring[ ] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

Abbiamo così inizializzato una stringa (array) di 6 valori di tipo **char**: la parola **Hello** più il carattere nullo **'\0'**.

Possiamo anche inizializzare un array di caratteri usando una **stringa costante**.
Alle stringhe costanti viene sempre aggiunto implicitamente un carattere nullo finale **'\0'.**

```
char mystring [ ] = "Hello";
```

In entrambi i casi la dimensione dell'array **mystring** è di 6 elementi di tipo **char**: i 5 caratteri di **Hello** e il carattere nullo finale ( '\0' ).

```
mystring[0]='H';
mystring[1]='e';
mystring[2]='l';
mystring[3]='l';
mystring[4]='o';
mystring[5] = '\0';
```

# Conversione di array di caratteri in string

La dichiarazione di array di caratteri è quella che si usa in linguaggio C, perché in C **non esiste il tipo string**.
Il tipo string esiste in C++. Nel linguaggio C++ il tipo string è una classe e non un tipo di dati primitivo.
Una stringa in C++ non è altro che un array di caratteri

```
char v[10] ;          //dichiarazione di array di caratteri
```

L'array può essere inizializzato in fase di dichiarazione

```
char v[ ] = "il mio libro";   // in questo caso non conviene specificare la dimensione perché viene
                              // calcolata in automatica (numero dei caratteri+1); il carattere aggiuntivo è "\0"
V="il mio libro";             Errore!!
```

# Conversione di array di caratteri in string

Le stringhe sono array di caratteri, quindi la conversione è automatica

```
char s1[]="ciao mondo";
string s2, s3;
s2=s1;                    // assegna l'array di caratteri  a s2
cout << s2;
s2[5]='m';                // utilizzo di s2 come array di caratteri
s3="ciao universo";
cout << endl<< s3[10];    // stampa 'r'
getchar();                // attende la pressione di un tasto
```

# Conversione di stringhe in array di caratteri

Le stringhe sono array di caratteri, quindi la conversione è automatica

```
// CONVERSIONE DI STRINGHE IN ARRAY DI CARATTERI
string s = “IL MIO LIBRO";                          //assegnazione di valore ad una stringa
cout << "inserisci la frase:"<<endl;
getline(cin, s) ;                                    //input di una stringa da tastiera
 int n = s.length();                                 //calcolo la lunghezza della stringa
char s2[n+1];                                        // dichiaro un array di caratteri della dimensione n+1

strcpy(s2, s.c_str());        // copia della stringa s nell’array di caratteri s2 (c_str() converte la stringa in array di char)

for (int i=0; i<n; i++)
     cout << "\t"<<s2[i];                  // stampa dell’array di char
```

# Caricamento di un array

- Gli elementi di un Array possono essere caricati in modo sequenziale (elemento dopo elemento) oppure casuale specificando l'indice

**Caricamento e stampa sequenziale**

```
#include <iostream>
#define MAX 10                          //dimensione massima dell'array
using namespace std;
int main(){
     int array[MAX];                    //dichiarazione dell'array
     for (int x=0; x<MAX; x++){                    //lettura degli elementi dell'array
          cout<<"Inserisci  l'elemento "<<x+1<<" dell'array: ";
          cin>>array[x];
     }
     for (int i=0; i<MAX; i++){                    // stampa dell'array
          cout<< i<<"-"<<array[i]<<endl;
     }
}
```

# Passaggio di array come parametri

Un array usato come parametro di una funzione **viene sempre passato per riferimento**

Questo permette di risparmiare spazio in memoria

**Sintassi per l'utilizzo di funzioni**

Scrittura della funzione e passaggio di un array con la dimensione; dati[ ] è l'array di interi e dim è la dimensione. Non c'è bisogno di passare l'array con il valore tra parentesi quadre, perché viene passato l'indirizzo del primo elemento.

```
void leggiArray(int dati[], int dim);
```

Nella chiamata della funzione nel mai non si mettono le parentesi quadre, ma solo il nome dall'array

```
leggiArray(dati,dim);
```

Gli array passati per riferimento , sono modificabili all'interno delle funzioni. Se si vuole che un array passato come parametro non sia modificabile in una funzione, si deve far precedere la definizione del parametro dal **qualificatore const** nella dichiarazione della funzione

```
void stampaArray(const int dati[], int dim);
```

# Funzioni Caricamento e Stampa di array

- L'aesercizio precedente può essere scritto con le funzioni per il caricamento e stampa del vettore

Funzione Caricamento sequenziale

```
void leggiVet(double vet[], int dim){
    int x;
    for (x = 0; x < dim; x++){
        printf("Inserire l'elemento di indice %d: ", x);
        scanf("%lf", &vet[x]);
    }
}
```

Funzione Stampa sequenziale

```
void stampaVet(const double vet[], int dim){
    int x;
    printf("Indice Elemento\n");
    for (x = 0; x < dim; x++) {
        printf("%6d %8.0lf\n", x, vet[x]);
    }
}
```

Qualificatore const per impedire la modifica dell'array

# MATRICI

Una matrice è un array bidimensionale in cui:
- Gli elementi sono tutti dello stesso tipo
- Il meccanismo di accesso diretto ai suoi elementi richiede l'uso di due indici

Es: Dichiarazione di una matrice di 200 elementi di 10 righe e 20 colonne

```
int mdati[10][20]
```

L'indice di riga varia da 0 a 9 e l'indice di colonna varia da 0 a 19

Esempi:

```
int  C[4][3];
float  f[M][N];
char  b[2][3]={{'a',  'b',  'c'},{'d',  'e',  'f'}}
int  A[  ][3]=  {  1,2,3,4,5,6,7,8,9  };
```

Il primo numero rappresenta la riga, il secondo numero la colonna

# Passaggio di matrici come parametri

Nel passaggio di una matrice bidimensionale ad una funzione bisogna passare alla funzione il numero di colonne effettivo altrimenti il compilatore non riesce a calcolare dove inizia la riga successiva della matrice.

Esempio:

```
const int N=100;
const int M=150;
int A[N][M], B[N][M], n=75, m=75;
//dichiarazione della funzione
somma(A[][M], B[][M], n, m);
```

Passaggio di un **array multidimensionale ad una funzione**

```
char vett[N1][N2]...[NP] = ...
tipo nomefunz(char vett[][N2]...[NP], ...);
```

Si devono passare alla funzione tutte le dimensioni a partire dalla seconda

# MATRICI

**Esempio.**

Una classe composta da 32 studenti ha sostenuto durante l'anno 5 compiti in classe. Supponiamo di voler scrivere un programma che calcoli la media dei voti ottenuti dagli studenti.

Si potrebbe allora dichiarare una matrice del tipo: ***int A[32][5]*** in cui inserire i voti riportati da ciascun studente in ciascun compito.
Siccome però in una classe ci potrebbero essere più studenti o I compiti potrebbero essere di più, conviene adoperare per maggior generalità la seguente dichiarazione:

*int const R=40;  int const C=8;  int A[R][C];  int n=32, m=5.*

# MATRICI

**Supponiamo di voler scrivere un programma che stampi per ogni studente la somma e la media dei voti ottenuti.**

dichiariamo una matrice A[][] di interi contenente …

su ogni riga i voti di ogni studente (32), …

su ogni colonna i voti di ogni compito

```
int const R=40;
int const C=8;
int A[R][C];
int n=32;
int m=5;
```

| Studente \ Compito | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 9 | 6 | 0 | 7 | 7 | NON USATA | | |
| 1 | 5 | 6 | 6 | 0 | 6 | | | |
| 2 | 8 | 6 | 7 | 0 | 0 | | | |
| 3 | 4 | 6 | 5 | 4 | 4 | | | |
| … | … | … | … | … | … | | | |
| | … | … | … | … | … | | | |
| 32 | 2 | 4 | 4 | 5 | 4 | | | |
| … | NON USATA | | | | | | | |
| 39 | | | | | | | | |

# MATRICI

**Descrizione algoritmo. Strategia risolutiva:**

*Per ogni riga i che rappresenta uno studente*
*Scandire le colonne relative alla riga i e sommare e contare ogni esame il cui risultato è >0*
*Stampare lo studente, il numero di esami e la media*

```
void stampaesami(int mat[][maxcol], int riga, int col){
   int i,j,conta;
   double somma;
   for(int i=0; i<riga; i++){
      conta=0; somma=0;
      for (int j=0; j<col; j++){
         if(mat[i][j] > 0) {
               conta=conta+1;
               somma=somma+mat[i][j];
         }
      }
      if(somma>0){
         cout << "studente "<<i<< " esami superati: "<<conta
               <<" media degli esami: "<<somma/conta<<endl;
      }
   }
}
```

Voto 0 è un compito non consegnato

```
int const R=40;
int const C=8;
int A[R][C];
int n=32;
int m=5;
```

A[3][2] è il voto del 3° compito del 4° studente

| Studente \ Compito | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 9 | 6 | 0 | 7 | 7 | NON USATA | | |
| 1 | 5 | 6 | 6 | 0 | 6 | | | |
| 2 | 8 | 6 | 7 | 0 | 0 | | | |
| 3 | 4 | 6 | 5 | 4 | 4 | | | |
| … | … | … | … | … | … | | | |
| … | … | … | … | … | … | | | |
| 32 | 2 | 4 | 4 | 5 | 4 | | | |
| … | NON USATA | | | | | | | |
| 39 | | | | | | | | |

# Licenza