

# C: Array

## Fondamenti di Programmazione

# Array

E' un tipo di struttura dati che può memorizzare una raccolta **sequenziale** di elementi dello stesso tipo

**Tipo derivato:** deve far riferimento ad un tipo *base*

- Quando si usa un array ?
  - Quando abbiamo necessità di trattare un **insieme omogeneo** di dati !

# Array

E' quindi una **variabile strutturata**, i cui elementi sono tutti dello stesso tipo (*tipo del vettore*)

## Dichiarazione di un array:

- Indicare nome, tipo e lunghezza 💡



```
int vett[10];
```

- Con la dichiarazione riserviamo lo spazio di memoria sufficiente per il nostro array
- In questo esempio, evitiamo di dover dichiarare 10 diverse variabili

# Array

## Accedere agli elementi di un array:

- Tutti gli elementi di un array hanno lo stesso nome:

```
nome_array
```

- Ogni elemento è identificato da un indice:

```
nome_array[indice]
```

Esempio:

```
vett[2] //permette di accedere all'elemento con indice 2
```

# Array

- Il primo elemento di un array ha indice `0` !
- L'ultimo elemento di un array ha indice `N-1`, dove `N` indica la dimensione dell'array

`[0] [1] [2] ... [N-1]`

Tentare di accedere con un indice diverso (fuori dall'intervallo) può non generare un errore 😞

# Array

## Assegnamento

```
int vett[10], num;  
  
vett[0]=5;  
  
num=vett[0]; //assegna 5 a num
```

# Array

Permettono di sviluppare programmi concisi ed efficienti

- Esempio:

```
int voti[10], somma=0, i;  
  
...  
  
for(i=0; i<10; ++i) // <- i parte da 0  
{  
    printf("Inserisci voto: ");  
    scanf("%d",&voti[i]); // <- conserviamo i voti  
    somma += voti[i];  
}
```

# Array

## Considerazione importante!

con `int voti[10];` stiamo dicendo che la lunghezza dell'array sarà 10!

- Se volessimo inserire 20 elementi?
  - Modifica noiosa e rischiosa!
  - Cambiare ogni occorrenza!
  - Possibili diversi significati del numero 10!

Utilizzo della direttiva `#define` (si usa un nome simbolico)

```
#define NVOTI 10  
int voti[NVOTI];
```



# Esempio

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

int main()
{
    int i,n,voti[NVOTI],somma=0;
    float media;
    printf("Quanti voti vuoi inserire? ");
    scanf("%d",&n);
    for(i=0;i<n;i++) {
        printf("\nInserisci voto %d: ",i+1);
        scanf("%d",&voti[i]);
        somma+=voti[i];
    }
    media=(float)somma/n;
    printf("\nLa media vale: %f\n",media);
}
```

# Esempio

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

...

for(i=0;i<n;i++) {
    printf("\nInserisci voto %d: ",i+1);
    scanf("%d",&voti[i]);
    somma+=voti[i];
}

...
```

- Vengono caricati `n` voti e non `NVOTI` !
  - Viene utilizzato meno spazio
  - La scelta avviene a cura del programmatore!

# Esempio

## Controllo dimensione Array

```
#include <stdio.h>
#define NVOTI 10
int main(){
    int i,n,voti[NVOTI],somma=0;
    float media;

    do {
        printf("Quanti voti vuoi inserire? ");
        scanf("%d",&n);
    } while(n<1 || n > NVOTI);
}
```

# Array

## Note

- Non è possibile assegnare un array ad un altro array:

`vett1=vett2;` NON E' AMMESSA!

- Sono ammesse anche altre inizializzazioni:

```
int vett[3]={1,2,3};
```

oppure

```
int vett[]={1,2,3};
```

Nel secondo caso la dimensione è pari a 3 👍

# Array

## Note

- Il C non effettua alcun controllo sui limiti degli array
- In altre parole è possibile inizializzare un array di dimensione N con più di N valori senza avere alcun messaggio di errore in compilazione
- **È compito del programmatore garantire che tutti gli array siano abbastanza grandi da contenere ciò per cui sono stati creati**

# Esercizio

Scrivere un programma che esegua la somma di N valori interi precedentemente inseriti in un array

# Soluzione

```
#include <stdio.h>
#define N 10
int main() {
    int i,n,vett[N],somma=0;
    do {
        printf("Lunghezza vettore:");
        scanf("%d",&n);
    } while(n<1 || n>N);
    for(i=0;i<n;++i) {
        printf("\nInserisci numero %d: ",i+1);
        scanf("%d",&vett[i]);
        somma+=vett[i];
    }
    printf("\nLa somma vale: %d\n",somma);
}
```

# Esercizio

Scrivere un programma che esegua la copia di un array in un nuovo array



# Soluzione

```
#include <stdio.h>
#define DIM 100

int main()
{
    int arr1[DIM], arr2[DIM];
    int i, n;

    printf("Numero elementi del primo array:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("elemento - %d : ",i+1);
        scanf("%d",&arr1[i]);
    }
}
```

# Soluzione

```
for(i=0; i<n; i++)
    arr2[i] = arr1[i];

printf("\nPrimo array:\n");
for(i=0; i<n; i++)
    printf("%3d", arr1[i]);
printf("\n\nSecondo array:\n");
for(i=0; i<n; i++)
    printf("%3d", arr2[i]);
}
```

# Controllo dimensione array

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

int main()
{
    int i,n,vett[N],somma=0;

    do {
        printf("Lunghezza vettore? ");
        scanf("%d",&n);
    } while(n<1 || n>N);

    for(i=0;i<n;++i) {
        printf("\nInserisci numero %d: ",i+1);
        scanf("%d",&vett[i]);
        somma+=vett[i];
    }
    printf("\nLa somma vale: %d\n",somma);
}
```

# Esercizio

Scrivere un programma che memorizzi  $N$  voti e determini la media, il voto maggiore e il voto minore

# Soluzione

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

int main()
{
    int i,n,voti[NVOTI],somma=0,min,max;
    float media;

    do {
        printf("Quanti n vuoi inserire? ");
        scanf("%d",&n);
    } while(n<1 || n>NVOTI);

    for(i=0;i<n;++i) {
        printf("\nInserisci voto %d: ",i+1);
        scanf("%d",&voti[i]);
        somma+=voti[i];
    }
    media=(float)somma/n;
    printf("\nLa media vale: %.1f\n",media);
}
```

# Soluzione

```
min=voti[0];
max=voti[0];
for(i=0;i<n;++i) {
    if(voti[i]<min) min=voti[i];
    if(voti[i]>max) max=voti[i];
}
printf("\nIl voto piu' alto e': %d\n",max);
printf("\nIl voto piu' basso e': %d\n",min);
```

# Esercizio

Scrivere un programma che calcoli la frequenza di N voti compresi tra 0 e 10

# Soluzione

```
#include <stdio.h>
#define VOTI 11

int main()
{
    int i, counter_voti[VOTI], voto, n_voti;

    for(i=0; i<VOTI; ++i)
        counter_voti[i]=0;

    printf("Quanti voti devi registrare?\n");
    scanf("%d", &n_voti);
```



# Soluzione

```
for(i=0;i<n_voti;++i) {  
    do {  
        printf("Inserisci voto %d: \n",i+1);  
        scanf("%d",&voto);  
    } while(voto<0 || voto>10);  
    ++counter_voti[voto];  
}  
printf("Voto; Frequenza\n");  
for(i=0;i<VOTI;++i)  
    printf("%3d -> %10d\n",i,counter_voti[i]);  
}
```

# Soluzione (tra 18 e 30)

```
#include <stdio.h>
#define VOTI 13
#define VMIN 18
#define VMAX 30

int main()
{
    int i, counter_voti[VOTI], voto, n_voti;

    for(i=0; i<VOTI; ++i)
        counter_voti[i]=0;
    printf("Quanti voti devi registrare?\n");
    scanf("%d", &n_voti);
```

## Soluzione (tra 18 e 30)

```
for(i=0;i<n_voti;++i) {  
    do {  
        printf("Inserisci voto %d: \n",i+1);  
        scanf("%d",&voto);  
    } while(voto<VMIN || voto>VMAX);  
    ++counter_voti[voto-VMIN];  
}  
printf("Voto; Frequenza\n");  
for(i=0;i<VOTI;++i)  
    printf("%3d -> %10d\n",i+VMIN,counter_voti[i]);  
}
```

# Esercizio

Scrivere un programma che esegua la ricerca di un elemento (letto da tastiera) in un array

# Soluzione - ricerca sequenziale

```
#include <stdio.h>
#define DIM 100
int main() {
    int v[DIM], n, i, elem, trovato=0;
    do {
        printf("\nInserisci dimensione array: ");
        scanf("%d", &n);
    } while(n<1 || n>DIM);
    printf("\nInserisci i %d elementi: ",n);
    for(i=0;i<n;i++) {
        printf("\nelemento di indice - %d: ",i);
        scanf("%d",&v[i]);
    }
    printf("\nInserisci elemento da ricercare: ");
    scanf("%d",&elem);
    i=0;
    while(trovato!=1 && i<n) {
        if(elem==v[i]) trovato=1;
        ++i;
    }
    if(trovato==1) printf("\nElemento in pos %d",i);
    else printf("\nElemento non presente");
}
```

# Esercizio

Scrivere un programma che ordini un array (in ordine crescente)

# Soluzione - bubble sort

```
#include <stdio.h>
#define DIM 100

int main()
{
    int v[DIM], n, i, j, tmp;

    do {
        printf("Inserisci dimensione array: \n");
        scanf("%d", &n);
    } while(n<1 || n>DIM);
    printf("Inserisci i %d elementi:\n",n);
    for(i=0;i<n;i++) {
        printf("elemento di indice - %d : ",i);
        scanf("%d",&v[i]);
    }
}
```

# Soluzione

```
for(i=0; i<n-1; i++) {  
    for(j=0; j<n-1; j++) {  
        if(v[j] > v[j+1]) {  
            tmp = v[j];  
            v[j] = v[j+1];  
            v[j+1] = tmp;  
        }  
    }  
}  
printf("\nElementi in ordine crescente:\n");  
for(i=0; i<n; i++)  
    printf("%d  ", v[i]);  
}
```



# Matrici

**Vettori:** array monodimensionali

**Matrici:** array bidimensionali

- Il C permette di definire array di qualsiasi dimensione

Per accedere ad un elemento di una matrice: `M[i][j]`

- Il primo indice identifica la riga, il secondo la colonna

Dichiarazione: `int M[NR][NC];`

# Esempio

```
#include <stdio.h>
#define DIM 3
int main()
{
    int m[DIM][DIM], i, j;

    printf("Inserisci elementi nella matrice:\n");
    for(i=0; i<DIM; i++) {
        for(j=0; j<DIM; j++) {
            printf("elemento - [%d], [%d] : ", i, j);
            scanf("%d", &m[i][j]);
        }
    }
    printf("\nLa matrice e': \n");
    for(i=0; i<DIM; i++) {
        printf("\n");
        for(j=0; j<DIM; j++)
            printf("%d\t", m[i][j]);
    }
}
```

# Esercizio

Scrivere un programma che calcoli la somma di due matrici quadrate di dimensione  $N \times N$

# Soluzione

```
#include <stdio.h>
#define DIM 50

int main()
{
    int m1[DIM][DIM], m2[DIM][DIM], sum[DIM][DIM], i, j, n;

    do {
        printf("Dimensione matrice quadrata: \n");
        scanf("%d", &n);
    } while(n<1 || n>DIM);

    printf("Elementi della prima matrice:\n");
    for(i=0; i<n; i++) {
        for(j=0; j<n; j++) {
            printf("elemento - [%d], [%d] : ", i, j);
            scanf("%d", &m1[i][j]);
        }
    }
}
```

```

printf("Elementi della seconda matrice:\n");
for(i=0;i<n;i++) {
    for(j=0;j<n;j++) {
        printf("elemento - [%d],[%d] : ",i,j);
        scanf("%d",&m2[i][j]);
    }
}

for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        sum[i][j]=m1[i][j]+m2[i][j];

printf("\n\nLa somma vale: \n");
for(i=0;i<n;i++){
    printf("\n");
    for(j=0;j<n;j++)
        printf("%d\t",sum[i][j]);
}
printf("\n\n");
}

```

# Esercizio

Scrivere un programma che calcoli la somma delle righe e delle colonne di una matrice quadrata di dimensione  $N \times N$

# Soluzione

```
#include <stdio.h>
#define DIM 100
int main()
{
    int i,j,k,m1[DIM][DIM],rsum[DIM],csum[DIM],n;

    do {
        printf("Dimensione matrice quadrata: \n");
        scanf("%d", &n);
    } while(n<1 || n>DIM);

    printf("Inserisci elementi della matrice:\n");
    for(i=0;i<n;i++) {
        for(j=0;j<n;j++) {
            printf("elemento - [%d],[%d] : ",i,j);
            scanf("%d",&m1[i][j]);
        }
    }
}
```

# Soluzione

```
for(i=0; i<n; i++) {  
    rsum[i]=0;  
    for(j=0; j<n; j++)  
        rsum[i]=rsum[i]+m1[i][j];  
}  
  
for(i=0; i<n; i++) {  
    csum[i]=0;  
    for(j=0; j<n; j++)  
        csum[i]=csum[i]+m1[j][i];  
}
```



# Soluzione

```
printf("La somma righe e colonne vale:\n");
for(i=0;i<n;i++) {
    for(j=0;j<n;j++)
        printf("% 4d",m1[i][j]);
    printf("% 8d",rsum[i]);
    printf("\n");
}
printf("\n");
for(j=0;j<n;j++) {
    printf("% 4d",csum[j]);
}
printf("\n\n");
}
```