# Dati strutturati: gli array

Prof. Salvatore Venticinque
Prof. Massimiliano Rak

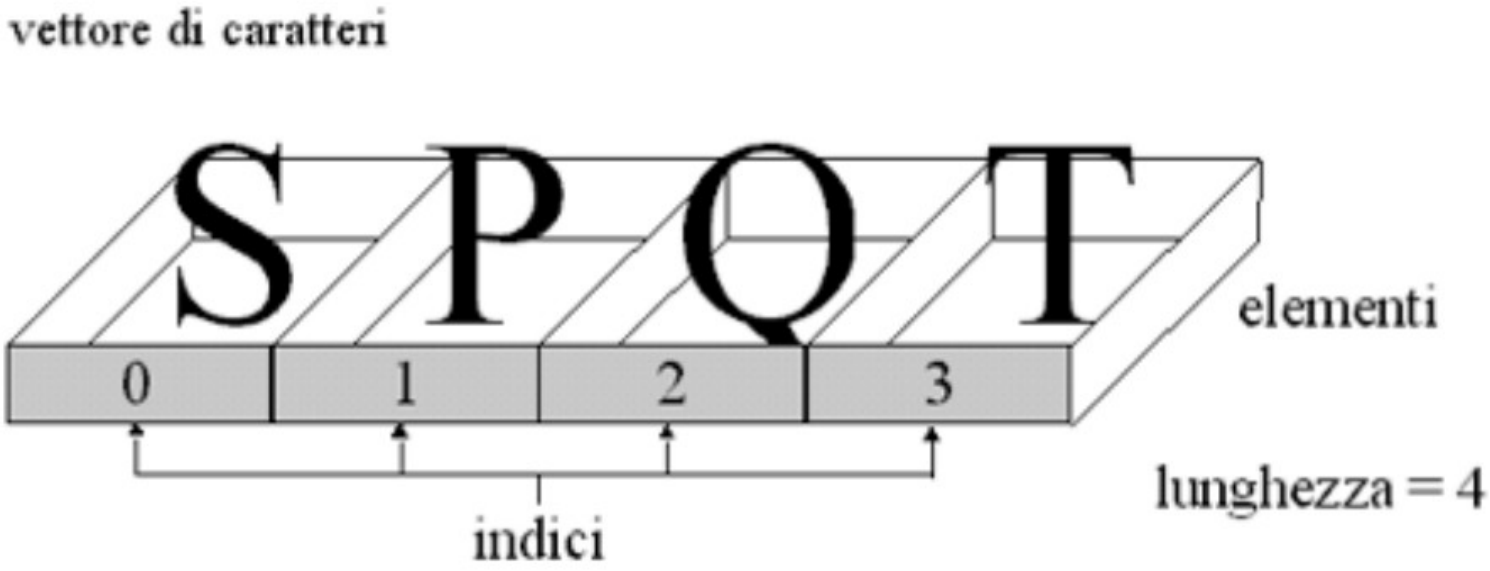
# Tipi strutturati

- Composizione di tipi semplici (char, int, long, float, ...)
- Es.
  - Sequenza
  - Record
  - Stack
  - Lista
  - Coda
  - Albero
  - Grafo ...

# Array

- Insieme omogeneo di dati
- Variabile di tipo array è:
  - strutturata
  - può memorizzare più valori tutti dello stesso tipo.
  - composizione sequenza
- Ogni elemento dell'array:
  - contiene un unico dato
  - è individuato da un numero progressivo, detto indice, che specifica la posizione all'interno del vettore
  - l'indice può assumere valori in [0,n-1]
  - La base dell'array è sempre zero.
  - La dimensione massima ***n*** dell'array è detta lunghezza, o dimensione

# Array


vettore di caratteri
S P Q T
elementi
0
1
2
3
lunghezza = 4
indici

# Dichiarazione array

- Il tipo dei dati contenuti nel vettore viene detto tipo del vettore,
- Per la dichiarazione di una variabile array devono essere definiti
  - il nome
  - il tipo
  - a dimensione

Es:

- int  a[6]

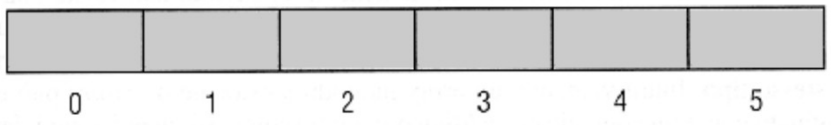


Figura 4.2 Struttura dell'array `a[6]`

# Utilizzo array

int a[6]

l'indice può quindi assumere i valori: 0, 1, 2, 3, 4,5.

Le istruzioni:

a[0] = 71;

a[1] = 4;

assegnano al ***primo*** elemento del vettore a il valore 71 e al ***secondo*** 4.

a[3] = b;

copia il ***valore*** della variabile b al ***quarto*** elemento

*N.B a[3] e b sono due diverse variabili che contengono lo stesso valore, ma I cui contenuti possono cambiare indipendentemente nelle seguenti istruzioni*

# Inizializzazione array

Per inizializzare l'array a tempo di compilazione:

valori tra parentesi graffe, separati da una virgola:

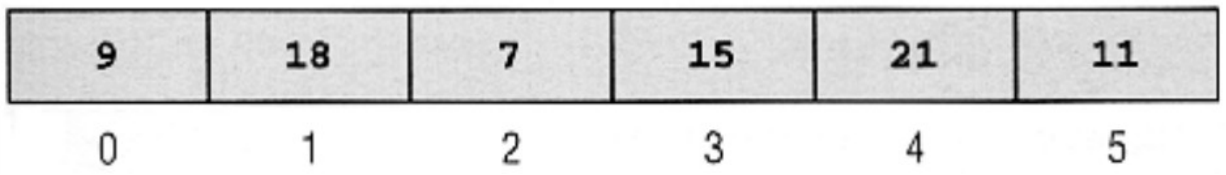
```
int voti[6] = {11, 18, 7, 15, 21, 9};
```

Nel seguente caso il compilatore deduce la dimensione

```
int voti[] = {11, 18, 7, 15, 21, 9};
```

# Inizializzazione

```
/* Inizializzazione dell'array */
for(i=0; i<6; i++) {
        printf("Inser. intero: ");
        scanf("%d", &a[i]);
}
```



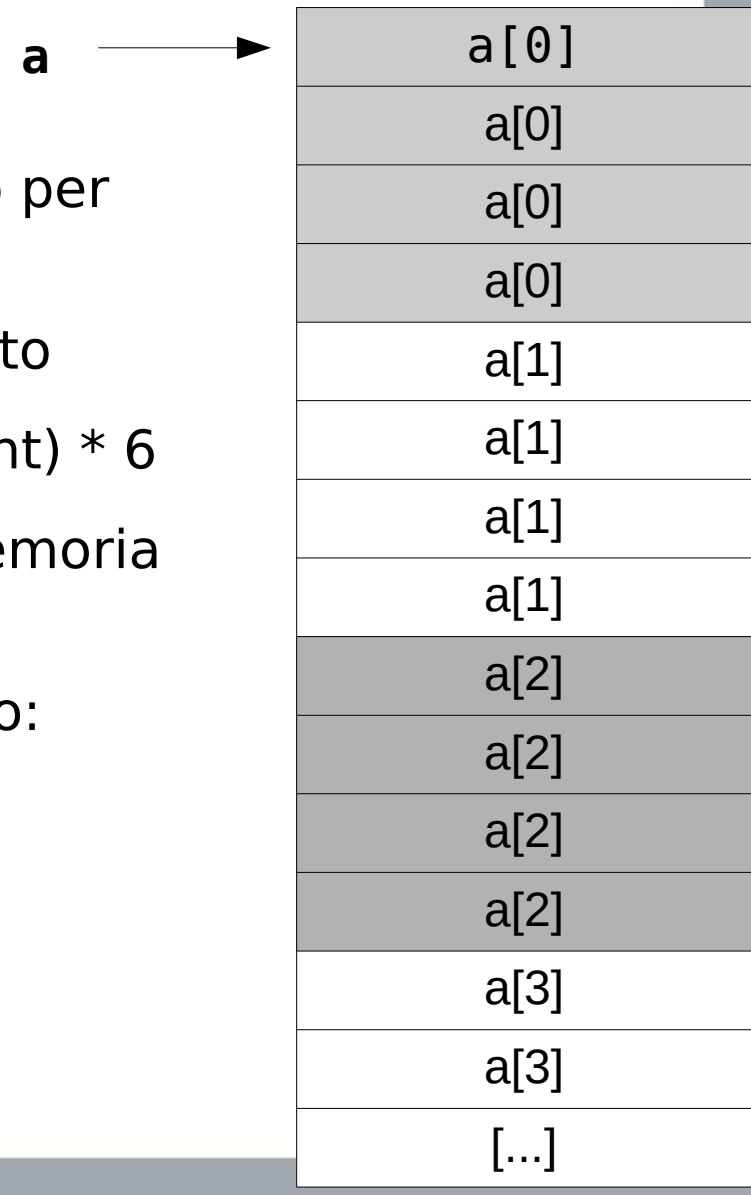
# Stampa vettore

```
/* Stampa dell'array */
for(i=0; i<6; i++) {
        printf("%d ", a[i]);
}
        printf("\n");
```

# Architettura hardware

int a[6]

- Il tipo definisce quanti byte occorrono per memorizzare ogni elemento
- Il nome è l'indirizzo del primo elemento
- Spazio di memoria necessario: sizof(int) * 6
- a[3]  indica il valore all'indirizzo di memoria a+3*sizeof(int)
- &a[3] è l'indirizzo del quarto elemento: a+3*sizeof(int)



# In una memoria a 32 bit

| | | | |
|---|---|---|---|
| a[0] | a[0] | a[0] | a[0] |
| a[1] | a[1] | a[1] | a[1] |
| a[2] | a[2] | a[2] | a[2] |
| a[3] | a[3] | a[3] | a[3] |
| a[4] | a[4] | a[4] | a[4] |
| a[5] | a[5] | a[5] | a[5] |
| Altri dati | Altri dati | Altri dati | Altri dati |
| Istruzione | Istruzione | Istruzione | Istruzione |
| Istruzione | Istruzione | Istruzione | Istruzione |

Il compilatore deve conoscere la dimensione massima per riservare spazio in memoria !!!

# Riempimento

- Il riempimento:
  - Rappresenta lo spazio effettivamente utilizzato
  - è numero di elementi inserito nell'array
  - <= dimensione massima
  - A differenza della dimensione può cambiare durante l'esecuzione

```
#define n 6

int main()
{
   float v[n];
   int r,i;
   do{
      printf(“inserisci riempimento (<=%d): ”,n);
      scanf(“%d”,&r);
      }while((r<0) || (r>n));
   for(i=0;i<r;i++)
      scanf(“%f”,&v[i]);
}
```

# Somma degli elementi

```
#define n 10
int main(){
   int v[n],r,i,somma;
   // lettura riempimento e elementi
   somma = 0;
   for(i=0;i<r;i++)
     somma = somma+v[i];
   printf("somma=%d",somma);
}
```

# Errore Array

```
#include<stdio.h>
int main(){
     int v[20];
     int i = 0;
     int n;

     while(i<n)
       printf("%d\n",v[i++]);
  }
```

```
Quanto vale n?
Se n troppo grande : Segmentation Fault!!!!
```

# Altri algoritmi

- Ricerca elemento
- Ricerca posizione elemento
- Massimo
- Minimo
- Media
- Ordinamento !!!
- Inversione
- Prodotto scalare 2 vettori

# Stringhe

- Una stringa è un vettore di caratteri
- Dichiarazione:

  char s[20]

- Dichiarazione e inizializzazione:

  char s1[20] =”macchina”

  char s2[]  = “macchina”

N.B.:

- La  stringa s1:
  - Dimensione massima: 20
  - Riempimento: 9
- La stringa s2:
  - Dimensione massima 9

# Stampa di una stringa

```
#include <stdio.h>

int main()
{
    char s1[] = "macchina";
    char s2[20] = "macchina";

    printf("%s\n",s1);
    printf("%s\n",s2);

}
```

**%s** stampa tutti I caratteri del vettore fino al carattere di fine stringa: byte 0

# Lettura Stringa

```
#include <stdio.h>

int main()
{
    char s[20];

    prtinf(“inserisci stringa\n”);
    scanf("%s",s);

}
```

# Lunghezza di una stringa

```
#include <stdio.h>


int main()
{
	char s[20] = "macchina";
	int len=0;
	while(s[len]!='\0')
		len++;
	printf("len=%d",len);
}
```

Problema ?????

# Lunghezza di una stringa

```
#include <stdio.h>


int main()
{
      char s[20] = "macchina";
      int len=0;
      while((len<20) &&  (s[len]!='\0'))
            len++;
      if(len<20)
        printf("len=%d",len);
      else
        printf(“fine stringa non trovato\n”);
}
```

Problema risolto

# Lettura Stringa

scanf legge fino al primo spazio

Utilizzare gets per leggere stringhe con spazi:

```
char s[20];
gets(s);
```

C'è un problema ...

# fgets

- **fgets** è una funzione pensata per i file

- Può essere utilizzata per leggere da tastiera
- Occorre specificare quanti caratteri leggere

Esempio:

```
char s[5]
  fgets(s,5,stdin);
  printf("%s",s)
```

# Esercizi stringhe

- Confronto stringhe
- Sottostringa
- Concatenazione stringhe
- Conteggio parole
- Eliminazione spazi
- Maiuscole2Minuscole