

## Unità didattica: Strutture dati statiche

[1-T]

**Titolo:** Richiami sulle principali strutture dati statiche  
(array, record)

Argomenti trattati:

- ✓ Richiami sul tipo di dato strutturato
- ✓ Tipo di dato primitivo e tipo di dato derivato
- ✓ Richiami sul tipo array ed esempio in C
- ✓ Richiamo sul tipo record ed esempio in C

Prerequisiti richiesti: fondamenti del linguaggio C, tipi di dati strutturati elementari (array, record)

# dato strutturato

=

**insieme di informazioni**

- ⊗ **logicamente collegate;**
- ⊗ **identificate da un unico nome**

***Esempi:***

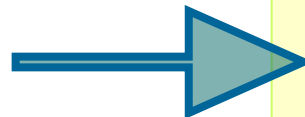
**mazzo di carte**



**Informazioni di  
ogni componente**

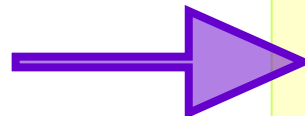
*seme, colore, valore*

**abbonato al telefono**



*nome, indirizzo,  
numero telefonico*

**utente ENEL**



*nome, contratto,  
consumo, pagamenti*

# Ogni **tipo di dato strutturato** stabilisce:

- ❖ se il numero di componenti è **fisso** (tipo di dato **statico**) oppure è **variabile** (tipo di dato **dinamico**);
- ❖ il tipo delle componenti;
- ❖ la modalità di accesso alle componenti;
- ❖ l'eventuale possibilità di inserimento/eliminazione di componenti;
- ❖ l'eventuale ordinamento (collegamento logico) delle componenti.

*I linguaggi di programmazione mettono a disposizione alcuni **tipi di dato** come **primitivi** e la possibilità di crearne di nuovi (**tipi di dato derivati**)*

I **tipi (di dato) primitivi** sono “gestiti” nel linguaggio stesso, cioè sono possibili, direttamente nel linguaggio, operazioni quali:



accesso alle informazioni,  
modifica,  
inserimento / eliminazione.

Per i **tipi di dato derivati**, il programmatore deve esprimere il **nuovo tipo** tramite quelli **primitivi** e realizzare le procedure per le operazioni sugli oggetti del nuovo tipo.

Il tipo di dato primitivo più comune è l'**array**.

Il **tipo di dato array** stabilisce che:

- ❖ il numero di componenti è fisso (tipo di dato **statico**);
- ❖ le componenti devono essere tutte dello stesso tipo;
- ❖ l'accesso alle componenti è **diretto** tramite indici (legati alla posizione delle componenti nella struttura).

# Esempio:

come si possono aggiungere informazioni in un array ordinato

1. Bisogna sovradimensionare l'array (spreco di memoria!)



Per ogni inserimento ...

2. Bisogna slittare le componenti (inefficiente!)

A	
D	
E	
M	
N	

A	
D	
E	
M	
N	

Analoghi slittamenti sono necessari, quando si elimina una componente, per ricompattare l'array

# Esempio 1: array in C

```
type array_name[size1][size2]...
```

```
int M[10];  
float A[3][3][3];  
char T[80][24];  
:  
M[i]=i;  
A[i][j][h]=h/(float)(1+i+j);  
if (T[h][k]==NULL) ...
```


# Il tipo di dato record stabilisce che:

- ❖ il numero di componenti è fisso (tipo di dato **statico**);
- ❖ le componenti (**campi**) possono essere di tipo diverso;
- ❖ l'accesso alle componenti è diretto tramite il nome.



## Esempio 2: record in C

```
struct struct_name  
{type c_1;  
  type c_2;  
  :  
  type c_s;  
};
```



oppure equivalentemente

```
typedef struct { . . . } struct_name;
```

tipo noto

identificatore  
nuovo tipo

```
enum MESI {Gen=1, Feb, Mar, ... , Nov, Dic};
```

```
...
```

```
struct DATA
```

```
{unsigned short giorno;
```

```
enum MESI mese;
```

```
char *anno;
```

```
} oggi, domani;
```

```
⋮
```

```
oggi.giorno=19;
```

```
oggi.mese=Mar;
```

```
(char *) (oggi.anno) = "2002";
```

```
domani=oggi;
```

```
domani.giorno=oggi.giorno+1;
```

```
typedef struct  
{unsigned short giorno;  
enum MESI mese;  
char *anno;  
} DATA;  
DATA oggi, domani;
```

