

# Lezione 12

## Appunti di Davide Scarlata 2024/2025


 Prof: Michele Garetto

 Mail:  [michele.garetto@unito.it](mailto:michele.garetto@unito.it)

 Corso:  C

 [Moodle corso C](#)

 [Moodle Lab matricole dispari](#)

 Data: 9/05/2025

---

## Union

```
typedef union {  
    tipox x;  
    tipoy y;  
} u;
```

## differenza tra union e struct

- union: occupa la dimensione in byte del tipo più grande

- struct: occupa la somma in byte delle dimensioni dei tipi

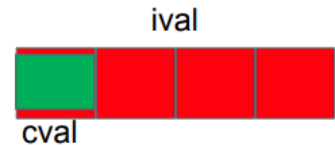
## USO DELLA MEMORIA

- Attenzione alla differenza tra **struct** e **union**:

```
struct mystruct {
    char cval;
    int ival;
} s;
```

```
union myunion {
    char cval;
    int ival;
} u;
```

- In memoria:



## A cosa servono le union?

In alcuni contesti applicativi è necessario gestire dati che hanno una stessa valenza ma che possono essere formattati secondo standard diversi: le union sono uno strumento che permette di definire tipi/variabili che possono accogliere tutti i diversi formati, riducendo la quantità di memoria occupata rispetto alle struct.

## Esempio di utilizzo delle union

```
enum nazione { ITA, USA }; // etichette degli standard usati
typedef struct TaggedUnion {
    enum nazione paese; // identifica il caso
    union { // tipi che dipendono dal caso
        nomeITA NI;
        nomeUSA NU;
    }
    n;
} nome;
```

## variabili puntatore a funzione

sintassi:

```
tipo_restituito (*nome)(tipo1, tipo2, ...);
```

uso di variabili puntatore a funzione:

```
int risultato;  
risultato = (*nome)(arg1, arg2, ...);
```

## dichiarazione di una funzione che restituisce un puntatore a funzione

```
typedef struct{  
    void *dato;  
    opcread creadato;  
    opstampa stampa;  
    opconfronta confronta;  
    tree left, right;  
}
```

come si dichiarano queste variabili?

```
typedef void *(*opcread)();  
typedef void (*opstampa)(void *a);  
typedef int (*opconfronta)(void *a, void *b);
```

```
tree t=(tree)malloc(sizeof(struct nodo));  
t->creadato = crea;  
t->stampa = stampa;  
t->confronta = confronta;  
t->left = NULL;  
t->right = NULL;  
t->dato = t->creadato();  
t->stampa(t->dato);
```

utilizzo di sprintf (stampa su stringa)

```
char s1[n];  
char s2[n];  
scanf("%s", s1);  
scanf("%s", s2);
```

```
risultato = (char *)malloc(sizeof(char)*(n1+n2+1));  
sprintf(risultato, "%s%s", s1, s2);
```