

I RECORD



Le strutture o record

I record o strutture sono strutture dati statiche non omogenee cui si fa riferimento usando un unico identificatore.

Il **tipo** di dato struttura deve essere dichiarato, in modo da definire quanti e quali dati dovranno essere aggregati nella struttura.

Dichiarazione del tipo record

```
struct nome_struttura{  
    tipo_variabile1 nome_variabile1;  
    tipo_variabile2 nome_variabile2;  
    ... };
```

Ogni singola variabile appartenente alla struttura è detta campo della struttura.

Dichiarazione del tipo record

```
struct studente{  
    int matricola;  
    string nome;  
    string cognome;  
    float media;  
    char corso;};
```

matricola	nome	cognome	media	corso
Intero	Stringa	Stringa	Float	Char

Visibilità del tipo record

```
#include < ... >
```

```
struct ...
```

← **Ambiente globale.**

Il tipo struct è visibile ovunque nel programma.

```
main()
```

```
{ struct ...  
... }
```

Ambiente locale della funzione.

Il tipo struct è visibile nella funzione.

```
fun()
```

```
{ struct ...  
... }
```

Dichiarazione di variabile di tipo record

La dichiarazione delle variabili di tipo struttura può essere fatta contemporaneamente alla dichiarazione del tipo

```
struct nome_struttura{  
    tipo_variabile1 nome_variabile1;  
    tipo_variabile2 nome_variabile2;  
    ...; } s, t;
```

oppure separatamente

```
struct nome_struttura s, t;
```

Assegnazione valori a record

Per assegnare i valori ad una struttura è necessario assegnarli ai singoli campi che la compongono.

E' possibile assegnare i valori in modo diretto:

```
struct nome_struttura{  
    tipo_variabile1 nome_variabile1;  
    tipo_variabile2 nome_variabile2;  
    ...; } s, t;  
  
s.nome_variabile1=...;  
  
s.nome_variabile2=...;  ....
```

Assegnazione valori a record

```
struct studente{    int matricola;  
                   string nome;  
                   string cognome;  
                   float media;  
                   char corso;}s;
```

```
s.matricola=1234;  
s.nome="Mario";  
s.cognome="Rossi";  
s.media=28.5;  
s.corso='B';
```

**L'ordine di assegnazione
dei valori ai campi è
indifferente.**

Assegnazione valori a record

E' possibile assegnare i valori dei campi di una struttura acquisendoli da tastiera:

```
struct nome_struttura{  
    tipo_variabile1 nome_variabile1;  
    tipo_variabile2 nome_variabile2;  
    ...; }s, t;
```

```
cin >> s.nome_variabile1;
```

```
cin >> s.nome_variabile2;
```

```
...
```

Visualizzazione valori di un record

Per visualizzare i valori di una struttura è necessario visualizzare i valori dei singoli campi.

```
struct nome_struttura{  
    tipo_variabile1 nome_variabile1;  
    tipo_variabile2 nome_variabile2;  
    ...; }s;  
  
cout << s.nome_variabile1;  
  
cout << s.nome_variabile2;  
  
...
```

Copia dei valori di un record

E' possibile effettuare la copia diretta dei valori contenuti in una struttura, dentro una seconda struttura, del medesimo tipo.

```
struct nome_struttura{
```

```
    tipo_variabile1 nome_variabile1;
```

```
    tipo_variabile2 nome_variabile2;
```

```
    ...; }s, t;
```

```
s=t;
```

Vettori di record

E' possibile generare array di tipo struttura.

Dopo aver dichiarato il tipo struttura si dichiara l'array:

```
struct nome_struttura nome_vettore[dimensione];
```

```
struct studente v[3];
```

	matricola	nome	cognome	media	corso
0	Intero	Stringa	Stringa	Float	Char
1					
2					

Assegnazione valori a vettori di record

E' necessario assegnare il valore ai singoli campi delle singole componenti.

```
for(i=0;i<N:i++)  
    {cin >> v[i].matricola;  
      cin >> v[i].nome;  
      cin >> v[i].cognome;  
      cin >> v[i].media;  
      cin >> v[i].corso;}
```

Visualizzazione valori di vettori di record

E' necessario visualizzare il valore dei singoli campi delle singole componenti.

```
for(i=0;i<N;i++)  
    {cout << v[i].matricola;  
      cout << v[i].nome;  
      cout << v[i].cognome;  
      cout << v[i].media;  
      cout << v[i].corso; }
```

Passaggio di record a funzione

Un record può essere passato per valore o per riferimento, attraverso l'uso dell'operatore &.

```
fun(a)
```

```
{printf("%d", a.matricola);}
```

	matricola	nome	cognome	media	corso
s	2738	Mario	Rossi	25.8	B

```
main()
```

```
{struct studente s;
```

```
...
```

```
fun(s);}
```

2738

Passaggio di record a funzione

Un record può essere passata per valore o per riferimento, attraverso l'uso dell'operatore &.

```
fun(struct studente &a)
{a.nome="Marco";}
```

```
main()
{struct studente s;
fun(s);
...}
```

	matricola	nome	cognome	media	corso
s	2738	Marco	Rossi	25.8	B

↑
a

Passaggio di record a funzione

Un vettore di record sarà sempre passato per riferimento.

```
fun(struct studente a[2])  
{ a[0].nome = "Marco";}
```

```
main()  
{ struct studente s[2];  
...  
fun(s);}
```

a

s

	matricola	nome	cognome	media	corso
	2738	Marco	Rossi	25.8	B
	6259	Piero	Bianchi	28	A

Passaggio di struttura a funzione

Un vettore di record sarà sempre passato per riferimento.

```
fun(struct studente a[2])  
{ a[1].nome = "Marco";}
```

```
main()  
{ struct studente s[2];  
  ...  
  fun(s);}
```

	matricola	nome	cognome	media	corso
s	2738	Mario	Rossi	25.8	B
	6259	Marco	Bianchi	28	A

Diagram illustrating the passing of a structure array by reference. The array `s` is shown with two records. The first record has matricola 2738, nome Mario, cognome Rossi, media 25.8, and corso B. The second record has matricola 6259, nome **Marco**, cognome Bianchi, media 28, and corso A. An arrow labeled `a` points to the first record, and another arrow labeled `a` points to the second record, indicating that the function receives a reference to the array.