

Titolo unità didattica: Introduzione al linguaggio C

[03]

Titolo modulo : Puntatori in C

[04-C]

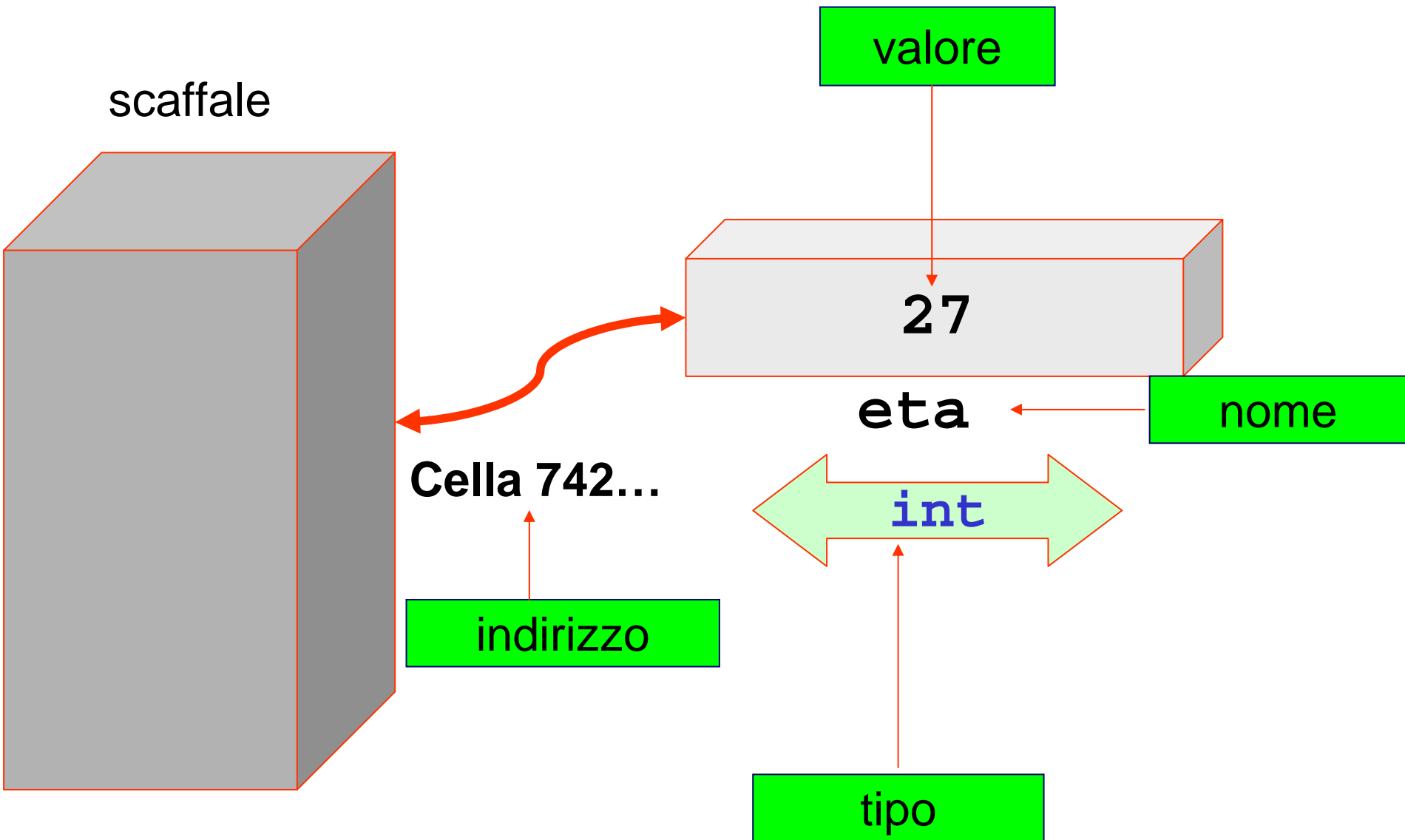
Accesso ai valori delle variabili attraverso gli indirizzi di memoria in C

Argomenti trattati:

- ✓ indirizzi di memoria delle variabili C
- ✓ puntatori in C
- ✓ operatore di indirizzo in C
- ✓ operatore di dereferenziazione in C

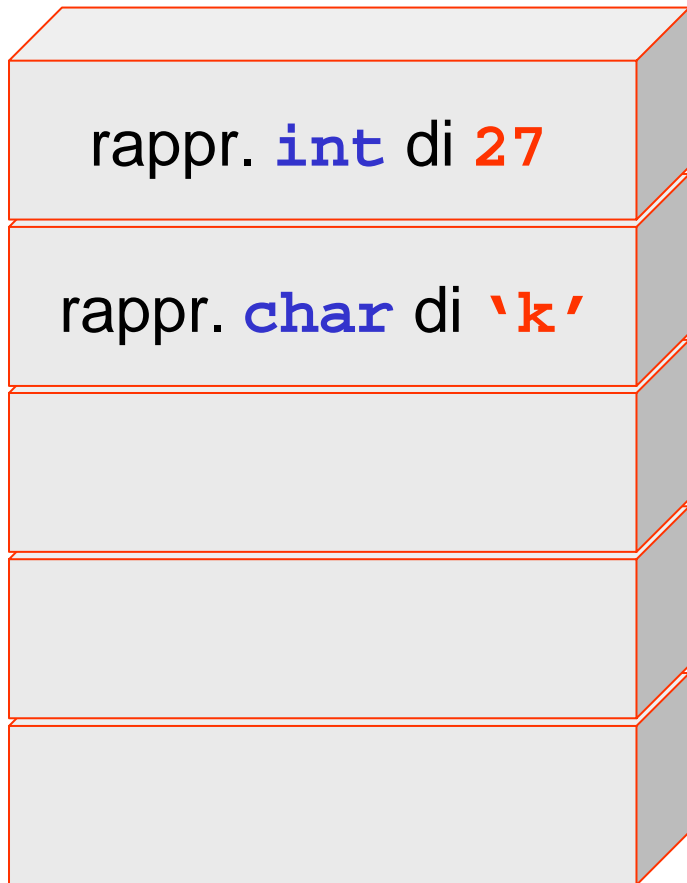
Prerequisiti richiesti: AP-03-03-C

metafora della scatola etichettata in uno scaffale



celle di memoria, indirizzi, valori memorizzati, variabili

memoria



indirizzo

01

02

03

04

05

```
int eta;  
char lettera;  
eta = 27;  
lettera = 'k';
```

alla variabile `eta` viene associata la cella di indirizzo 01
alla variabile `lettera` viene associata la cella di indirizzo 02
la rappresentazione di `27` viene memorizzata nella cella di indirizzo 01
la rappresentazione di `'k'` viene memorizzata nella cella di indirizzo 02

celle di memoria, indirizzi, valori memorizzati, variabili

memoria



indirizzo

01

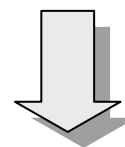
02

03

04

05

in C è possibile conoscere l'indirizzo della cella associata a una variabile



operatore `&`,
operatore indirizzo di

`&eta`
`&lettera`

indicano:
`indirizzo` della variabile `eta`
`indirizzo` della variabile `lettera`

in C è possibile assegnare l'indirizzo di una variabile a una **variabile puntatore**

un **puntatore** è una variabile che contiene l'indirizzo di un'altra variabile

dichiarazione di puntatore

<tipo> * <puntatore> ;

```
int *ipunt ;
```

```
float *c ;
```

```
char *r ;
```

ipunt è un puntatore a una variabile di tipo **int**

c è un puntatore a una variabile di tipo **float**

r è un puntatore a una variabile di tipo **char**

puntatori in C

```
int *ipunt
```

ipunt è un puntatore a una variabile di tipo intero

ipunt contiene l'indirizzo di una variabile di tipo intero

il valore puntato da **ipunt** è un dato di tipo intero

puntatori in C

```
int *ipunt
```

operatore di
dereferenziazione

ipunt è la **variabile puntatore** (il suo valore è un **indirizzo**)

***ipunt** è il **valore memorizzato** nella cella di **quell'indirizzo**

puntatori in C

assegnazione di un indirizzo a un puntatore

```
<puntatore> = &<variabile>;
```

```
int eta;  
int *ipunt;  
ipunt = &eta;
```

il valore del puntatore **ipunt** è l'indirizzo della variabile **eta**

il puntatore **ipunt** punta alla variabile **eta**

puntatori in C

accesso al **valore puntato** da un puntatore
(accesso **indiretto** al valore di una variabile)

***<puntatore>**

```
int eta;  
int *ipunt;  
ipunt = &eta;  
eta = 27;
```

il **valore** della variabile **eta** può essere ottenuto

- ✓ direttamente: **eta**
- ✓ indirettamente: ***ipunt**

puntatori in C

accesso al **valore puntato** da un puntatore
(accesso **indiretto** al valore di una variabile)

***<puntatore>**

```
int eta;  
int *ipunt;  
ipunt = &eta;  
eta = 27;  
printf("valore di eta=%d/n", eta);
```

accesso diretto
a eta

```
int eta;  
int *ipunt;  
ipunt = &eta;  
eta = 27;  
printf("valore di eta=%d/n", *ipunt);
```

accesso indiretto
a eta

l'operatore di **dereferenziazione** *
quando viene applicato a un **puntatore**
indica il **valore puntato**

alloca memoria per **x**

alloca memoria per
un indirizzo di un **int**

associa **5** a **x**

associa l'indirizzo di **x**
a **y**

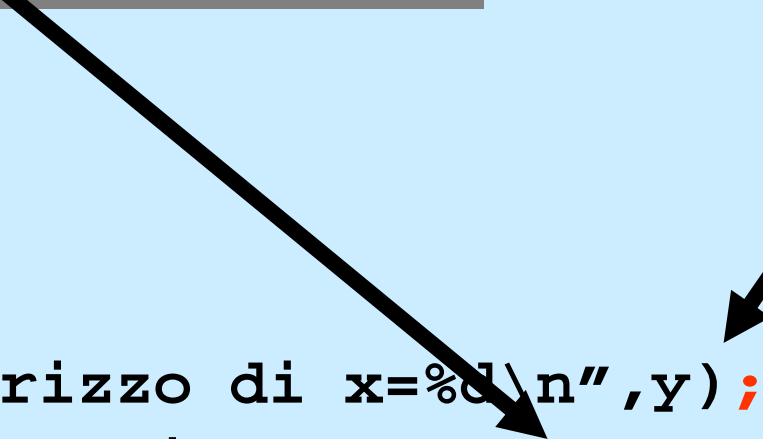
```
#include <stdio.h>
void main ()
{
  int x;
  int *y;
  x = 5;
  y = &x;
  printf ("indirizzo di x=%d\n", x);
  printf ("valore di x=%d\n", *y);
}
```

l'operatore di **dereferenziazione** *****
quando viene applicato a un **puntatore**
indica il **valore puntato**

stampa il valore puntato da **y**,
cioè il valore di **x**

stampa il valore di **y**,
cioè, l'indirizzo di **x**

```
{  
int x;  
int *y;  
x = 5;  
y = &x;  
printf ("indirizzo di x=%d\n", y);  
printf ("valore di x=%d\n", *y);  
}
```



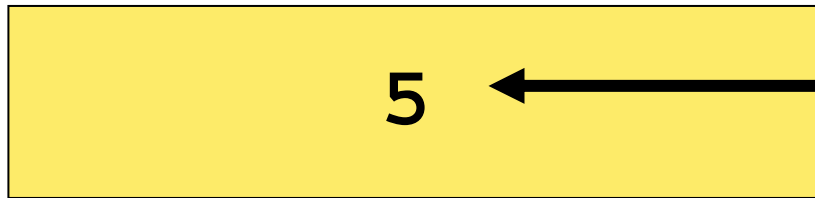
l'operatore di **dereferenziazione** *
quando viene applicato a un **puntatore**
indica il **valore puntato**

```
#include <stdio.h>
void main ()
{
    int x;
    int *y;
    x = 5;
    y = &x;
    printf ("indirizzo di x=%d\n",y);
    printf ("valore di x=%d\n",*y);
}
```

```
indirizzo di x=1245052
valore di x=5
```

cella di memoria di \mathbf{x}

indirizzo **1245052**



$\mathbf{x} = 5;$

valore di \mathbf{x}

cella di memoria di \mathbf{y}

indirizzo 3306521



$\mathbf{y} = \&\mathbf{x};$

valore di \mathbf{y} ,
cioè indirizzo di \mathbf{x}

l'operatore di **dereferenziazione** `*`
quando viene applicato a un **puntatore** indica il **valore puntato**

```
#include <stdio.h>
void main ()
{
    int x;
    int *y;
    x = 5;
    y = &x;
    *y = 6;
    *y = (*y)+1;
    printf ("valore di x=%d\n",x);
    printf ("valore di x=%d\n",*y);
}
```

associa 6 alla variabile
puntata da **y**, cioè **x**

incrementa di uno il
valore puntato da **y**,
cioè il valore di **x**

```
valore di x=7
valore di x=7
```

Esempio

```
/* dichiarazione di puntatore a intero */  
int *ipunt;  
  
/* dichiarazione di variabili intere */  
int a = 5, b;  
  
ipunt = &a;      /* ipunt punta ad a */  
  
b = *ipunt;      /* assegnare a b il  
                 valore della variabile puntata da ipunt,  
                 cioè il valore di a, ovvero 5 */  
  
*ipunt = 9;      /* assegnare 9 alla  
                 variabile puntata da ipunt, ovvero ad a  
                 */
```


priorità degli operatori
indirizzo & e deferenzaiazione *

priorità più elevata degli operatori aritmetici

Esempio

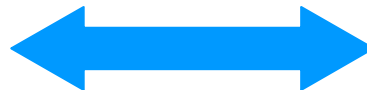
```
int a=2,b=1,c,*p;  
p = &b;  
c = a+*p;
```



```
c = a+(*p);
```

sono associativi a destra

```
c = *&b;
```



```
c = *(&b);
```

priorità degli operatori

indirizzo & e deferenzaione *

***** e **&** sono uno l'inverso dell'altro

➤ data la dichiarazione

```
int a;
```

***&a** è equivalente ad **a**

valore

➤ data la dichiarazione

```
int *ip;
```

&*ip è equivalente a **ip**

indirizzo

&*ip non può essere usato a sinistra di =