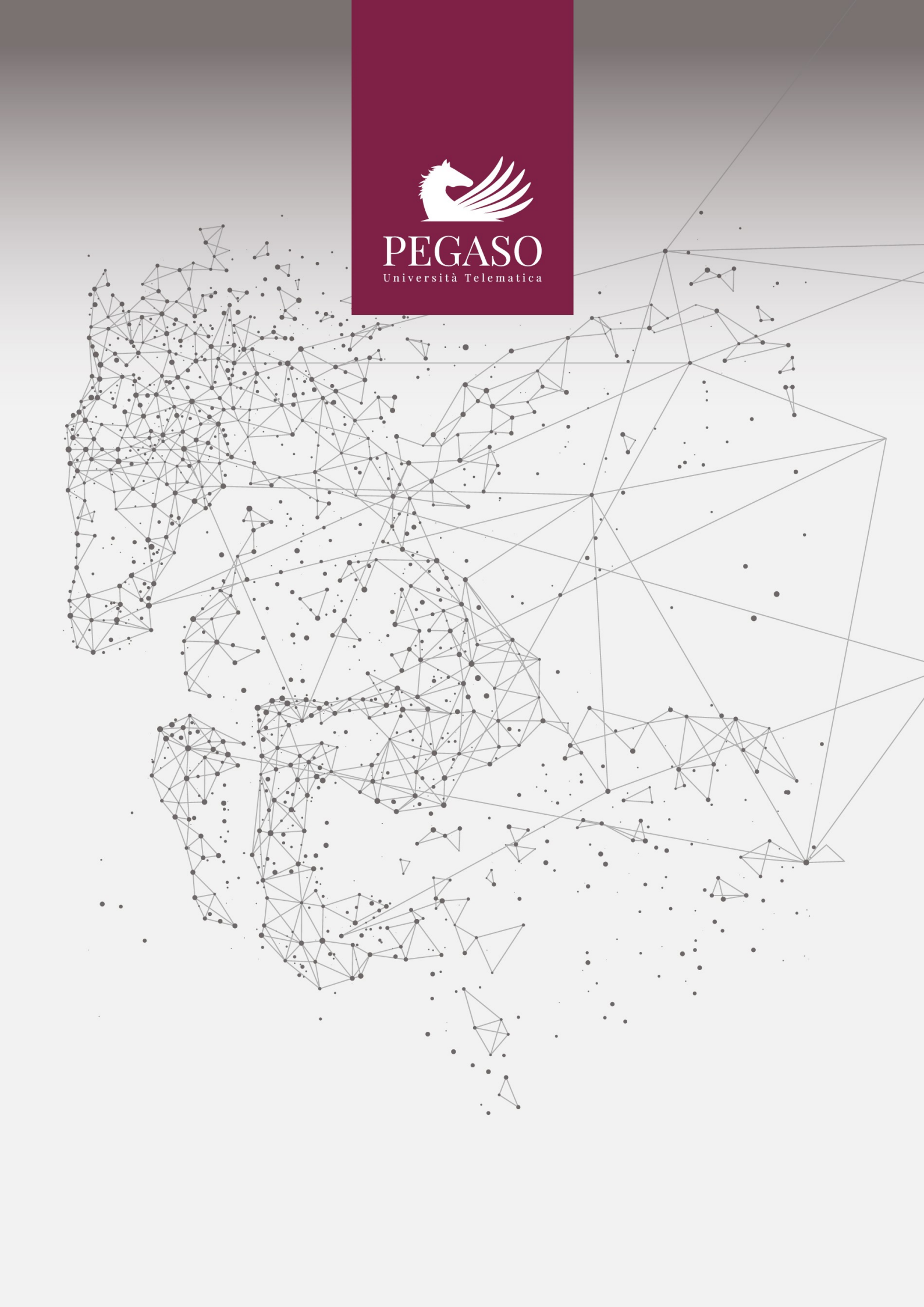

PEGASO
Università Telematica

PEGASO
Università Telematica

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

Indice

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

1. Passare argomenti a funzioni per riferimento

Vi sono due modi di passare argomenti a una funzione: per valore e per riferimento.

Ricordiamo che nel **passaggio per valore**, la funzione chiamata riceve una copia del valore delle variabili passate dalla funzione chiamante.

Tuttavia, ciò comporta un consumo di tempo e di memoria.

Inoltre, talvolta è opportuno restituire alla funzione chiamante valori modificati.

Con return si può restituire alla funzione chiamante un solo valore modificato da una funzione chiamata.

È possibile usare un passaggio per riferimento per consentire a una funzione di "restituire" più valori alla sua funzione chiamante.

Usare `&` e `*` per realizzare il passaggio per riferimento

In C si usano i puntatori e l'operatore di indirezione (`*`) per realizzare il passaggio per riferimento.

Quando si chiama una funzione con argomenti che devono essere modificati, vengono passati gli indirizzi degli argomenti.

Ciò si realizza applicando l'operatore di indirizzo (`&`) alla variabile (nella funzione chiamante) il cui valore sarà modificato.

Come abbiamo visto, gli array non vengono passati usando l'operatore &, perché il C passa automaticamente la locazione iniziale nella memoria dell'array (il nome di un array è equivalente a `&arrayName[0])`.

Quando l'indirizzo di una variabile viene passato a una funzione, si può usare nella funzione l'operatore di indirezione (`*`) per modificare il valore in quella locazione nella memoria della funzione chiamante.

PEGASO
Università Telematica

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

Passaggio per valore

I programmi successivi presentano due versioni di una funzione che eleva al cubo un intero: `cubeByValue e cubeByReference`.

Nella riga 14 della prima versione, la funzione `main` passa per valore la variabile `number` alla funzione `cubeByValue`.

La funzione `cubeByValue` eleva al cubo il suo argomento e passa il nuovo valore indietro a `main` usando un'istruzione `return`.

Il nuovo valore è assegnato a `number` in `main` (riga 14).

```
1  // Eleva al cubo una variabile
2  // usando il passaggio per valore.
3  #include <stdio.h>
4
5  int cubeByValue(int n); // prototipo
6
7  int main(void)
8  {
9     int number = 5; // inizializza number
10
11    printf("The original value of number is %d", number);
12
13    // passa number per valore a cubeByValue
14    number = cubeByValue(number);
15
16    printf("\nThe new value of number is %d\n", number);
17 }
18
19 // calcola e restituisci il cubo di un argomento intero
20 int cubeByValue(int n)
```

PEGASO
Università Telematica

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

```
21  {
22     return n * n * n; //  restituisci il cubo di n
23  }
```

```
The original value of number is 5

The new value of number is 125
```

Passaggio per riferimento

Nel programma successivo la funzione `main` passa la variabile `number` per riferimento (riga 15) – viene passato l'indirizzo di `number` – alla funzione `cubeByReference`.

La funzione `cubeByReference` riceve come parametro un puntatore a un `int` chiamato `nPtr` (riga 21): la funzione dereferenzia il puntatore ed eleva al cubo il valore al quale punta `nPtr` (riga 23), quindi assegna il risultato a `*nPtr` (che è in realtà `number` in `main`), cambiando così il valore di `number` in `main`.

```
 1 // Eleva al cubo una variabile
 2  // usando il passaggio per riferimento.
 3
 4  #include <stdio.h>
 5
 6  void cubeByReference(int *nPtr); // prototipo di funzione
 7
 8  int main(void)
 9  {
10     int number = 5; // inizializza number
11
12     printf("The original value of number is %d", number);
13
```

PEGASO
Università Telematica

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

```
14     // passa l'indirizzo di number a cubeByReference
15     cubeByReference(&number);
16
17     printf("\nThe new value of number is %d\n", number);
18  }
19
20  // eleva al cubo *nPtr; di fatto modifica number in main
21  void cubeByReference(int *nPtr)
22  {
23     *nPtr = *nPtr * *nPtr * *nPtr; // calcola il cubo di *nPtr
24  }
```

```
The original value of number is 5

The new value of number is 125
```

Usare un parametro puntatore per ricevere un indirizzo

Una funzione che riceve un indirizzo come argomento deve essere definita con un parametro puntatore per ricevere l'indirizzo.

Ad esempio, nel programma precedente l'intestazione per la funzione `cubeByReference` (riga 21) è:

```
void cubeByReference(int *nPtr)
```

L'intestazione specifica che `cubeByreference` riceve l'indirizzo di una variabile intera come argomento, memorizza l'indirizzo localmente in `nPtr` e non restituisce alcun valore.

Parametri puntatore nei prototipi di funzione

Il prototipo di funzione per `cubeByReference` (riga 6) specifica un parametro `int *`.

PEGASO
Università Telematica

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

Come con altri tipi di variabili, non è necessario includere i nomi dei puntatori nei prototipi di funzione. I nomi inclusi per fini di documentazione sono ignorati dal compilatore C.

Funzioni che ricevono array unidimensionali

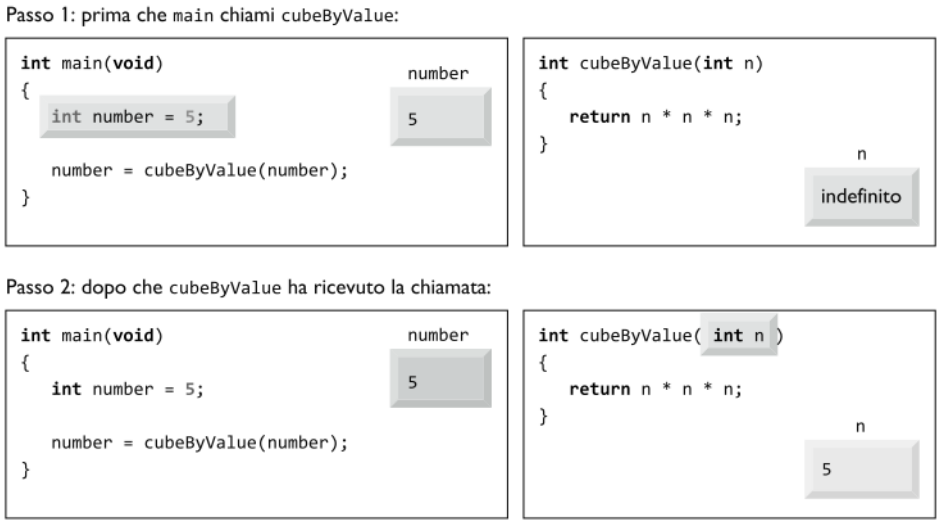
Per una funzione che si aspetta come argomento un array unidimensionale, il prototipo e l'intestazione della funzione possono usare la notazione usata per i puntatori mostrata nella lista dei parametri della funzione `cubeByReference` (riga 21).

Il compilatore non distingue tra una funzione che riceve un puntatore e una che riceve un array unidimensionale.

Ciò, naturalmente, significa che la funzione deve "sapere" quando riceve un array o semplicemente una variabile singola per la quale si sta eseguendo il passaggio per riferimento.

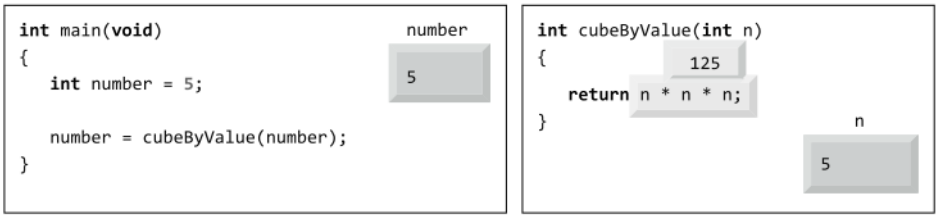
Quando il compilatore incontra un parametro di funzione per un array unidimensionale della forma `int b[]`, il compilatore converte il parametro nella notazione per i puntatori `int *b`. Le due forme sono intercambiabili.

Analisi di un tipico passaggio per valore:

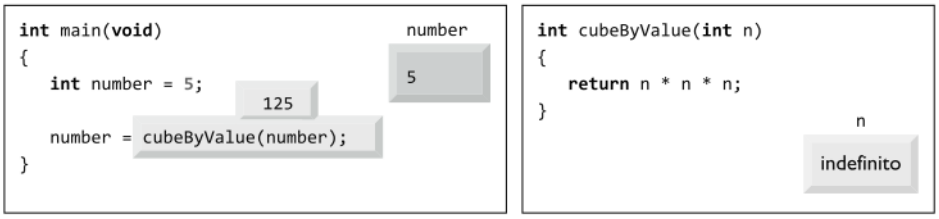


Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

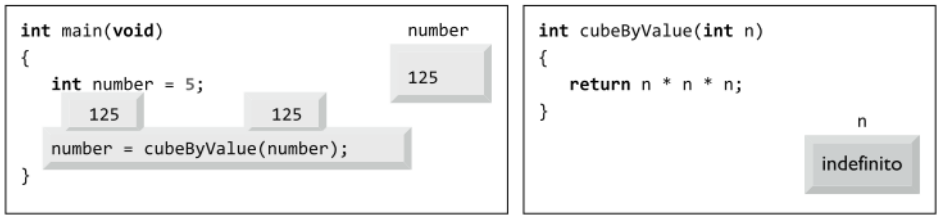
Passo 3: dopo che `cubeByValue` ha elevato al cubo il parametro `n` e prima che `cubeByValue` torni alla funzione `main`:



Passo 4: dopo che `cubeByValue` è tornata alla funzione `main` e prima che si assegni il risultato a `number`:

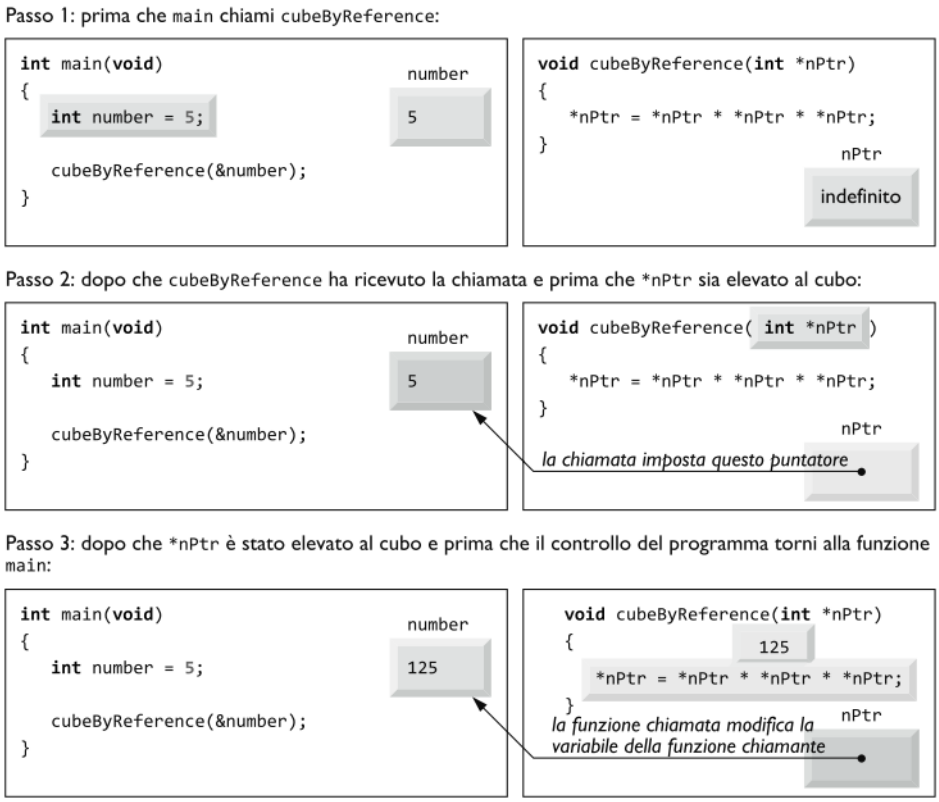


Passo 5: dopo che `main` ha completato l'assegnazione a `number`:



Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

Analisi di un tipico passaggio per riferimento con un argomento puntatore:



😊 *Prevenzione di errori*

Usate il passaggio per valore per passare argomenti a una funzione, a meno che la funzione chiamante non necessiti esplicitamente che la funzione chiamata modifichi il valore della variabile dell'argomento nell'ambiente della stessa funzione chiamante.

Ciò previene la modifica accidentale degli argomenti passati dalla funzione chiamante e costituisce inoltre un ulteriore esempio del principio del privilegio minimo.

PEGASO
Università Telematica

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).

Riferimenti bibliografici

- Paul Deitel, Harvey Deitel, "Il linguaggio C – Fondamenti e tecniche di programmazione", Libro edito da Pearson Italia. Include anche utili esercizi di autovalutazione.

PEGASO
Università Telematica

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright. Ne è severamente vietata la riproduzione o il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore (L. 22.04.1941/n. 633).