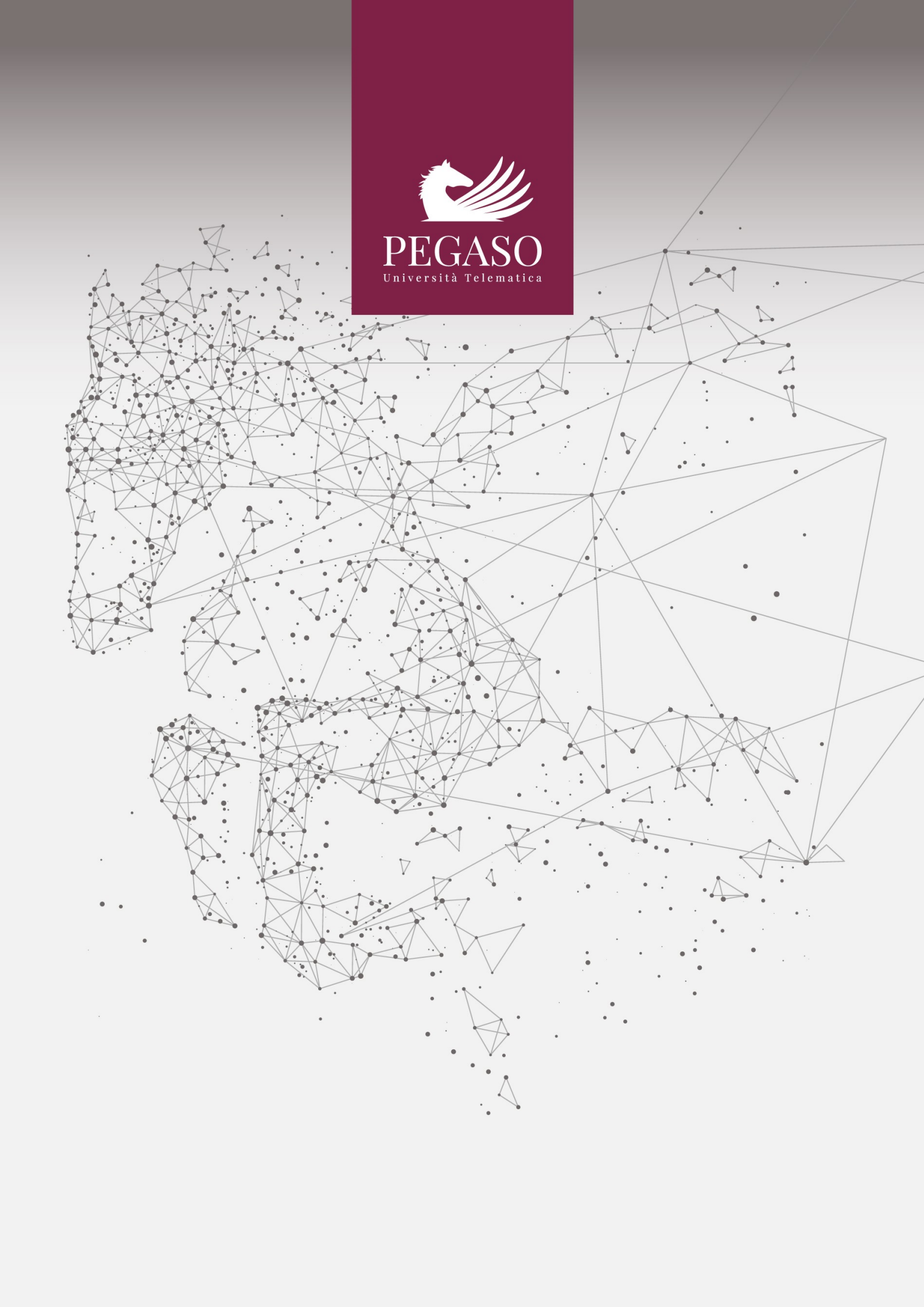

PEGASO
Università Telematica

Indice

1. Espressioni con puntatori

I puntatori sono operandi validi nelle espressioni aritmetiche, nelle espressioni di assegnazione e nelle espressioni di confronto.

Tuttavia, non tutti gli operatori normalmente usati in queste espressioni sono validi in combinazione con le variabili puntatore.

Operatori utilizzabili per l'aritmetica dei puntatori

È possibile incrementare (++) o decrementare (--) un puntatore.

È possibile aggiungere un intero a un puntatore (+ o +=), sottrarre un intero da un puntatore (- o –=) e sottrarre un puntatore da un altro puntatore.

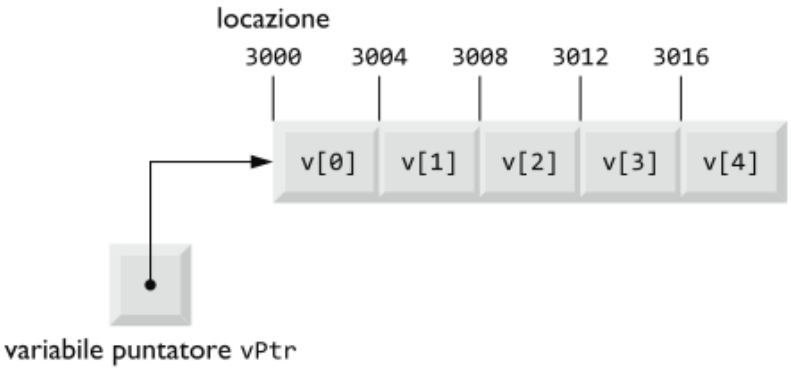
Quest'ultima operazione ha senso solo quando entrambi i puntatori puntano agli elementi dello stesso array.

Indirizzare un puntatore a un array

Supponete che sia stato definito l'array `int v[5]` e che il suo primo elemento sia alla locazione 3000 nella memoria.

Supponete che il puntatore `vPtr` sia stato inizializzato in modo da puntare `a v[0]` (ossia il valore di vPtr è 3000).

La Figura seguente illustra questa situazione per una macchina con interi di 4 byte.



La variabile `vPtr` può essere inizializzata per puntare all'array `v` con l'una o l'altra delle istruzioni:

```
vPtr = v;
vPtr = &v[0];
```

Portabilità

Poiché i risultati dell'aritmetica dei puntatori dipendono dalla dimensione degli oggetti a cui punta un puntatore, l'aritmetica dei puntatori è dipendente dalla macchina e dal compilatore.

Aggiungere un intero a un puntatore

Nell'aritmetica convenzionale, 3000 + 2 produce il valore 3002.

Questo non avviene normalmente con l'aritmetica dei puntatori.

Quando un intero è sommato o sottratto da un puntatore, il puntatore non è incrementato o decrementato semplicemente di quell'intero, ma di quell'intero moltiplicato per le dimensioni dell'oggetto a cui il puntatore si riferisce.

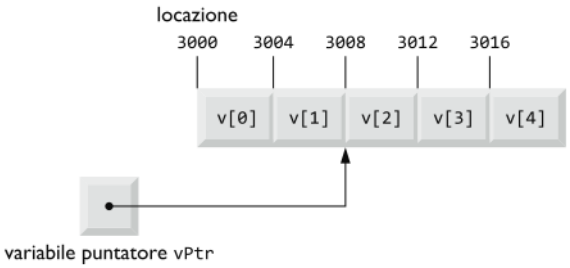
Il numero di byte dipende dal tipo di dati dell'oggetto.

Ad esempio, l'istruzione

```
vPtr += 2;
```

darebbe il valore 3008 (3000 + 2 * 4), supponendo che un intero sia memorizzato in 4 byte di memoria.

Nell'array `v`, `vPtr` punterebbe adesso a `v[2]`



Se un intero fosse memorizzato in 2 byte di memoria, il calcolo precedente darebbe la locazione di memoria 3004 (3000 + 2 * 2).

Se l'array fosse di un tipo differente di dati, l'istruzione precedente incrementerebbe il puntatore di due volte il numero di byte che gli occorrono per memorizzare un oggetto di quel tipo di dati.

Quando si usa l'aritmetica dei puntatori con un array di caratteri, i risultati saranno coerenti con l'aritmetica regolare, perché ogni carattere è lungo 1 byte.

☹ *Errore comune di programmazione*

Usare l'aritmetica dei puntatori per un puntatore che non fa riferimento a un elemento in un array.

Sottrarre un intero da un puntatore

Se `vPtr` fosse stato incrementato fino a 3016, il che significa che punterebbe a `v[4]`, l'istruzione

```
vPtr -= 4;
```

riporterebbe `vPtr` al valore 3000, cioè all'inizio dell'array.

☹ *Errore comune di programmazione*

Uscire fuori dai limiti di un array quando si usa l'aritmetica dei puntatori.

Incrementare e decrementare un puntatore

Se un puntatore è incrementato o decrementato di uno, è possibile usare gli operatori di incremento (++) e di decremento (--).

Entrambe le istruzioni

```
++vPtr;
```

```
vPtr++;
```

incrementano il puntatore facendolo puntare alla locazione successiva nell'array.

Tutte e due le istruzioni

```
--vPtr;
```

```
vPtr--;
```

decrementano il puntatore facendolo puntare all'elemento precedente dell'array.

2. Espressioni con puntatori – parte II

Sottrarre un puntatore da un altro

Le variabili puntatore possono essere sottratte l'una dall'altra.

Ad esempio, se `vPtr` contiene l'indirizzo 3000 e `v2Ptr` contiene l'indirizzo 3008, l'istruzione

```
x = v2Ptr - vPtr;
```

assegnerebbe a `x` il numero degli elementi dell'array da `vPtr` a `v2Ptr`, in questo caso 2 (non 8).

L'aritmetica dei puntatori è indefinita, a meno che non sia eseguita su un array.

Non possiamo presumere che due variabili dello stesso tipo siano memorizzate in modo contiguo nella memoria, a meno che non siano elementi adiacenti di un array.

☹ *Errore comune di programmazione*

Sottrarre o confrontare due puntatori che non fanno riferimento a elementi nello stesso array.

Assegnare puntatori ad altri puntatori

Un puntatore può essere assegnato a un altro puntatore se entrambi hanno lo stesso tipo. L'eccezione a questa regola è costituita dal puntatore a `void` (cioè `void *`), il quale è un puntatore generico che può rappresentare qualunque tipo di puntatore.

A tutti i tipi di puntatore è possibile assegnare un puntatore a `void` e a un puntatore a `void` è possibile assegnare un puntatore di qualsiasi tipo (compreso un altro puntatore a `void`).

In entrambi i casi non è necessaria alcuna operazione di cast.

Puntatore a void

Un puntatore a void non può essere dereferenziato.

Si può fare infatti la seguente considerazione:

il compilatore sa che un puntatore a `int` fa riferimento a 4 byte di memoria su una macchina con interi a 4 byte, ma un puntatore a `void` contiene semplicemente una locazione di memoria per un tipo di dati sconosciuto (il numero esatto di byte a cui il puntatore fa riferimento non è conosciuto dal compilatore).

Il compilatore deve conoscere il tipo di dati per determinare il numero di byte che rappresentano il valore di riferimento.

☹ *Errore comune di programmazione*

Assegnare un puntatore di un tipo a un puntatore di un altro tipo se nessuno dei due è del tipo `void *` *è un errore di sintassi.*

☹ *Errore comune di programmazione*

Dereferenziare un puntatore `void *` *è un errore di sintassi.*

Confrontare i puntatori

È possibile confrontare i puntatori usando gli operatori di uguaglianza e relazionali, ma tali confronti non hanno senso, a meno che i puntatori non puntino a elementi dello stesso array.

I confronti di puntatori comparano gli indirizzi memorizzati nei puntatori.

Un confronto di due puntatori che puntano a elementi nello stesso array potrebbe evidenziare, ad esempio, che un puntatore punta a un elemento dell'array con indice più alto rispetto all'altro puntatore.

Il confronto di puntatori è comunemente usato per determinare se un puntatore è NULL.

☹ *Errore comune di programmazione*

Confrontare due puntatori che non fanno riferimento a elementi nello stesso array.

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.