Capitolo 2

Array

Andrea Gini

Dopo aver introdotto i tipi primitivi, è giunto il momento di analizzare in profondità un altro strumento importantissimo in un linguaggio di programmazione: l'array. Gli array (o vettori) sono collezioni di variabili indicizzate, che permettono di gestire in maniera relativamente semplice grosse porzioni di memoria, e di effettuare su di essa calcoli ripetitivi. Gli array tornano utili in tutte le situazioni in cui si ha l'esigenza di manipolare un gruppo di variabili dello stesso tipo che contengono valori tra loro correlati.

Si immagini di dover scrivere un programma che calcoli la media delle temperature giornaliere; se si dovessero prendere in considerazione 6 misurazioni all'ora, una ogni 10 minuti, servirebbero ben 144 variabili.

```
int temp1 = 15;          // ore 0.00
int temp2 = 16;          // ore 0.10
int temp3 = 16;          // ore 0.20
int temp4 = 16;          // ore 0.30
....
int temp144 = 14;        // ore 11.50
```

Oltre alla scarsa praticità di dover dichiarare 144 variabili, non esiste nessun modo pratico per effettuare calcoli che abbraccino tutto l'insieme dei valori: l'unico sistema per calcolare la media sarebbe quello di realizzare una gigantesca espressione aritmetica del tipo:

```
int media = (temp1 + temp2 + temp3 + .... + temp143 + temp144) / 144;
```

In casi come questo è utile ricorrere a un array, uno strumento concettualmente simile a una tabella, che accomuna sotto un unico nome un insieme di variabili dello stesso tipo:

```
int[] temp = new int[144];
```

La creazione e l'utilizzo degli array presentano alcune differenze rispetto all'impiego delle normali variabili. Nei prossimi paragrafi verranno illustrate una a una.

Dichiarazione di array

La dichiarazione di un array ha una sintassi un po' più complessa della dichiarazione di variabile semplice. Come per le variabili è necessario indicare un tipo e un nome, con la differenza che, dopo aver specificato il tipo, è necessario posporre una coppia di parentesi quadre:

```
int[] vettoreDiInteri;
```

Assegnamento

La variabile vettoreDiInteri appena dichiarata non è un vettore, ma solamente un riferimento (in inglese, reference) a un vettore. Il vettore vero e proprio è un'entità separata, che occupa un certo spazio nella memoria e che deve essere creata in modo opportuno. Prima di essere inizializzata, la variabile ha il valore null, una costante che indica che la variabile non referenzia alcun vettore.

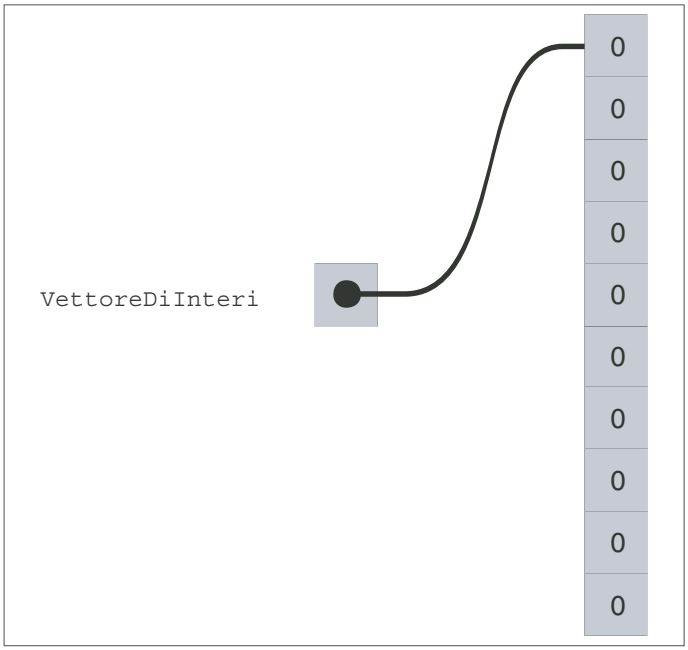
Figura 2.1 – *La variabile con cui si fa riferimento a un vettore ha inizialmente valore null.*

Per creare un vettore è necessario utilizzare la parola riservata new, come nell'esempio seguente:

```
vettoreDiInteri = new int[10];
```

Il valore tra parentesi quadre è la dimensione del vettore: è possibile specificare un qualsiasi valore intero positivo. Il vettore appena creato è formato da dieci elementi, inizializzati a zero.

Figura 2.2 – *Un vettore è un oggetto di memoria composto da un certo numero di elementi, ognuno dei quali può contenere un valore.*



Dereferenziazione

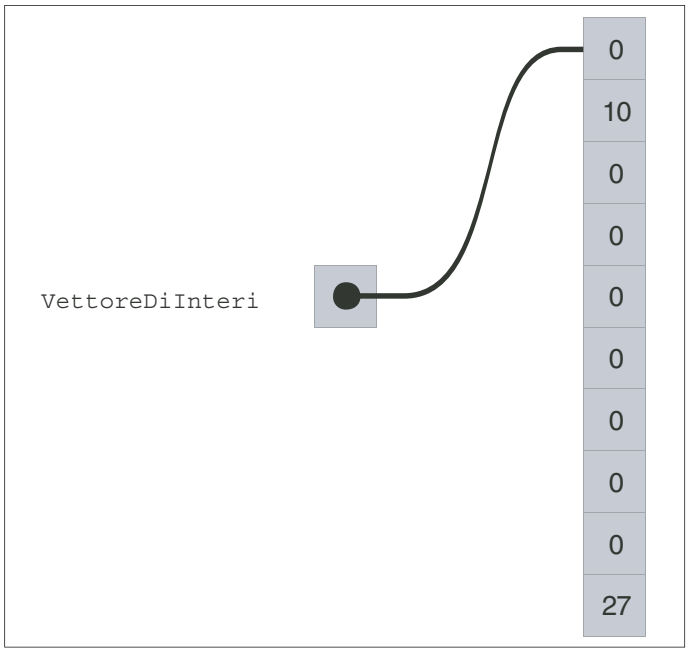
La dereferenziazione è l'operazione che permette di assegnare un valore a un elemento del vettore. Per dereferenziare un elemento di un vettore, occorre specificare il nome dell'array seguito dal numero dell'elemento tra parentesi quadre:

```
vettoreDiInteri[1] = 10;
```

Gli elementi di un vettore si contano a partire da zero: pertanto, se si desidera assegnare il valore 27 al decimo elemento del vettore, è necessario scrivere:

```
vettoreDiInteri[9] = 27;
```

Figura 2.3 – *Lo stesso vettore della figura 2.2, dopo aver dereferenziato il secondo e il decimo elemento.*

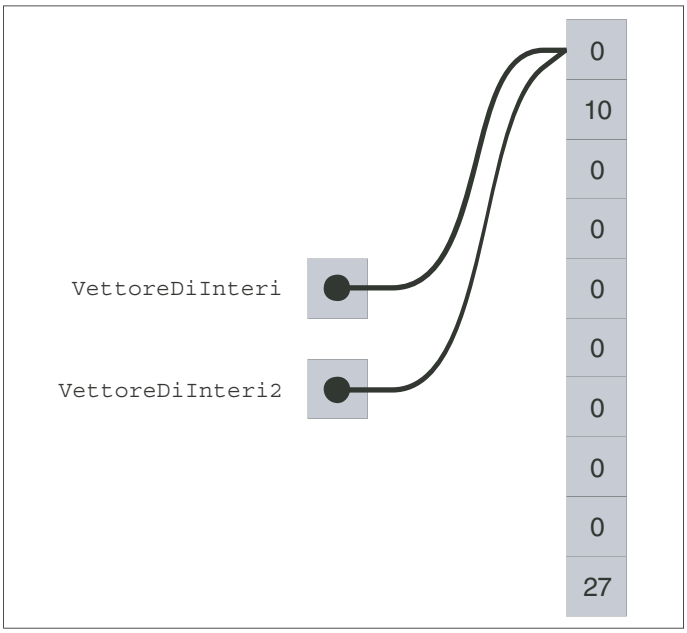


Differenza tra assegnamento e dereferenziazione

Quando si lavora su vettori, bisogna avere ben chiara la differenza tra dereferenziazione e assegnamento. L'assegnamento è un'operazione che agisce direttamente sulla variabile, provocandone un cambiamento di valore. La dereferenziazione, invece, è un'operazione indiretta: essa non opera sulla variabile, ma sull'oggetto di memoria puntato da essa. Se si crea un nuovo reference e gli si assegna il valore di un reference che punta a un vettore già esistente, si verifica una situazione in cui due variabili puntano allo stesso vettore, come nell'esempio seguente illustrato in figura 2.4:

```
int[] vettoreDiInteri2;
vettoreDiInteri2 = vettoreDiInteri;
```

Figura 2.4 – *Due variabili che fanno riferimento allo stesso vettore.*



In una situazione come questa le operazioni vettoreDiInteri[5] = 10 e vettoreDiInteri[5] = 10 avranno entrambe il risultato di porre a 10 l'elemento numero 5 dell'unico vettore puntato dalle due variabili. Per procedere all'effettiva copia di un vettore, è necessario dapprima creare un vettore delle stesse dimensioni, quindi copiare uno a uno gli elementi del primo nel secondo. Questa operazione può essere eseguita con un ciclo while, come nell'esempio seguente:

```
// crea un vettore e lo inizializza
int[] v1 = new int[5];
v1[0] = 10;
v1[1] = 12;
v1[2] = 14;
v1[3] = 16;
v1[4] = 18;

// crea un vettore della stessa dimensione di v1
int[] v2 = new int[5];
int i = 0;
while(i < 5) {
  // copia il valore della i-esima cella
  // di v1 nella i-esima cella di v2
  v2[i] = v1[i];
}
```

Inizializzazione automatica di un vettore

Un vettore può essere inizializzato con una serie di valori, in modo simile a come si può fare con le variabili. L'istruzione:

```
int[] vettore = {10,12,14,16,18};
```

equivale alla sequenza di istruzioni:

```
int[] vettore = new int[5];
vettore[0] = 10;
vettore[1] = 12;
vettore[2] = 14;
vettore[3] = 16;
vettore[4] = 18;
```

Lunghezza di un vettore

I vettori creati con l'operatore new hanno esattamente la dimensione specificata nella dichiarazione. Gli indici sono numerati a partire da 0, per cui l'ultimo elemento avrà l'indice pari alla dimensione del vettore meno uno. Per esempio, in un vettore da 10 elementi gli indici sono compresi tra 0 e 9. Il programmatore può essere interessato a conoscere la dimensione di un array a runtime: per questo scopo, ogni vettore dispone di un'apposita costante length, accessibile tramite l'operatore '.':

```
int vettoreDiInteri[] = new int[10];
System.out.print("La dimensione del vettore è ");
System.out.println(vettoreDiInteri.length);
```

Un esempio di manipolazione di vettori

Il vettore è uno strumento potentissimo, che permette di lavorare su porzioni di memoria anche molto grandi usando un numero ridotto di istruzioni. I cicli while permettono di valutare uno a uno gli elementi di un array, e di effettuare qualche tipo di operazione su di essi. Per calcolare la media dei valori contenuti in un ipotetico vettore vettoreDiInteri, si può usare un frammento di codice di questo tipo:

```
int i = 0;
int somma = 0;
int media = 0;

while(i<vettoreDiInteri.length) {
```

```
    somma = somma + vettoreDiInteri[i];
    i++;
  }
  media = somma / sommaDiInteri.length;
```

Il seguente esempio permette di togliersi una soddisfazione: quella di scrivere un programma che sfrutti una parte importante della memoria del proprio computer.

Un vettore di byte di dimensione 1024 occupa esattamente un kilobyte (KB) di memoria. Un vettore di int della stessa dimensione ne occupa 4, dal momento che un int è grande 4 byte. Un vettore di interi da 64 megabyte ha una dimensione che può essere calcolata moltiplicando 64 per 1048576 (pari a 1024 al quadrato) e dividendo per quattro. Una volta creato un simile vettore, lo si può riempire di valori casuali scelti tra 0 e 10000; infine, è possibile calcolare la somma di tutti i valori e la relativa media aritmetica. Si noti l'uso di una variabile di tipo long per memorizzare la somma di tutti i numeri: è facile comprendere che una variabile intera non avrebbe la dimensione sufficiente a contenere il risultato.

```
public class MemoryConsumer {

  public static void main(String argv[]) {

    long sum = 0;
    long average = 0;

    // calcola la dimensione del vettore.
    // Se si dispone di poca memoria, impostare
    // un valore più basso nella variabile megaBytes.
    int megaBytes = 64;
    int dim = megaBytes * 1048576 / 4;
    int[] bigArray = new int[dim];


    // riempie il vettore di valori casuali
    int i = 0;
    while ( i < bigArray.length ) {
      bigArray[i] = (int)(32000 * Math.random());
      i++;
    }

    // calcola la somma di tutti i valori
    i = 0;
    while ( i < bigArray.length ) {
      sum = sum + bigArray[i];
      i++;
    }

    // calcola la media
```

```
    average = sum / bigArray.length;

    // stampa i risultati
    System.out.print("La somma di tutti i numeri presenti nel vettore è ");
    System.out.println(sum);
    System.out.print("La media della somma di tutti i numeri presenti nel vettore è ");
    System.out.println(average);
  }
}
```

Il programma deve essere salvato, come di consueto, in un file dal nome "MemoryConsumer.java"; per compilarlo bisogna digitare il comando:

```
javac MemoryConsumer.java
```

mentre per eseguirlo bisogna ricorrere all'istruzione:

```
java MemoryConsumer
```

Dopo qualche istante, il programma stamperà un output del tipo:

```
La somma di tutti i numeri presenti nel vettore è 239999200405
La media della somma di tutti i numeri presenti nel vettore è 15999
```

Per poter eseguire questo programma, è necessario disporre di un computer con almeno 256 MB di RAM. Se non si dispone di memoria sufficiente, il computer segnalerà un errore:

```
java.lang.OutOfMemoryError
        <<no stack trace available>>
Exception in thread "main"
```

In questo caso, si provi a diminuire il valore della variabile 'megaBytes', portandolo per esempio a 32, quindi si provi a compilare ed eseguire nuovamente.

Vettori multidimensionali

Il linguaggio Java consente di creare vettori bidimensionali, ricorrendo a una sintassi del tipo:

```
int i[][] = new int[10][15];
```

I vettori bidimensionali sono concettualmente simili a una tabella rettangolare, dotata di righe e colonne. Il seguente programma crea una tavola pitagorica in un vettore bidimensionale, quindi la stampa sullo schermo:

```
public class Tabelline2 {

  public static void main(String argv[]) {
    int[][] tabellina = new int[11][11];

    // crea la tabellina in un vettore bidimensionale
    int i = 0;
    int j = 0;
    while(i <= 10) {
      while(j <= 10) {
        int prodotto = i*j;
        tabellina[i][j] = prodotto;
        j = j + 1;
      }
      i = i + 1;
      j = 0;
    }

    // stampa il contenuto del vettore
    i = 0;
    j = 0;
    while(i <= 10) {
      while(j <= 10) {
        int prodotto = i*j;
        System.out.print(tabellina[i][j]);
        System.out.print("\t");
        j = j + 1;
      }
      i = i + 1;
      j = 0;
      System.out.println();
    }
  }
}
```

È possibile definire vettori con un numero qualsiasi di dimensioni:

```
int v1[][][] = new int[10][15][5];
int v2[][][][] = new int[10][15][12][5];
```

Tali strutture, in ogni caso, risultano decisamente poco utilizzate.

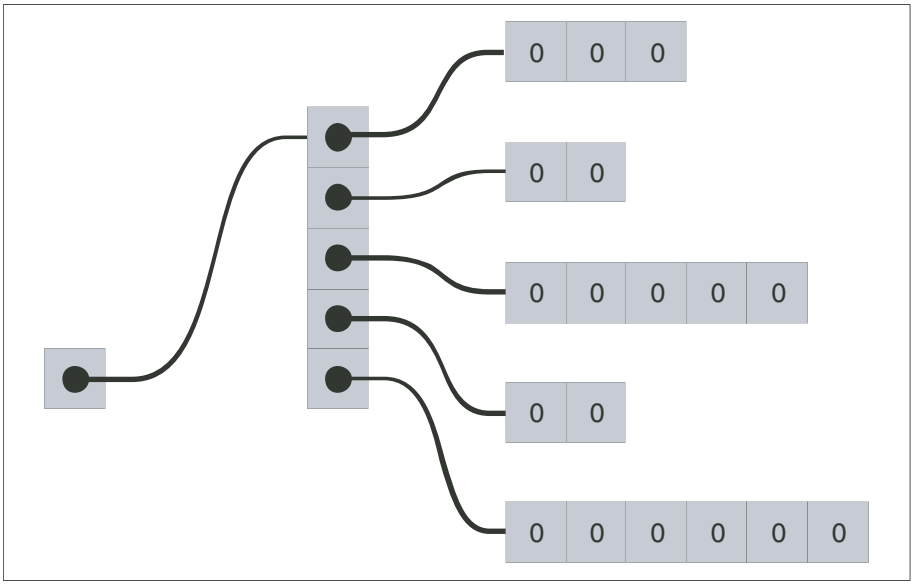
Vettori incompleti

I vettori n-dimensionali vengono implementati in Java come array di array. Questa scelta implementativa consente di realizzare tabelle non rettangolari, come nell'esempio seguente:

```
// crea un vettore con una componente incompleta
int tabella[][] = new int[5][];

tabella[0] = new int[3];
tabella[1] = new int[2];
tabella[2] = new int[5];
tabella[3] = new int[2];
tabella[4] = new int[6];
```

Figura 2.5 – *Rappresentazione in memoria di un vettore non rettangolare.*



Per dichiarare un vettore incompleto come quello dell'esempio è necessario specificare la prima componente in fase di creazione, mentre la seconda verrà precisata successivamente creando uno a uno i sotto-vettori. Anche in questo caso ci si trova in presenza di un costrutto scarsamente utilizzato, che tuttavia vale la pena di conoscere.

Inizializzazione automatica di un vettore multidimensionale

Un vettore può essere inizializzato con una serie di valori, in modo simile a come si può fare con i vettori semplici. Naturalmente è necessario ricorrere a un costrutto un po' più complesso, che tenga conto della particolare struttura di questi vettori. La seguente istruzione, per esempio, crea un vettore a tre componenti in verticale, in cui la prima riga ha tre colonne, la seconda due

e la terza quattro, e contemporaneamente inizializza gli elementi con i valori specificati, come si può vedere in figura 2.6:

```
int[][] vettore = { { 10,12,14} , {16,18} , {20,22,24,26} };
```

Figura 2.6 – *Un altro esempio di vettore non rettangolare.*