10 - Programmare con gli Array

Programmazione e analisi di dati
Modulo A: Programmazione in Java

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa
http://pages.di.unipi.it/milazzo
milazzo@di.unipi.it

Corso di Laurea Magistrale in Informatica Umanistica
A.A. 2017/2018

Programmare con gli array

Lo scopo di questa lezione è illustrare alcuni schemi di programmi

- cioè soluzioni a problemi di programmazione che ricorrono frequentemente nello sviluppo di programmi
- sono essenzialmente degli algoritmi, espressi nel linguaggio di programmazione scelto (Java)

Ci concentreremo su problemi che richiedono cicli e array per essere risolti

- Quindi: schemi di programmi iterativi

Per ogni schema di programma che consideremo:

- Vedremo un esempio di problema e il programma che lo risolve
- Ragioneremo sul programma dell'esempio per estrarne lo schema di funzionamento
- Menzioneremo altri esempi di problemi che possono essere risolti seguendo lo stesso schema

Innanzitutto: quando gli array non servono? (1)

Innanzitutto è bene capire che non sempre serve usare un array.

Esempio: programma che chiede all'utente di inserire 10 numeri e ne calcola la media

```
import java.util.Scanner;

public class Media {  // soluzione che usa un array
  public static void main(String[] args) {

    Scanner input = new Scanner(System.in);
    int[] valori = new int[10];

    // memorizza tutti i valori in un array
    for (int i=0; i<10; i++)
      valori[i] = input.nextInt();

    // somma tutti gli elementi dell'array
    int somma = 0;
    for (int x : valori) somma+=x;

    // calcola e stampa la media
    int media = somma/10;
    System.out.println("La media e': " + media);
  }
}
```

Innanzitutto: quando gli array non servono? (2)

Prima di creare un array e iniziare a riempirlo di valori, pensare bene se c'e' un modo per ottenere lo stesso risultato lavorando on-the-fly

- ossia utilizzando i dati man mano che arrivano, senza memorizzarli

```
import java.util.Scanner;

public class Media2 {  // soluzione che NON usa un array
  public static void main(String[] args) {

    Scanner input = new Scanner(System.in);

    // somma on-the-fly tutti gli elementi inseriti
    int somma = 0;
    for (int i=0; i<10; i++)
      somma += input.nextInt();

    // calcola e stampa la media
    int media = somma/10;
    System.out.println("La media e': " + media);
  }
}
```

Innanzitutto: quando gli array non servono? (3)

Altro esempio dello stesso tipo...

Esempio: scrivere un programma che stampa gli elementi di un array in ordine inverso

```
import java.util.Scanner;

public class Inverti {  // soluzione che usa un secondo array
  public static void main(String[] args) {

    int[] valori = { 5, 7, 9, 11, 23, 8 };
    int last = valori.length-1; // posizione dell'ultimo elemento

    // creo un nuovo array in cui copio i valori in ordine inverso
    int[] valori2 = new int[valori.length];
    for (int i=0; i<valori.length; i++)
      valori2[i] = valori[last-i];

    // stampa il contenuto del secondo array
    for (int x : valori2)
      System.out.println(x);
}
```

Innanzitutto: quando gli array non servono? (4)

Ma... serve veramente il secondo array?

- I dati sono già memorizzati nel primo. Basta eseguire le stampe in ordine inverso...

```
import java.util.Scanner;

public class Inverti2 {  // soluzione che NON usa un secondo array
  public static void main(String[] args) {

    int[] valori = { 5, 7, 9, 11, 23, 8 };

    // scorre l'array dall'ultimo elemento al primo
    // notare l'uso degli indici nel ciclo for
    for (int i=(valori.length-1); i>=0; i--)
      System.out.println(valori[i]);

}
```

Innanzitutto: quando gli array non servono? (5)

In quali di questi esempi è necessario un array?

1. Chiedere all'utente di inserire 10 numeri e stampare il minimo tra essi
2. Chiedere all'utente di inserire 10 stringhe e stampare la stringa ottenuta concatenandole tutte l'una dopo l'altra
3. Chiedere all'utente di inserire 10 numeri e, successivamente, ristamparli tutti nell'ordine
4. Chiedere all'utente di inserire 10 caratteri e stampare `Trovata Vocale!` se tra essi è presente almeno una vocale
5. Chiedere all'utente di inserire 10 numeri e verificare se i primi 5 sono uguali agli ultimi 5 (ad es. `1,3,4,7,9,1,3,4,7,9`)

Innanzitutto: quando gli array non servono? (5)

In quali di questi esempi è necessario un array?

1. Chiedere all'utente di inserire 10 numeri e stampare il minimo tra essi
2. Chiedere all'utente di inserire 10 stringhe e stampare la stringa ottenuta concatenandole tutte l'una dopo l'altra
3. Chiedere all'utente di inserire 10 numeri e, successivamente, ristamparli tutti nell'ordine
4. Chiedere all'utente di inserire 10 caratteri e stampare `Trovata Vocale!` se tra essi è presente almeno una vocale
5. Chiedere all'utente di inserire 10 numeri e verificare se i primi 5 sono uguali agli ultimi 5 (ad es. `1,3,4,7,9,1,3,4,7,9`)

Soluzione: NNSNS

Quanto deve essere grande l'array nel caso 5?

Schemi di programmi su array

D'ora in poi assumeremo che i dati su cui vogliamo lavorare siano contenuti in un array

Vedremo solo frammenti di programmi da inserire nel seguente scheletro

```
import java.util.Scanner;

public class NomeProgramma {
  public static void main(String[] args) {

    Scanner input = new Scanner(System.in);
    int[] valori = int[10]; // l'array in questione

    // inizializzazione da parte dell'utente
    for (int i=0; i<10; i++)
      valori[i] = input.nextInt();

    // I FRAMMENTI DI PROGRAMMI CHE VEDREMO
    // DOVRANNO ESSERE INSERITI QUI

  }
}
```

Scansione sequenziale di un array (1)

Esempio: Sommare tutti i valori di un array

```
int somma=0;
for (int x: valori)
  somma+=x;

System.out.println(somma);
```

Si scandisce tutto l'array dall'inizio alla fine "portandosi dietro" la somma dei valori incontrati

- All'`i`-esima iterazione la variabile (accumulatore) `somma` rappresenta in modo sintetico tutti gli elementi dell'array tra le posizioni `0` e `i-1`
- In altre situazioni la variabile da "portarsi dietro" può essere
  - un contatore (Esempio: stampare quanti zeri sono contenuti nell'array)
  - un valore (Esempio: stampare il valore minimo)
  - un indice (Esempio: stampare la posizione del valore minimo)
  - combinazioni delle precedenti (Esempi: stampare valore e posizione del minimo; stampare minimo e massimo)

Scansione sequenziale di un array (2)

Schema di programma: scansione sequenziale

```
1. Dichiara e inizializza variabili
   opportune X,Y,... (da "portarsi dietro")

2. Per ogni elemento E dell'array
     aggiorna le variabili X,Y,... considerando E

3. Stampa il risultato sulla base dei valori finali
   di X,Y,...
```

Scansione sequenziale di un array (3)

Altri esempi in cui può essere applicato questo schema:

1. Dato un array di stringhe stampare la stringa più lunga
2. Verificare se un dato elemento è presente in un array (terminologia: problema della ricerca di un elemento)
3. In un array di caratteri, stampare il numero di occorrenze di ogni vocale (es: `"Ci sono 5 a, 3 e, 7 i, 0 o, 9 u"`)
4. Verificare se i valori di un array sono in ordine crescente

Che informazioni bisogna "portarsi dietro" in ogni caso?

Scansione a doppio indice di un array (1)

Esempio: Inversione di un array: ridisporre gli elementi dell'array in ordine inverso (dall'ultimo al primo)

Da così:

`valori`	7	6	12	667	-33	0	13	-34	4	100
	0	1	2	3	4	5	6	7	8	9

A così:

`valori`	100	4	-34	13	0	-33	667	12	6	7
	0	1	2	3	4	5	6	7	8	9

- Idee su come fare?

Scansione a doppio indice di un array (2)

Idea: scambio il primo elemento con l'ultimo, il secondo con il penultimo, ecc...

Attenzione: come si scambiano i valori di due elementi dell'array?

- **NON** in questo modo:

```
valori[0] = valori[9];
valori[9] = valori[0]; //AAARGH!! che fine ha fatto valori[0]?
```

- **CI VUOLE** una variabile d'appoggio:

```
int tmp = valori[0];
valori[0] = valori[9];
valori[9] = tmp;
```

Lo stesso vale quando si vogliono scambiare i valori di due normali variabili

Scansione a doppio indice di un array (3)

Ecco la soluzione:

```
for (int i=0; i<(valori.length/2); i++) { //notare la guardia

  // posiz. elem. da scambiare con quello in posizione i
  int j=(valori.length-1)-i;

  int tmp = valori[i];
  valori[i] = valori[j];
  valori[j] = tmp;
}
```

oppure

```
for (int i=0, j=valori.length-1; i<j; i++,j--) { //notare la guardia
  int tmp = valori[i];
  valori[i] = valori[j];
  valori[j] = tmp;
}
```

In questo caso stiamo usando due indici, `i` e `j`, per muoverci nell'array

- In questo caso `i` e `j` sono entrambi aggiornati ad ogni iterazione
- In altri casi `i` e `j` possono essere aggiornati a turno (es. successivo)
- Notare che questo ciclo fa `valori.length/2` iterazioni, ma ad ogni iterazioni vengono letti 2 valori. Totale: tutti i `valori.length` valori vengono letti

Scansione a doppio indice di un array (4)

Altro esempio: Si supponga di avere un array `valori` di 10 elementi in cui gli elementi della prima metà e della seconda metà sono disposti in ordine crescente. Copiare tutti gli elementi di `valori` in un nuovo array `valori2` in modo che siano tutti ordinati.

Se questo è `valori`:

`valori`	3	6	12	66	74	1	13	15	91	100
	0	1	2	3	4	5	6	7	8	9

Questo deve essere `valori2`:

`valori2`	1	3	6	12	13	15	66	74	91	100
	0	1	2	3	4	5	6	7	8	9

- Idee su come fare?

Scansione a doppio indice di un array (5)

Idea: confronto `valori[0]` con `valori[5]` e copio il minore in `valori2`. Incremento l'indice del valore che ho copiato, ripeto il confronto e continuo così .....

```
int valori2 = new int[10];
int i=0, j=5, k=0;  //k serve per scorrere valori2
do {
  // copia il minore e incrementa i o j
  if (valori[i]<valori[j]) {
    valori2[k] = valori[i];
    i++;
  } else {
    valori2[k] = valori[j];
    j++;
  }

  // incrementa k
  k++;

} while ((i<5)&&(j<10));

//uno tra i e j non ha concluso il suo lavoro...
//solo uno di questi due cicli viene eseguito
while (i<5) { valori2[k]= valori[i]; i++; k++; }
while (j<10) { valori2[k]= valori[j]; j++; k++; }
```

Scansione a doppio indice di un array (6)

Schema di programma: scansione a doppio indice

1. Dichiara e inizializza due indici i e j

2. In un ciclo
   usa i e j per confrontare/scambiare i valori dell'array
   aggiorna i e/o j

3. Gestisci eventuali elementi dell'array non considerati

4. Stampa il risultato

Scansione a doppio indice di un array (6)

Altri esempi in cui può essere applicato questo schema:

1. Verificare se un array di `char` è palindromo, ossia la sequenza di caratteri letti da sinistra a destra, oppure da destra a sinistra è la stessa. (Il concetto di palindromo è di solito riferito alle stringhe: ad es. `"ailatiditalia"`, `"itopinonavevanonipoti"`, `"inotipiedideipitoni"`,...)

Scansione di un array con cicli annidati (1)

Nello schema precedente ogni elemento dell'array veniva confrontato con un altro elemento.

Spesso capita di dover confrontare ogni elemento dell'array con tutti gli altri elementi

Esempio: Un programma che verifica se esistono due elementi uguali nell'array

Come in questo caso:

`valori`	7	6	12	667	-33	0	12	-34	4	100
	0	1	2	3	4	5	6	7	8	9

Bisogna per forza confrontare ogni elemento con tutti gli altri

Scansione di un array con cicli annidati (2)

Idea: prendo il primo elemento e lo confronto con tutti i successivi, prendo il secondo elemento e lo confronto con tutti i successivi, ...

```
// flag che diventa true quando si trova un elemento ripetuto
boolean trovato=false;
// per ogni elemento (ultimo escluso)...
for (int i=0; i<9; i++) {
  // ... considera tutti i successivi (da i+1) ...
  for (int j=i+1; j<10; j++) {
     // ... e confronta
    if (valori[i]==valori[j]) trovato=true;
  }
}
```

Note:

- quando si considera il secondo elemento non c'e' bisogno di confrontarlo con il primo (già fatto!), e così via...
- per evitare tali controlli superflui non si può usare `for-each` (non consente di gestire gli indici)
- si possono usare anche cicli `while` e `do-while` (ad esempio, per interrompere la ricerca appena `trovato` diventa `true`)

Scansione di un array con cicli annidati (3)

Schema di programma: scansione con cicli annidati

```
1. Dichiara e inizializza un indice i

2. Cicla per i che va da 0 alla penultima posizione
   In ogni iterazione del ciclo:

   a. Dichiara e inizializza un indice j

   b. Cicla per j che va da i+1 all'ultima posizione
      In ogni iterazione del ciclo:
          Usa i e j per confrontare/scambiare gli elementi

3. Stampa il risultato
```

Scansione di un array con cicli annidati (4)

Altro esempio: Ordinamento di un array (algoritmo *Selection sort*)

Da così:

`valori`	7	6	12	667	33	10	13	34	4	100
	0	1	2	3	4	5	6	7	8	9

A così:

`valori`	4	6	7	10	12	13	33	34	100	667
	0	1	2	3	4	5	6	7	8	9

Scansione di un array con cicli annidati (5)

L'algoritmo Selection sort è (più o meno) quello che molti usano per sistemarsi le carte in mano...

Scansione di un array con cicli annidati (6)

Idea: cerco il valore minimo e lo metto in prima posizione (scambiandoli), cerco il minimo dei rimanenti e lo metto in seconda posizione (scambiandoli), ...

Animazione: in rosso i valori da scambiare e in grassetto i valori "sistemati"

`valori`	7	6	12	667	33	10	13	34	4	100
	0	1	2	3	4	5	6	7	8	9
`valori`	**4**	6	12	667	33	10	13	34	7	100
	0	1	2	3	4	5	6	7	8	9
`valori`	**4**	**6**	12	667	33	10	13	34	7	100
	0	1	2	3	4	5	6	7	8	9
`valori`	**4**	**6**	**7**	667	33	10	13	34	12	100
	0	1	2	3	4	5	6	7	8	9

eccetera...

Scansione di un array con cicli annidati (7)

Ecco la soluzione (Selection sort):

```
for (int i=0; i<9; i++) {

  // cerca il minimo nelle posizioni tra i e 10...
  int posmin=i; //... portandosi dietro l'indice
  for (int j=i; j<10; j++) {
    if (valori[j]<valori[posmin])
      posmin=j;
  }

  // scambia il minimo con l'elemento in posizione i
  int tmp = valori[i];
  valori[i] = valori[posmin];
  valori[posmin] = tmp;
}
```

Note:

- La ricerca del minimo segue lo schema "scansione sequenziale" portandosi dietro un indice `posmin`
- Il `for` interno fa partire `j` dal valore `i`, non `i+1` (piccolo aggiustamento dello schema "scansione con cicli annidati")

Scansione di un array con cicli annidati (8)

Altri esempi in cui può essere applicato questo schema:

1. Altri algoritmi di ordinamento iterativi (Ad esempio: Insertion sort e Bubble sort)
2. Stampare tutte le coppie possibili di valori dell'array
3. Determinare il numero di copie dell'elemento che compare più volte nell'array

In tutti questi esempi il tipo dei valori dell'array può essere numerico, `char` o `String`

Breve confronto tra gli schemi proposti (1)

Abbiamo visto

- Scansione sequenziale
- Scansione a doppio indice
- Scansione con cicli annidati

I tre schemi servono per risolvere problemi diversi

La "scansione sequenziale" usa un solo indice, mentre gli altri ne usano due

- I due indici sono gestiti da un unico ciclo nella "scansione a doppio indice"
- Sono invece gestiti da due cicli diversi nella "scansione con cicli annidati"

Breve confronto tra gli schemi proposti (2)

Nella "scansione sequenziale" e "a doppio indice" ogni elemento dell'array viene usato una volta sola (o comunque poche volte)

- Nella "scansione con cicli annidati" ogni elemento può essere usato fino a `length` volte

Questo incide sul tempo di esecuzione del programma (o meglio sul numero di operazioni che dovrà eseguire – detto anche complessità computazionale)

- Un programma che usa la "scansione sequenziale" o la "scansione a doppio indice" esegue un numero di operazioni proporzionale alla dimensione dell'array (complessità lineare)
- Un programma che usa la "scansione con cicli annidati" esegue un numero di operazioni proporzionale al quadrato della dimensione dell'array (complessità quadratica)

Al crescere della dimensione dell'array, il numero di operazioni dell'approccio "scansione con cicli annidati" cresce più velocemente che negli altri casi (per approfondimenti: libro di Crescenzi, Gambosi e Grossi)