Gli Array

Programmazione

Corso di Laurea in Informatica
Corso di Laurea in Informatica per la Comunicazione Digitale

Università di Camerino

Prof. Michele Loreti

Anno Accademico 2024/25

Motivazione

Problema

- alcuni problemi riguardano collezioni di dati correlati tra loro, non è ragionevole avere una singola variabile per ogni dato
  - gioco del Sudoku (e altri giochi basati su “tabelle”)
  - rappresentazione di matrici
- in alcuni problemi non è nota a priori la quantità di dati su cui occorre operare e non è possibile elaborarli “un po’ alla volta”
  - calcolo della media aritmetica vs calcolo della varianza
  - ricerca e ordinamento

Soluzione

- introduciamo un nuovo tipo di dato contenitore (**array**) per la rappresentazione di **sequenze** di dati di lunghezza arbitraria

Creazione e tipo di un array

Creazione di un array

```
int[] a = { 2, 3, 5, 7 };
```

- crea un **array** di 4 elementi `int`
- gli elementi sono 2, 3, 5 e 7 in quest'ordine
- utile per array molto piccoli e dei quali si conoscono gli elementi

```
int[] a = new int[4];
```

- crea un **array** di 4 elementi `int`
- gli elementi *non sono definiti* (possono esistere convenzioni)
- indispensabile per creare array dei quali non si conosce a priori il contenuto o la dimensione

Note

- un **tipo array** ha la forma `tipo[]` dove "`tipo`" è il tipo dei suoi **elementi**
- "`tipo`" può essere a sua volta un tipo array (array bi/multidimensionali)
- `new` è una parola chiave di Java, non può essere usata come identificatore
- in generale possono esserci **espressioni** al posto delle costanti 2, 3, 4, ...

Lunghezza di un array

- la **dimensione** (o **lunghezza**) di un array è il numero dei suoi elementi
- la dimensione è fissata al momento della creazione dell'array e non varia più per tutta la vita dell'array
- la dimensione di un array non fa parte del suo tipo (es. due array con elementi di tipo `int`, uno di lunghezza 4 e l'altro di lunghezza `4000` hanno lo stesso tipo `int[]`)
- se a è una variabile (o espressione) di tipo array, allora `a.length` è un valore di tipo `int` che indica la dimensione di a

Esempi

```
int[] a = { 2, 3, 5, 7 };
println("Lunghezza di a = " + a.length); // stampa 4
```

```
int[] a = new int[4];
println("Lunghezza di a = " + a.length); // stampa 4
```

Nota: `length` **non** è una parola chiave di Java, pertanto può essere usato come identificatore di metodi, argomenti o variabili

Accesso agli elementi di un array

- gli elementi di un array di lunghezza n sono **indicizzati** da 0 fino a $n - 1$
  - il **primo** elemento ha indice 0
  - il **secondo** elemento ha indice 1
  - ...
  - l'**ultimo** elemento ha indice $n - 1$
- se `a` è una variabile (o espressione) di tipo array e `i` una variabile (o espressione) di tipo `int`, allora `a[i]` è l'elemento di `a` con indice `i`

Esempio

```
int[] a = { 2, 3, 5, 7 };
for (int i = 0; i < a.length; i++)
  println("a[" + i + "] = " + a[i]);
```

```
a[0] = 2
a[1] = 3
a[2] = 5
a[3] = 7
```

Modifica degli elementi di un array

- la forma `a[i]` può apparire anche **a sinistra** di un operatore di assegnamento, nel qual caso viene **modificato** (cioè **sovrascritto**) l'elemento con indice `i` dell'array a

Esempio

```
int[] a = { 2, 3, 5, 7 };
for (int i = 0; i < a.length; i++)
  a[i] = a[i] + 1;
for (int i = 0; i < a.length; i++)
  println("a[" + i + "] = " + a[i]);
```

```
a[0] = 3
a[1] = 4
a[2] = 6
a[3] = 8
```

Esempio: media aritmetica

Problema

Dati n elementi x_1, ..., x_n, calcolare la quantità $\frac{\sum_{i=1}^{n} x_i}{n}$

Soluzione

Esempio: media aritmetica

Problema

Dati n elementi x_1, ..., x_n, calcolare la quantità $\frac{\sum_{i=1}^{n} x_i}{n}$

Soluzione

```
double media_aritmetica(int[] a) {
    double s = 0;
    for (int i = 0; i < a.length; i++)
  s = s + a[i];
    return s / a.length;
}

int[] a = { 2, 3, 5, 7 };
println("Media = " + media_aritmetica(a));
```

Esempio: varianza

Problema

Dati n elementi x_1, ..., x_n, calcolare la quantità $\frac{\sum_{i=1}^{n}(x_i-\overline{x})^2}{n}$ dove $\overline{x}$ è la media aritmetica degli x_i

Soluzione

Esempio: varianza

Problema

Dati n elementi x_1, ..., x_n, calcolare la quantità $\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}$ dove $\overline{x}$ è la media aritmetica degli x_i

Soluzione

```
double varianza(int[] a) { // a.length > 0;
    double m = media_aritmetica(a);
    double r = 0;
    for (int i = 0; i < a.length; i++)
  r = r + Math.pow(a[i] - m, 2);
    return r / a.length;
}

int[] a = { 2, 3, 5, 7 };
println("Varianza = " + varianza(a));
```

Esempio: verifica dell’ordinamento

Problema

Data una sequenza di elementi x_1, ..., x_n verificare se è ordinata in modo non decrescente, ovvero se $x_1 \leq x_2 \leq \cdots \leq x_n$

Soluzione

Esempio: verifica dell'ordinamento

Problema

Data una sequenza di elementi x_1, ..., x_n verificare se è ordinata in modo non decrescente, ovvero se $x_1 \leq x_2 \leq \cdots \leq x_n$

Soluzione

```
boolean ordinato(int[] a) {
    for (int i = 0; i < a.length - 1; i++)
  if (a[i] > a[i + 1]) return false;
    return true;
}

int[] a = { 2, 3, 5, 7 };
println("Ordinato = " + ordinato(a));
int[] b = { 2, 5, 3, 7 };
println("Ordinato = " + ordinato(b));
```

Esempio: presenza di elementi ripetuti

Problema

Data una sequenza x_1, ..., x_n verificare se esiste un elemento presente più di una volta, ovvero se esistono i e j con $i \neq j$ tali che $x_i = x_j$

Soluzione

Esempio: presenza di elementi ripetuti

Problema

Data una sequenza x_1, ..., x_n verificare se esiste un elemento presente più di una volta, ovvero se esistono i e j con $i \neq j$ tali che $x_i = x_j$

Soluzione

```
boolean ripetizioni(int[] a) {
    for (int i = 0; i < a.length; i++)
  for (int j = i + 1; j < a.length; j++)
      if (a[i] == a[j]) return true;
    return false;
}

int[] a = { 2, 3, 5, 7 };
println("Ripetizioni = " + ripetizioni(a));
int[] b = { 2, 7, 5, 7 };
println("Ripetizioni = " + ripetizioni(b));
```

Organizzazione in memoria di un array

- il contenuto di un array è memorizzato in un'area di memoria **contigua** nell'**heap**
- gli elementi di un array sono memorizzati **uno di seguito all'altro** in tale area
- un argomento o variabile di tipo array contiene solo un **riferimento** all'area di memoria allocata
- il "costo" per accedere a un elemento dell'array è molto basso, essenzialmente occorre calcolare `base + dimensione elemento × indice` dove
  - `base` è il riferimento all'array
  - `dimensione elemento` è il numero di byte occupati da ogni elemento ed è determinato dal tipo dell'array (es. per un array di tipo `int[]` la dimensione di ogni elemento è di 4 byte)
  - `indice` è l'indice dell'elemento a cui si vuole accedere

Argomenti di tipo array

```
void modifica_array(int[] a) {
    a[0]++;
}

void stampa_array(int[] a) {
    print("[");
    for (int i = 0; i < a.length; i++)
  print(" " + a[i]);
    println(" ]");
}

int[] a = { 2, 3, 5, 7 };
stampa_array(a);
modifica_array(a);
stampa_array(a);
```

Risultato

```
[ 2 3 5 7 ]
[ 3 3 5 7 ]
```

Morale

- passare un array come argomento significa passare un **riferimento** alla regione di memoria (nell'heap) in cui l'array è allocato
- il "costo" del passaggio di un array come argomento è **indipendente** dalla dimensione dell'array (il contenuto non è copiato)
- la modifica di un argomento di tipo array **si riflette** sul chiamante (e su chiunque altro abbia un riferimento allo stesso array)

Confronto tra array

```
int[] a = { 2, 3, 5, 7 };
int[] b = { 2, 3, 5, 7 };
println("A == A ==> " + (a == a));
println("A != A ==> " + (a != a));
println("A == B ==> " + (a == b));
println("A != B ==> " + (a != b));
```

```
Risultato

A == A ==> true
A != A ==> false
A == B ==> false
A != B ==> true
```

Morale

- confrontare due array significa confrontarne i riferimenti, non il contenuto
- se si vuole confrontare il contenuto di due array si può definire un metodo
- gli operatori relazionali <, >, <= e >= non sono applicabili agli array

```
boolean array_uguali(int[] a, int[] b) {
    if (a.length != b.length) return false;
    for (int i = 0; i < a.length; i++)
        if (a[i] != b[i]) return false;
    return true;
}
```

Array di array

Creazione di un array bidimensionale

```
int[][] a = { { 2, 3, 5, 7 },
              { 11, 13, 17 } };
println("A    " + a.length);
println("A[0] " + a[0].length);
println("A[1] " + a[1].length);
```

```
A    2
A[0] 4
A[1] 3
```

- crea un **array** con due elementi di tipo `int[]`
- il primo array interno ha 4 elementi di tipo `int`
- il secondo array interno ha 3 elementi di tipo `int`

```
int[][] a = new int[2][3];

println("A    " + a.length);
println("A[0] " + a[0].length);
println("A[1] " + a[1].length);
```

```
A    2
A[0] 3
A[1] 3
```

- crea un **array** di 2 elementi di tipo `int[]`
- ciascun array interno ha 3 elementi di tipo `int`

Esempio: creazione della matrice identità

Problema

Definire un metodo con un argomento `n` di tipo `int` e tipo di ritorno `int[][]` che crea la matrice identità di dimensione `n`

Soluzione

```
int[][] identita(int n) { // n >= 0;
    int[][] m = new int[n][n];
    for (int i = 0; i < n; i++)
  for (int j = 0; j < n; j++)
      m[i][j] = i == j ? 1 : 0;
    return m;
}
```

Nota

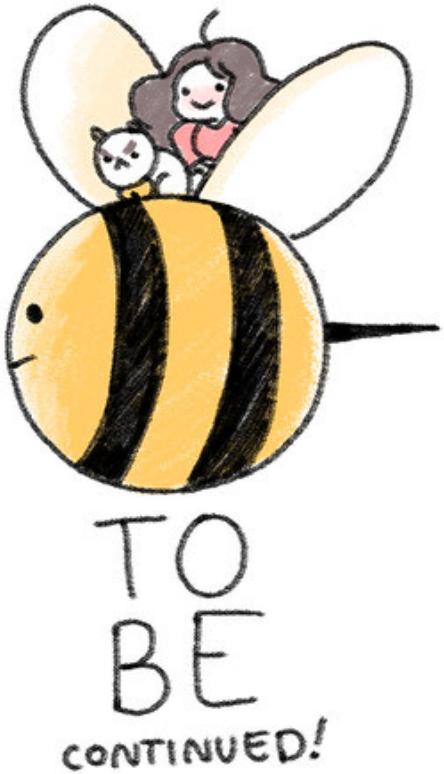
- `m[i]` è la **riga** con indice `i` della matrice
- `m[i][j]` è l'**elemento** con indice `j` della riga con indice `i` della matrice

Organizzazione in memoria di un array di array

```
void metodo() {
    ...
    int[][] a = { {  2,  3,  5, 7 },
                  { 11, 13, 17    } };
    ...
}
```

Esercizi

1. Definire il metodo `void inverti(int[] a)` che **inverte l'ordine** degli elementi in a. Ripetere l'esercizio, questa volta definendo un metodo `int[] inverti(int[] a)` che **non modifica** a.
2. Definire i metodi `int min_array(int[] a)` e `int max_array(int[] a)` che determinano rispettivamente l'elemento **più piccolo** e **più grande** di a.
3. Date due sequenze $a_1, \ldots, a_m$ e $b_1, \ldots, b_n$ la loro **concatenazione** è la sequenza $a_1, \ldots, a_m, b_1, \ldots, b_n$. Definire un metodo `int[] concatena(int[] a, int[] b)` che calcola la concatenazione di a e b.
4. Date due sequenze $a_1, \ldots, a_m$ e $b_1, \ldots, b_n$ di elementi sui quali è definita una nozione d'ordine $\leq$ diciamo che la prima sequenza è **lessicograficamente minore o uguale** della seconda se $m \leq n$ e $a_i \leq b_i$ per ogni $1 \leq i \leq m$. Definire un metodo `boolean le(int[] a, int[] b)` che determina se a è lessicograficamente minore o uguale a b.
5. **Difficile!** Definire un metodo `int[][] permutazioni(int n)` che calcola tutte le permutazioni dei numeri interi da 1 a n. **Suggerimento**: definire un metodo ausiliario `int[] inserisci(int k, int[] a, int i)` che crea un array uguale ad a ad eccezione del fatto che l'elemento k è stato inserito in posizione `i`.


TO
BE
CONTINUED!