Programmazione strutturata IV e Gli array

Stefano Mizzaro

Dipartimento di matematica e informatica
Università di Udine
http://www.dimi.uniud.it/mizzaro
mizzaro@dimi.uniud.it

Programmazione, lezione 7
18 ottobre 2004

Dove siamo

- Programmazione strutturata
- Strutture di controllo
  - Sequenza: `;` e `{}`
  - Selezione: `if`, `if/else`, `switch/case`
  - Iterazione: `while`, `do/while`, `for`
- Sviluppo incrementale
- Equivalenze, `++` e `--`, errori tipici, regole di buona programmazione
- **`Leggi.java`**

Oggi

- **`break`** e **`continue`**
- Esempi
- Introduzione agli array

`break` e `continue`

- Interrompono un ciclo (il ciclo che li racchiude)
- **`break`**: esce dal ciclo
- **`continue`**: passa all'iterazione successiva
  - ("dopo la `}`" e "prima della `}`")
- Parole riservate
- Istruzioni
- Etichettati

Esempi

```
for (int i = 1; i <= 10; i++) {
  if (i == 5)
    break;
  System.out.println(i);
}
System.out.println("Fuori dal ciclo");
```

```
for (i = 1; i <= 10; i++) {
  if (i == 5)
    continue;
  System.out.println(i);
}
System.out.println("Fuori dal ciclo");
```


Esempio di `break` etichettato

```
pippo: {
  for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10; j++) {
      System.out.print('*');
      if ((i == 7) && (j == 9))
        break pippo;
    }
    System.out.println();
  }
}
```

Un esercizio

- Scrivere un programma per calcolare il massimo comun divisore di due numeri naturali
- Facciamolo insieme…
- Diciamo, per fissare le idee, che i due numeri sono memorizzati nelle variabili **`x`** e **`y`**

Prima idea

- Usiamo un ciclo che fa assumere alla variabile **`i`** tutti i valori da 1 fino al minimo dei due numeri **`x`** e **`y`**
- Per ogni valore di **`i`** verifichiamo se **`i`** è un divisore sia di **`x`** sia di **`y`**
- Se lo è, lo memorizziamo in una variabile **`mcd`**
- L'ultimo divisore che troviamo (che memorizziamo in **`mcd`**) è il massimo
- …schema…

Prima versione

```
"Per i che va da 1 al minimo fra x e y"
  "se i divide x e y allora metto i in mcd"
"Stampo mcd"
```

```
for (i = 1; i <= "minimo fra x e y"; i++) {
  if ("i e' un divisore di x e y")
    mcd = i;
}
"Stampo mcd"
```

```
for (i = 1; i <= ((x < y)? x : y); i++) {
  if ((x % i == 0) && (y % i == 0))
    mcd = i;
}
System.out.println(mcd);
```


Seconda idea

- Mai accontentarsi…
- Ma se invece di incrementare **`i`** e "aspettare" di trovare l'ultimo divisore… facciamo "al contrario"?
  - Il ciclo che fa assumere a **`i`** tutti i valori dal minimo di **`x`** e **`y`** fino a 1
  - Per ogni valore di **`i`** verifichiamo se **`i`** è un divisore sia di **`x`** sia di **`y`**
  - Il primo divisore che troviamo è il massimo
- …schema…

Seconda versione

```
for (i = ((x < y)? x : y); i >= 1; i--)
  if ((x % i == 0) && (y % i == 0)) {
    mcd = i;
    break;
}
System.out.println(mcd);
```

```
mcd = 1;
i = ((x < y)? x : y);
while (i >= 1 && mcd == 1) {
  if ((x % i == 0) && (y % i == 0))
    mcd = i;
  i--;
}
System.out.println(mcd);
```


Confronto

- Qual è la migliore?
- La seconda è più "veloce"
  - Nella prima **`i`** deve assumere sempre tutti i $min(x,y)$ valori
  - Nella seconda assume tutti i $min(x,y)$ valori solo se x e y sono primi fra loro
- Meno iterazioni del ciclo = più veloce
- Meno iterazioni = meno valori assunti da **`i`**

Terza idea

- Mai accontentarsi!!!!
- Ci sono altri valori di `i` che possiamo escludere? Cioè: ci sono valori di `i` che non sono senz'altro un MCD?
- Sì! Dato un numero k, quanto grande è, al massimo, il suo massimo divisore?
  - (escluso k stesso)
  - (quindi non dimentichiamoci di $min(x,y)$...)
- ... schema...

Terza versione

```
min = ((x < y)? x : y);
if ((x % min == 0) && (y % min == 0))
  mcd = min;
else
  for (i = min/2; i >= 1; i--)
    if ((x % i == 0) && (y % i == 0)) {
      mcd = i;
      break;
    }
System.out.println(mcd);
```


Quarta idea!

- MAI ACCONTENTARSI!!!!
- E ricordare che ragionare a livello del codice non è sempre il metodo migliore...
- Idea completamente diversa, basata su una proprietà algebrica (Euclide)
  - Se $x = y \Rightarrow mcd(x,y) = x = y$
  - Se $x > y \Rightarrow mcd(x,y) = mcd(x - y, y)$
- Possiamo sfruttarla per decrementare `x` e `y` fino a quando sono uguali
- ... schema...

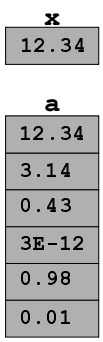
Quarta versione

```
while (x != y)
  if (x > y)
    x = x - y;
  else
    y = y - x;
System.out.println(x);
```

- Più elegante
- Più sintetico
- Più efficiente
- (e ce ne sarebbe anche una quinta...)

Gli array

- (Cap. 4)
- Variabile:
  - Scatola
  - Con nome, valore e tipo
- Array:
  - Sequenza di scatole, ognuna delle quali
    - "Ha un nome"
    - Contiene un valore
    - È di un tipo



Esempio

- Programma che deve memorizzare 10 valori di temperatura
  - Uso 10 variabili `double temperatura1, temperatura2, ..., temperatura10`??
  - Uso un array `temperature` di 10 posizioni di `double`
- Posso comunque accedere ad ogni singola "scatola" tramite la posizione (la prima, la seconda, ...)

	temperature
0	20.5
1	20.0
2	19.8
3	19.4
4	20.4
5	21.6
6	22.0
7	21.9
8	21.0
9	21.1

In Java

```
double[] temperature;
temperature = new double[10];
temperature[0] = 20.5;
temperature[1] = 20.0;
temperature[2] = 19.8;
temperature[3] = 19.4;
temperature[4] = 20.4;
temperature[5] = 21.6;
...
System.out.print(temperature[1]);
System.out.print(temperature[5]);
...
temperature[5] = temperature[4];
temperature[5] = temperature[5]+1;
temperature[5+1] = temperature[5];
...
```

temperature

0	20.5
1	20.0
2	19.8
3	19.4
4	20.4
5	21.6
6	22.0
7	21.9
8	21.0
9	21.1

Sintassi

- Dichiarazione
  - *tipo*`[]` *nome*`;`
  - Es.: `double[] temperature;`
- Allocazione
  - *nome* `= new` *tipo*`[`*lunghezza*`];`
  - Es.: `temperature = new double[10];`
- Uso
  - *nome*`[`*posizione*`]`
  - Es.: `temperature[5+1] = temperature[5];`

Vantaggio degli array

- Gestione uniforme
- Ad es., per azzerare tutte le temperature…

```
temperature[0] = 0;
temperature[1] = 0;
temperature[2] = 0;
...
temperature[9] = 0;
```

```
for (int i = 0; i <= 9; i++)
  temperature[i] = 0;
```


Gestione uniforme

- Calcolare la media delle temperature

```
double[] temperature;
temperature = new double[10];
double media = 0;
...
for (int i = 0; i <= 9; i++) {
  media = media + temperature[i];
System.out.print(media/10);
```

- Fatelo con le 10 variabili…
- … e poi fatelo per 100 temperature…

Dichiarazione

- Simile alle dichiarazioni di variabili
- Si usano le `[]`
  - dopo il tipo (meglio! "Array di tipo…")
  - o dopo il nome

```
int[] a1;
int i;
char[] a2, a3;
char c;
double[] a4;
```

```
int a1[], i;
char a2[], a3[], c;
double a4[];
```


Allocazione

- Non c'era per le variabili, serve per allocare la memoria necessaria per gli elementi
- Si usano
  - `new`
  - `[`*lunghezza*`]` (numero di elementi)

```
a1 = new int[10];
a2 = new char[12];
a3 = new char[5];
a4 = new double[7];
```

- Dichiarazione e allocazione si possono fare insieme

```
int a1[] = new int[10], i;
char a2[] = new char[12], a3[] = new char[5], c;
double[] a4 = new double[7];
```

Inizializzazione

- È possibile assegnare valori iniziali a un array

```
int[] giorniMese =
          {31,28,31,30,31,30,31,31,30,31,30,31};
```

~~int[] giorniMese;~~
~~giorniMese={31,28,31,30,31,30,31,31,30,31,30,31};~~

```
int[] giorniMese;
giorniMese =
 new int[] {31,28,31,30,31,30,31,31,30,31,30,31};
```


Uso

- Per ora, ogni posizione è come una variabile
  - (si può anche lavorare su tutto un array)
  - Si usano le `[]` e un **indice** (un'espressione `int`)
  - Però l'indice può essere un'espressione! (`for`, …)
- L'indice del primo elemento è 0 (zero)!!!

```
a1[4] = 5;
if (a1[4] > i) {
  a2[10] = 'p';
  a4[0] = a4[1];
}
```


Il `.length`

- C'è modo di sapere la lunghezza di un array (il numero di locazioni)
- ***nomearray*`.length`**
- "th", non "ht"!!!!!!!!!!!!!

```
char[] a = new char[12];
char[] b = {'a', 's', 'd'};
System.out.println(a.length);
System.out.println(b.length);
System.out.println(b[b.length - 2]);
System.out.println(b[a.length/6]);
```


Esercizi

- Azzerare un array di `int` , diciamo di nome `a` (ossia, assegnare zero a tutti gli elementi di `a`)
- Assegnare 0, 1, 2, … agli elementi di `a`
- Incrementare di 1 ogni elemento dell'array `a`
- Assegnare 1 agli elementi di `a` di indice dispari e 2 a quelli di indice pari

Le parole riservate (29/50)

- Note
  - `boolean break byte case char continue default do double else false final float for if int long short switch true void while`
- Facili
  - `const goto`
- Usate
  - `class import public static throws new`

Le parole riservate – cont.

- Ancora ignote
  - `abstract catch extends finally implements instanceof interface native null package private protected return super synchronized this throw transient try volatile`

Riassunto

- ~1/4 del corso
- Mattoni
- Programmazione strutturata
- Array (rudimenti)
- Ho seguito il libro, fino a §4.3, pg. 85
- Eserciziario: capp. 1 e 2
- Prossima lezione…
  - Sempre array

Stefano Mizzaro - Prog. strutt. 4 & Array 31