La programmazione strutturata - III

Stefano Mizzaro

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

Programmazione, lezione 6
10 marzo 2015

Oggi

- Riassunto
- **`for`**
- Cicli annidati
- Esempi, esercizi, errori tipici
- Programmazione strutturata
- **`break`** e **`continue`**

Dove siamo

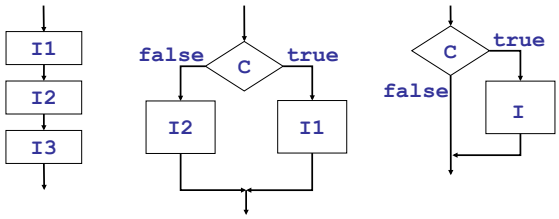
- Strutture di controllo della programmazione strutturata
  - Sequenza
    - `;` e `{}`
  - Selezione
    - **`if, if/else, switch/case`**
  - Iterazione (ripetizione)
    - **`while, do/while,`** `for`
    - `(break, continue)`
- Sviluppo incrementale per raffinamenti successivi
  - "Step-wise refinement"

Sequenza e selezione

```
I1;
I2;
I3;
```

```
if (C)
  I1
else
  I2
```

```
if (C)
  I
```

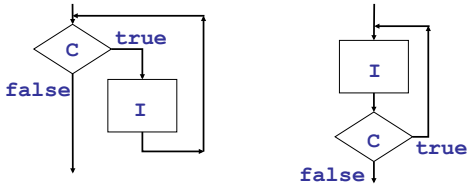


Iterazione: `while` e `do/while`

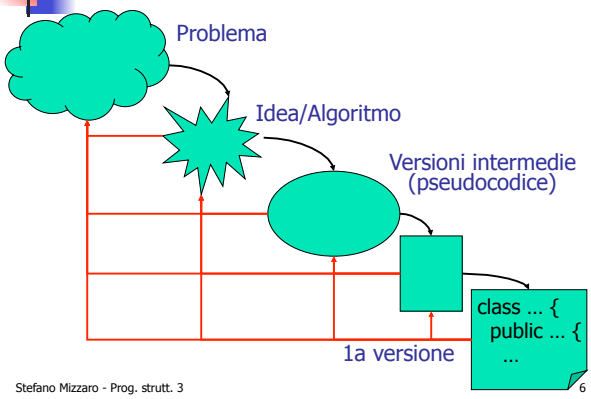
```
while (C)
  I
```

```
do {
  I
} while (C);
```

"Corpo"

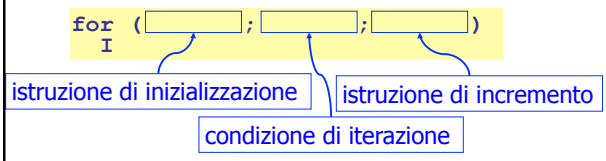


Raffinamenti successivi



Il `for`

- Terza e ultima istruzione di iterazione



- (e `I` è l'istruzione che viene iterata)
- Intestazione e corpo del ciclo `for`

Esempi

```
for (i = 0; i <= 10; i = i + 1)
  System.out.println(i);
```

```
for (i = 0; i <= 10; i = i + 1) {
  System.out.print(i + " ");
  System.out.println(i * i);
}
```


Significato del `for`

- Quindi il significato di

  `for(Init; C; Inc) I`

  è:
  - Nell'istruzione di inizializzazione `Init` viene inizializzato il valore di una variabile di controllo
  - Se la condizione di iterazione `C` è **`true`**, il corpo del ciclo (`I`) viene eseguito
  - Al termine dell'esecuzione di `I` viene eseguita l'istruzione di incremento `Inc`
  - …e `C` e `Inc` "riguardano" la variabile di controllo

Cicli `for` e `while`

- Si può simulare un **`for`** con un **`while`**:

```
for (I1; C; I2)
  I3;
```

⇒

```
I1;
while (C) {
  I3;
  I2;
}
```

- Prima la condizione `C`
- L'incremento `I2` alla fine

Vantaggio del `for`

- Gestione unitaria della variabile di controllo
  - Inizializzazione
  - Incremento
  - Controllo
- Con un **`while`** o con un **`do/while`** invece devo cercare in giro per il codice
- E anche la dichiarazione può essere fatta insieme all'inizializzazione…

Dichiarazione nel `for`

- Si può dichiarare la variabile di controllo del ciclo nell'intestazione del **`for`**:

```
for (int i = 0; i <= 10; i = i + 1)
  System.out.println(i);
```

- È comodo (1 sola riga invece di 4!), ma attenzione: `i` "esiste" solo nel corpo del **`for`**

```
for (int i = 0; i <= 10; i = i + 1)
  System.out.println(i);
System.out.println(i);
```

Incrementi e decrementi…

- L'incremento può non essere unitario…

```
for (i = 0; i <= 10; i = i + 2)
  System.out.println(i);
```

- … o essere un decremento

```
for (i = 10; i >= 1; i = i - 1)
  System.out.println(i);
```


Errori tipici

```
for (int i = 0; i <= 10; i = i + 1);
  System.out.println(i);
```

- … quello che sembra nel corpo del **`for`** non lo è…
  - si evita se si mette la `{` (perché `);{` "suona strano"…)

```
for (int i = 10; i <= 1; i = i - 1)
  System.out.println(i);
```

- … la condizione è invertita
- …

Uso del `for`

- Non modificate la variabile di controllo nel corpo del **`for`** → usate il **`while`**
- Evitare cose del tipo

```
for ( ; ; )
  I
```

⇒

```
while (true)
  I
```


Operatori di incremento e decremento

- Le forme più tipiche dell'istruzione di incremento in un **`for`** sono
  `i = i + 1` e **`i = i - 1`**
- 2 "istruzioni" più comode
  - `++i` (o `--i`): pre-incrementa (decrementa) `i`
  - `i++` (o `i--`): post-incrementa (o decrementa) `i`
- (e non solo nei **`for`**…)
- Ma NON sono istruzioni! `++` e `--` sono operatori, e quindi `++i` (e …) sono espressioni

Sono operatori!

- Quindi possono stare dentro a espressioni
- Anche se spesso sono usati da soli:

```
for (int i = 1; i <= 10; i++)
  System.out.println(i);
```

```
for (int i = 10; i >= 1; i--)
  System.out.println(i);
```

- Pre e post: quanto vale `i`?

```
i = a;
a=a+1;
```

```
int a, i;
a = 5;
i = a++;
```

```
int a, i;
a = 5;
i = ++a;
```

```
a=a+1;
i = a;
```


Achtung!

- Errore tipico all'esame:
  - Usare `++i` e `--i` (o `i++` e `i--`) invece di `i+1` e `i-1`
  - Non sono solo espressioni, hanno anche un effetto/assegnamento
- Si può fare tutto senza!
- … quindi pensateci 2 volte…
  - (uso "sicuro" : nell'incremento dei **`for`**)

Scaletta

- Riassunto
- `for`
- Cicli annidati
- Esempi, esercizi, errori tipici
- Programmazione strutturata
- `break` e `continue`

Cicli annidati

- Cicli dentro ad altri cicli

```
for (i = 1; i <= 5; i++) {
  for (j = 1; j <= 10; j++)
    System.out.print('+');
  System.out.println();
}
```

```
for (i = 1; i <= 5; i++) {
  for (j = 1; j <= i; j++)
    System.out.print('+');
  System.out.println();
}
```

- Ragionare "a scatole"…
- Astrazione

Quale istruzione uso?

- Selezione (`if` o `switch`)
  - Condizione e numero di alternative
- Iterazione (`while, do/while, for`)
  - `for`: cicli "standard"
    - Variabile di controllo che assume valori equidistanti fino a una soglia
    - Numero di iterazioni noto a priori
  - `while, do/while`: almeno un'iterazione?
- Non è sempre facile scegliere…

Altri errori tipici (1/2)

- Ciclo infinito
  - Per mancata modifica variabile (dimenticanza…)

```
i = 0;
while (i <= 10) {
  System.out.println(i);
}
```

  - Per errata condizione (`>` o `<` meglio di `!=`)

```
i = 1;
while (i != 10) {
  System.out.println(i);
  i = i + 2;
}
```


Altri errori tipici (2/2)

- "Errore di uno"
  - Stampare numeri da 1 a 10

```
i = 1;
do {
  System.out.println(i);
  i++;
} while (i < 10);
```

```
i = 1;
do {
  i++;
  System.out.println(i);
} while (i < 10);
```

```
i = 0;
do {
  i++;
  System.out.println(i);
} while (i < 10);
```

```
i = 1;
while (i <= 10) {
  System.out.println(i);
  i++;
}
```

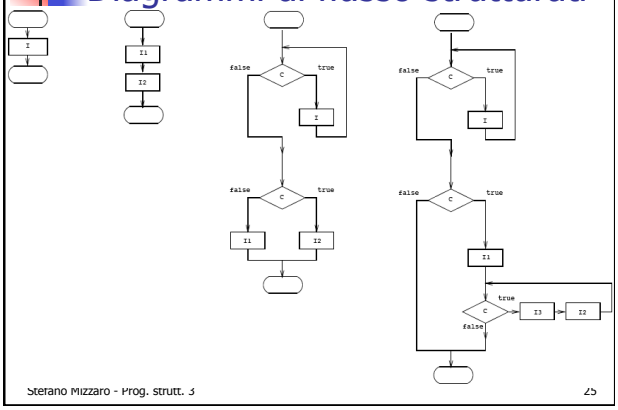
```
for (i = 1;i <= 10;i++)
  System.out.println(i);
```

```
i = 0;
while (i <= 9) {
  i++;
  System.out.println(i);
}
```


Scaletta

- Riassunto
- `for`
- Cicli annidati
- Esempi, esercizi, errori tipici
- Programmazione strutturata
- `break` e `continue`

Diagrammi di flusso strutturati



La programmazione strutturata

- L'istruzione di salto:
  - Etichetta (label)
  - "`goto`" o "`jump`"
- `if` e salto permettono di simulare tutti i cicli
- Ma… e il viceversa?

▪ Sequenza ▪ Selezione ▪ Salto	vs.	▪ Sequenza ▪ Selezione ▪ Iterazione

3 domande…

- Iterazione vs. salto
  1. Si perde qualcosa senza il salto?
- `while`, `do/while` e `for`
  2. Ma servono tutte e tre?
- Perché
  3. Ma perché queste scelte in Java (e in altri linguaggi)?

… e 3 risposte

- Böhm e Jacopini (1966)
  1. Con l'iterazione e senza il salto si riescono a rappresentare gli stessi algoritmi
  2. Con una sola fra `while`, `do/while` e `for` si simulano le altre 2
     - (bastano sequenza e `while`. Ex.: simulare `if/else` con `while`…)
- Dijkstra (1968)
  3. Salti = minore comprensibilità (spaghetti logic/programming)

Scaletta

- Riassunto
- `for`
- Cicli annidati
- Esempi, esercizi, errori tipici
- Programmazione strutturata
- `break` e `continue`

`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();
  }
}
```


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 while`
- Facili
  - `const goto`
- Usate
  - `class import public static throws new void`

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

- Strutture di controllo, programmazione strutturata
  - Sequenza
    - `;` e `{}`
  - Selezione
    - `if, if/else, switch/case`
  - Iterazione (ripetizione)
    - `while, do/while, for`
    - `break, continue`
- Sviluppo incrementale

Prossima lezione

- Esempi ed esercizi
- Array