

La programmazione strutturata - III

Stefano Mizzaro

Dipartimento di matematica e informatica
Università di Udine
<http://www.dimi.uniud.it/~mizzaro>
mizzaro@dimi.uniud.it
Programmazione, lezione 5
26 gennaio 2004

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

Stefano Mizzaro - Prog. strutt. 3

2

Oggi

- Esempi ed esercizi
- Input di dati più comodo
- do/while
- for
- Cicli annidati
- Esempi, esercizi, errori tipici
- Programmazione strutturata

Stefano Mizzaro - Prog. strutt. 3

3

Esempi ed esercizi

- (Scrivete un programma che visualizza tutti i numeri fra 1 e 100 che siano multipli di 2 o di 3)

```
i = 1;
while (i <= 100) {
  if (i % 2 == 0 || i % 3 == 0)
    System.out.println(i);
  i = i + 1;
}
```

Stefano Mizzaro - Prog. strutt. 3

4

Esempi ed esercizi

- Scrivete un programma che visualizza tutti i numeri fra 1 e 100 che siano multipli di 2 e di 3

```
i = 1;
while (i <= 100) {
  if (i % 2 == 0 && i % 3 == 0)
    System.out.println(i);
  i = i + 1;
}
```

- MEGLIO! Multipli di 2 e di 3?? Ah, multipli di 6!!

```
i = 6;
while (i <= 100) {
  System.out.println(i);
  i = i + 6;
}
```

Stefano Mizzaro - Prog. strutt. 3

5

L'input

- Come inserire dati durante l'esecuzione
- La **System.in.read()** non è il massimo:
 - Legge un solo carattere
 - Ne restituisce il codice ASCII/Unicode
- È possibile "estendere" il Java con istruzioni di input più comode
 - Lo abbiamo fatto per voi... scaricate il file **Leggi.java**, sul sito Web del corso

Stefano Mizzaro - Prog. strutt. 3

6

Uso di `Leggi.java`

- `Leggi.java` va messo (per ora) nella stessa directory del programma che lo usa!!
- Potete scrivere nei vostri programmi nuove istruzioni per la lettura di:

- `boolean`, `byte`, `short`, `int`, `long`, `float`, `double`, `char`, `String`

```
byte b;
short s;
int i;
long l;
float f;
double d;
boolean t;
char c;
String x;

b = Leggi.unByte();
s = Leggi.unoShort();
i = Leggi.unInt();
l = Leggi.unLong();
f = Leggi.unFloat();
d = Leggi.unDouble();
t = Leggi.unBoolean();
c = Leggi.unChar();
x = Leggi.unoString();
```

Stefano Mizzaro - Prog. strutt. 3

7

Esempio

- Programma per trovare il massimo fra due numeri interi

```
...
int a, b;
a = Leggi.unInt();
b = Leggi.unInt();
if (a>b)
    System.out.println(
        "Il massimo e' il primo");
else
    System.out.println(
        "Il massimo e' il secondo");
...
```

Stefano Mizzaro - Prog. strutt. 3

8

Il `do/while`

- Un'altra istruzione di iterazione
- `do I while (C)`;
 - "Fai I mentre C è vera"
- Differenza dal `while`:
 - Il controllo del valore di C viene effettuato alla fine dell'iterazione, anziché all'inizio
 - Con `while (C) I`, potrebbe darsi che I non sia eseguita neanche una volta
 - Con `do I while (C)`, sicuramente I viene eseguita almeno una volta

Stefano Mizzaro - Prog. strutt. 3

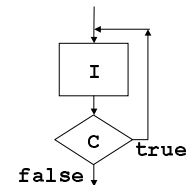
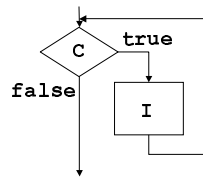
9

Diagramma di flusso

- Differenze fra `while` e `do/while`

```
while (C)
    I
```

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



Stefano Mizzaro - Prog. strutt. 3

10

Esempio

- Stampare i numeri da 1 a 10 con un `do/while`

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

```
i = 0;
do {
    i = i + 1;
    System.out.println(i);
} while (i < 10);
```

Stefano Mizzaro - Prog. strutt. 3

11

Puntualizzazioni

- Le parentesi graffe sono superflue ("`do`" e "`while`" fanno già da parentesi)
 - Meglio metterle per leggibilità
 - `} while (C)` non è senz'altro un inizio di ciclo `while` (sarebbe `while (C)`)
- `do/while` è superfluo, ma fa comodo
- Si usa meno del `while` perché "non uniforme"

Stefano Mizzaro - Prog. strutt. 3

12

Simulare il **do/while** con il **while**

- Bisogna essere sicuri che **I** sia eseguita almeno una volta...

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

⇒

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

boolean primo;
primo = true;
while (C || primo) {
  primo = false;
  I;
}
```

Stefano Mizzaro - Prog. strutt. 3

13

Simulare il **while** con il **do/while**

- Più difficile
- Non è detto che **I** sia eseguita almeno una volta...
- ... bisogna innanzitutto controllare **C**...

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

⇒

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

Stefano Mizzaro - Prog. strutt. 3

14

Esempio: controllo input

- Se volete leggere un carattere che DEVE essere `'s'` o `'n'`:

```
do {
  c = System.in.read();
} while ((c != 'n') && (c != 's'));
```

Leggi.unChar()

- ...meglio del **while** perché la prima iterazione deve essere "diversa", non uniforme

Stefano Mizzaro - Prog. strutt. 3

15

Il **for**

- Terza e ultima istruzione di iterazione

```
for ( [ ] ; [ ] ; [ ] )
  I
```

istruzione di inizializzazione istruzione di incremento

condizione di iterazione

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

Stefano Mizzaro - Prog. strutt. 3

16

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);
}
```

Stefano Mizzaro - Prog. strutt. 3

17

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

Stefano Mizzaro - Prog. strutt. 3

18

Cicli **for** e **while**

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

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

⇒

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

Stefano Mizzaro - Prog. strutt. 3

19

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...

Stefano Mizzaro - Prog. strutt. 3

20

Dichiarazione nel **for**

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

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

- È comodo, ma attenzione: **i** esiste solo nel corpo del **for**

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

Stefano Mizzaro - Prog. strutt. 3

21

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);
```

Stefano Mizzaro - Prog. strutt. 3

22

Errori tipici

- ... quello che sembra nel corpo del **for** non lo è...

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

- ... la condizione è invertita

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

- ...

Stefano Mizzaro - Prog. strutt. 3

23

Uso del **for**

- Non modificare MAI la variabile di controllo nel corpo del **for** → usare il **while**
- Evitare cose del tipo

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

⇒

```
while (true)
  I
```

Stefano Mizzaro - Prog. strutt. 3

24

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`...)
- Sono operatori...

Stefano Mizzaro - Prog. strutt. 3

25

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`?

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

Stefano Mizzaro - Prog. strutt. 3

26

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

Stefano Mizzaro - Prog. strutt. 3

27

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();
}
```

Stefano Mizzaro - Prog. strutt. 3

28

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?

Stefano Mizzaro - Prog. strutt. 3

29

Altri errori tipici (1/2)

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

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

- Per errata condizione

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

Stefano Mizzaro - Prog. strutt. 3

30

Altri errori tipici (2/2)

■ "Errore di uno"

<pre>i = 1; do { System.out.println(i); i++; } while (i < 10);</pre>	<pre>i = 1; do { i++; System.out.println(i); } while (i < 10);</pre>
<pre>i = 0; do { i++; System.out.println(i); } while (i < 10);</pre>	<pre>i = 1; while (i <= 10) { System.out.println(i); i++; }</pre>
<pre>for (i = 1; i <= 10; i++) System.out.println(i);</pre>	<pre>i = 0; while (i <= 9) { i++; System.out.println(i); }</pre>

Stefano Mizzaro - Prog. strutt. 3

31

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

Stefano Mizzaro - Prog. strutt. 3

32

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 il Java (e altri linguaggi) hanno effettuato queste scelte?

Stefano Mizzaro - Prog. strutt. 3

33

... e 3 risposte

- Böhm e Jacopini (1966)
 1. Con l'iterazione e senza il salto si riesce 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. Salto = minore comprensibilità (spaghetti logic/programming)

Stefano Mizzaro - Prog. strutt. 3

34

Riassunto

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

Stefano Mizzaro - Prog. strutt. 3

35

Prossima lezione

- **break** e **continue**
- Esempi ed esercizi
- Array

Stefano Mizzaro - Prog. strutt. 3

36