

Programmazione strutturata IV e Gli array

Stefano Mizzaro

Dipartimento di matematica e informatica
Università di Udine
<http://www.dimi.uniud.it/mizzaro/>
mizzaro@uniud.it
Programmazione, lezione 7
16/03/2015

Comunicazioni

- No lezione il 30-31 marzo
- Lab.?
- Iscrizione al gruppo google con "password", comunicata a lezione

S. Mizzaro - Prog. strutt. 4 & Array

2

Dove siamo

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

S. Mizzaro - Prog. strutt. 4 & Array

3

Oggi

- Esempio
- Introduzione agli array

S. Mizzaro - Prog. strutt. 4 & Array

4

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

S. Mizzaro - Prog. strutt. 4 & Array

5

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...

S. Mizzaro - Prog. strutt. 4 & Array

6

Schema 1

S. Mizzaro - Prog. strutt. 4 & Array 7

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);
```

```
class ... {
    public ... {
        ...
    }
}
```

S. Mizzaro - Prog. strutt. 4 & Array

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* verificiamo se *i* è un divisore sia di *x* sia di *y*
- Il primo divisore che troviamo è il massimo
- ...schema...

S. Mizzaro - Prog. strutt. 4 & Array 9

Schema 2

S. Mizzaro - Prog. strutt. 4 & Array 10

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);
```

```
class ... {
    public ... {
        ...
    }
}
```

S. Mizzaro - Prog. strutt. 4 & Array


Confronto

- Qual è la migliore?
- La seconda versione 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*

S. Mizzaro - Prog. strutt. 4 & Array 12

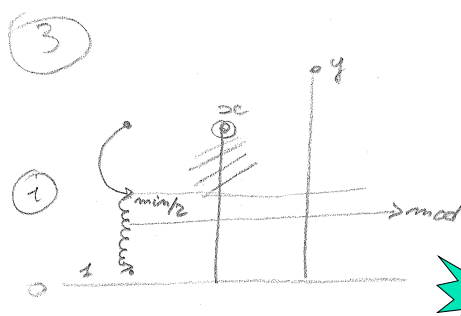

Terza idea

- 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...



S. Mizzaro - Prog. strutt. 4 & Array 13

Schema 3

S. Mizzaro - Prog. strutt. 4 & Array 14

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);
    
```


class ... {
 public ... {
 ...

- Anche qui il **break** "interrompe"
- **min** evita calcoli inutili
- (e se **min** è dispari?)

S. Mizzaro - Prog. strutt. 4 & Array 15

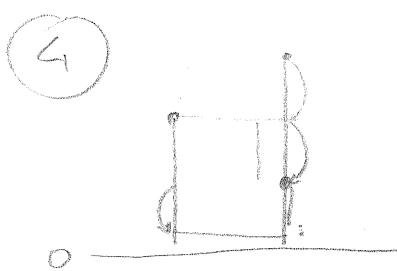

Quarta idea!

- 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...



S. Mizzaro - Prog. strutt. 4 & Array 16

Schema 4

S. Mizzaro - Prog. strutt. 4 & Array 17

Quarta versione

```

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

class ... {
 public ... {
 ...

- Più elegante
- Più sintetico
- Più efficiente
- Meglio di così cosa vogliamo?
 - ...

S. Mizzaro - Prog. strutt. 4 & Array 18

Quinta versione

```

while (x != 0 && y != 0)
  if (x > y)
    x = x % y;
  else
    y = y % x;
System.out.println(x==0?y:x);
    
```

```

class ... {
public ... {
...
    
```

- Fare più volte $x = x - y$ è equivalente a fare un'unica volta $x = x \% y$
- Quando l'ultimo resto è zero, un numero è multiplo dell'altro, e il risultato è nell'altra variabile
- Esempi di while \neq for...

S. Mizzaro - Prog. strutt. 4 & Array 19

Schema 5

S. Mizzaro - Prog. strutt. 4 & Array 20

Schemi 1-5

S. Mizzaro - Prog. strutt. 4 & Array 21

Morali

- Mai accontentarsi
 - Un programma è (quasi) sempre migliorabile
 - Soprattutto la prima versione funzionante
- Saper riflettere al livello dell'idea, non del codice
 - "tornare indietro"
- Usare conoscenze da altri campi

S. Mizzaro - Prog. strutt. 4 & Array 22

Gli array

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

	x
	12.34
	a
0	12.34
1	3.14
2	0.43
3	3E-12
4	0.98
5	0.01

S. Mizzaro - Prog. strutt. 4 & Array 23

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

S. Mizzaro - Prog. strutt. 4 & Array 24

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

S. Mizzaro - Prog. strutt. 4 & Array 25

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];`

S. Mizzaro - Prog. strutt. 4 & Array 26

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;

```

S. Mizzaro - Prog. strutt. 4 & Array 27

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...
- (N.B. Brutti i letterali...)

S. Mizzaro - Prog. strutt. 4 & Array 28

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[];

```

S. Mizzaro - Prog. strutt. 4 & Array 29

Allocazione

- Non c'era per le variabili, serve per allocare la memoria necessaria per gli elementi
- Si usano
 - `new`
 - [`lunghezza`] (numero di elementi)
- Dichiarazione e allocazione si possono fare insieme

```

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

```

```

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

```

S. Mizzaro - Prog. strutt. 4 & Array 30

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};
```

S. Mizzaro - Prog. strutt. 4 & Array

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];
}
```

S. Mizzaro - Prog. strutt. 4 & Array

32

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]);
```

S. Mizzaro - Prog. strutt. 4 & Array

33

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

S. Mizzaro - Prog. strutt. 4 & Array

34

Esercizi

- Azzerare un array di `int`, diciamo di nome `a` (ossia, assegnare zero a tutti gli elementi di `a`)

```
for (int i = 0; i < a.length; i++)
    a[i] = 0;
```

- Assegnare 0, 1, 2, ... agli elementi di `a`

```
for (int i = 0; i < a.length; i++)
    a[i] = i;
```

- Incrementare di 1 ogni elemento dell'array `a`

```
for (int i = 0; i < a.length; i++)
    a[i] = a[i] + 1; // o a[i]++, o ++a[i]
```

S. Mizzaro - Prog. strutt. 4 & Array

35

Esercizi

- Assegnare 1 agli elementi di `a` di indice dispari e 2 a quelli di indice pari

```
for (int i = 0; i < a.length; i++)
    if (i % 2 != 0)
        a[i] = 1;
    else
        a[i] = 2;
```

```
for (int i = 0; i < a.length; i++)
    a[i] = (i % 2 != 0) ? 1 : 2;
```

```
for (int i = 0; i < a.length; i++)
    a[i] = 2 - (i % 2);
```

S. Mizzaro - Prog. strutt. 4 & Array

36

Osservazioni

- Il primo elemento di un array ha indice ZERO
- L'ultimo elemento di un array `a` ha indice `a.length - 1` (MENO UNO!)
- Nei cicli for che "spazzolano" un array `a`,
 - `i < a.length` invece di `i <= a.length - 1`
 - è più comodo, ed è frequente...
 - ... ma ricordarsi gli "Errori di uno" e fare attenzione!

S. Mizzaro - Prog. strutt. 4 & Array

37

Il `final` e le dimensioni degli array

```
int[] a = new int[10];
```

```
final int N = 10;
int[] a = new int[N];
```

- Modifiche più semplici nel secondo caso

```
for (int i = 0; i < N; i++)
```

- (es.: da 10 a 1000...)

- Ma in Java c'è il `.length`...

```
for (int i = 0; i < a.length; i++)
  ...
```

S. Mizzaro - Prog. strutt. 4 & Array

38

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

S. Mizzaro - Prog. strutt. 4 & Array

39