Laboratorio di Informatica
Lezione 3

Cristian Consonni

14 ottobre 2015

Outline

Chi sono

Cristian Consonni

- **DISI - Dipartimento di Ingegneria e Scienza dell'Informazione**
- **Pagina web** del laboratorio:
  http://disi.unitn.it/~consonni/teaching
- **Email**: cristian.consonni@unitn.it
- **Ufficio**: Povo 2 - Open Space 9
  - Per domande: scrivetemi una mail
  - Ricevimento: su appuntamento via mail

Outline for section 1

Metodo della bisezione

Negli esercizi della scorsa volta abbiamo calcolato gli zeri di una **funzione**:

$$f(x) = x^3 - x - 2 \tag{1}$$

Funzioni (I)

Possiamo definire delle funzioni per rendere il codice:

- più **leggibile**
- più **riutilizzabile**
- più **compatto** e semplice da modificare

Funzioni (II)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

analizziamo la sintassi parola per parola.

Funzioni (III)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

Le parole chiave public e static modificano la visibilità della funzione. Per ora ignoratele e scrivete sempre così.

Funzioni (IV)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

Questo è il **tipo del valore di ritorno** (*return type*) della funzione, vuol dire, in questo caso, che la funzione restituisce un double.

Funzioni (V)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

Dopo il tipo di ritorno si specifica il **nome** della funzione, come per le variabili si tratta dell'identificativo da utilizzare per utilizzare una funziona.

Funzioni (VI)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

Le funzioni sono caratterizzate dalle parentesi tonde `f()` che servono per potere *chiamare* una funzioone.

Funzioni (VII)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

All'interno delle parentesi tonde si indicano gli argometi della funzione detti **parametri formali**.

Funzioni (VIII)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

Ogni *parametro formale* è caratterizzato da un proprio **tipo**...

Funzioni (IX)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

e da un proprio **nome**.

Funzioni (X)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x1, int x2) {
    return(x1 * x1 * x1 - x2 - 2);
}
```

Una funzione può avere più parametri formali ciascuno caratterizzato dal proprio tipo e nome.

Funzioni (XI)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

Le parentesi graffe {} delimitano il **corpo** (o *scope*) della funzione.

Funzioni (XII)

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

La parola riservata return indica quale valore sarà restituito dalla funzione.

Funzioni: riassumendo

Dichiarazione e implementazione di una funzione:

```
public static double f(double x) {
    return(x * x * x - x - 2);
}
```

- `public static` → visibilità
- `double` → tipo del valore di ritorno
- `f` → nome della funzione
- `double x` → parametri formali
- `return` → valore di ritorno

Procedure

Le procedure sono funzioni che non restituiscono alcun valore. Per indicare questo fatto si usa come valore di ritorno **void**:

```
public static void f(double x) {
  println("Questa è una procedura");
}
```

(Nelle slide userò printf e println invece di System.out.printf o System.out.println).

- `public static` → visibilità
- `void` → nessun valore ritornarto, è una **procedura**
- `f` → nome della procedura
- `double x` → parametri formali
- `return` → valore di ritorno

Funzioni e procedure: esempio (I)

```
public class FunzioniProcedure {

  // Questa è una funzione di due parametri x e y
  // che restituisce un intero
  public static int somma(int x, int y) {
      return x + y;
  }

  ...
```

Funzioni e procedure: esempio (II)

```
...

// Questa è una procedura di due parametri x e y
public static void stampaSomma(int x, int y) {

  printf("== Somma di due numeri ==%n");
  printf("Il primo parametro (x) vale %d.%n", x);
  printf("Il secondo parametro (y) vale %d.%n", y);
  printf("La somma di %d e %d è %d.%n", x, y, x + y);

  // La riga seguente può essere omessa,
  // io preferisco indicarla per chiarezza
  return;
}

...
```

Funzioni e procedure: esempio (III)

```
...

// Questo è il main, come al solito
public static void main(String[] args) {

  int a = 7;
  int b = 11;
  int risultato;

  // Chiamata della funzione
  risultato = somma(a, b);
  printf("risultato: %d.%n", risultato);

  // Chiamata della procedura
  stampaSomma(a, b);
 }
}
```

Parametri attuali

Nella slide precedente abbiamo visto

```
public static int somma(int x, int y) {
...
public static void stampaSomma(int x, int y) {
```

definizioni della funzione `somma` e della procedura `stampaSomma`.
Entrambe le definizioni hanno come **parametri formali** le variabli `x` e `y`.

Le chiamate:

```
risultato = somma(x, y);
...
stampaSomma(x, y);
```

hanno come **parametri attuali** le variabli `a` e `b`.

Parametri passati per valore (I)

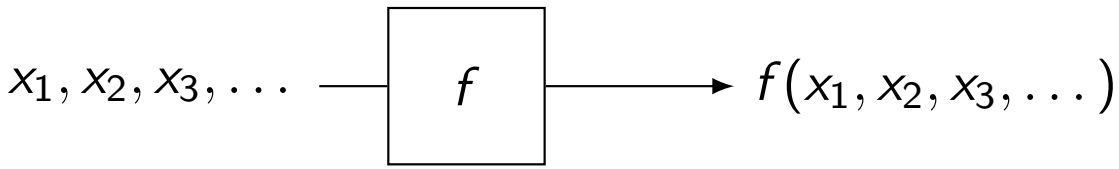
```
public class StampaParola {

  // di solito in Java si usa il CamelCase
  public static void stampaParolaInVerticale(String parola) {

    // String.toUpperCase() rende una stringa tutta
    // maiuscola
    parola = parola.toUpperCase();

    // String.length restituisce la lunghezza della parola
    for(int i = 0; i < parola.length(); i++) {

    // parola.charAt(i) restituisce l'i-esimo carattere
    // della parola si parte a contare da zero.
    println(parola.charAt(i));
    }
  }
...
```

Parametri passati per valore (II)

```
...

 public static void main(String[] args) {

   String parola = "Ciao";

   printf("La parola da stampare è: %s.%n", parola);
   stampaParolaInVerticale(parola);
   printf("La parola stampata è: %s.%n", parola);

 }
}
```

Funzioni: diagramma

Possiamo immaginare una funzione come una "scatola nera" che rende in ingresso i parametri e restituisce un valore:



`return_type f`

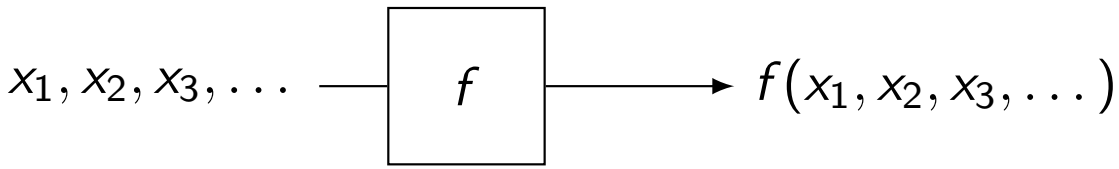
`type1` x_1`, type2` x_2`, type3` x_3`, ...`

Attenzione

In un linguaggio imperativo è possibile che una funzioni modifichi il valore di alcune variabili in ingresso!

Funzioni: diagramma

Possiamo immaginare una funzione come una "scatola nera" che rende in ingresso i parametri e restituisce un valore:



`return_type f`

`type1` x_1`, type2` x_2`, type3` x_3`, ...`

Attenzione

In un linguaggio imperativo è possibile che una funzioni modifichi il valore di alcune variabili in ingresso!

Outline for section 2

Iterazione e ricorsione (I)

Una distizione importante per quanto riguarda le funzione è la seguente:

- funzioni **iterative**
- funzioni **ricorsive**

Una funzione **ricorsiva** è una funzione che **richiama se stessa**.
Altrimenti è **iterativa**.
Le funzioni viste finora sono iterative.

Iterazione e ricorsione (II)

Le funzioni ricorsive si trovano molto comunemente anche in matematica. Esempi:

- fattoriale: $n! = n \cdot n-1 \cdot \dots \cdot 3 \cdot 2 \cdot 1 := n \cdot (n-1)!$
- successione di Fibonacci: $F_n = F_{n-1} + F_{n-2}$

ovviamente la definzione deve avere alcuni **casi limite** altrimenti si andrebbe avanti all'infinito.

Per i casi di cui sopra:

- fattoriale: $n = 0 \Rightarrow 0! = 1$ (il fattoriale non è definito per numeri interi negativi)
- successione di Fibonacci: $F_0 = 0$ e $F_1 = 1$[a]

[a] Tradizionalmente si definiva $F_0 = 1$ e $F_1 = 1$, ma ora si definisce di solito come sopra. Si veda anche `https://oeis.org/A000045`

Definizione ricorsiva del fattoriale

```
public class FattorialeRicorsivo {

  public static int fattoriale(int n) {
    if (n == 0) {
      return 1;
    }
    return n * fattoriale(n - 1);
  }

  public static void main(String[] args) {
    int numero = 10;
    int risultato = 0;

    risultato = fattoriale(numero);
    printf("Il fattoriale di %d è: %d", numero, risultato);
  }
}
```

Outline for section 3

Variabili globali e locali

All'interno di una funzione possono essere definite nuove *variabili*. Queste variabili sono **locali** ovvero esistono e sono utilizzatibili solo all'interno della funzione stessa.

```
...
public static int somma(int x, int y) {
    int ris;
    ris = x + y;
    return ris;
}

public static void main(String[] args) {
  int a = 7;
  int b = 11;
  int risultato;

  // Qui *non posso* usare ris!
  ...
```

Variabili globali e locali

Le parentesi graffe {} definiscono uno **scope** (lett. *ambito*) ovvero un blocco di codice.

```
public static void main(String[] args) {
  int fatt  = 1;
  int N = 10;

  for (int n = 2; n < N; n++) {
    // La variabile n esiste solo all'interno di questo
    // blocco di codice
    ...
  }
```

Variabili globali e locali

Come avete già visto potete usare variabili che avete definito in blocco più esterno dentro blocchi più interni.

```
public static void main(String[] args) {
  int fatt  = 1;
  int N = 10;

  for (int n = 2; n < N; n++) {
    // fatt è definita nel blocco più esterno, quindi
    // la posso usare anche qui
    fatt = fatt * n;
  }
```

Variabili globali e locali

Posso definere delle variabili che possano essere usate in ogni scope, queste si dicono variabili **globali**.

```
public class GlobaliLocali {

  int numero = 0;

...
```

queste variabili possono essere modificate da ogni funzione e quindi vanno usate con attenzione. È meglio limitare il più possibile lo scope di una variabile.
Cercate di dare nomi significativi alle variabili globali altrimenti sembra che "piovano dal cielo". Solitamente le variabili globali (specie se rappresentano una costante) si scrivono con nomi tutti in `MAIUSCOLO`.

Outline for section 4

Esercizi (parte I)

1. Scrivere una funzione testa se un numero intero è primo (restituendo un booleano);
2. Scrivere una procedura che stampi una parola (o una frase) al contrario. Ad esempio:
   - "gatto" $\rightarrow$ ottag;
   - "anna" $\rightarrow$ anna;
3. scrivere una funzione che calcoli la somma dei numeri interi da 1 a N;
4. riscrivere la funzione precedente in modo ricorsivo;

Outline for section 5

Array (I)

0	1	2	3	4	5	6	7	8	9
a"	"b"	"c"	"d"	"e"	"f"	"g"	"h"	"i"	"j"

Un **array** (o **vettore**) è una *struttura dati* composto da un **numero finito** di elementi tutti dello **stesso tipo**.

Array (I)

Simili al concetto matematico di vettore:

$$v = \begin{pmatrix} 1 \\ 0 \\ 4 \\ 7 \end{pmatrix}$$

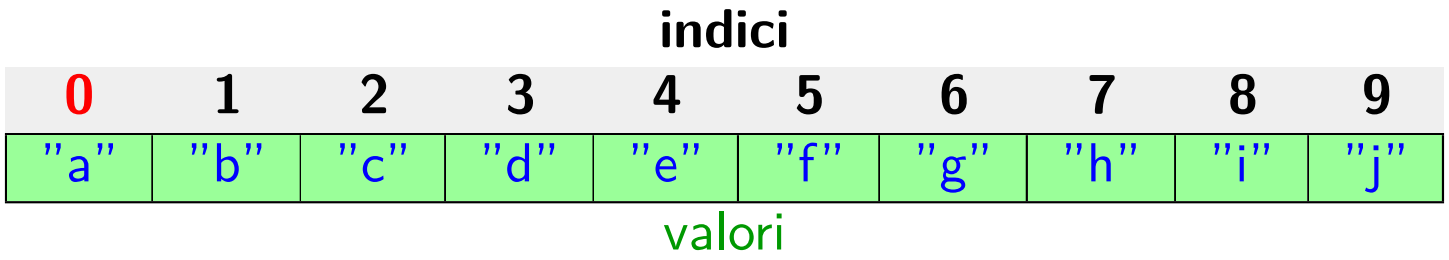
Array (II)

1. Dichiarazione di un array:
   - `tipo nome[]`
   - `tipo[] nome`
2. Allocazione:
   - `nome = new tipo[dimensione]`
3. Diachiarazione e inizializzazione con literals:
   - `tipo[] nome ={val1, val2, ...}`

Array (III)

```
public class Array {

  public static void main(String[] args) {

    int[] vettore;
    vettore = new int[10];

  ...
```

Array (IV)



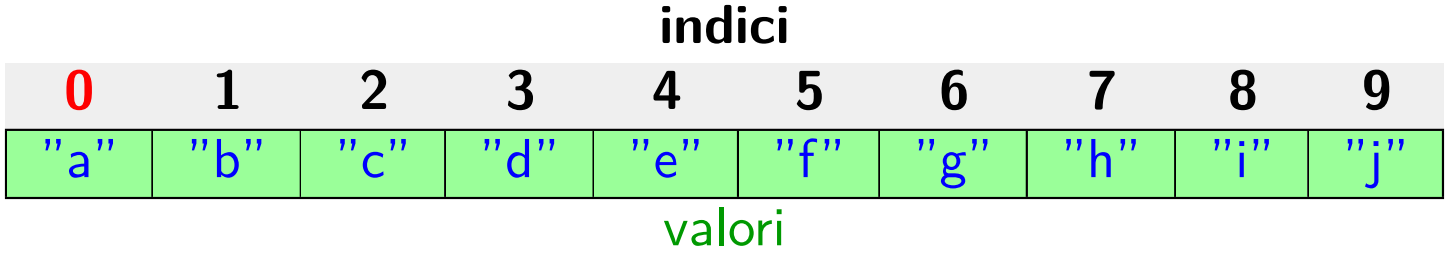
Per accedere all'elemento con indice `i` dell'array si usa la sintassi:

```
nome[i]
```

Notate che:

- gli indici partono da **zero** e arrivono fino alla **lunghezza dell'array meno uno**
- il primo elemento è accessibile con `nome[0]`
- l'ultimo elemento è accessibile con `nome[len - 1]` (dove `len` è la lunghezza dell'array.

Array (V)



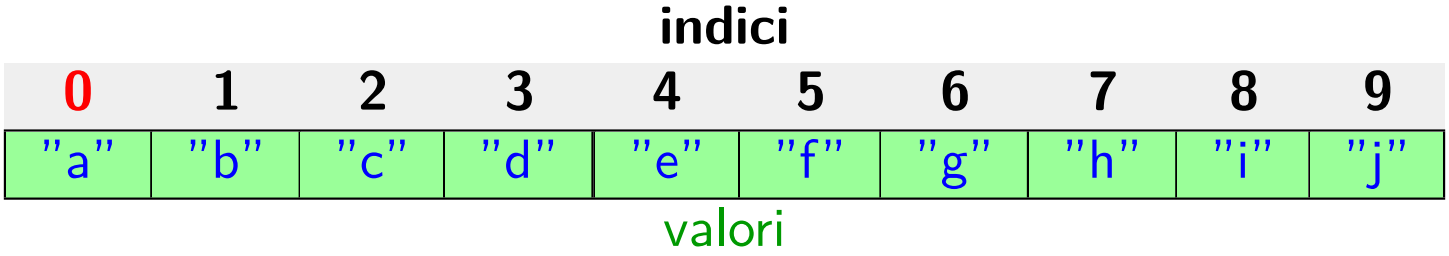
Per accedere all'elemento con indice `i` dell'array si usa la sintassi:

```
valore = nome[i]
```

Esempi:

- `nome[0]` → "a"
- `nome[4]` → "e"
- `nome[10]` → **Errore: out of range**

Array (VI)



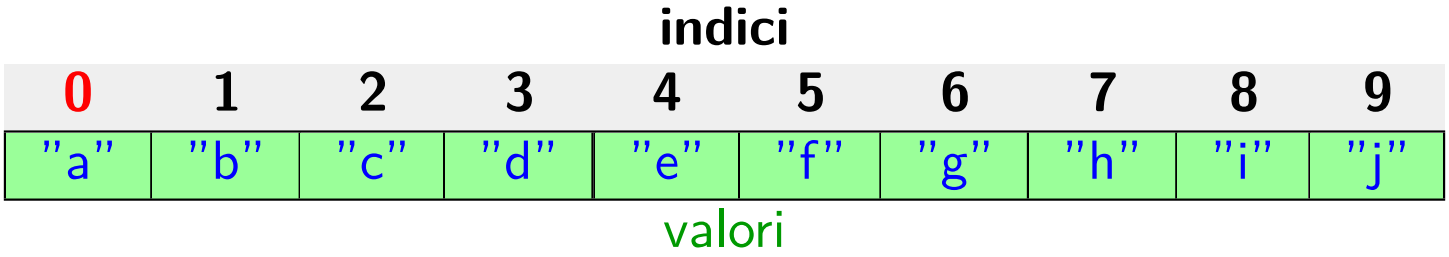
Per impostare il valore dell'elemento con indice `i` dell'array si usa la sintassi:

```
nome[i] = valore
```

Esempi:

- `nome[0] = "z"`
- `nome[4] = "x"`
- `nome[10]` → **Errore: out of range**

Array (VI)



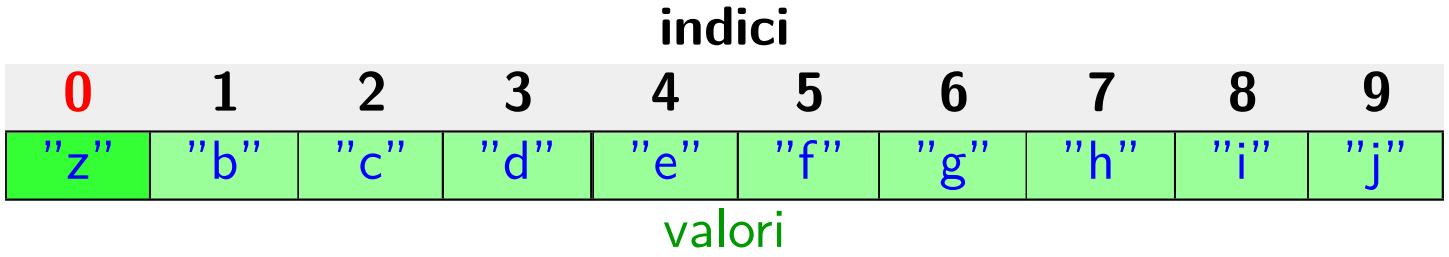
Per impostare il valore dell'elemento con indice `i` dell'array si usa la sintassi:

```
nome[i] = valore
```

Esempi:

- `nome[0] = "z"`
- `nome[4] = "x"`
- `nome[10]` → **Errore: out of range**

Array (VI)



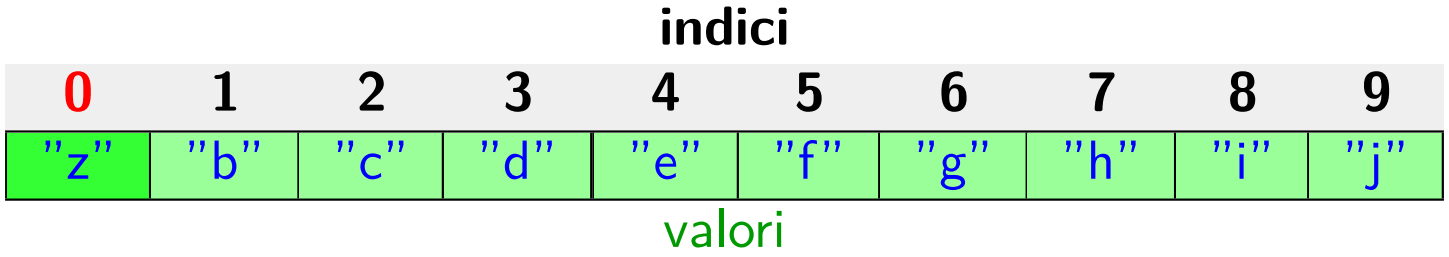
Per impostare il valore dell'elemento con indice `i` dell'array si usa la sintassi:

```
nome[i] = valore
```

Esempi:

- `nome[0] = "z"`
- `nome[4] = "x"`
- `nome[10]` → **Errore: out of range**

Array (VI)



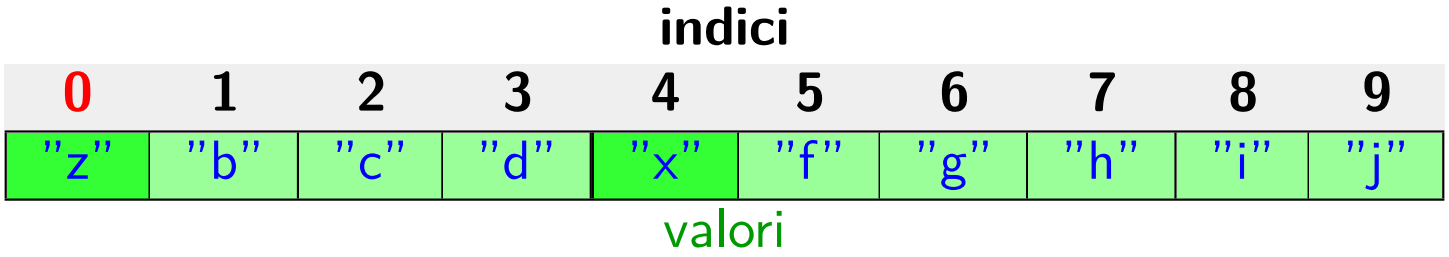
Per impostare il valore dell'elemento con indice `i` dell'array si usa la sintassi:

```
nome[i] = valore
```

Esempi:

- `nome[0] = "z"`
- `nome[4] = "x"`
- `nome[10]` → **Errore: out of range**

Array (VI)



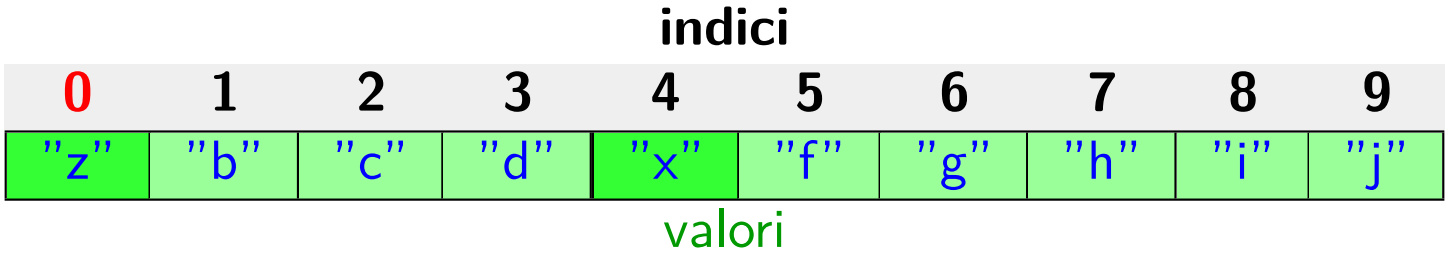
Per impostare il valore dell'elemento con indice `i` dell'array si usa la sintassi:

```
nome[i] = valore
```

Esempi:

- `nome[0] = "z"`
- `nome[4] = "x"`
- `nome[10]` → **Errore: out of range**

Array (VI)



Per impostare il valore dell'elemento con indice `i` dell'array si usa la sintassi:

```
nome[i] = valore
```

Esempi:

- `nome[0] = "z"`
- `nome[4] = "x"`
- `nome[10]` → **Errore: out of range**

Array e funzioni

Gli array possono essere usati come argomenti delle funzioni

```
public class ArrayFun {

  public static void somma(int[] v) {

    int s = 0,
    for(int i = 0; i < v.length; i++)  {

    }
  ...
```

Attenzione

A differenza delle altre variabili, se un array viene modificato in una funzione resta modificato per tutto il programma

Outline for section 6

Matrici (I)

Analogamente ai vettori, possiamo introdurre le matrici:

$$A = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 4 & 5 \\ 3 & 7 & 0 \end{pmatrix}$$

Matrici (II)

In Java sono vettori di vettori:

```
public class Matrix {

  public static void main(String[] args) {

    double [ ] [ ] mat = new int[ 5 ] [ 4 ] ;
...
```

Outline for section 7

Esercizi (I)

- inizializzare un array di 10 interi e inserire in ciascun elemento il valore 5 (utilizzare un ciclo for)
- inizializzare un array di interi e assegnare a ciascun elemento un valore uguale alla sua posizione
- Dopo aver inizializzato l'array precedente, copiare il suo contenuto in ordine nverso in un altro array della stessa dimensione.
  - $(1, 2, 3) \rightarrow (3, 2, 1)$

Suggerimento, con `arr.length` si può ottenere la lunghezza dell'array:

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

for (int i = 0; i < arr.length; i++) {
   ...
}
```

Esercizi (I)

- Implementare una funzione che dato in ingresso un vettore di numeri interi restituisca quel vettore ordinato in ordine crescente.

Esistono tantissimi algoritmi di ordinamento, ecco l'idea di base per uno di essi:

- array v di lunghezza N, si fa scorrere l'indice i da 1 a $N-1$ ripetendo i seguenti passi:
  1. si cerca il più piccolo elemento della sottosequenza $v[1..N]$
  2. si scambia questo elemento con l'elemento i-esimo

Un'animazione descrittiva è disponibile a questo indirizzo:

`https://upload.wikimedia.org/wikipedia/commons/9/94/Selection-Sort-Animation.gif`

Esercizi (II)

- Scrivere una funzione che calcoli il prodotto scalare (`double`) di due vettori (di `double`) :

$$\langle v, w \rangle = \sum_{i=1}^{N} v_i \cdot w_i \tag{2}$$

Esercizi (III)

- Scrivere un programma che data in ingresso una matrice di numeri casuali ne calcoli la matrice trasposta.
  - Scrivere una funzione che stampi una matrice
  - Scrivere una funzione che calcoli la matrice trasposta data una matrice

Data la matrice A, la matrice trasposta - indicata con A^T - si ottiene scambiando le righe con le colonne:

$$A = \begin{pmatrix} 1 & 9 & 3 \\ 0 & 4 & 5 \\ 2 & 7 & 0 \end{pmatrix} \qquad A^T = \begin{pmatrix} 1 & 0 & 2 \\ 9 & 4 & 7 \\ 3 & 5 & 0 \end{pmatrix}$$

Esercizi (IV)

- Scrivere un programma che dati in ingresso i tre coefficienti a, b e c risolva le equazioni di secondo grado usando la formula.
  - il programma deve gestire i casi in cui non esistano soluzioni reali (discriminante $\Delta = (b^2 - 4ac)$ negativo),

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Esercizio: trovare tutti i numeri primi fino a N (I)

- Nell'esercizio sulla primalità abbiamo scritto una funzione che ci permette di testare se un dato numero n è primo.
- Ammettiamo di voler scrivere un programma che ci dica tutti i numeri primi fino a 100.

Il crivello di Eratostene (I)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (II)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (III)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (IV)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (V)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (VI)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (VII)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (VI)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (VII)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (VIII)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (IX)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (X)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (XI)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (XII)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (XIII)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (XIV)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (XV)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (XVI)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Il crivello di Eratostene (XVII)

Il **crivello di Eratostene** è un algoritmo per trovare tutti i numeri primi da 1 a un intero massimo N.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Esercizio: trovare tutti i numeri primi fino a N (II)

- Scrivere un programma che riempa un vettore di numeri da 1 a 100 e usando l'algoritmo del crivello di Eratostene crei alla fine un altro vettore contenente i numeri primi fino a 100.