



## Problemi, algoritmi e oggetti *prima parte*

Capitolo 5 (a)  
ottobre 2011

### Contenuti

- ◆ Problemi e algoritmi
  - comprensione del problema
  - identificazione di un algoritmo per il problema
  - dall'algoritmo all'oggetto
  - qualità degli algoritmi
- ◆ Introduzione agli algoritmi (per esempi)
  - un linguaggio per scrivere algoritmi
  - progettazione di algoritmi
  - problemi di ingresso-uscita
  - lettura e somma di due numeri interi
  - somma di una sequenza di numeri interi
  - lunghezza di una sequenza
  - somma dei pari e dei dispari in una sequenza
  - somma di una sequenza di dieci numeri
  - somma dei primi N numeri interi positivi
  - traccia dell'esecuzione di un metodo o di un algoritmo
  - la formula di Gauss
- ◆ Attraversamento di labirinti
  - attraversamento di labirinti lineari
  - attraversamento di labirinti

## Risoluzione di problemi

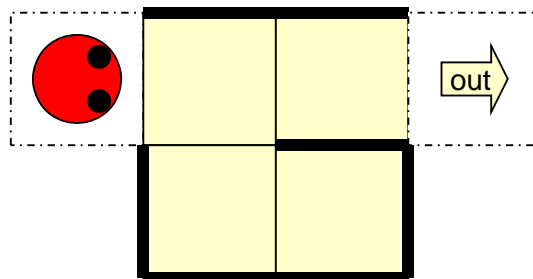
- un problema è un compito che si vuole far risolvere automaticamente a un calcolatore
  - i problemi sono di solito parametrici
- che vuol dire risolvere un problema?
  - comprendere il problema
  - definire un algoritmo (un procedimento risolutivo) per il problema
  - implementare l'algoritmo

Questo capitolo introduce, anche mediante degli esempi, le nozioni di problema e algoritmo

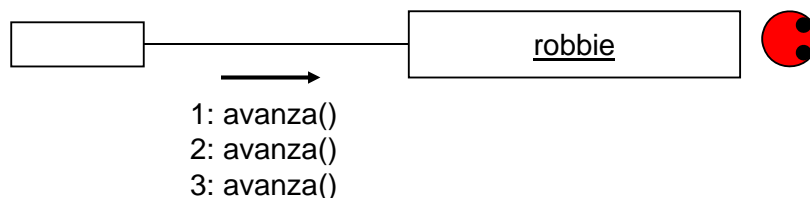
## Problemi e algoritmi

Il problema dell'attraversamento di un labirinto semplice

- si vuole far attraversare al robot **robbie** un labirinto semplice
  - inizialmente **robbie** si trova all'ingresso del labirinto



- una possibile soluzione

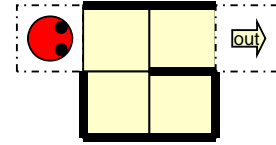


## Comprensione del problema

I problemi vengono descritti (in modo formale) dalla loro **specificità**

Problema

- attraversamento di un labirinto semplice



Insieme di ingresso

- il robot **robbie**

Pre-condizione

- il robot **robbie** si trova all'ingresso di un labirinto semplice, nella direzione di ingresso

Insieme di uscita

- vuoto

Post-condizione

- il robot **robbie** si trova all'uscita del labirinto semplice

## Specificità di un problema

La **specificità** di un **problema**

- **insieme di ingresso** o **parametri** – i parametri del problema
  - un insieme di **dati di ingresso** e un insieme di **oggetti di ingresso**
- **pre-condizione** – condizione relativa all'insieme di ingresso
  - proprietà soddisfatte dai dati di ingresso e dagli oggetti di ingresso (considerati nel loro stato iniziale)
- **insieme di uscita** o **risultati** – le informazioni che devono essere calcolate nella risoluzione del problema
  - **dati di uscita** e **oggetti di uscita**
- **post-condizione** – condizione relativa all'insieme di uscita
  - proprietà soddisfatte dai dati di uscita e dallo stato finale degli oggetti coinvolti dal problema, anche con riferimento ai dati di ingresso e allo stato iniziale degli oggetti di ingresso

## Esempio: equazione di secondo grado

### Problema

- calcolare le radici di un'equazione di secondo grado in forma normale,  $A \cdot X^2 + B \cdot X + C = 0$

### Insieme di ingresso

- i coefficienti reali A, B e C dell'equazione di secondo grado

### Pre-condizione

- nessuna

### Insieme di uscita

- la radici X1 e X2

### Post-condizione

- $A \cdot X1^2 + B \cdot X1 + C$  vale 0,  $A \cdot X2^2 + B \cdot X2 + C$  vale 0

## Identificazione di un algoritmo per il problema

### Un **algoritmo** per un problema

- una sequenza di istruzioni che permette di far evolvere gli oggetti di interesse da uno stato iniziale che soddisfa la pre-condizione a uno stato finale che soddisfa la post-condizione
  - sulla base di ulteriori dati di ingresso
  - calcolando eventuali dati in uscita

## Algoritmi ed esecutori automatici

Gli algoritmi vanno espressi in termini delle istruzioni di un **esecutore automatico** (ad es., un calcolatore, un oggetto)

- ciascuna istruzione deve poter essere eseguita dall'esecutore in tempo finito
- l'intera sequenza di istruzioni deve poter sempre essere eseguita in tempo finito
  - sempre = per ogni possibile insieme di ingresso che soddisfa la pre-condizione del problema

## Identificazione di un algoritmo

Un procedimento risolutivo per il problema dell'attraversamento di un labirinto semplice

1. fai avanzare **robbie** tre volte

Può essere considerato un algoritmo?

- è un procedimento corretto
- ma (ad essere pignoli) non è sufficientemente dettagliato
  - non è chiaro come far *avanzare tre volte* il robot
  - il robot sa solo *avanzare*

## Identificazione di un algoritmo

Il procedimento risolutivo descritto in modo più dettagliato

1. *fai avanzare robbie tre volte*
  - 1.1 *fai avanzare robbie*
  - 1.2 *fai avanzare robbie*
  - 1.3 *fai avanzare robbie*

Questo procedimento può essere considerato un algoritmo?

- è un procedimento corretto
- è sufficientemente dettagliato?
  - sì – per un esecutore umano
  - no – per un calcolatore (o per un oggetto)

## Identificazione di un algoritmo

Il procedimento risolutivo descritto in modo ancora più dettagliato

1. *fai avanzare robbie tre volte*
    - 1.1 *fai avanzare robbie*  
**robbie.avanza();**
    - 1.2 *fai avanzare robbie*  
**robbie.avanza();**
    - 1.3 *fai avanzare robbie*  
**robbie.avanza();**
- questo è sicuramente un algoritmo

## Algoritmi

Un algoritmo è una sequenza di “passi”, in cui ciascun “passo”

- è un’istruzione di un linguaggio di programmazione
- oppure, può essere facilmente codificato mediante un linguaggio di programmazione

Sono da considerarsi dunque algoritmi entrambi i seguenti procedimenti

1. *fai avanzare **robbie** tre volte*
  - 1.1 *fai avanzare **robbie***  
**robbie.avanza();**
  - 1.2 *fai avanzare **robbie***  
**robbie.avanza();**
  - 1.3 *fai avanzare **robbie***  
**robbie.avanza();**
1. *fai avanzare **robbie** tre volte*
  - 1.1 *fai avanzare **robbie***  
**robbie.avanza();**
  - 1.2 *fai avanzare **robbie***  
**robbie.avanza();**
  - 1.3 *fai avanzare **robbie***  
**robbie.avanza();**

## Esempio: equazione di secondo grado

Algoritmo per calcolare le radici di un’equazione di secondo grado in forma normale

1. *calcola il discriminante DELTA dell’equazione*  
 $DELTA = B^2 - 4AC$
2. *calcola le radici X1 e X2 dell’equazione*  
 $X1 = (-B + \text{SQRT}(DELTA)) / 2A$   
 $X2 = (-B - \text{SQRT}(DELTA)) / 2A$

## Dall'algoritmo all'oggetto

Un algoritmo può essere codificato come metodo di un oggetto

```
import fiji.robot.*;
/* Oggetto per l'attraversamento di labirinti semplici. */
public class AttraversatoreLabirintoSemplice {
    /* Fa attraversare a robbie un labirinto semplice. */
    public static void attraversaLabirintoSemplice(
        Robot robbie) {
        // pre: robbie all'ingresso di un labirinto semplice

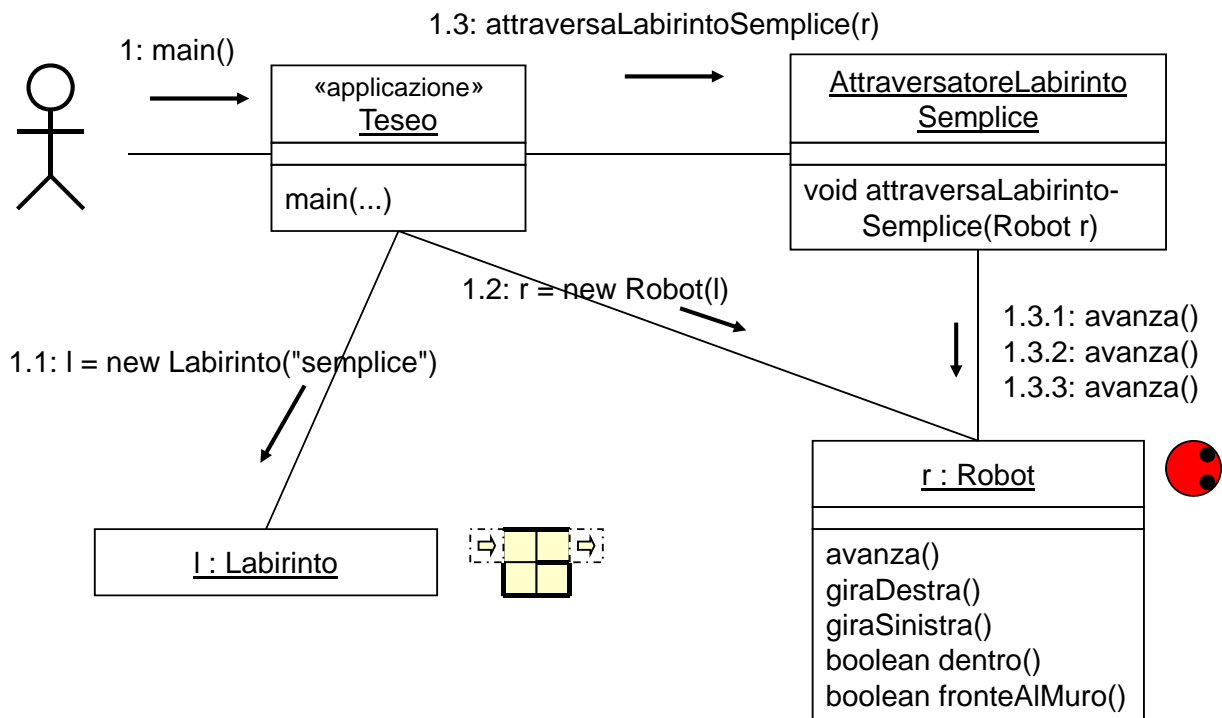
        /* 1. fai avanzare robbie tre volte */
        /* 1.1 fai avanzare robbie */
        robbie.avanza();
        /* 1.2 fai avanzare robbie */
        robbie.avanza();
        /* 1.3 fai avanzare robbie */
        robbie.avanza();
    }
}
```

## Uso dell'oggetto

```
import fiji.robot.*;
/* Crea un robot e gli fa attraversare un labirinto
 * semplice. */
public class Teseo {
    public static void main(String[] args) {
        Labirinto l; // un labirinto semplice
        Robot r; // robot nel labirinto l

        /* crea il robot in un labirinto semplice */
        l = new Labirinto("semplice");
        r = new Robot(l);
        /* fa attraversare il labirinto l a r */
        AttraversatoreLabirintoSemplice.
            attraversaLabirintoSemplice(r);
        /* ora r è all'uscita di l */
    }
}
```

## Uso dell'oggetto



17

Problemi, algoritmi e oggetti

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Qualità degli algoritmi

Due qualità fondamentali di un algoritmo

- **correttezza**
  - l'algoritmo permette effettivamente di risolvere il problema
- **efficienza**
  - l'esecuzione dell'algoritmo richiede un uso limitato di risorse

Altre qualità degli algoritmi

- leggibilità
  - essere facilmente comprensibile
- modificabilità
- parametricità
- riusabilità
- essere consegnato nei tempi previsti
- ...

18

Problemi, algoritmi e oggetti

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Correttezza

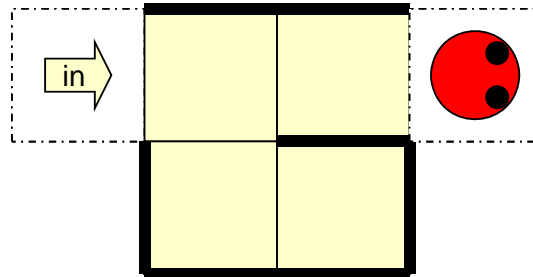
Il procedimento illustrato in precedenza risolve il problema dell'attraversamento di un labirinto semplice

1. *fai avanzare **robbie** tre volte*

1.1 *fai avanzare **robbie***

1.2 *fai avanzare **robbie***

1.3 *fai avanzare **robbie***



Diciamo allora che è un procedimento **corretto**

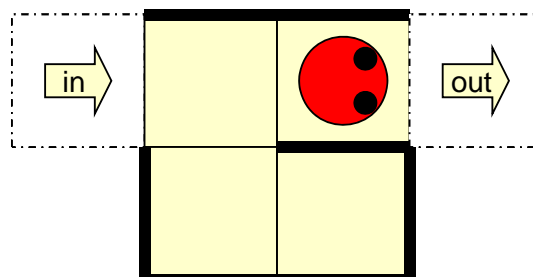
## Correttezza

Il seguente procedimento è invece **non corretto**

1. *fai avanzare **robbie** due volte*

1.1 *fai avanzare **robbie***

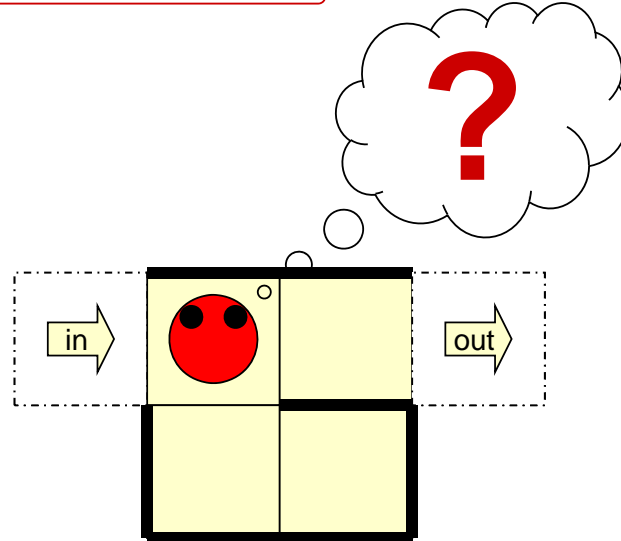
1.2 *fai avanzare **robbie***



## Correttezza

Un altro esempio di procedimento non corretto

1. fai avanzare **robbie**
2. fai girare a sinistra **robbie**
3. fai avanzare **robbie**

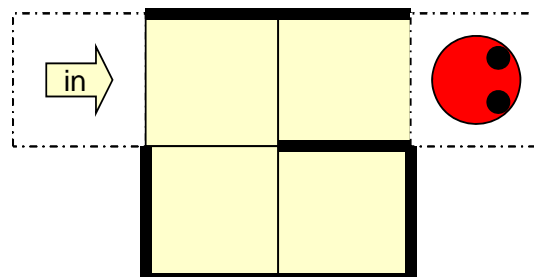


## Efficienza

Un algoritmo è tanto più efficiente quanto meno risorse richiede per la sua esecuzione

- una risorsa importante è il tempo di esecuzione

1. *fai avanzare **robbie** due volte*  
`robbie.avanza();`  
`robbie.avanza();`
2. *fai fare a **robbie** un giro su se stesso*  
`robbie.giraSinistra();`  
`robbie.giraSinistra();`  
`robbie.giraSinistra();`  
`robbie.giraSinistra();`
3. *fai avanzare **robbie***  
`robbie.avanza();`



## Introduzione agli algoritmi (per esempi)

Le nozioni di problema e di algoritmo vengono ora approfondite mediante lo studio di alcuni esempi, che illustrano gli aspetti principali nella risoluzione di problemi

- un algoritmo è una sequenza di istruzioni che, se eseguite ordinatamente, permettono di risolvere un certo problema
- gli algoritmi vengono scritti mediante un *linguaggio per esprimere algoritmi* – basato sull'uso di un numero limitato di tipologie di istruzioni
- la *progettazione* degli algoritmi avviene solitamente *per raffinamenti successivi*
- la *verifica delle qualità* di un algoritmo (in particolare, la correttezza e l'efficienza) richiede strumenti adeguati – saranno studiati solo in capitoli successivi

## Un linguaggio per scrivere algoritmi

Gli algoritmi vengono solitamente espressi mediante linguaggi intermedi tra linguaggi di programmazione e linguaggio naturale

- a metà strada tra un metodo e una descrizione testuale

In genere, questi linguaggi si basano sulle seguenti tipologie di **istruzioni**

- istruzioni semplici
  - invio di un messaggio a un oggetto
  - calcolo di un'espressione e assegnazione
- istruzioni di controllo (istruzioni strutturate)
  - sequenza
  - istruzione condizionale
  - istruzione ripetitiva

## Sequenza

Una **sequenza** (o **istruzione composta** o **blocco**) è un gruppo di istruzioni che devono essere eseguite in sequenza

- in sequenza – una alla volta, una dopo l'altra

Sintassi

```
{  
    istruzione-1  
    istruzione-2  
    ...  
    istruzione-n  
}
```

Semantica (significato)

- esegui prima l'**istruzione-1**, poi l'**istruzione-2**, ..., poi l'**istruzione-n**

## Istruzione condizionale

L'**istruzione condizionale** (**istruzione if-else**) consente di eseguire un'istruzione scelta tra due (**istruzione-1** e **istruzione-2**) condizionatamente al verificarsi (o meno) di una condizione **condizione**

Sintassi

```
if (condizione)  
    istruzione-1  
else  
    istruzione-2           oppure  
if (condizione) {  
    sequenza-istruzioni-1  
} else {  
    sequenza-istruzioni-2  
}
```

Semantica

1. valuta la condizione **condizione**
2. se la condizione **condizione** si è verificata allora esegui l'istruzione **istruzione-1**, altrimenti esegui l'istruzione **istruzione-2**

## Istruzione condizionale

Una variante, l'**istruzione if** – consente di eseguire un'istruzione (**istruzione**) oppure di non fare niente, condizionatamente al verificarsi (o meno) di una condizione **condizione**

Sintassi

```
if (condizione)  
    istruzione           oppure           if (condizione) {  
                                sequenza-istruzioni  
                                }
```

Semantica

1. valuta la condizione **condizione**
2. se la condizione **condizione** si è verificata allora esegui l'istruzione **istruzione**, altrimenti non fare nient'altro

## Istruzione condizionale – esempio

Calcola il maggiore tra due numeri interi **A** e **B**

```
if ( A è maggiore di B )  
    maggiore = A;  
else  
    maggiore = B;  
il maggiore tra A e B vale maggiore
```

## Istruzione condizionale – esempio

Il triangolo di lati **A**, **B** e **C** è equilatero, isoscele o scaleno?

```
latiUguali = 0
if ( A è uguale a B )
    latiUguali = latiUguali + 1;
if ( A è uguale a C )
    latiUguali = latiUguali + 1;
if ( B è uguale a C )
    latiUguali = latiUguali + 1;
if ( latiUguali è uguale a 3 )
    tipoTriangolo = "equilatero"
else if ( latiUguali è uguale a 1 )
    tipoTriangolo = "isoscele"
else
    tipoTriangolo = "scaleno"
il triangolo è di tipo tipoTriangolo
```

29

Problemi, algoritmi e oggetti

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Istruzione ripetitiva

L'**istruzione ripetitiva** (**istruzione while**) consente di eseguire ripetutamente un'istruzione **istruzione** fintanto che la condizione **condizione** risulta verificata

Sintassi

```
while ( condizione )
    istruzione
oppure
while ( condizione ) {
    sequenza-istruzioni
}
```

- l'istruzione **istruzione** è il **corpo** dell'istruzione ripetitiva

Semantica

1. valuta la condizione **condizione**
2. se la condizione **condizione** si è verificata allora esegui l'istruzione **istruzione** e torna a eseguire questi due passi
  - se invece la condizione **condizione** non si è verificata, allora considerata terminata l'esecuzione dell'istruzione ripetitiva

30

Problemi, algoritmi e oggetti

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Istruzione ripetitiva – esempio

Dati due numeri naturali **A** e **B**, calcola **A** elevato alla **B**

```
potenza = 1;  
moltiplicazioni = 0;  
while ( moltiplicazioni è minore di B ) {  
    potenza = potenza * A;  
    moltiplicazioni = moltiplicazioni + 1;  
}  
A elevato alla B vale potenza
```

## Istruzioni di controllo – esempio

Dati due numeri interi positivi **A** e **B**, calcola il massimo comun divisore tra **A** e **B**

```
M = A;  
N = B;  
while ( M è diverso da N ) {  
    if ( M è maggiore di N )  
        M = M - N;  
    else  
        N = N - M;  
}  
M è il massimo comun divisore tra A e B
```

## Progettazione di algoritmi

Scrivere un algoritmo è un'attività complessa

Una metodologia efficace è la strategia top-down (dall'alto verso il basso)

- la strategia **top-down** consiste nel decomporre iterativamente un problema da risolvere in sotto-problemi, fino a quando ciascun sotto-problema non può essere risolto in modo elementare
- la soluzione congiunta di tutti i sotto-problemi costituisce una soluzione per il problema iniziale

Un'altra metodologia efficace è basata sul “riuso” e l’“adattamento” di alcuni algoritmi (noti) di base

## Una metodologia per la progettazione di algoritmi

Progettazione di algoritmi per **raffinamenti successivi**

- scrivi una versione iniziale dell'algoritmo
  - una sequenza di uno o più passi
- raffina ciascun passo dell'algoritmo, fintato che l'algoritmo non è completo di tutti i dettagli
  - il raffinamento di un passo consiste nella sostituzione del passo con un'istruzione semplice o con un'istruzione di controllo

La progettazione di un algoritmo per raffinamenti successivi è un'attività iterativa

## Una metodologia per la progettazione di algoritmi

Questa metodologia è stata in effetti già adottata

1. fai avanzare **robbie** tre volte

1. *fai avanzare* **robbie** *tre volte*

1.1 fai avanzare **robbie**

1.2 fai avanzare **robbie**

1.3 fai avanzare **robbie**

1. *fai avanzare* **robbie** *tre volte*

1.1 *fai avanzare* **robbie**

**robbie.avanza();**

1.2 *fai avanzare* **robbie**

**robbie.avanza();**

1.3 *fai avanzare* **robbie**

**robbie.avanza();**

■ raffinamenti, commenti e numerazione gerarchica

## Un'altra metodologia per la progettazione di algoritmi

E' possibile anche progettare nuovi algoritmi per adattamento di altri algoritmi (noti)

- in alcuni casi, per risolvere un problema P, è possibile
  - partire da un algoritmo conosciuto A' che risolve un problema P' simile a P
  - adattare (modificare) A' in modo che risolva il problema P anziché P'
- attenzione, questo non è sempre possibile

La modalità di progettazione di algoritmi per adattamento richiede la conoscenza di un certo numero di algoritmi di base

- da utilizzare, appunto, per l'adattamento

## Problemi di ingresso-uscita

I problemi di ingresso-uscita sono una classe ristretta di problemi

- i dati di ingresso del problema vengono letti dalla tastiera
- i dati di uscita calcolati risolvendo il problema vengono visualizzati sullo schermo

Un esempio di problema di ingresso-uscita

- leggere dalla tastiera una coppia di numeri A e B, calcolarne la somma e visualizzarla sullo schermo

```
Scrivi due numeri interi
```

```
10 15
```

```
La somma dei due numeri è 25
```

- problemi di questo tipo portano alla scrittura di applicazioni (piuttosto che di metodi)

## Problemi di ingresso-uscita

Specifici di un **problema di ingresso-uscita**

- **dati di ingresso**
  - i parametri del problema – devono essere letti dalla tastiera
- **pre-condizione**
  - proprietà dei dati di ingresso
  - si assume che sia verificata – ovvero, si ipotizza che l'utente inserisca dei dati di ingresso che soddisfano la pre-condizione
- **dati di uscita**
  - i risultati del problema – devono essere visualizzati sullo schermo
- **post-condizione**
  - proprietà dei dati di uscita rispetto ai dati di ingresso

## Lettura e somma di due numeri interi

Si consideri il seguente semplice problema di ingresso-uscita

- si vuole leggere dalla tastiera una coppia di numeri A e B, calcolarne la somma e visualizzarla sullo schermo

Problema

- lettura e somma di due numeri interi

Dati di ingresso

- una coppia di numeri interi, A e B

Pre-condizione

- nessuna

Dati di uscita

- un numero S

Post-condizione

- S è la somma di A e B

## Lettura e somma di due numeri interi

Si osservi la natura parametrica di questo problema

- risolvere questo problema vuol dire identificare e implementare un algoritmo in grado di leggere e calcolare la somma di due numeri interi A e B
  - indipendentemente dagli specifici valori di A e B
- in generale, i problemi di interesse per l'informatica sono parametrici
  - i risultati da calcolare dipendono da dati i cui valori non sono noti al momento in cui si vuole affrontare e risolvere il problema

## Algoritmo per la lettura e somma di due numeri interi

Algoritmo per il problema della lettura e somma di due numeri interi

1. leggi (dalla tastiera) i due numeri interi **a** e **b**
2. calcola la somma **somma** di **a** e **b**
3. visualizza **somma** (sullo schermo)

Una sequenza di tre passi

- il procedimento non è ancora sufficientemente dettagliato, e quindi i tre passi devono essere ulteriormente raffinati

## Raffinamento del passo 1

1. leggi (dalla tastiera) i due numeri interi **a** e **b**

Il passo 1 può essere raffinato usando un'istruzione di controllo per sequenza

1. *leggi (dalla tastiera) i due numeri interi **a** e **b***
  - 1.1 leggi il numero intero **a** dalla tastiera
  - 1.2 leggi il numero intero **b** dalla tastiera

## Raffinamento dei passi 1.1 e 1.2

I passi ottenuti dal raffinamento del passo 1 possono essere ulteriormente raffinati sulla base delle seguenti considerazioni

- per la lettura dalla tastiera è possibile usare un oggetto **in** di tipo **Scanner**
  - per ora, trascuriamo l'aspetto della creazione di questo oggetto
- è poi possibile effettuare la lettura di un numero intero usando l'operazione **int nextInt()** dell'oggetto **in**

La seguente porzione dell'algoritmo è stata ottenuta mediante lo svolgimento di due raffinamenti (uno per ciascun passo)

1. *leggi (dalla tastiera) i due numeri interi **a** e **b***
  - 1.1 *leggi il numero intero **a** dalla tastiera*  
**a = in.nextInt();**
  - 1.2 *leggi il numero intero **b** dalla tastiera*  
**b = in.nextInt();**

## Raffinamento dei passi 2 e 3

2. calcola la somma **somma** di **a** e **b**
3. visualizza **somma** (sullo schermo)

Il passo 2 può essere raffinato usando un'istruzione semplice di assegnazione

2. *calcola la somma **somma** di **a** e **b***  
**somma = a + b;**

Il passo 3 può essere raffinato usando un'istruzione semplice di invio di un messaggio all'oggetto **System.out**

3. *visualizza **somma** (sullo schermo)*  
**System.out.println(somma);**

## Algoritmo per la lettura e somma di due numeri interi

In sintesi, il problema della lettura e somma di due numeri interi può essere risolto dal seguente algoritmo

1. leggi (dalla tastiera) i due numeri interi **a** e **b**
  - 1.1 leggi il numero intero **a** dalla tastiera  
**a = in.nextInt();**
  - 1.2 leggi il numero intero **b** dalla tastiera  
**b = in.nextInt();**
2. calcola la somma **somma** di **a** e **b**  
**somma = a + b;**
3. visualizza **somma** (sullo schermo)  
**System.out.println(somma);**

## Variabili per l'algoritmo

L'algoritmo per la lettura e somma di due numeri interi fa uso delle seguenti variabili

- la variabile intera **a**
  - rappresenta il primo numero intero letto dalla tastiera
- la variabile intera **b**
  - rappresenta il secondo numero intero letto dalla tastiera
- la variabile intera **somma**
  - rappresenta la somma di **a** e **b**

## Oggetti per l'algoritmo

Inoltre, l'algoritmo per la lettura e somma di due numeri interi fa uso dei seguenti oggetti

- l'oggetto **in** di tipo **Scanner** – rappresenta la tastiera, da cui vanno letti i dati
  - per quest'oggetto è necessario dichiarare una variabile **in**, di tipo **Scanner**
  - l'implementazione dell'algoritmo dovrà occuparsi anche della creazione di questo oggetto, nell'ambito di un'assegnazione

```
in = new Scanner( System.in );
```

- l'oggetto **System.out** – rappresenta lo schermo, su cui vanno visualizzati i risultati
  - non è necessaria nessuna variabile per quest'oggetto
  - l'implementazione dell'algoritmo non dovrà occuparsi della creazione di questo oggetto

## Programma per la lettura e somma di due numeri interi

L'algoritmo per la lettura e somma di due numeri interi può essere codificato da un'applicazione Java

- l'algoritmo viene implementato dal metodo **main** di una classe applicazione

```
import java.util.*;
/* Applicazione che legge dalla tastiera due numeri interi
 * e ne calcola e visualizza la somma. */
public class SommaDueNumeri {
    public static void main(String[] args) {

        ...

    }
}
```

## Programma per la lettura e somma di due numeri interi

```
int a;           // il primo numero intero
int b;           // il secondo numero intero
int somma;      // la somma di a e b
Scanner in;     // per la lettura dalla tastiera

/* crea l'oggetto che rappresenta la tastiera */
in = new Scanner( System.in );

/* legge i due numeri interi a e b */
System.out.println("Scrivi due numeri interi");
/* legge due numeri interi a e b */
a = in.nextInt();
b = in.nextInt();
/* calcola la somma di a e b e la visualizza */
somma = a+b;
System.out.print("La somma dei due numeri è ");
System.out.println(somma);
```

## Somma di una sequenza di numeri interi

Si consideri il seguente problema di ingresso-uscita

- si vuole leggere dalla tastiera una sequenza di numeri interi, separati da spazi e terminata da una 'X' (o un altro carattere), calcolare la somma degli elementi della sequenza e visualizzarla sullo schermo

Scrivi una sequenza di numeri interi

10 15 0 -2 X

La somma dei numeri è 23

## Lettura e somma di una sequenza di numeri interi

### Problema

- lettura e somma di una sequenza di numeri interi

### Dati di ingresso

- una sequenza di numeri interi  $A_1, A_2, \dots, A_N$

### Pre-condizione

- $N \geq 0$ , i numeri sono separati da spazi e terminati da una 'X'

### Dati di uscita

- un numero S

### Post-condizione

- S è uguale alla somma  $A_1 + A_2 + \dots + A_N$

## Lettura e somma di una sequenza di numeri interi

### Parametricità di questo problema

- la lunghezza della sequenza
- il valore degli elementi della sequenza

Il problema della lettura e somma di una sequenza di numeri interi non può essere risolto da un algoritmo che sia semplicemente una sequenza di istruzioni semplici

- il numero degli elementi della sequenza non è noto a priori

La versione iniziale dell'algoritmo è la seguente

1. leggi una sequenza di numeri interi e calcolane la somma  
**somma**
2. *visualizza* **somma**  
**System.out.println(somma);**

## Raffinamento del passo 1

1. leggi una sequenza di numeri interi e calcolane la somma  
**somma**

Raffinamento di questo passo

1. *leggi una sequenza di numeri interi e calcolane la somma*  
**somma**

1.1 *inizialmente* **somma** vale zero

**somma = 0;**

1.2 *finché ci sono altri numeri, leggili e sommalili a* **somma**

**while ( ci sono altri numeri nella sequenza ) {**

leggi un elemento della sequenza **numero** dalla tastiera

incrementa **somma** di **numero**

**}**

È già possibile provare ad applicare questo algoritmo

- ad es., alla sequenza di ingresso **1 3 5 X**

## Uso della variabile **somma**

La variabile **somma** è usata nell'algoritmo per memorizzare la somma degli elementi che sono stati letti dalla tastiera

- inizialmente, quando nessun elemento della tastiera è stato letto, vale 0
- dopo ciascuna operazione di lettura e incremento, è uguale alla somma degli elementi letti fino a quel momento
- dopo la lettura dell'intera sequenza, è uguale alla somma di tutti gli elementi della sequenza
  - che è proprio il valore che deve essere calcolato

## Discussione

L'uso dell'istruzione ripetitiva permette di risolvere il problema

- l'algoritmo legge tutti gli elementi della sequenza
  - il corpo dell'istruzione ripetitiva viene eseguito tante volte quanti sono gli elementi della sequenza
- l'algoritmo calcola la somma degli elementi letti

## Ulteriori raffinamenti

Alcuni ulteriori raffinamenti sono immediati

*1.2 finché ci sono altri numeri nella sequenza, leggili e sommalì a **somma***

```
while ( ci sono altri numeri nella sequenza ) {  
    leggi un elemento della sequenza numero dalla tastiera  
    numero = in.nextInt();  
    incrementa somma di numero  
    somma = somma + numero;  
}
```

Che cosa vuol dire **somma = somma + numero?**

## Raffinamento della condizione

**while** ( ci sono altri numeri nella sequenza )

Rimane da raffinare la condizione dell'istruzione ripetitiva

- bisogna sapere che l'oggetto **in** fornisce una operazione **boolean hasNextInt()** che verifica se il prossimo dato in ingresso – più precisamente, il prossimo “token” – è un numero intero (oppure no)
- inoltre, è richiesto che l'utente inserisca una 'X' al termine della sequenza di numeri
  - una 'X' NON è un numero
- dunque è possibile usare l'espressione **in.hasNextInt()** per sapere se ci sono ancora altri numeri nella sequenza
- questa espressione può essere usata come condizione per la nostra istruzione ripetitiva

## Algoritmo per la somma di una sequenza di numeri interi

1. *leggi una sequenza di numeri interi e calcolane la somma*

**somma**

1.1 *inizialmente **somma** vale zero*

**somma = 0;**

1.2 *finché ci sono altri numeri nella sequenza, leggili e sommalili a **somma***

**while** ( *ci sono altri numeri nella sequenza* )

**while** ( **in.hasNextInt()** ) {

*leggi un elemento della sequenza **numero** dalla tastiera*

**numero = in.nextInt();**

*incrementa **somma** di **numero***

**somma = somma + numero;**

}

2. *visualizza **somma***

**System.out.println(somma);**

## Programma per la somma di una sequenza di numeri

```
int numero;        // elemento corrente della sequenza
int somma;        // somma degli elementi della sequenza
Scanner in;       // per la lettura dalla tastiera

/* crea l'oggetto che rappresenta la tastiera */
in = new Scanner( System.in );
```

## Programma per la somma di una sequenza di numeri

```
/* leggi una sequenza di numeri,
 * separata da spazi e terminata da una 'X'
 * (o un altro carattere), e calcolane la somma */
System.out.println(
    "Scrivi una sequenza di numeri interi");
/* inizialmente somma vale zero */
somma = 0;
/* finché ci sono altri numeri nella sequenza,
 * leggili e sommali a somma */
while ( in.hasNextInt() ) { // finché il prossimo
    // elemento è un numero
    /* leggi un numero della sequenza */
    numero = in.nextInt();
    /* incrementa somma di numero */
    somma = somma + numero;
}
/* visualizza somma */
System.out.print("La somma dei numeri è ");
System.out.println(somma);
```

## Lunghezza di una sequenza

Si consideri ora il seguente problema di ingresso-uscita

- si vuole leggere dalla tastiera una sequenza di numeri interi, separati da spazi e terminata da una 'X' (o un altro carattere), calcolare la lunghezza della sequenza (ovvero, il numero di elementi della sequenza) e visualizzarla sullo schermo

Scrivi una sequenza di numeri interi

10 15 0 -2 X

La lunghezza della sequenza è 4

## Lunghezza di una sequenza

Problema

- lettura di una sequenza di numeri interi e calcolo della lunghezza

Dati di ingresso

- una sequenza di numeri interi  $A_1, A_2, \dots, A_N$

Pre-condizione

- $N \geq 0$ , i numeri sono separati da spazi e terminati da una 'X'

Dati di uscita

- un numero  $L$

Post-condizione

- $L$  è uguale alla lunghezza  $N$  della sequenza  $A_1, A_2, \dots, A_N$

## Lunghezza di una sequenza

Una variante del problema della lettura e somma di una sequenza di numeri interi

- bisogna leggere dalla tastiera una sequenza di numeri interi
  - ma non bisogna calcolarne la somma
  - bisogna piuttosto contare il numero degli elementi

Questo problema può essere risolto adattando l'algoritmo per la somma di una sequenza di numeri

- anziché gestire la variabile **somma**
- va gestita una variabile **lunghezza** che
  - inizialmente deve valere zero
  - ogni volta che viene letto un elemento deve essere incrementata di uno
- **lunghezza** viene usata per contare il numero degli elementi della sequenza che sono stati letti
  - al termine, è uguale alla lunghezza della sequenza

## Algoritmo per il calcolo della lunghezza di una sequenza

1. *leggi una sequenza di numeri interi e calcolane la lunghezza*  
**lunghezza**

1.1 *inizialmente lunghezza vale zero*

**lunghezza = 0;**

1.2 *finché ci sono altri numeri nella sequenza, leggili e incrementa lunghezza*

```
while ( in.hasNextInt() ) {
```

*leggi un elemento della sequenza numero dalla tastiera*

```
numero = in.nextInt();
```

*incrementa lunghezza*

```
lunghezza = lunghezza + 1;
```

```
}
```

2. *visualizza lunghezza*

```
System.out.println(lunghezza);
```

## Esercizi – validazione e implementazione

Verificare l'algoritmo per il calcolo della lunghezza di una sequenza

- ad esempio, se la sequenza è **1 3 5 0 X**

Implementare l'algoritmo per il calcolo della lunghezza di una sequenza

## Somma dei pari e dei dispari in una sequenza

Si consideri il seguente problema di ingresso-uscita, che è una variante dei problemi precedenti

- si vuole leggere dalla tastiera una sequenza di numeri interi, separati da spazi e terminata da una 'X' (o un altro carattere), calcolare la somma degli elementi di valore pari e la somma degli elementi di valore dispari, e visualizzare la somma dei pari e la somma dei dispari sullo schermo

**Scrivi una sequenza di numeri interi**

**10 15 0 -2 X**

**La somma dei pari è 8**

**La somma dei dispari è 15**

Una variante della lettura e somma di una sequenza

- ci sono due dati di uscita anziché uno solo

## Somma dei pari e dei dispari in una sequenza

Per risolvere il problema bisogna gestire una coppia di variabili

- **sommaPari**
  - la somma degli elementi di valore pari della sequenza
- **sommaDispari**
  - la somma degli elementi di valore dispari della sequenza

Queste due variabili devono essere gestite come segue

- inizialmente valgono entrambe zero
- per ogni numero della sequenza
  - se il numero è pari, allora deve essere usato per incrementare **sommaPari**
  - altrimenti il numero è dispari, e deve essere usato per incrementare **sommaDispari**

## Somma dei pari e dei dispari in una sequenza

In questo caso, all'interno dell'istruzione ripetitiva, deve essere usata una istruzione condizionale **if-else**

- per ogni elemento della sequenza deve essere eseguita una tra due istruzioni al verificarsi (o meno) di una condizione
  - una possibile istruzione è l'incremento di **sommaPari**
  - l'altra possibile istruzione è l'incremento di **sommaDispari**
  - comunque deve essere eseguita esattamente una tra queste due istruzioni

## Uso dell'istruzione if-else

*leggi gli elementi della sequenza e calcola la somma dei pari e la somma dei dispari*

**sommaPari = 0;**

**sommaDispari = 0;**

*finché ci sono altri numeri nella sequenza, leggili e calcola la somma dei pari e la somma dei dispari*

**while ( in.hasNextInt() ) {**

**numero = in.nextInt();**

*se numero è pari, incrementa sommaPari di numero, altrimenti incrementa sommaDispari di numero*

**if ( numero è pari )**

**sommaPari = sommaPari + numero;**

**else**

**sommaDispari = sommaDispari + numero;**

**}**

## Condizione per verificare se un numero è pari

Rimane da raffinare la condizione “**numero è pari**”

- l'operatore **%** calcola il resto della divisione tra due numeri interi
- un numero è pari se diviso per due dà resto zero
- la condizione dell'istruzione condizionale può essere scritta come **numero%2==0**

## Algoritmo per la somma dei pari e dei dispari

*leggi gli elementi della sequenza e calcola la somma dei pari e la somma dei dispari*

**sommaPari = 0;**

**sommaDispari = 0;**

*finché ci sono altri numeri nella sequenza, leggili e calcola la somma dei pari e la somma dei dispari*

**while ( in.hasNextInt() ) {**

**numero = in.nextInt();**

*se numero è pari, incrementa sommaPari di numero, altrimenti incrementa sommaDispari di numero*

**if ( numero%2==0 )**

**sommaPari = sommaPari + numero;**

**else**

**sommaDispari = sommaDispari + numero;**

**}**

## Programma per la somma dei pari e la somma dei dispari

```
int numero;    // elemento corrente della sequenza
int sommaPari; // somma degli elementi pari
int sommaDispari; // somma degli elementi dispari
Scanner in;    // per la lettura dalla tastiera

/* crea un oggetto per la lettura dalla tastiera */
in = new Scanner( System.in );
```

## Programma per la somma dei pari e la somma dei dispari

```
/* leggi una sequenza di numeri interi e calcola
 * la somma dei pari e la somma dei dispari */

/* inizialmente le somme valgono zero */
sommaPari = 0;
sommaDispari = 0;

/* finché ci sono altri numeri nella sequenza, leggili
 * e calcola la somma dei pari e la somma dei dispari */
while ( in.hasNextInt() ) {
    /* leggi un elemento della sequenza */
    numero = in.nextInt();
    /* se numero è pari, incrementa sommaPari di numero,
     * altrimenti incrementa sommaDispari di numero */
    if ( numero%2==0)    // se numero è pari
        /* incrementa sommaPari di numero */
        sommaPari = sommaPari + numero;
    else    // altrimenti numero è dispari
        /* incrementa sommaDispari di numero */
        sommaDispari = sommaDispari + numero;
}

/* visualizza le somme */
System.out.println(sommaPari);
System.out.println(sommaDispari);
```

## Esercizio – conta gli zeri

Risolvere il seguente problema di ingresso-uscita

- leggere dalla tastiera una sequenza di numeri interi, separati da spazi e terminata da una 'X' (o un altro carattere), calcolare il numero di zeri della sequenza (ovvero, il numero di elementi della sequenza che sono uguali a zero) e visualizzare il numero di zeri sullo schermo

Scrivi una sequenza di numeri interi

10 15 0 -2 X

Il numero di zeri nella sequenza è 1

- bisogna usare l'istruzione if

## Calcolo degli zeri di una sequenza

```
int numero;    // elemento corrente della sequenza
int zeri;      // numero di zeri nella sequenza
Scanner in;    // per la lettura dalla tastiera

/* crea un oggetto per la lettura dalla tastiera */
in = new Scanner( System.in );

/* leggi una sequenza di numeri interi e
 * calcolane il numero di zeri */
/* inizialmente zeri vale zero */
zeri = 0;
/* finché ci sono altri numeri nella sequenza,
 * leggili e calcola il numero di zeri */
while ( in.hasNextInt() ) {
    /* leggi un elemento della sequenza */
    numero = in.nextInt();
    /* se numero è uguale a zero, incrementa zeri */
    if (numero==0)    // se numero è uguale a zero
        /* incrementa zeri */
        zeri = zeri + 1;
}
/* visualizza zeri */
System.out.println(zeri);
```

75

Problemi, algoritmi e oggetti

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Esercizio – media aritmetica

Scrivere e implementare un algoritmo che legge una sequenza non vuota di numeri interi e ne calcola la media aritmetica

- $\text{media aritmetica} = \text{somma degli elementi} / \text{numero di elementi}$

76

Problemi, algoritmi e oggetti

Fondamenti di informatica: Oggetti e Java  
Luca Cabibbo

## Somma di una sequenza di dieci numeri

Si consideri il seguente problema di ingresso-uscita

- si vuole leggere dalla tastiera una sequenza di numeri interi di lunghezza fissata (ad esempio, di lunghezza dieci), calcolare la somma degli elementi e visualizzarla sullo schermo

Scrivi una sequenza di dieci numeri interi

1 12 -13 2 0

-4 8 0

4 10

La somma dei dieci numeri è 20

## Letture e somma di una sequenza di dieci numeri interi

Problema

- lettura di una sequenza di dieci numeri interi e calcolo della somma degli elementi

Dati di ingresso

- una sequenza di dieci numeri interi  $A_1, A_2, \dots, A_{10}$

Pre-condizione

- nessuna

Dati di uscita

- un numero  $S$

Post-condizione

- $S$  è la somma degli elementi della sequenza  $A_1, A_2, \dots, A_{10}$

## Lettura e somma di una sequenza di dieci numeri interi

Una soluzione poco parametrica e poco modificabile

- dieci variabili
- dieci istruzioni di lettura
- una somma di dieci valori

E' preferibile una soluzione parametrica basata su un'istruzione ripetitiva

- si suppone che la lunghezza della sequenza sia nota a priori
- cerchiamo un algoritmo parametrico rispetto a questa lunghezza

## Lettura e somma di una sequenza di dieci numeri interi

L'algoritmo può essere realizzato gestendo le seguenti variabili

- **numero**
  - valore dell'elemento corrente della sequenza
- **somma**
  - la somma degli elementi della sequenza che sono stati già letti
- **numeriLetti**
  - il numero degli elementi della sequenza che sono stati già letti

La gestione delle variabili **somma** e **numeriLetti** è come segue

- inizialmente valgono entrambe zero
- per ogni **numero** letto
  - **somma** deve essere incrementata di **numero**
  - **numeriLetti** deve essere incrementata di uno

## Algoritmo per la lettura e somma di dieci numeri

*leggi una sequenza di dieci numeri interi e calcolane la somma  
inizializza la somma e il numero di elementi letti*

**somma = 0;**

**numeriLetti = 0;**

*leggi i dieci elementi della sequenza e calcolane la somma*

**while ( ci sono altri elementi nella sequenza ) {**

*leggi un elemento **numero** della sequenza*

**numero = in.nextInt();**

*incrementa **somma di numero***

**somma = somma + numero;**

*incrementa **numeriLetti***

**numeriLetti = numeriLetti + 1;**

**}**

Bisogna raffinare la condizione dell'istruzione ripetitiva

- quante volte viene valutata la condizione?
- quanto vale **numeriLetti** durante ciascuna valutazione della condizione?

## Algoritmo per la lettura e somma di dieci numeri

*inizializza la somma e il numero di elementi letti*

**somma = 0;**

**numeriLetti = 0;**

*leggi i dieci elementi della sequenza e calcolane la somma*

**while ( numeriLetti < 10 ) {**

*leggi un elemento **numero** della sequenza*

**numero = in.nextInt();**

*incrementa **somma di numero***

**somma = somma + numero;**

*incrementa **numeriLetti***

**numeriLetti = numeriLetti + 1;**

**}**

## Programma per la lettura e somma di dieci numeri

```
int numero;    // elemento corrente della sequenza
int somma;    // somma degli elementi della sequenza
int numeriLetti; // numero degli elementi letti
Scanner in;    // per la lettura dalla tastiera

/* crea un oggetto per la lettura dalla tastiera */
in = new Scanner( System.in );

/* leggi una sequenza di dieci numeri interi e
 * calcolane la somma */
/* inizializza la somma e il numero di elementi letti */
somma = 0;
numeriLetti = 0;
/* leggi i dieci elementi della sequenza e
 * calcolane la somma */
while (numeriLetti<10) { // ci sono altri elementi
    /* leggi un elemento della sequenza */
    numero = in.nextInt();
    /* incrementa somma di numero */
    somma = somma + numero;
    /* incrementa numeriLetti */
    numeriLetti = numeriLetti + 1;
}
/* visualizza somma */
System.out.println(somma);
```

## Ricapitolando – lettura e somma di una seq. di numeri

### 1. leggi una sequenza di numeri interi e calcolane la somma

#### **somma**

#### 1.1 inizialmente **somma** vale zero

```
somma = 0;
```

#### 1.2 finché ci sono altri numeri nella sequenza, leggili e sommalili a **somma**

```
while ( ci sono altri numeri nella sequenza )
```

```
while (in.hasNextInt() ) {
```

```
    leggi un elemento della sequenza numero dalla tastiera
```

```
    numero = in.nextInt();
```

```
    incrementa somma di numero
```

```
    somma = somma + numero;
```

```
}
```

### 2. visualizza **somma**

```
System.out.println(somma);
```

## Ricapitolando – lettura e somma di una seq. di 10 numeri

*inizializza la somma e il numero di elementi letti*

**somma = 0;**

**numeriLetti = 0;**

*leggi i dieci elementi della sequenza e calcolane la somma*

**while ( numeriLetti < 10 ) {**

*leggi un elemento numero della sequenza*

**numero = in.nextInt();**

*incrementa somma di numero*

**somma = somma + numero;**

*incrementa numeriLetti*

**numeriLetti = numeriLetti + 1;**

**}**

## Somma dei primi N numeri interi positivi

Si vuole risolvere il problema di calcolare la somma dei primi N numeri interi positivi, dato un numero naturale N

- un problema nella forma più generale
- la sua soluzione deve portare alla scrittura di un metodo in cui
  - il numero naturale N è un parametro del metodo
  - la somma dei primi N numeri interi positivi deve essere restituita dal metodo

## Somma dei primi N numeri interi positivi

### Problema

- somma dei primi N numeri interi positivi

### Insieme di ingresso

- un numero N

### Pre-condizione

- N è un numero naturale, ovvero  $N \geq 0$

### Insieme di uscita

- un numero S

### Post-condizione

- S è uguale alla somma  $1+2+ \dots +N$  dei primi N numeri positivi (S deve essere uguale a zero nel caso in cui N sia uguale a zero)

## Somma dei primi N numeri interi positivi

La somma dei primi N numeri interi positivi può essere calcolata da un algoritmo che utilizza le seguenti variabili

- una variabile **somma**, gestita come segue
  - **somma** inizialmente vale zero
  - a **somma** deve essere sommato ciascuno dei primi N numeri interi positivi – ovvero, ciascun numero intero compreso tra 1 e N
- una variabile **i**
  - a cui si vogliono far assumere come valori tutti i numeri interi compresi tra 1 e N – ovvero, i valori che devono essere sommati a **somma**

## Algoritmo per la somma dei primi N numeri interi positivi

La prima formulazione dell'algoritmo è la seguente

*inizializza somma*

**somma = 0;**

per ogni valore di **i** compreso tra 1 e N, incrementa **somma** di **i**

*incrementa somma di i*

**somma = somma + i;**

*il risultato è somma*

L'algoritmo non è completamente dettagliato

- deve essere eseguito un raffinamento, introducendo una istruzione ripetitiva

## Algoritmo per la somma dei primi N numeri interi positivi

Bisogna introdurre nell'algoritmo dei passi che permettano di far assumere a **i** tutti i valori interi compresi tra 1 e N

*inizializza somma*

**somma = 0;**

per ogni valore di **i** compreso tra 1 e N, incrementa **somma** di **i**

*inizializza i*

**i = 1;**

**while ( i non ha assunto tutti i valori compresi tra 1 e N ) {**

*incrementa somma di i*

**somma = somma + i;**

*incrementa i*

**i = i + 1;**

**}**

*il risultato è somma*

## Esecuzione dell'istruzione ripetitiva

Prima di raffinare la condizione dell'istruzione ripetitiva, è utile comprendere come opera l'algoritmo

- ad esempio, nel caso in cui  $N$  vale 3
- quante volte viene valutata la condizione?
- quanto vale  $i$  durante ciascuna valutazione della condizione?

## Algoritmo per la somma dei primi $N$ numeri interi positivi

*inizializza **somma***

**somma = 0;**

*per ogni valore di  $i$  compreso tra 1 e  $N$ , incrementa **somma** di  $i$*

*inizializza  $i$*

**$i = 1;$**

***while** (  $i$  non ha assunto tutti i valori compresi tra 1 e  $N$  )*

**while** (  $i \leq N$  ) {

*incrementa **somma** di  $i$*

**somma = somma +  $i$ ;**

*incrementa  $i$*

**$i = i + 1;$**

}

*il risultato è **somma***

## Metodo per la somma dei primi N numeri interi positivi

L'algoritmo può essere codificato come metodo di un oggetto

```
/* Oggetto che sa calcolare la somma dei primi N numeri
 * interi positivi. */
class SommatorePositivi {
    /* Calcola la somma dei primi n numeri
     * interi positivi. */
    public static int sommaPrimiPositivi(int n) {

        ...

    }
}
```

## Metodo per la somma dei primi N numeri interi positivi

```
/* Calcola la somma dei primi n numeri
 * interi positivi. */
public static int sommaPrimiPositivi(int n) {
    // pre: n>=0
    int somma;    // somma dei primi n numeri positivi
    int i;        // per iterare tra 1 e n

    /* inizializza somma */
    somma = 0;
    /* per ogni valore di i compreso tra 1 e n,
     * incrementa somma di i */
    /* inizializza i */
    i = 1;
    while (i<=n) {    /* finché i non ha assunto tutti i
                       * valori compresi tra 1 e n */
        /* incrementa somma di i */
        somma = somma + i;
        /* incrementa i */
        i = i + 1;
    }
    /* il risultato è somma */
    return somma;
}
```

## Uso del metodo sommaPrimiPositivi

Il metodo **int sommaPrimiPositivi(int n)** può essere ad esempio utilizzato come segue

```
int sommaPrimiQuattro;    // somma dei primi quattro
                          // numeri interi positivi

/* calcola la somma dei primi quattro numeri positivi
 * la visualizza sullo schermo */
sommaPrimiQuattro =
    SommatorePositivi.sommaPrimiPositivi(4);
System.out.println(sommaPrimiQuattro);
```

## Traccia dell'esecuzione di un metodo o di un algoritmo

La **traccia** dell'esecuzione di un metodo o algoritmo è una tabella (costruita dinamicamente) in cui sono elencati

- le istruzioni eseguite
- il valore di ciascuna variabile, dopo l'esecuzione di ciascuna istruzione

Si consideri ad esempio il caso in cui **n** vale 4

istruzione eseguita	n	somma	i	i<=n
<i>invocazione</i>	4			
<b>somma = 0</b>		<b>0</b>		
<b>i = 1</b>			<b>1</b>	
<b>i&lt;=n ?</b>				<b>true</b>
<b>somma = somma+i</b>		<b>1</b>		
<b>i = i+1</b>			<b>2</b>	

## Esecuzione del metodo

istruzione eseguita	n	somma	i	i<=n
... segue ...	4	1	2	
i<=n ?				true
somma = somma+i		3		
i = i+1			3	
i<=n ?				true
somma = somma+i		6		
i = i+1			4	
i<=n ?				true
somma = somma+i		10		
i = i+1			5	
i<=n ?				false
return somma				

L'esecuzione del metodo restituisce il valore 10

## La formula di Gauss

Esiste una nota formula (formula di Gauss) per il calcolo della somma  $S_N$  dei primi  $N$  numeri interi positivi (valida per  $N$  naturale)

$$S_N = 1+2+ \dots + N = N (N+1) / 2$$

```
/* Calcola la somma dei primi n numeri positivi. */
public static int sommaPrimiPositiviGauss(int n) {
    // pre: n >= 0
    int somma;    // somma dei primi n numeri positivi

    /* calcola la somma dei primi n numeri positivi */
    somma = n*(n+1)/2;
    /* il risultato è somma */
    return somma;
}
```

- uno stesso problema può essere risolto da più algoritmi, anche molto diversi tra loro

## Confronto tra i due algoritmi

Quante operazioni sono richieste per lo svolgimento dei due algoritmi?

- la soluzione iterativa per il calcolo della somma dei primi  $N$  numeri interi positivi richiede lo svolgimento di  $2N+2$  assegnazioni e di  $N+1$  confronti
- la soluzione basata sulla formula di Gauss richiede lo svolgimento di una sola assegnazione

La soluzione basata sulla formula di Gauss è sempre più efficiente di quella iterativa

- “sempre” va interpretato nel senso di “per ogni possibile valore di  $N$  che soddisfa la pre-condizione del problema” – ovvero per ogni numero naturale  $N$

## Esercizio – come fare per eseguire un'istruzione $N$ volte

Definire un algoritmo – o meglio, uno “schema algoritmico” – che consente di eseguire una certa istruzione  $I$  per esattamente  $N$  volte

- dove  $N$  è un numero naturale
- finora sono stati proposti due diversi schemi algoritmici di questo tipo

## Esercizio – fattoriale

Definire un algoritmo e poi un metodo per il calcolo del fattoriale di un numero naturale N

- il fattoriale di un numero naturale N è il prodotto dei primi N numeri interi positivi, ovvero dei numeri interi compresi tra 1 e N
  - ad esempio, il fattoriale di 5 è uguale a  $1*2*3*4*5$ , che vale 120
  - per definizione, il fattoriale di 0 è uguale a 1

## Conversione Celsius-Fahrenheit

Si vuole visualizzare sullo schermo una tabella di conversione di temperature da gradi Celsius a gradi Fahrenheit

- vale la formula  $T_F = T_C * 1.8 + 32$
- per temperature (Celsius) da 0 a 100 gradi – ogni 5 gradi

Celsius	Fahrenheit
0.0	32.0
5.0	41.0
10.0	50.0
15.0	59.0
...	...
100.0	212.0

## Conversione Celsius-Fahrenheit

La prima formulazione dell'algoritmo è la seguente

per ogni valore di **tc** compreso tra 0 e 100, per incrementi di 5

**tf = tc\*1.8+32;**

visualizza **tc** e **tf**

## Conversione Celsius-Fahrenheit

```
double tc;    // temperatura in gradi Celsius
double tf;    // temperatura in gradi Fahrenheit

/* intestazione della tabella */
System.out.println("          Celsius          Fahrenheit ");

/* corpo della tabella */
tc = 0;
while (tc <= 100) {
    tf = tc*1.8+32;
    System.out.println("\t" + tc + "\t\t" + tf);
    tc = tc + 5;
}
```

## Esercizi – prodotto

Definire un algoritmo per calcolare il prodotto di due numeri naturali A e B mediante somme ripetute

- A per B si ottiene sommando A per B volte ad una variabile che inizialmente vale zero

Definire un algoritmo per calcolare il prodotto di due numeri interi A e B mediante somme ripetute

## Esercizio – Massimo di una sequenza di numeri interi

Scrivere un'applicazione che legge dalla tastiera una sequenza di dieci numeri interi e calcola l'elemento di valore massimo

```
Scrivi una sequenza di dieci numeri interi
1 12 -13 2 0 -4 8 0 4 10
Il massimo è 12
```

Il problema va risolto mediante il seguente algoritmo, che però va ulteriormente raffinato:

```
leggi il primo elemento della sequenza e assumi che sia
il massimo provvisorio
per nove volte {
    leggi un elemento della sequenza
    se l'elemento letto è maggiore del massimo provvisorio,
    allora assumi questo elemento come nuovo
    massimo provvisorio
}
```

il massimo provvisorio è il massimo della sequenza

## Esercizio – Massimo di una sequenza di numeri interi?

Si consideri nuovamente il problema di leggere dalla tastiera una sequenza di dieci numeri interi e calcolare l'elemento di valore massimo

Dimostrare che il seguente procedimento NON risolve tale problema:

assumi che il massimo provvisorio della sequenza valga zero per dieci volte {

leggi un elemento della sequenza

se l'elemento letto è maggiore del massimo provvisorio, allora assumi questo elemento come nuovo massimo provvisorio

}

il massimo provvisorio è il massimo della sequenza

Suggerimento: cercare un contro-esempio