LA PROGRAMMAZIONE STRUTTURATA

Il concetto e la definizione di algoritmo

INTRODUZIONE

Nella programmazione strutturata la soluzione ad un problema è l'**algoritmo**.

Un **algoritmo** fa uso di due classi di strumenti:

- le **strutture dati**
- le **strutture di controllo del flusso**

IL CONCETTO DI ALGORITMO

Sebbene la parola **algoritmo** suoni un po' difficile
la nostra **vita quotidiana** è piena di algoritmi.

Chiedere informazioni.

PROBLEMA

1. Prosegui diritto per 100 metri
2. Gira a destra
3. Alla fine della strada gira a destra
4. Al secondo semaforo gira a sinistra
5. Alla rotatoria prendi la terza uscita
6. Dopo 500 metri sei arrivato

ALESSANDRO URSOMANDO

LA PROGRAMMAZIONE STRUTTURATA
CC BY

IL CONCETTO DI ALGORITMO

Sebbene la parola **algoritmo** suoni un po' difficile
la nostra **vita quotidiana** è piena di algoritmi.

Lavare il bucato con la lavatrice.

PROBLEMA

1. Se l'oblò è chiuso, aprilo
2. Metti i panni nel cestello
3. Chiudi l'oblò
4. Accendi la lavatrice
5. Imposta il programma
6. Metti il detersivo
7. Metti l'ammorbidente
8. Avvia il ciclo

ALESSANDRO URSOMANDO

IL CONCETTO DI ALGORITMO

Sebbene la parola **algoritmo** suoni un po' difficile
la nostra **vita quotidiana** è piena di algoritmi.

Preparare la crema pasticcera.

PROBLEMA

1. Riscalda 500 ml di latte
2. Lavora 6 tuorli con 150 gr di zucchero e 50 gr di farina
3. Versa il composto nella casseruola
4. Continua a mescolare finché la crema assume la densità desiderata.

Vedremo in seguito che **tutti** gli strumenti
utili alla creazione di algoritmi sono importati dalla **vita quotidiana**.

DEFINIZIONE DI ALGORITMO

Un algoritmo è una **sequenza**
finita di **passi elementari**
che risolve una **classe di problemi.**

ALGORITMO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

DEFINIZIONE DI ALGORITMO

Un algoritmo è una **sequenza finita** di **passi elementari** che risolve una **classe di problemi.**

ALGORITMO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

ALESSANDRO URSOMANDO

DEFINIZIONE DI ALGORITMO

Un algoritmo è una **sequenza finita** di **passi elementari** che risolve una **classe di problemi.**

ALGORITMO

Un **passo elementare** è un'istruzione **chiara**, inequivocabile e **non ulteriormente scomponibile**

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Un'istruzione si dice non ulteriormente scomponibile quando è atomica

ALESSANDRO URSOMANDO

DEFINIZIONE DI ALGORITMO

QUAND'È CHE UN'ISTRUZIONE È CHIARA?

MEDICO

INFERMIERA

BIDELLO

Dipende dall'**esecutore**

ALESSANDRO URSOMANDO

DEFINIZIONE DI ALGORITMO

Un algoritmo è una **sequenza finita** di **passi elementari** che risolve una **classe di problemi.**

ALGORITMO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Un **passo elementare** è un'istruzione **chiara**, inequivocabile e **non ulteriormente scomponibile**

Il concetto di algoritmo si porta dietro il concetto di **esecutore**

L'esecutore di un algoritmo codificato in un linguaggio di programmazione è il **computer**

ALESSANDRO URSOMANDO

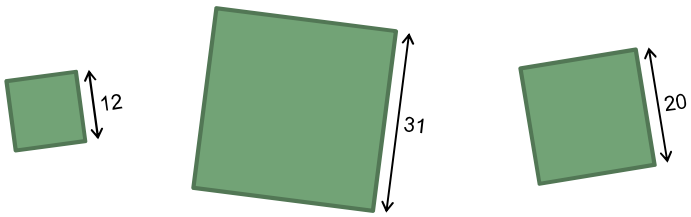
DEFINIZIONE DI ALGORITMO

Un algoritmo è una **sequenza finita** di **passi elementari** che risolve una **classe di problemi.**

ALGORITMO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Se il problema è **l'area del quadrato**, l'algoritmo non calcola l'area di **uno specifico** quadrato ma di un quadrato **qualsiasi**



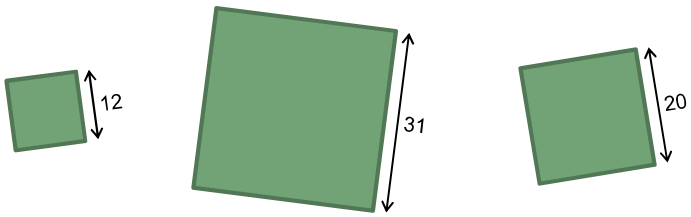
DEFINIZIONE DI ALGORITMO

Un algoritmo è una **sequenza finita** di **passi elementari** che risolve una **classe di problemi.**

ALGORITMO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Se il problema è **l'area del quadrato**, l'algoritmo non calcola l'area di **uno specifico** quadrato ma di un quadrato **qualsiasi**



Si dice anche che l'algoritmo è **generico.**

LA PROGRAMMAZIONE STRUTTURATA

Strutture dati

INFORMAZIONE E DATO

Un'**informazione** archiviata in un computer prende il nome di **dato**.

INFORMAZIONE E DATO

Un'**informazione** archiviata in un computer prende il nome di **dato**.

Il **dato** viene tenuto in una **struttura dati**
della quale conosciamo il **tipo**,
il **nome**,
il **contenuto**
ed il suo **indirizzo**
in memoria centrale o in memoria di massa

DATO

INFORMAZIONE E DATO

Un'**informazione** archiviata in un computer prende il nome di **dato**.

Il **dato** viene tenuto in una **struttura dati**
della quale conosciamo il **tipo**,
il **nome**,
il **contenuto**
ed il suo **indirizzo**
in memoria centrale o in memoria di massa

DATO

Intero, decimale, stringa, booleano..

INFORMAZIONE E DATO

Un'**informazione** archiviata in un computer prende il nome di **dato**.

Il **dato** viene tenuto in una **struttura dati**
della quale conosciamo il **tipo**,
il **nome**,
il **contenuto**
ed il suo **indirizzo**
in memoria centrale o in memoria di massa

DATO

INFORMAZIONE E DATO

Un'**informazione** archiviata in un computer prende il nome di **dato**.

Il **dato** viene tenuto in una **struttura dati**
della quale conosciamo il **tipo**,
il **nome**,
il **contenuto**
ed il suo **indirizzo**
in memoria centrale o in memoria di massa

DATO

INFORMAZIONE E DATO

Un'**informazione** archiviata in un computer prende il nome di **dato**.

Il **dato** viene tenuto in una **struttura dati**
della quale conosciamo il **tipo**,
il **nome**,
il **contenuto**
ed il suo **indirizzo**
in memoria centrale o in memoria di massa

DATO

Classifichiamo i dati in: dati di **input**, dati di **output** e dati **interni**

ALESSANDRO URSOMANDO

LA PROGRAMMAZIONE STRUTTURATA → STRUTTURE DATI
CC BY

INFORMAZIONE E DATO

Un'**informazione** archiviata in un computer prende il nome di **dato**.

Il **dato** viene tenuto in una **struttura dati**
della quale conosciamo il **tipo**,
il **nome**,
il **contenuto**
ed il suo **indirizzo**
in memoria centrale o in memoria di massa

DATO

Classifichiamo i dati in: dati di **input**, dati di **output** e dati **interni**

ALESSANDRO URSOMANDO

INFORMAZIONE E DATO

Un'**informazione** archiviata in un computer prende il nome di **dato**.

Il **dato** viene tenuto in una **struttura dati**
della quale conosciamo il **tipo**,
il **nome**,
il **contenuto**
ed il suo **indirizzo**
in memoria centrale o in memoria di massa

DATO

Classifichiamo i dati in: dati di **input**, dati di **output** e dati **interni**

Classifichiamo le strutture dati come **semplici** (o **scalari**),
strutture dati **complesse** (più dati sotto un solo nome),
strutture dati **su memoria di massa** (i file)
strutture dati **dinamiche** (pile, code, alberi e grafi)

STRUTTURE DATI SEMPLICI (O SCALARI) LE VARIABILI

Una variabile è un'area della RAM riservata per un uso specifico. Ci si riferisce ad essa mediante un **nome**; essa ha *sempre* un **valore**; i valori ammessi sono tutti dello stesso **tipo**.

VARIABILE

Dichiarare una variabile vuol dire riservare lo spazio in memoria.
Ogni variabile viene dichiarata una volta sola.

L'**assegnazione** è la procedura mediante la quale si definisce il valore della variabile. Il valore di una variabile di solito cambia diverse volte.

STRUTTURE DATI SEMPLICI (O SCALARI) LE COSTANTI

Una variabile è un'area della RAM riservata per un uso specifico. Ci si riferisce ad essa mediante un **nome**; essa ha *sempre* un **valore**; i valori ammessi sono tutti dello stesso **tipo**.

VARIABILE

Una costante è un'area della RAM con un **nome**, un **valore** ed un **tipo**. Differisce dalla variabile perché una volta assegnatole un valore questo non cambia *mai più*.

COSTANTE

ALESSANDRO URSOMANDO

STRUTTURE DATI COMPLESSE

Le strutture dati complesse sono un unico oggetto che contiene più valori

I valori possono essere **omogenei** (cioè dello stesso tipo)..

..o di tipo diverso

ALESSANDRO URSOMANDO

STRUTTURE DATI COMPLESSE

Le strutture dati complesse sono un unico oggetto che contiene più valori

I valori possono essere **omogenei** (cioè dello stesso tipo)..

..o di tipo diverso

Le strutture dati complesse hanno un nome collettivo

Per accedere alle singole celle si usano indici numerici o etichette tra parentesi quadre o anche la **dot notation**

Le strutture complesse possono essere bidimenisionali..

ALESSANDRO URSOMANDO

LA PROGRAMMAZIONE STRUTTURATA → STRUTTURE DATI
CC BY

STRUTTURE DATI COMPLESSE

Le strutture dati complesse sono un unico oggetto che contiene più valori

I valori possono essere **omogenei** (cioè dello stesso tipo)..

..o di tipo diverso

Le strutture dati complesse hanno un nome collettivo

Per accedere alle singole celle si usano indici numerici o etichette tra parentesi quadre o anche la **dot notation**

Le strutture complesse possono essere bidimenisionali..

tridimenisionali o n-dimensionali

ALESSANDRO URSOMANDO

FILE

I file sono l'unico modo per archiviare informazioni in modo definitivo (quindi su memoria di massa e non in RAM).

I linguaggi di programmazione di alto livello forniscono il modo di interagire con i file (che sono gestiti dal sistema operativo): sarà quindi possibile creare, aprire in lettura, aprire in lettura/scrittura e chiudere i file.

FILE

ALESSANDRO URSOMANDO

PUNTATORI

Un **puntatore** è una variabile il cui contenuto è **l'indirizzo** di un'altra variabile.

INDIRIZZO

Scaffale 11 – Posto 8

ALESSANDRO URSOMANDO

LA PROGRAMMAZIONE STRUTTURATA

Strutture di controllo del flusso: la sequenza

PASSI ELEMENTARI

Un passo elementare può servire a:

- ❑ Presentare un output
- ❑ Modificare il valore di una struttura dati
- ❑ Acquisire un input
- ❑ Dirigere il flusso di esecuzione
- ❑ Invocare una sub-routine

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

In qualsiasi linguaggio di programmazione
ci sono queste **e solo queste** tipologie di istruzione

Se impariamo a gestire
queste tipologie di istruzione
poi sapremo programmare in qualsiasi linguaggio

ISTRUZIONI DI OUTPUT

Un passo elementare può servire a:

- ❑ **Presentare un output**
- ❑ Modificare il valore di una struttura dati
- ❑ Acquisire un input
- ❑ Dirigere il flusso di esecuzione
- ❑ Invocare una sub-routine

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Ogni linguaggio ha una (o più) istruzioni di output

```
System.out.print("Ciao mondo");
```

```
System.out.println("Ciao mondo");
```


MODIFICARE IL VALORE DI UNA STRUTTURA DATI

Un passo elementare può servire a:

- ❑ Presentare un output
- ❑ **Modificare il valore di una struttura dati**
- ❑ Acquisire un input
- ❑ Dirigere il flusso di esecuzione
- ❑ Invocare una sub-routine

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Ogni linguaggio ha una istruzione di assegnazione

```
<nome_variabile> = "Alessandro";
```

ISTRUZIONI DI INPUT

Un passo elementare può servire a:

- ❑ Presentare un output
- ❑ Modificare il valore di una struttura dati
- ❑ **Acquisire un input**
- ❑ Dirigere il flusso di esecuzione
- ❑ Invocare una sub-routine

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Ogni linguaggio ha una (o più) istruzioni di input

In Java vi darò io un file..

STRUTTURE DI CONTROLLO DEL FLUSSO SEQUENZA

Un passo elementare può servire a:

- ❑ **Presentare un output**
- ❑ **Modificare il valore di una struttura dati**
- ❑ **Acquisire un input**
- ❑ Dirigere il flusso di esecuzione
- ❑ Invocare una sub-routine

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

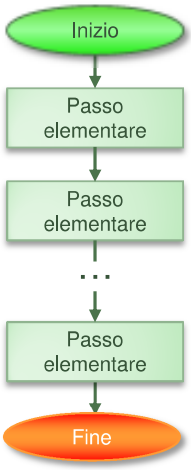
Con queste sole istruzioni siamo già pronti a fare dei piccoli programmini con la struttura di controllo del flusso più semplice: la **sequenza**.

STRUTTURE DI CONTROLLO DEL FLUSSO
SEQUENZA

La **sequenza** è la struttura di controllo più banale.

I passi vengono eseguiti uno dopo l'altro incondizionatamente

SEQUENZA



Vediamo qualche **flow chart** di esempio.

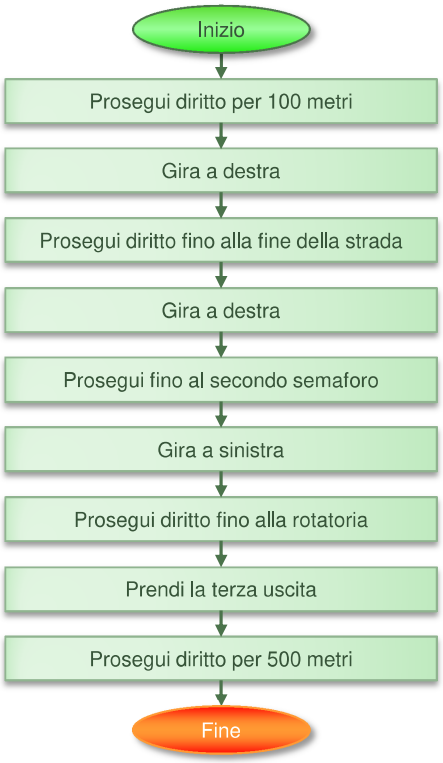
ALESSANDRO URSOMANDO

LA PROGRAMMAZIONE STRUTTURATA → CONTROLLO DEL FLUSSO
CC BY

STRUTTURE DI CONTROLLO DEL FLUSSO
SEQUENZA

Chiedere informazioni.

PROBLEMA DEL MONDO REALE



ALESSANDRO URSOMANDO

STRUTTURE DI CONTROLLO DEL FLUSSO SEQUENZA

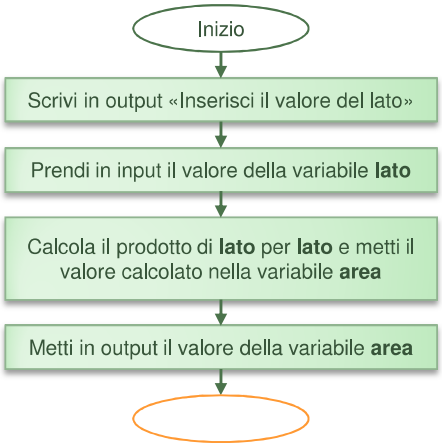
Svegliarsi.

PROBLEMA DEL MONDO REALE

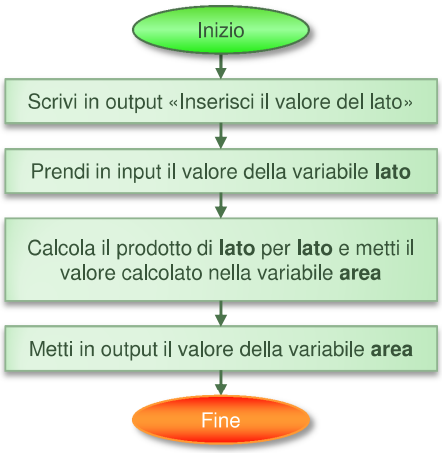
STRUTTURE DI CONTROLLO DEL FLUSSO SEQUENZA

Calcolare l'area di un quadrato.

PROBLEMA DI INFORMATICA



STRUTTURE DI CONTROLLO DEL FLUSSO
SEQUENZA



STRUTTURE DI CONTROLLO DEL FLUSSO
SEQUENZA

Vediamo come
una cosa del genere potrebbe
essere implementata in Java.

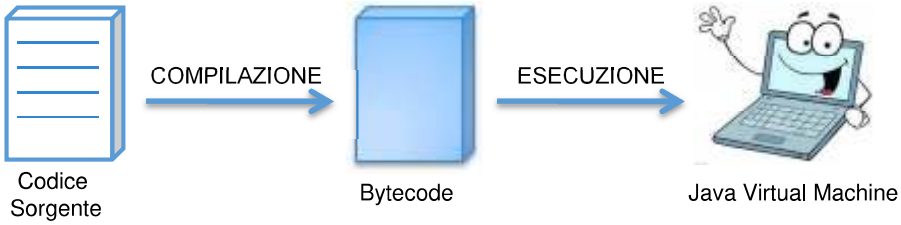
JAVA

Introduzione

INTRODUZIONE

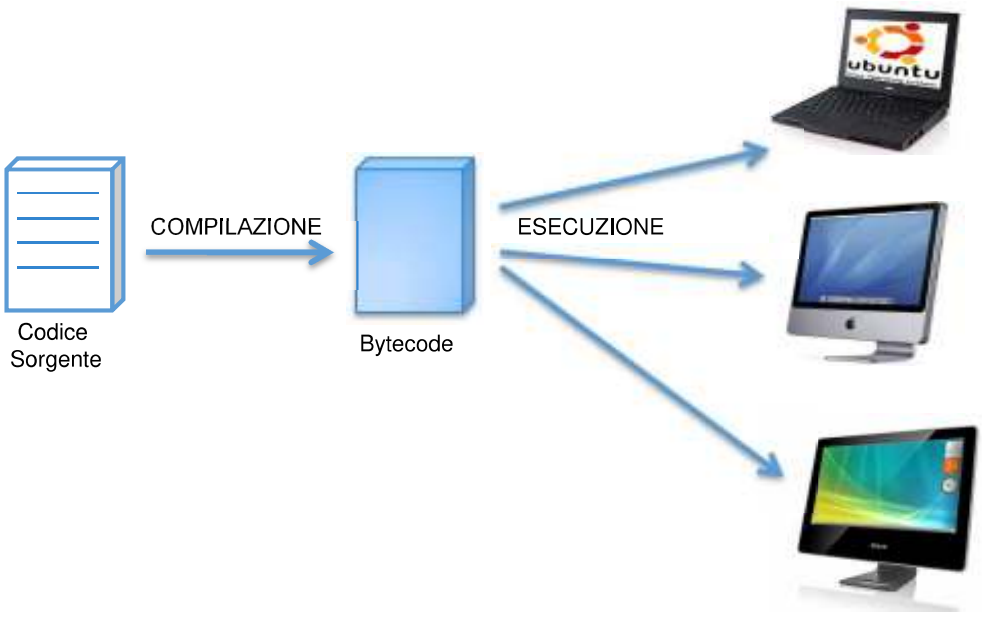
INTRODUZIONE

CROSS-PLATFORM



INTRODUZIONE

CROSS-PLATFORM



INTRODUZIONE

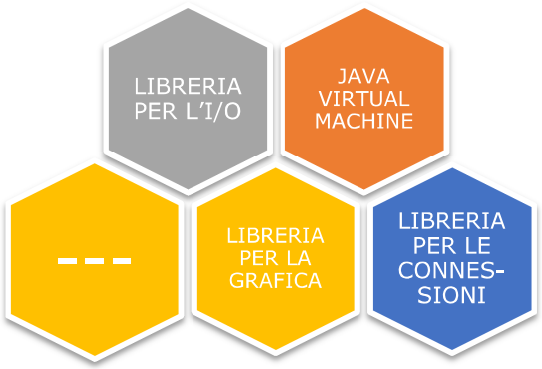
Cross Platform

Usi

Versioni e IDE

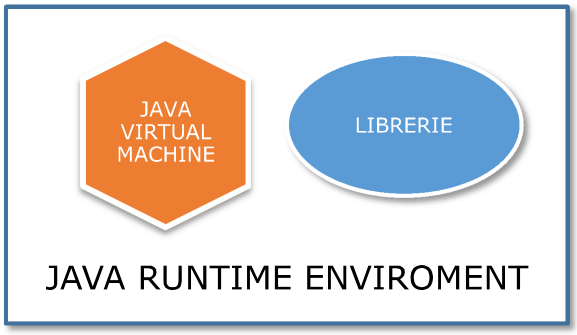
INTRODUZIONE

VERSIONI E IDE



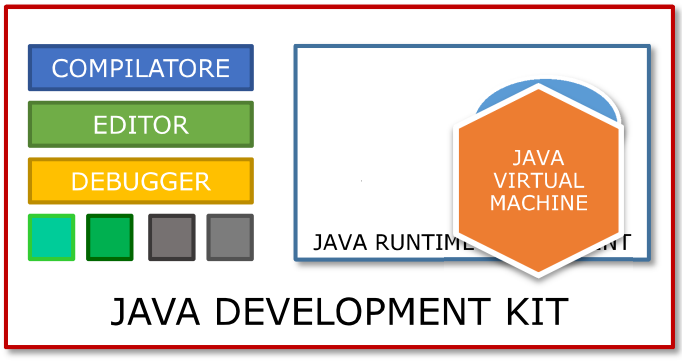
INTRODUZIONE

VERSIONI E IDE



INTRODUZIONE

VERSIONI E IDE



INTRODUZIONE

VERSIONI E IDE

- IDE per sviluppare in Java
- Multi-Piattaforma
- Licenza GNU
- Kent University

BlueJ

JAVA

I primi esercizi

INTRODUZIONE

Abbiamo accennato alle istruzioni di output
e all'istruzione di assegnazione

```
System.out.print("Ciao mondo");
```

```
System.out.println("Ciao mondo");
```

```
<nome_variabile> = "Alessandro";
```

Per l'input,
una volta incluso nel progetto il file «Lettore.java»

```
int num = Lettore.leggiUnIntero();
```

```
String s = Lettore.leggiUnaStringa();
```

```
double num = Lettore.leggiUnDouble();
```

```
char c = Lettore.leggiUnCarattere();
```

Parliamo un attimo di **tipi di dati** e **operatori**
e siamo pronti a fare qualche esercizio

TIPI DI DATO

TIPO	BIT	VALORI
boolean	1	-
char	16	unicode [0; 2^{16}-1]
byte	8	[-128; 127]
short	16	[-2^{15}; 2^{15}-1]
int	32	[-2^{31}; 2^{31}-1]
long	64	[-2^{63}; 2^{63}-1]
float	32	precisione singola
double	64	precisione doppia

TIPI DI DATO

TIPO	BIT	VALORI
boolean	1	-
char	16	unicode [0; 2^{16}-1]
byte	8	[-128; 127]
short	16	[-2^{15}; 2^{15}-1]
int	32	[-2^{31}; 2^{31}-1]
long	64	[-2^{63}; 2^{63}-1]
float	32	precisione singola
double	64	precisione doppia

circa 65 mila caratteri diversi

TIPI DI DATO

TIPO	BIT	VALORI
boolean	1	-
char	16	unicode [0; 2^{16}-1]
byte	8	[-128; 127]
short	16	[-2^{15}; 2^{15}-1]
int	32	[-2^{31}; 2^{31}-1]
long	64	[-2^{63}; 2^{63}-1]
float	32	precisione singola
double	64	precisione doppia

NUMERI INTERI

un unsigned byte assume valori in [0; 255]

TIPI DI DATO

TIPO	BIT	VALORI
boolean	1	-
char	16	unicode [0; 2^{16}-1]
byte	8	[-128; 127]
short	16	[-2^{15}; 2^{15}-1]
int	32	[-2^{31}; 2^{31}-1]
long	64	[-2^{63}; 2^{63}-1]
float	32	precisione singola
double	64	precisione doppia

NUMERI INTERI

2^{15} vale circa 32k

TIPI DI DATO

TIPO	BIT	VALORI
boolean	1	-
char	16	unicode [0; 2^{16}-1]
byte	8	[-128; 127]
short	16	[-2^{15}; 2^{15}-1]
int	32	[-2^{31}; 2^{31}-1]
long	64	[-2^{63}; 2^{63}-1]
float	32	precisione singola
double	64	precisione doppia

NUMERI INTERI

2^{31} vale circa 2 miliardi

TIPI DI DATO

TIPO	BIT	VALORI
boolean	1	-
char	16	unicode [0; 2^{16}-1]
byte	8	[-128; 127]
short	16	[-2^{15}; 2^{15}-1]
int	32	[-2^{31}; 2^{31}-1]
long	64	[-2^{63}; 2^{63}-1]
float	32	precisione singola
double	64	precisione doppia

NUMERI INTERI

2^{63} è un numero a 19 cifre!!

TIPI DI DATO

TIPO	BIT	VALORI
boolean	1	-
char	16	unicode [0; 2^{16}-1]
byte	8	[-128; 127]
short	16	[-2^{15}; 2^{15}-1]
int	32	[-2^{31}; 2^{31}-1]
long	64	[-2^{63}; 2^{63}-1]
float	32	precisione singola
double	64	precisione doppia

NUMERI RAZIONALI

da -2 milioni a +2 milioni (circa)

TIPI DI DATO

TIPO	BIT	VALORI
boolean	1	-
char	16	unicode [0; 2^{16}-1]
byte	8	[-128; 127]
short	16	[-2^{15}; 2^{15}-1]
int	32	[-2^{31}; 2^{31}-1]
long	64	[-2^{63}; 2^{63}-1]
float	32	precisione singola
double	64	precisione doppia

NUMERI RAZIONALI

da
-2 miliardi
a
+2 miliardi
(circa)

OPERATORI

OPERATORI	SIGNIFICATO
/*..*/ /**..*/ //..	Commenti
=	Assegnazione
+ - * / %	Aritmetici
+	Concatenazione
+= -= *= /= %=	Assegnazione combinati
++ --	Auto-incremento / decremento
== != > >= < <=	Confronto
&& || !	Logici

(LABORATORIO)

- Hello world!
- Calcoli banali
- Cifre decimali
- Input di valori di vario tipo

LA PROGRAMMAZIONE STRUTTURATA

Strutture di controllo del flusso: la selezione

DIRIGERE IL FLUSSO DI ESECUZIONE

Un passo elementare può servire a:

- ❑ Presentare un output
- ❑ Modificare il valore di una struttura dati
- ❑ Acquisire un input
- ❑ **Dirigere il flusso di esecuzione**
- ❑ Invocare una sub-routine

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Qualsiasi algoritmo, anche quelli del mondo reale, per adattarsi alle circostanze potrebbe **non** richiedere l'esecuzione in sequenza di tutti i suoi passi elementari.

ALESSANDRO URSOMANDO

STRUTTURE DI CONTROLLO DEL FLUSSO SELEZIONE

Lavare il bucato con la lavatrice.

PROBLEMA

1. Se l'oblò è chiuso, aprilo
2. Metti i panni nel cestello
3. Chiudi l'oblò
4. Accendi la lavatrice
5. Imposta il programma
6. Metti il detersivo
7. Metti l'ammorbidente
8. Avvia il ciclo

Qui, il passo uno è scomponibile in due passi, il secondo dei quali potrebbe non essere eseguito tutte le volte.

Questo è un esempio di **selezione**.

1. Se l'oblò è chiuso
2. apri l'oblò
3. Metti i panni nel cestello
4. Chiudi l'oblò
5. Accendi la lavatrice
6. Imposta il programma
7. Metti il detersivo
8. Metti l'ammorbidente
9. Avvia il ciclo

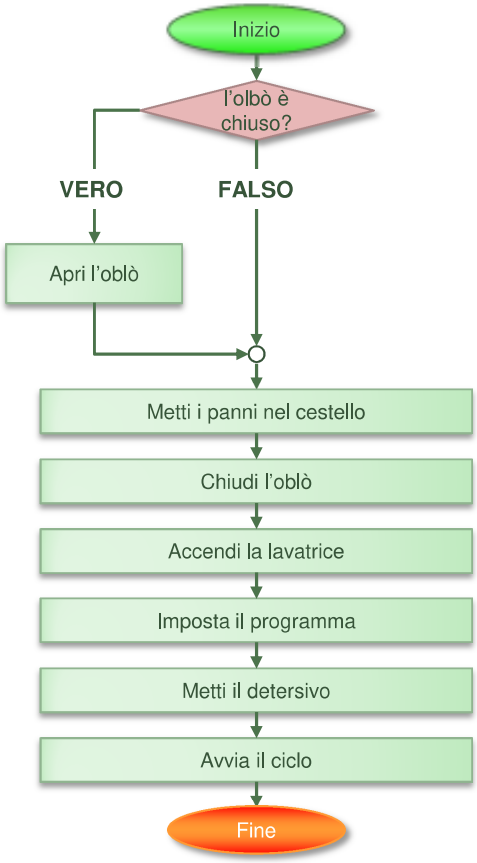
ALESSANDRO URSOMANDO

STRUTTURE DI CONTROLLO DEL FLUSSO
SELEZIONE

Lavare il bucato con la lavatrice.

PROBLEMA

Questo è il caso più semplice di selezione: la **selezione a una via**.

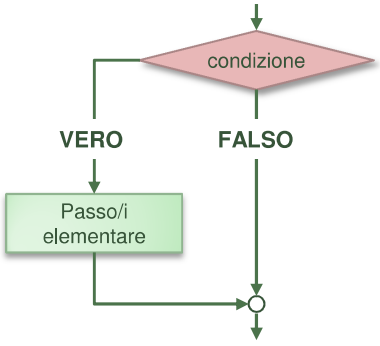


ALESSANDRO URSOMANDO

STRUTTURE DI CONTROLLO DEL FLUSSO
SELEZIONE A UNA VIA

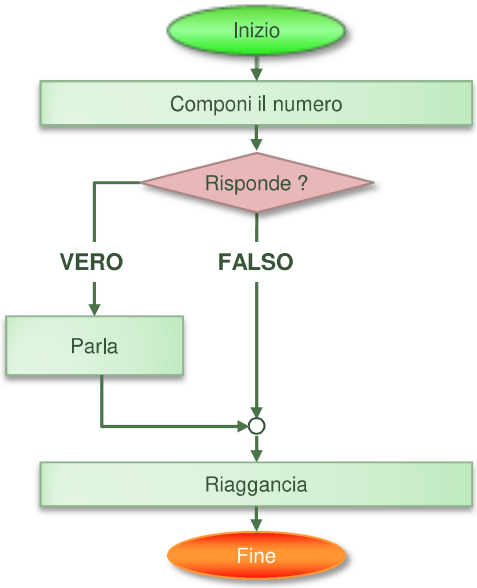
I passi vengono eseguiti solo dopo avere verificato una condizione, dopodiché ci si ricongiunge al flusso principale

SELEZIONE



ALESSANDRO URSOMANDO

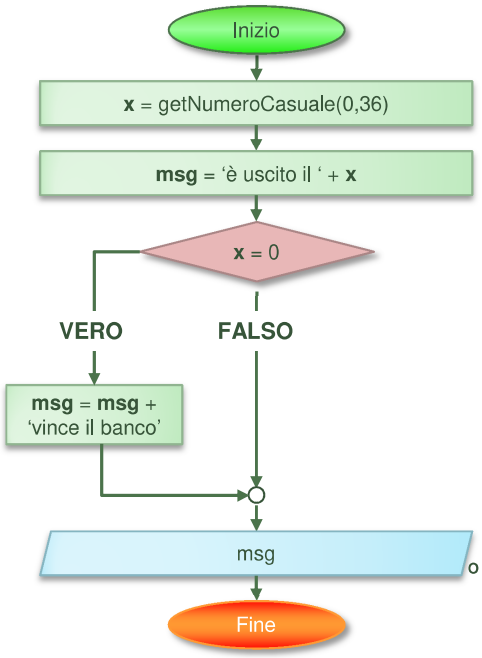
STRUTTURE DI CONTROLLO DEL FLUSSO
SELEZIONE A UNA VIA



Fare una telefonata.

PROBLEMA DEL MONDO REALE

STRUTTURE DI CONTROLLO DEL FLUSSO
SELEZIONE A UNA VIA



Simulare una sessione di gioco alla roulette.

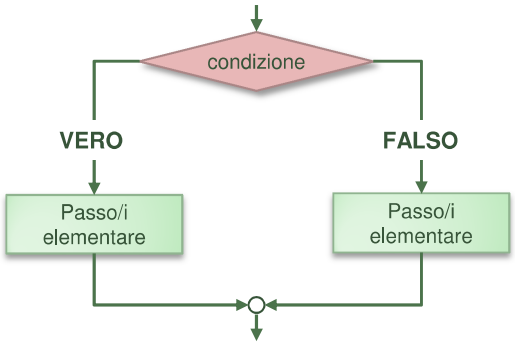
PROBLEMA DI INFORMATICA

STRUTTURE DI CONTROLLO DEL FLUSSO
SELEZIONE A DUE VIE

La **selezione a due vie** prevede delle operazioni da svolgere sia nel caso in cui la condizione risulti vera, sia nel caso risulti falsa.

I passi vengono eseguiti solo dopo avere verificato una condizione, dopodiché ci si ricongiunge al flusso principale

SELEZIONE

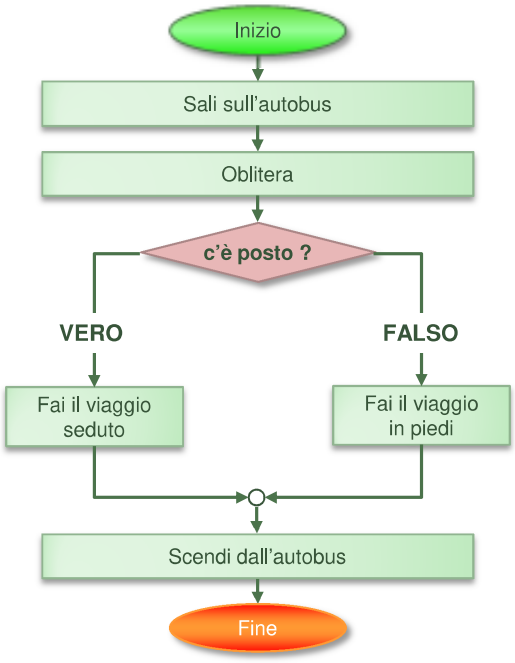


ALESSANDRO URSOMANDO

STRUTTURE DI CONTROLLO DEL FLUSSO
SELEZIONE A DUE VIE

Fare un viaggio in autobus.

PROBLEMA DEL MONDO REALE

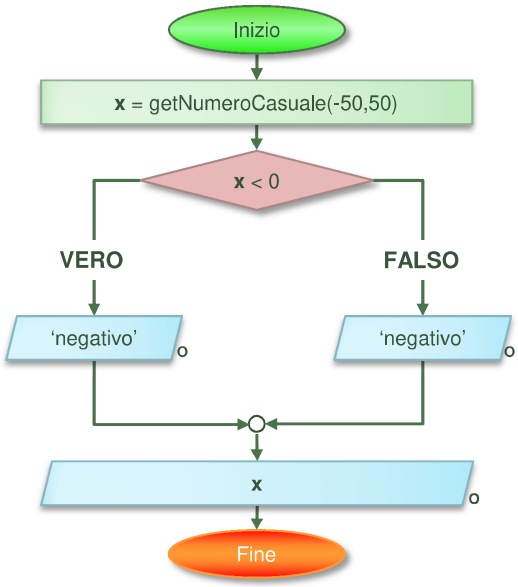


ALESSANDRO URSOMANDO

STRUTTURE DI CONTROLLO DEL FLUSSO
SELEZIONE A DUE VIE

Riconoscere il segno di un numero casuale.

PROBLEMA DI INFORMATICA

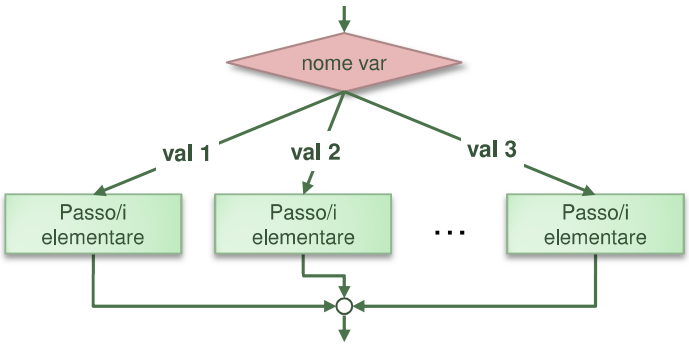


SELEZIONE

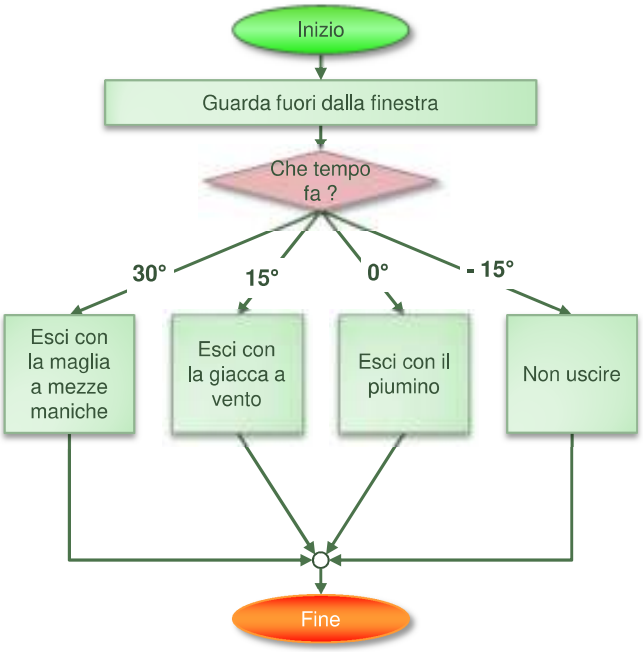
La **selezione ad n vie** prevede delle operazioni da svolgere in diversi casi.

I passi vengono eseguiti solo dopo avere verificato una condizione

SELEZIONE



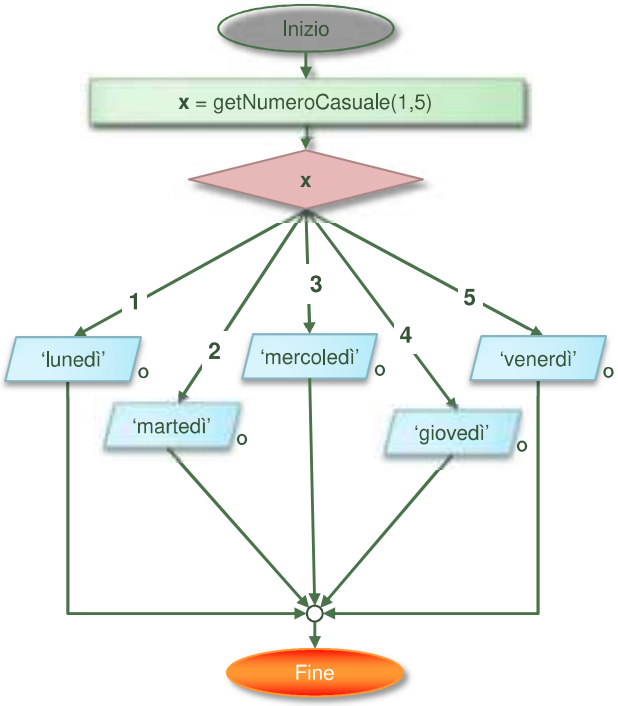
SELEZIONE AD N VIE



Decidere cosa indossare.

PROBLEMA DEL MONDO REALE

SELEZIONE AD N VIE



Estrarre un giorno lavorativo a caso.

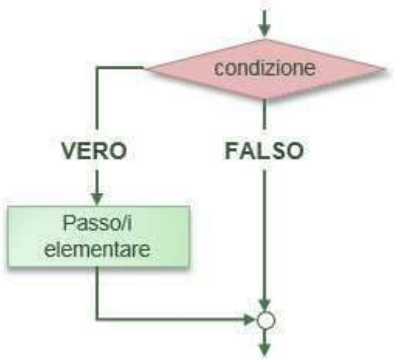
PROBLEMA DI INFORMATICA

JAVA

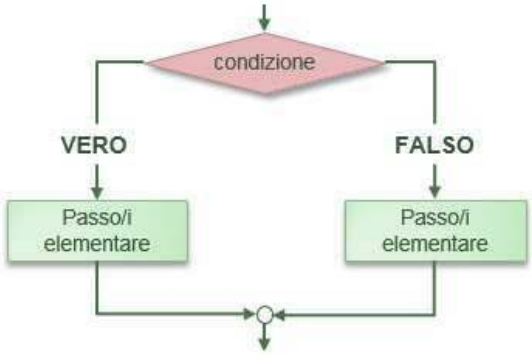
Codifica di istruzione condizionali

ISTRUZIONI CONDIZIONALI

```
if (<condizione>){
    // istruzioni
}
```



ISTRUZIONI CONDIZIONALI

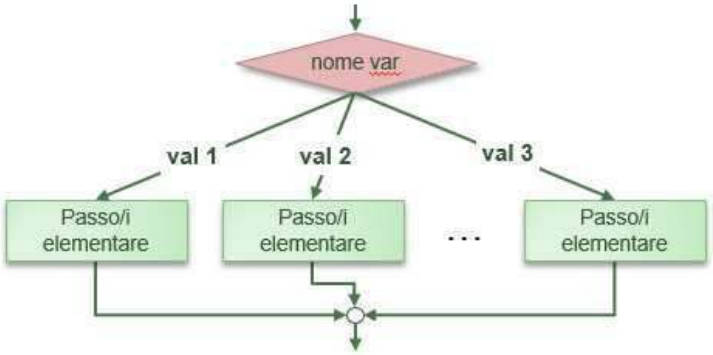



```
if (<condizione>){
    // istruzioni
}
```

```
if (<condizione>){
    // istruzioni
} else {
    // istruzioni
}
```

```
if (<condizione>){
    // istruzioni
} else if (<condizione>){
    // istruzioni
} else {
    // istruzioni
}
```


ISTRUZIONI CONDIZIONALI



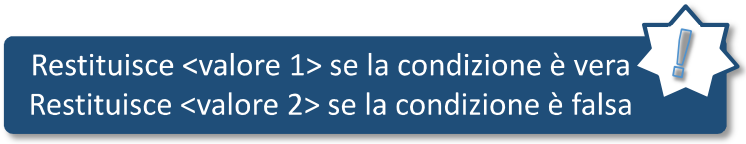

```
if (<condizione>){
    // istruzioni
}
```

```
if (<condizione>){
    // istruzioni
} else {
    // istruzioni
}
```

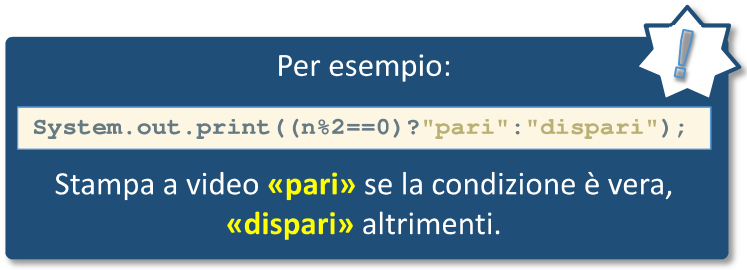
```
if (<condizione>){
    // istruzioni
} else if (<condizione>){
    // istruzioni
} else {
    // istruzioni
}
```

```
switch (<espressione>){
   case (<valore 1>):
        // istruzioni
        // break (opzionale)
   case (<valore 2>):
        // istruzioni
        // break (opzionale)
   // altri case
   // calusola opzionale
   default:
        // istruzioni
}
```

OPERATORE TERNARIO




```
(<condizione>)?(<valore 1>):(<valore 2>)
```



(LABORATORIO)

- Esercizi con condizioni semplici
- Esercizi con condizioni avanzate
- Le meraviglie dello switch

LA PROGRAMMAZIONE STRUTTURATA

Strutture di controllo del flusso: iterazione (1/2)

DIRIGERE IL FLUSSO DI ESECUZIONE

Un passo elementare può servire a:

- ❑ Presentare un output
- ❑ Modificare il valore di una struttura dati
- ❑ Acquisire un input
- ❑ **Dirigere il flusso di esecuzione**
- ❑ Invocare una sub-routine

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Qualsiasi algoritmo, anche quelli del mondo reale, per adattarsi alle circostanze potrebbe **non** richiedere l'esecuzione in sequenza di tutti i suoi passi elementari.

STRUTTURE DI CONTROLLO DEL FLUSSO
ITERAZIONE

Preparare la crema pasticcera.

PROBLEMA

1. Riscalda 500 ml di latte
2. Lavora 6 tuorli con 150 gr di zucchero e 50 gr di farina
3. Versa il composto nella casseruola
4. Continua a mescolare finché la crema assume la densità desiderata.

Qui, il passo quattro prevede la ripetizione di una certa attività.

1. Riscalda 500 ml di latte
2. Lavora 6 tuorli con 150 gr di zucchero e 50 gr di farina
3. Versa il composto nella casseruola
4. Mescola (un giro completo)
5. Se la densità non ti soddisfa
6. torna al punto 4

Abbiamo fatto l'esperienza dell'**iterazione**.

ALESSANDRO URSOMANDO

ITERAZIONE

Sebbene il concetto di iterazione sia semplice qui la faccenda si complica perché esistono 3 tipi di cicli.

I passi elementari vengono eseguiti più volte

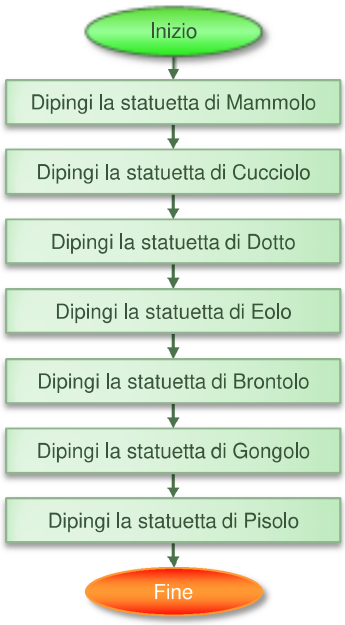
ITERAZIONE

La difficoltà sta nel capire in che circostanze è più opportuno usare l'uno o l'altro.

Cominciamo, come al solito, dal più semplice.

ALESSANDRO URSOMANDO

ITERAZIONE



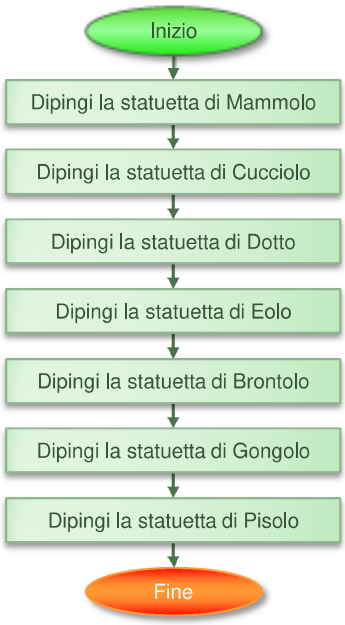
Il concetto di ciclo è stato introdotto per alleggerire situazioni come questa.

Dipingere le statuette dei nani

PROBLEMA DEL MONDO REALE

ALESSANDRO URSOMANDO

ITERAZIONE

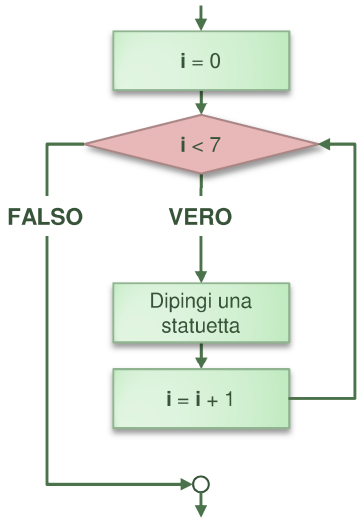


Il concetto di ciclo è stato introdotto per alleggerire situazioni come questa.

Una volta isolata l'istruzione (o le istruzioni) da eseguire **più volte** potremmo scrivere un algoritmo che semplicemente esegua quelle istruzioni (o quella istruzione) quante volte sono necessarie.

ALESSANDRO URSOMANDO

ITERAZIONE



Il concetto di ciclo è stato introdotto per alleggerire situazioni come questa.

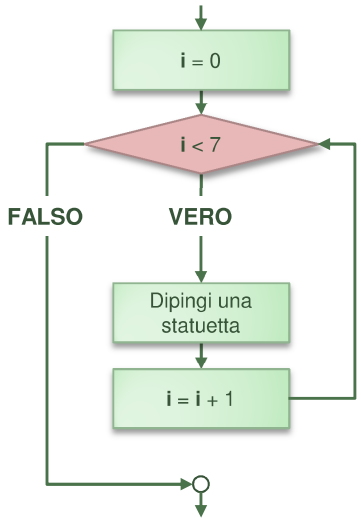
Una volta isolata l'istruzione (o le istruzioni) da eseguire **più volte** potremmo scrivere un algoritmo che semplicemente esegua quelle istruzioni (o quella istruzione) quante volte sono necessarie.

Per far ciò useremo una **variabile** con la quale contiamo quante volte abbiamo eseguito il ciclo.

Questo tipo di variabile viene denominata **contatore** e di solito il suo nome è **i**.

ALESSANDRO URSOMANDO

CICLO DI RIPETIZIONE SU CONTATORE



Il concetto di ciclo è stato introdotto per alleggerire situazioni come questa.

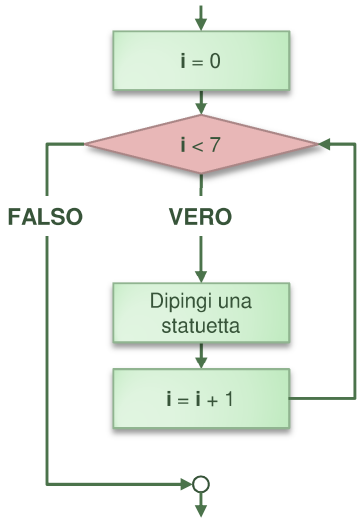
Una volta isolata l'istruzione (o le istruzioni) da eseguire **più volte** potremmo scrivere un algoritmo che semplicemente esegua quelle istruzioni (o quella istruzione) quante volte sono necessarie.

Per far ciò useremo una **variabile** con la quale contiamo quante volte abbiamo eseguito il ciclo.

Questo tipo di variabile viene denominata **contatore** e di solito il suo nome è **i**.

ALESSANDRO URSOMANDO

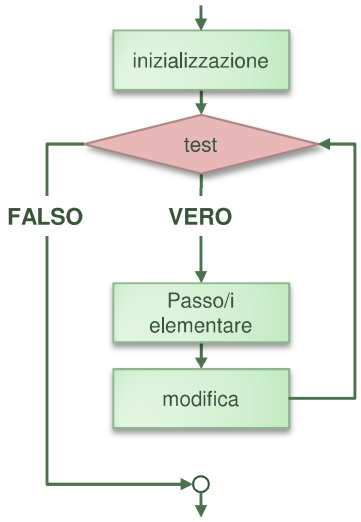
CICLO DI RIPETIZIONE SU CONTATORE



La variabile contatore viene quindi **inizializzata** prima di fare il ciclo, **testata** prima di ripetere il ciclo e **modificata** una volta eseguito il ciclo.

ALESSANDRO URSOMANDO

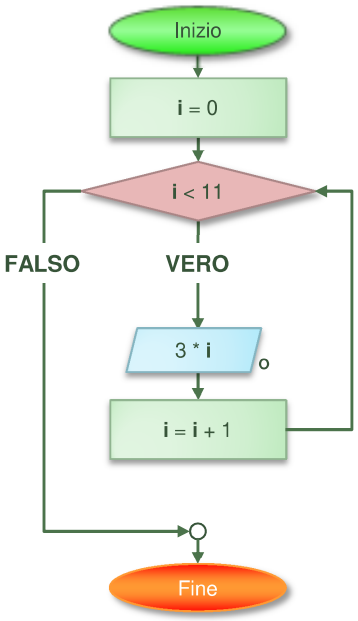
CICLO DI RIPETIZIONE SU CONTATORE



La variabile contatore viene quindi **inizializzata** prima di fare il ciclo, **testata** prima di ripetere il ciclo e **modificata** una volta eseguito il ciclo.

ALESSANDRO URSOMANDO

CICLO DI RIPETIZIONE SU CONTATORE



Stampare la tabellina del 3.

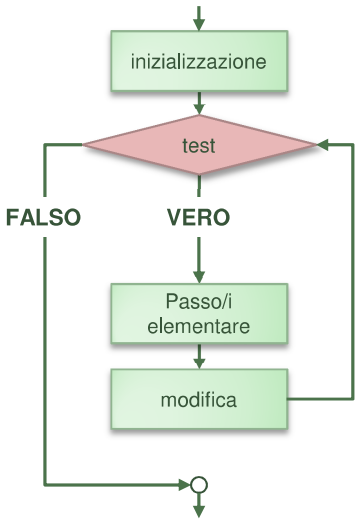
PROBLEMA DI INFORMATICA

ALESSANDRO URSOMANDO

JAVA

Codifica del ciclo di ripetizione su contatore

ISTRUZIONI ITERATIVA SU CONTATORE

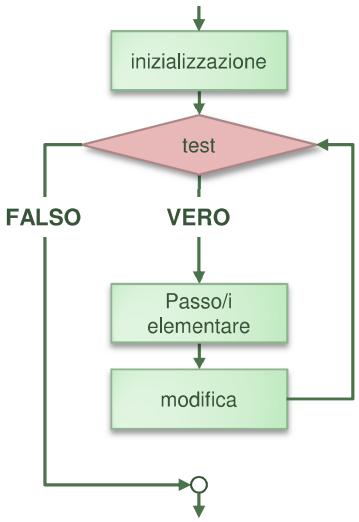



```
for (<start>; <test>; <modifica>){
    // passi elementari
}
```

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

Quanto vale la variabile contatore dopo il ciclo?

ISTRUZIONI ITERATIVA SU CONTATORE

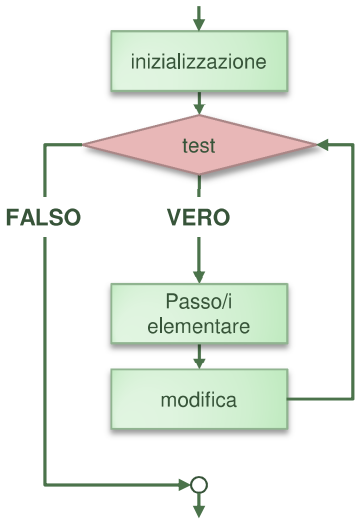



```
for (<start>; <test>; <modifica>){
    // passi elementari
}
```

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

Quante volte vengono eseguiti inizializzazione, test, modifica e blocco?

ISTRUZIONI ITERATIVA SU CONTATORE



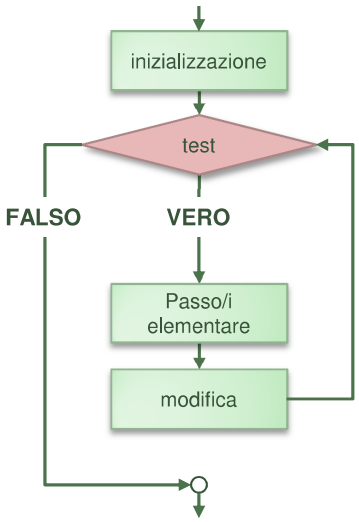

```
for (<start>; <test>; <modifica>){
    // passi elementari
}
```

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

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

Cosa succede qui?

ISTRUZIONI ITERATIVA SU CONTATORE




```
for (<start>; <test>; <modifica>){
    // passi elementari
}
```

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

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

```
for (int i=10; i>0; i++) {
    System.out.println(i);
}
```

E qui?

JAVA

Le stringhe

STRINGHE

Per le stringhe non esiste il tipo scalare ma solo il tipo oggetto..
Però è un oggetto speciale e lo potete usare come uno scalare.

```
String s = "Alessandro";
```

Sulla nostra stringa adesso possiamo fare varie operazioni.
Per esempio possiamo contare i caratteri:

```
int x = s.length();                    // x contiene 10
```

Possiamo isolare un carattere:

```
char c = s.charAt(0);                  // c contiene 'A'
```

Estrarre una sottostringa:

```
String t1 = s.substring(4);            // t1 contiene 'sandro'
String t2 = s.substring(4,6);          // t2 contiene 'san'
```

STRINGHE

Sempre con la stessa stringa di riferimento

```
String s = "Alessandro";
```

Verificare se contiene o meno una sottostringa

```
boolean b1 = s.contains("Ale");                    // b1 contiene true
boolean b2 = s.contains("Alex");                   // b2 contiene false
```

Verificare l'uguaglianza con un'altra stringa

```
boolean b1 = s.equals("Alessandro");               // b1 contiene true
boolean b2 = s.equals("Ursomando");                // b2 contiene false
```

Eliminare gli spazi prima e dopo

```
String t1 = "  Alessandro     ";
String t2 = t1.trim();                          // t2 contiene "Alessandro"
```


JAVA

Random

RANDOM

La funzione **Math.random()** restituisce un numero **reale** a caso nell'intervallo]0,1[.

```
double x = Math.random();      // un reale tra 0 e 1 (p.e. 0,56423)
```

Simbologia intervalli	
[a,b]	Estremi compresi
]a, b[	Estremi esclusi
[a,b[	Estremo iniziale incluso
]a,b]	Estremo finale incluso

RANDOM

La funzione **Math.random()** restituisce un numero **reale** a caso nell'intervallo]0,1[.

```
double x = Math.random();      // un reale tra 0 e 1 (p.e. 0,56423)
```

Ma noi (di certo) avremo bisogno di valori casuali di altro tipo.

ESEMPIO 1

Un colore del semaforo a caso.

ESEMPIO 2

Una temperatura terrestre a caso.

Come ottenere ciò che ci serve a partire da ciò che abbiamo?

RANDOM

ESEMPIO 1

Un colore del semaforo a caso.

Dovremo **mappare** ciascuno dei valori possibili (gli infiniti numeri reali in]0,1[ ) in uno dei valori desiderati (verde, giallo e rosso)

In questo caso basterà moltiplicare il valore casuale per 3 e trascurare la parte decimale: otterremo valori nell'intervallo [0,2] ai quali abbineremo i colori.

Vediamo qualche esempio.

x	x * 3	⌊ x * 3 ⌋	colore
0.012	0.036	0	rosso
0.350	1.050	1	giallo
0.412	1.236	1	giallo
0.781	2.343	2	verde
0.854	2.562	2	verde

RANDOM

La funzione **Math.random()** restituisce un numero **reale** a caso nell'intervallo]0,1[.

```
double x = Math.random();      // un reale tra 0 e 1 (p.e. 0,56423)
```

In generale, se vogliamo un valore a caso tra **k** valori possibili dobbiamo moltiplicare il valore casuale per **k** e trascurare la parte decimale.

```
double x = (int)(Math.random()*k);      // [0, k-1]
```

RANDOM

ESEMPIO 2

Una temperatura terrestre a caso.

Dovremo **mappare** ciascuno dei valori possibili (gli infiniti numeri reali in]0,1[ ) in uno dei valori desiderati [-98; +56].

Antartide - 4/07/2018: -98

Death valley - 10/07/1913: +56

In questo caso i valori desiderati sono 155 (98 negativi, lo zero e 56 positivi).

Moltiplicando per 155 il valore casuale distribuirò gli infiniti valori reali possibili in 155 "pacchetti" di valori la cui parte intera sarà tra 0 e 154.

Sommando il valore iniziale dell'intervallo otterrò valori all'interno dell'intervallo.

x	x * 155	⌊ x * 155 ⌋	⌊ x * 155 ⌋ - 98
0.012	1.86	1	-97
0.350	54,25	54	-44
0.412	63,86	63	-35
0.781	121,055	121	23
0.854	132,37	132	34

RANDOM

La funzione **Math.random()** restituisce un numero **reale** a caso nell'intervallo]0,1[.

```
double x = Math.random();      // un reale tra 0 e 1 (p.e. 0,56423)
```

In generale, se vogliamo un valore a caso tra **k** valori possibili dobbiamo moltiplica il valore casuale per **k** e trascurare la parte decimale.

```
double x = (int)(Math.random()*k);      / [0, k-1]
```

Concludendo, per ottenere un intero nell'intervallo **[a,b]** di dimensione **k**: moltiplico il valore casuale per **k**, trascuro la parte decimale e sommo **a**.

```
double x = (int)(Math.random()*k) + a;  // [a, b]
```

(LABORATORIO)

- Esercizi con i numeri casuali
  - Prodotto un numero casuale a 3 cifre verificare se è palindromo
  - Produrre e stampare un numero casuale palindromo a 5 cifre

(LABORATORIO)

- Esercizi di base per il for
  - Acquisito un intero n produrre e stampare un numero casuale a n cifre
- Esercizi per le stringhe
  - Verificare se una stringa avuta in input è palindroma controllando carattere per carattere (NOTE: ricordarsi di eliminare eventuali spazi bianchi a inizio/fine stringa).
  - Verificare se una stringa avuta in input è palindroma producendo la stringa al contrario e verificando l'uguaglianza con la stringa di partenza (NOTE: ricordarsi di eliminare eventuali spazi bianchi a inizio/fine stringa).
  - Avuta una stringa in input, produrre e stampare la stessa stringa criptata secondo il metodo che segue. Convertire ogni lettera con la cifra più simile (la «I» diventa 1, la «A» diventa 4, la «E» diventa 3m, la «O» diventa 0.

LA PROGRAMMAZIONE STRUTTURATA

Strutture di controllo del flusso: iterazione (2/2)

DIRIGERE IL FLUSSO DI ESECUZIONE

Un passo elementare può servire a:

- ❑ Presentare un output
- ❑ Modificare il valore di una struttura dati
- ❑ Acquisire un input
- ❑ **Dirigere il flusso di esecuzione**
- ❑ Invocare una sub-routine

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

Abbiamo già visto il **ciclo di ripetizione su contatore**, vediamo quello su **condizione**.

CICLO DI RIPETIZIONE SU CONDIZIONE

Se non sappiamo a priori **quante** volte dobbiamo eseguire il ciclo, sicuramente sappiamo **quando** eseguirlo e **quando** no.

I passi elementari vengono eseguiti più volte

ITERAZIONE

Anche questa volta ci sono due casi.

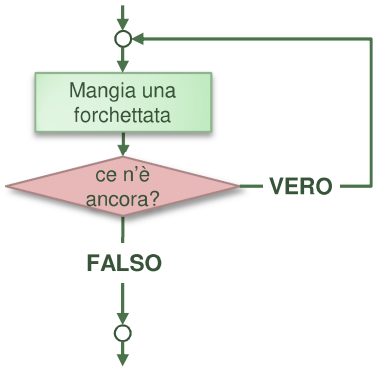
Prima eseguiamo il ciclo e poi facciamo il test? Oppure prima facciamo il test e poi eseguiamo il ciclo ?

ALESSANDRO URSOMANDO

CICLO DI RIPETIZIONE SU CONDIZIONE

Mangiare un piatto di spaghetti.

PROBLEMA DEL MONDO REALE

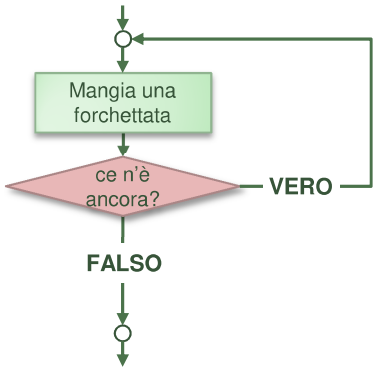


ALESSANDRO URSOMANDO

CICLO DI RIPETIZIONE SU CONDIZIONE CON CONDIZIONE IN CODA

Mangiare un piatto di spaghetti.

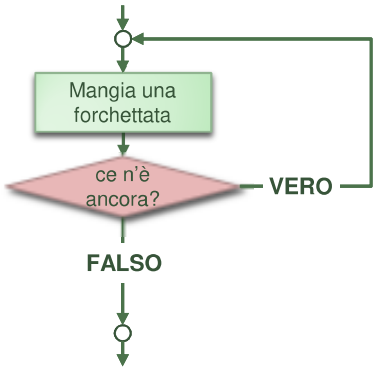
PROBLEMA DEL MONDO REALE



CICLO DI RIPETIZIONE SU CONDIZIONE CON CONDIZIONE IN CODA

Il ciclo su condizione con verifica della condizione in coda si presta in due casi:

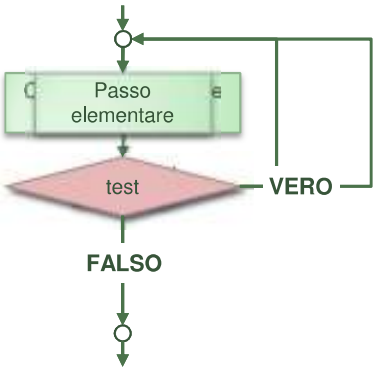
- Quando ha senso porsi la domanda **solo dopo** avere già eseguito una prima volta il ciclo (come per gli spaghetti).
- Quando c'è un obiettivo da raggiungere mediante uno o più **tentativi** (vediamo un esempio).



CICLO DI RIPETIZIONE SU CONDIZIONE CON CONDIZIONE IN CODA

Chiamare indietro qualcuno.

PROBLEMA DEL MONDO REALE



CICLO DI RIPETIZIONE SU CONDIZIONE

Quando ha senso porsi la domanda **prima** di eseguire la prima volta il ciclo..

I passi elementari vengono eseguiti più volte

ITERAZIONE

..allora si pone la verifica della condizione in testa.

Questo tipo di ciclo si chiama **ciclo di ripetizione su condizione con verifica della condizione in testa**.

CICLO DI RIPETIZIONE SU CONDIZIONE CON CONDIZIONE IN TESTA

Quando ha senso porsi la domanda **prima** di eseguire la prima volta il ciclo..

I passi elementari vengono eseguiti più volte

ITERAZIONE

..allora si pone la verifica della condizione in testa.

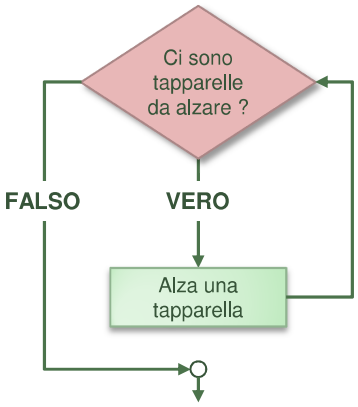
Questo tipo di ciclo si chiama **ciclo di ripetizione su condizione con verifica della condizione in testa**.

ALESSANDRO URSOMANDO

CICLO DI RIPETIZIONE SU CONDIZIONE CON CONDIZIONE IN TESTA

Alzare tutte le tapparelle di casa
(considerando che potrebbero essere già tutte aperte)

PROBLEMA DEL MONDO REALE

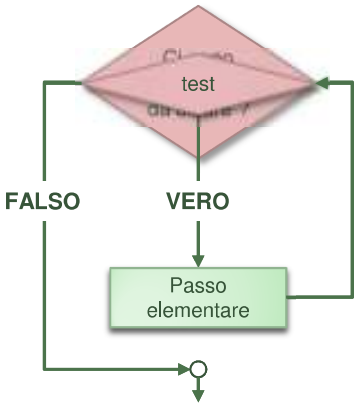


ALESSANDRO URSOMANDO

CICLO DI RIPETIZIONE SU CONDIZIONE CON CONDIZIONE IN TESTA

Alzare tutte le tapparelle di casa
(considerando che potrebbero essere già tutte aperte)

PROBLEMA DEL MONDO REALE



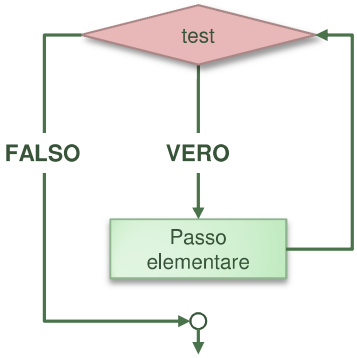
ALESSANDRO URSOMANDO

JAVA

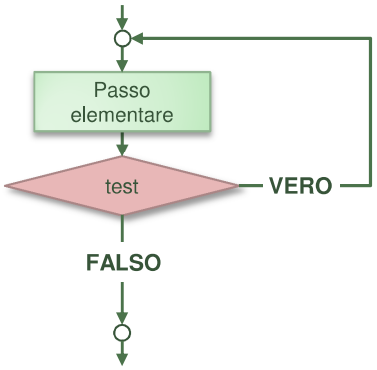
Codifica del ciclo di ripetizione su condizione

CICLO DI RIPETIZIONE SU CONDIZIONE

```
while (<test>) {
    // passi elementari
}
```




```
do {
    // passi elementari
} while (<test>)
```



In quale ciclo sono sicuro che almeno una volta il blocco viene sempre eseguito?

(LABORATORIO)

- Acquisire un valore numerico a tre cifre con il controllo dell'input
- Acquisito un numero intero, contare e stampare le cifre che compongono quel numero
- Acquisita una stringa di almeno dieci caratteri, stampare a video la prima parola contenuta in quella stringa (p.e. se la stringa è «Ciao mi chiamo Alessandro» l'output sarà «ciao»)
- Produrre un numero casuale a 3 cifre, con le tre cifre tutte diverse tra loro
- Produrre un anno bisestile casuale compreso tra il 1950 e 2050
- Produrre e stampare i primi 5 numeri primi maggiori di 100

JAVA

I vettori

I VETTORI

I vettori in Java sono una struttura dati **omogenea** e **statica**

La dimensione del vettore è stabilita durante la dichiarazione

Tutte le celle contengono valori dello stesso tipo

L'istruzione per dichiarare un vettore di 10 interi è questa:

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

L'accesso a una cella sia in lettura che in scrittura avviene così

```
vettore[i]
```

(LABORATORIO)

- Acquisita una stringa di almeno 5 parole e 10 lettere stampare a video l'occorrenza di ciascuna vocale presente nella stringa (100).
- Riempire un vettore con dieci numeri interi casuali a tre cifre. Successivamente stampare il vettore. Successivamente cercare il maggiore e stamparlo.(101)
- Riempire un vettore con dieci numeri interi casuali a due cifre. Successivamente stampare il vettore. Successivamente calcolare la media e stamparla (102)
- Riempire un vettore con i primi dieci numeri primi in [100,200]. Successivamente stampare il vettore. (103)

(LABORATORIO)

- Acquisisci 5 cifre (con il controllo dell'input) in un vettore di interi. Successivamente produci e stampa il numero rappresentato da quelle cinque cifre (attenzione alla prima cifra). (104)
- Acquisisci in input dieci nomi di gatto e dieci pesi, sistema i valori in due vettori paralleli. Successivamente stampa il nome del gatto più pesante.(105)
- Riempi un vettore con dieci interi a due cifre. Ordina e stampa il vettore. (106)

LA PROGRAMMAZIONE STRUTTURATA

Sviluppo top-down

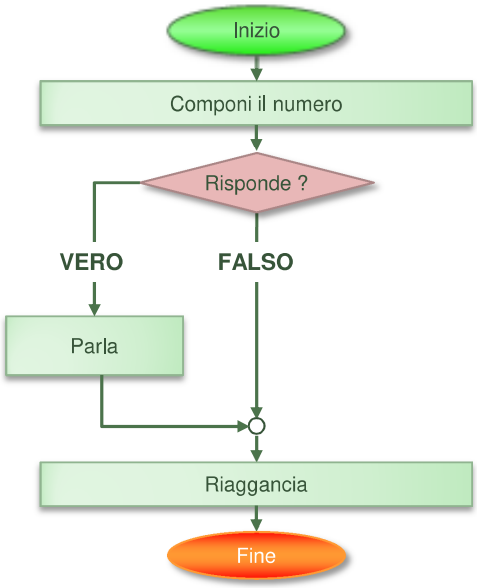
PASSI ELEMENTARI

Un passo elementare può servire a:

- ❑ Presentare un output
- ❑ Acquisire un input
- ❑ Modificare il valore di una struttura dati
- ❑ Dirigere il flusso di esecuzione
- ❑ **Invocare una sub-routine**

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare
13. passo elementare
14. passo elementare

INVOCARE UNA SUB-ROUTINE

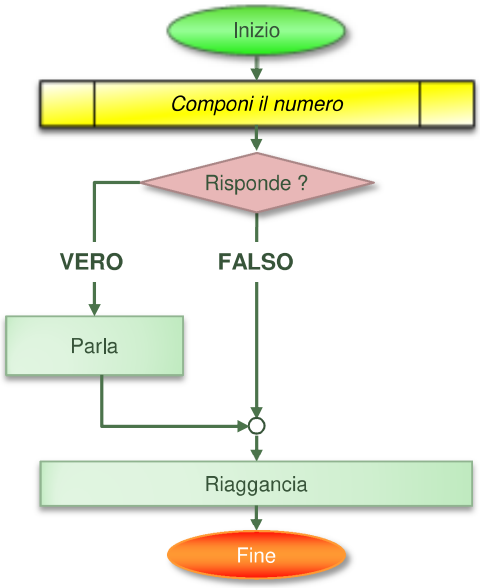


Fare una telefonata.

PROBLEMA DEL MONDO REALE

In realtà, il passo "Componi il numero" non è esattamente un passo elementare in quanto scomponibile.

INVOCARE UNA SUB-ROUTINE

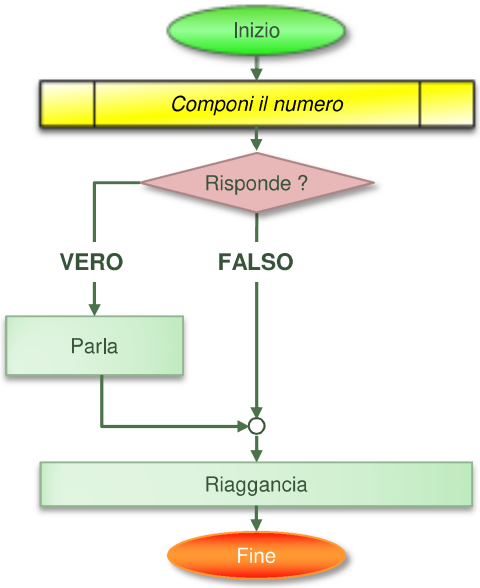


Fare una telefonata.

PROBLEMA DEL MONDO REALE

In realtà, il passo "Componi il numero" non è esattamente un passo elementare in quanto scomponibile.

INVOCARE UNA SUB-ROUTINE

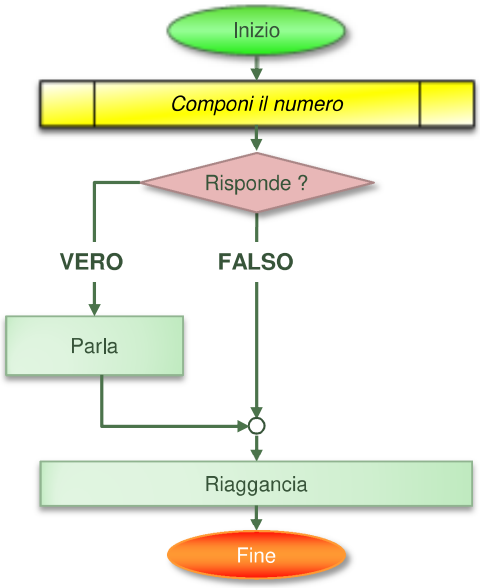


Fare una telefonata.

PROBLEMA DEL MONDO REALE

Creiamo dunque la sub-routine «Componi il numero»

INVOCARE UNA SUB-ROUTINE

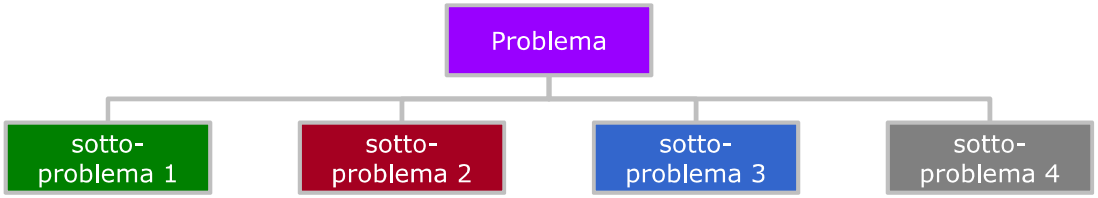


Questa strategia di sviluppo si chiama «top-down»

SVILUPPO TOP-DOWN

Per sviluppo **top-down** si intende un approccio alla soluzione di un problema che prevede l'individuazione di sotto-problemi (più piccoli e quindi più semplici da risolvere).

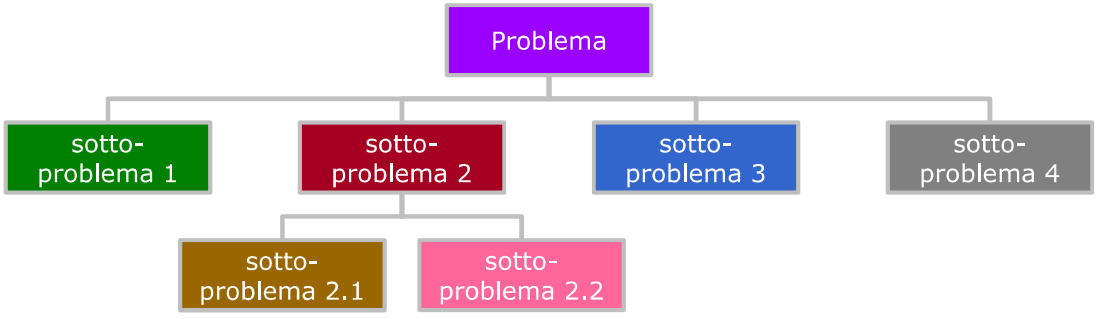
SVILUPPO TOP-DOWN



ALESSANDRO URSOMANDO

SVILUPPO TOP-DOWN

Ovviamente un sotto-problema
può essere a sua volta scomposto in sotto-problemi

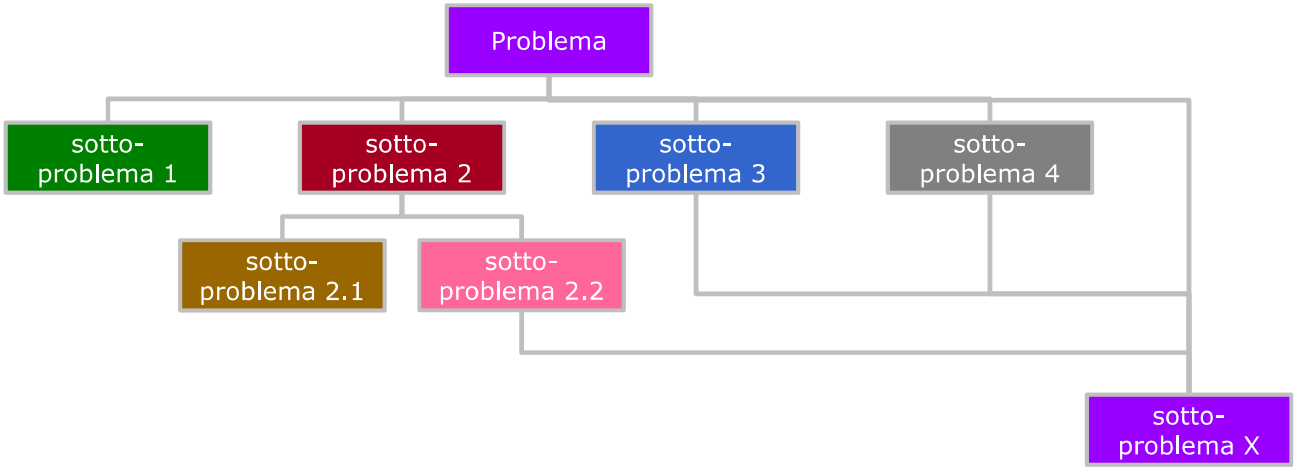


ALESSANDRO URSOMANDO

SVILUPPO TOP-DOWN

Ovviamente un sotto-problema
può essere a sua volta scomposto in sotto-problemi

Ovviamente un sotto-problema può ricorrere più di una volta



SVILUPPO TOP-DOWN

Evidentemente ogni **sotto-problema**
sarà risolto con la realizzazione di un **sotto-programma**.

Lo svolgimento di un compito ben preciso (sebbene piccolo)
può essere svolto in autonomia,
ma più spesso necessita di scambiare dei dati con chi si serve del sotto-programma.

Per esempio,
la composizione di un numero
richiede che si fornisca il numero da comporre!

SVILUPPO TOP-DOWN

Evidentemente ogni **sotto-problema** sarà risolto con la realizzazione di un **sotto-programma**.

Lo svolgimento di un compito ben preciso (sebbene piccolo) può essere svolto in autonomia, ma più spesso necessita di scambiare dei dati con chi si serve del sotto-programma.

I dati che un sotto-programma riceve in input da chi lo invoca si chiamano **parametri**.

Il dato che un sotto-programma fornisce in output a chi lo ha chiamato è il **valore restituito**.

ESEMPIO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare

1859. passo elementare
1860. passo elementare
1861. passo elementare
1862. passo elementare
1863. passo elementare
1864. passo elementare
1865. passo elementare

Supponiamo di avere un programma (che questo sia il programma principale o un sotto-programma non ha alcuna importanza)

Supponiamo poi di avere un sotto-programma qualsiasi.

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare

127. passo elementare
128. passo elementare
129. passo elementare
130. passo elementare
131. passo elementare

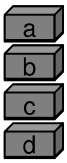
ESEMPIO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare
12. passo elementare

1859. passo elementare
1860. passo elementare
1861. passo elementare
1862. passo elementare
1863. passo elementare
1864. passo elementare
1865. passo elementare

Quando progettiamo un sotto-programma dobbiamo – tra l'altro – indicare i parametri di cui necessita (**parametri formali**).

Supponiamo che esso si riferisca a questi parametri con i nomi a, b ,c, d.



1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare

127. passo elementare
128. passo elementare
129. passo elementare
130. passo elementare
131. passo elementare

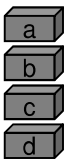
ESEMPIO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare

139. Invoca il sotto-programma qui a lato con i seguenti parametri: n1, n2, n3, n4 e metti il valore restituito nella variabile x.

1859. passo elementare
1860. passo elementare
1861. passo elementare
1862. passo elementare
1863. passo elementare
1864. passo elementare
1865. passo elementare

L'invocazione (da parte del programma chiamante) avviene indicando il nome del sotto-programma, fornendo i parametri (**parametri attuali**) con cui lavorare ed indicando cosa fare del valore restituito.



1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare

127. passo elementare
128. passo elementare
129. passo elementare
130. passo elementare
131. passo elementare

ESEMPIO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare

139. Invoca il sotto-programma qui a lato con i seguenti parametri: n1, n2, n3, n4 e metti il valore restituito nella variabile x.

1859. passo elementare
1860. passo elementare
1861. passo elementare
1862. passo elementare
1863. passo elementare
1864. passo elementare
1865. passo elementare

Osserviamo che i nomi di parametri attuali e formali **possono** anche **non** coincidere.

Deve **assolutamente** coincidere il tipo di ogni parametro. E l'ordine.

a
b
c
d

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare

127. passo elementare
128. passo elementare
129. passo elementare
130. passo elementare
131. passo elementare

ESEMPIO

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare

139. Invoca il sotto-programma qui a lato con i seguenti parametri: n1, n2, n3, n4 e metti il valore restituito nella variabile x.

1859. passo elementare
1860. passo elementare
1861. passo elementare
1862. passo elementare
1863. passo elementare
1864. passo elementare
1865. passo elementare

Il controllo del flusso passa così al sotto-programma..

n1
n2
n3
n4

a
b
c
d

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare

127. passo elementare
128. passo elementare
129. passo elementare
130. passo elementare
131. passo elementare

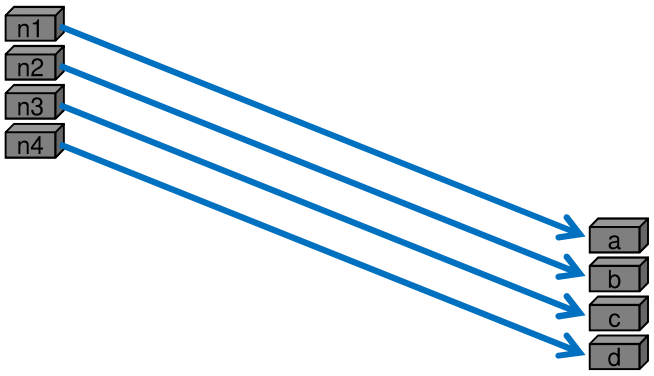
ESEMPIO

..nel cui parametro a è stato **copiato** il valore del parametro n1 e così via.

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare

139. Invoca il sotto-programma qui a lato con i seguenti parametri: n1, n2, n3, n4 e metti il valore restituito nella variabile x.

1859. passo elementare
1860. passo elementare
1861. passo elementare
1862. passo elementare
1863. passo elementare
1864. passo elementare
1865. passo elementare



1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare

127. passo elementare
128. passo elementare
129. passo elementare
130. passo elementare
131. passo elementare

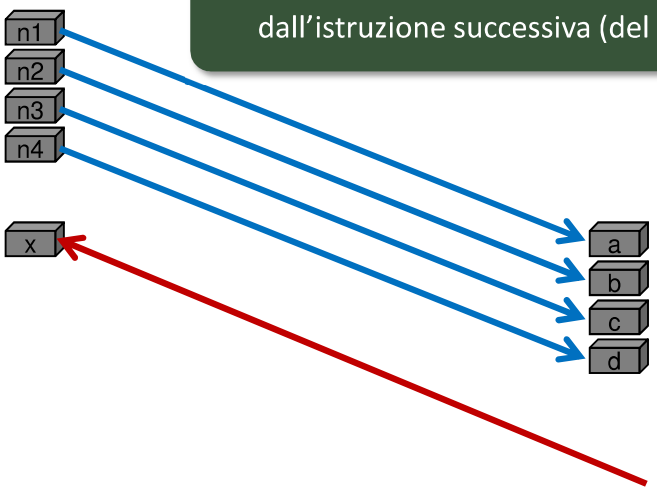
ESEMPIO

Quando il sotto-programma ha eseguito la sua ultima istruzione, questi fornisce al programma chiamante il suo **valore di restituzione** e l'esecuzione riprende dall'istruzione successiva (del chiamante).

1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare
6. passo elementare
7. passo elementare
8. passo elementare
9. passo elementare
10. passo elementare
11. passo elementare

139. Invoca il sotto-programma qui a lato con i seguenti parametri: n1, n2, n3, n4 e metti il valore restituito nella variabile x.

1859. passo elementare
1860. passo elementare
1861. passo elementare
1862. passo elementare
1863. passo elementare
1864. passo elementare
1865. passo elementare



1. passo elementare
2. passo elementare
3. passo elementare
4. passo elementare
5. passo elementare

127. passo elementare
128. passo elementare
129. passo elementare
130. passo elementare
131. passo elementare

SVILUPPO TOP-DOWN

Nello scenario che abbiamo appena descritto, possiamo avere qualsiasi numero di parametri ma sempre **uno ed un solo** valore restituito!

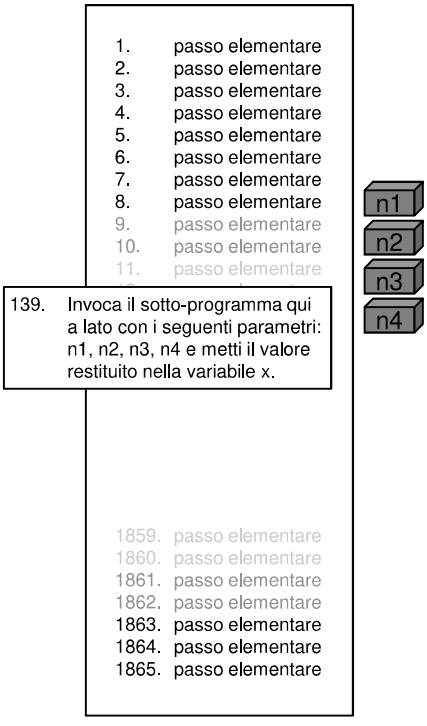
Se il sotto-programma necessita di restituire **altre informazioni** oltre al cosiddetto valore restituito, dobbiamo lavorare sui parametri.

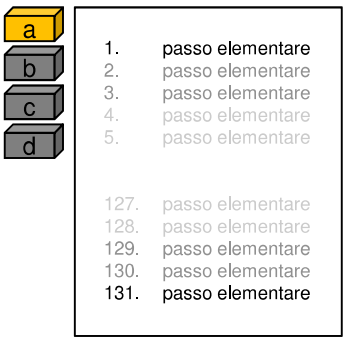
Nello scenario che abbiamo appena descritto il passaggio di parametri avviene **per copia**. Ovvero i valori dei parametri attuali vengono copiati nei parametri formali.

Ma se – nel progettare il sotto-programma – usiamo delle **variabili di tipo puntatore** come parametri (invece di usare delle variabili semplici), allora il passaggio di parametri avviene **per indirizzo**.

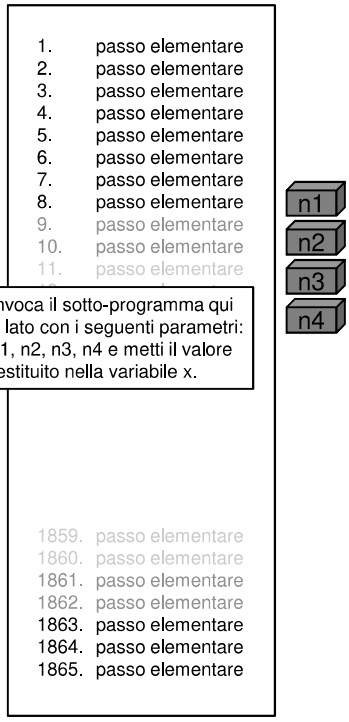
ESEMPIO

Supponiamo che il parametro formale a sia stato dichiarato di tipo **puntatore**.

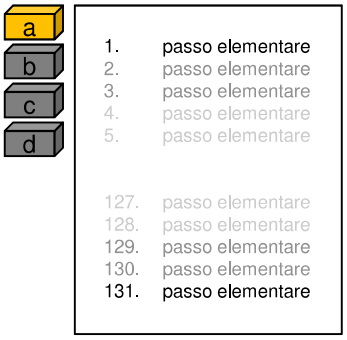




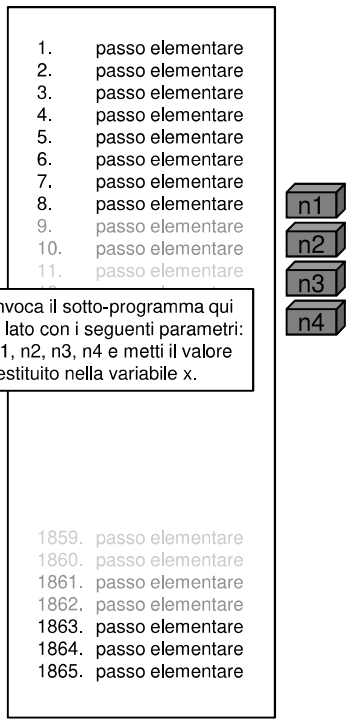
ESEMPIO



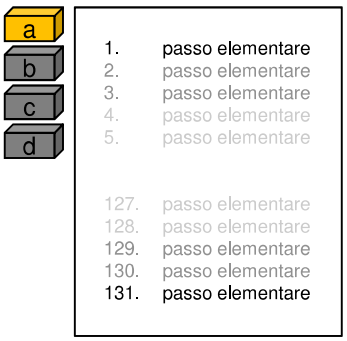
Ciò vuol dire che all'atto dell'invocazione nella **variabile a** non viene copiato il **valore di n1**, bensì il suo **indirizzo**!



ESEMPIO



Così ogni volta che modifichiamo il valore di **a** in realtà modifichiamo il valore della variabile puntata da **a** (che è **n1**).



JAVA

Le funzioni

LE FUNZIONI

La firma di una funzione in Java è questa:

```
<tipoRestituito> nomeFunzione (<tipoParam> <nomeParam>, <tipoParam> <nomeParam>, ...)
```

Facciamo un esempio.

La funzione restituirà un valore booleano

```
boolean isPrimo (int num) {
  if (num<2) return false;
  for (int i=2; i<=num/2; i++) {
    if (num%i == 0) {
      return false;
    }
  }
  return true;
}
```

LE FUNZIONI

La firma di una funzione in Java è questa:

```
<tipoRestituito> nomeFunzione (<tipoParam> <nomeParam>, <tipoParam> <nomeParam>, ...)
```

Facciamo un esempio.

La funzione restituirà un valore booleano

La funzione si chiama isPrimo

```
boolean isPrimo (int num) {
  if (num<2) return false;
  for (int i=2; i<=num/2; i++) {
    if (num%i == 0) {
      return false;
    }
  }
  return true;
}
```


LE FUNZIONI

La firma di una funzione in Java è questa:

```
<tipoRestituito> nomeFunzione (<tipoParam> <nomeParam>, <tipoParam> <nomeParam>, ...)
```

Facciamo un esempio.

La funzione restituirà un valore booleano

La funzione si chiama isPrimo

La funzione ha un solo parametro (formale)

```
boolean isPrimo (int num) {
  if (num<2) return false;
  for (int i=2; i<=num/2; i++) {
    if (num%i == 0) {
      return false;
    }
  }
  return true;
}
```

LE FUNZIONI

La firma di una funzione in Java è questa:

```
<tipoRestituito> nomeFunzione (<tipoParam> <nomeParam>, <tipoParam> <nomeParam>, ...)
```

Facciamo un esempio.

```
boolean isPrimo (int num) {
  if (num<2) return false;
  for (int i=2; i<=num/2; i++) {
    if (num%i == 0) {
      return false;
    }
  }
  return true;
}
```

- La funzione restituirà un valore booleano
- La funzione si chiama isPrimo
- La funzione ha un solo parametro (formale)
- La funzione chiude la sua esecuzione e restituisce un valore al chiamante con un return

LE FUNZIONI

La firma di una funzione in Java è questa:

```
<tipoRestituito> nomeFunzione (<tipoParam> <nomeParam>, <tipoParam> <nomeParam>, ...)
```

Facciamo un esempio.

```
boolean isPrimo (int num) {
  if (num<2) return false;
  for (int i=2; i<=num/2; i++) {
    if (num%i == 0) {
      return false;
    }
  }
  return true;
}
```

```
// altre istruzioni
if (isPrimo(x)) {
   // altre istruzioni
}
// altre istruzioni
```

- La funzione restituirà un valore booleano
- La funzione si chiama isPrimo
- La funzione ha un solo parametro (formale)
- La funzione chiude la sua esecuzione e restituisce un valore al chiamante con un return
- L'invocazione avviene semplicemente

LE FUNZIONI

La firma di una funzione in Java è questa:

```
<tipoRestituito> nomeFunzione (<tipoParam> <nomeParam>, <tipoParam> <nomeParam>, ...)
```

Facciamo un esempio.

```
boolean isPrimo (int num) {
  if (num<2) return false;
  for (int i=2; i<=num/2; i++) {
    if (num%i == 0) {
      return false;
    }
  }
  return true;
}
```

```
// altre istruzioni
if (isPrimo(x)) {
   // altre istruzioni
}
// altre istruzioni
```

- La funzione restituirà un valore booleano
- La funzione si chiama isPrimo
- La funzione ha un solo parametro (formale)
- La funzione chiude la sua esecuzione e restituisce un valore al chiamante con un return
- L'invocazione avviene semplicemente
- Il programma chiamante passa il parametro attuale. In questo caso una variabile il cui valore viene copiato nel parametro formale

CC BY

LE FUNZIONI

La firma di una funzione in Java è questa:

```
<tipoRestituito> nomeFunzione (<tipoParam> <nomeParam>, <tipoParam> <nomeParam>, ...)
```

Facciamo un esempio.

```
boolean isPrimo (int num) {
  if (num<2) return false;
  for (int i=2; i<=num/2; i++) {
    if (num%i == 0) {
      return false;
    }
  }
  return true;
}
```

```
// altre istruzioni
if (isPrimo(x)) {
   // altre istruzioni
}
// altre istruzioni
```

- La funzione restituirà un valore booleano
- La funzione si chiama isPrimo
- La funzione ha un solo parametro (formale)
- La funzione chiude la sua esecuzione e restituisce un valore al chiamante con un return
- L'invocazione avviene semplicemente
- Il programma chiamante passa il parametro attuale. In questo caso una variabile il cui valore viene copiato nel parametro formale
- Quando la funzione conclude il suo lavoro ci restituisce il suo valore (in questo caso true o false)

LE FUNZIONI

La firma di una funzione in Java è questa:

```
<tipoRestituito> nomeFunzione (<tipoParam> <nomeParam>, <tipoParam> <nomeParam>, ...)
```

Facciamo un esempio.

```
boolean isPrimo (int num) {
  if (num<2) return false;
  for (int i=2; i<=num/2; i++) {
    if (num%i == 0) {
      return false;
    }
  }
  return true;
}
```

```
// altre istruzioni
if (isPrimo(x)) {
   // altre istruzioni
}
// altre istruzioni
```

```
// altre istruzioni
boolean flag = isPrimo(199);
// altre istruzioni
```

- La funzione restituirà un valore booleano
- La funzione si chiama isPrimo
- La funzione ha un solo parametro (formale)
- La funzione chiude la sua esecuzione e restituisce un valore al chiamante con un return
- L'invocazione avviene semplicemente
- Il programma chiamante passa il parametro attuale. In questo caso una variabile il cui valore viene copiato nel parametro formale
- Quando la funzione conclude il suo lavoro ci restituisce il suo valore (in questo caso true o false)
- Qui invochiamo la funzione con un parametro attuale costante ed archiviamo il valore restituito in una variabile

(LABORATORIO)

- Realizzare la funzione isPrimo ed usarla per presentare un numero primo a caso in [500..1000] (200)
- Riempire un vettore con dieci numeri interi casuali in [0..9] tutti diversi tra loro. Successivamente stampare il vettore.(201)
- Realizzare una funzione che produce un numero casuale in un range [x..y] avuto come parametro. Usare la funzione per produrre un numero casuale tra due numeri avuti in input dall'utente. Stampare il valore prodotto (202-)

(LABORATORIO)

- Realizzare una funzione per verificare se un anno è bisestile o meno e usarla per riempire un vettore di 10 anni bisestili a caso in [1950..2050]. Successivamente ordinare il vettore. Infine stampare il vettore. (203-)
- Riempi un vettore con dieci interi in [0,99] senza ripetizioni. Successivamente crea un vettore di 100 booleani e inserisci true nelle celle indicate dal primo vettore, false altrimenti. Infine stampa il primo vettore e scorri il secondo vettore in modo da stampare tutti gli indici delle celle che contengono true.(204)

(LABORATORIO)

- Produrre un vettore **cifre** di 10 interi casuali tra zero e 9. Sulla base di questo vettore realizzare un terzo vettore **inLettere** – parallelo a **cifre** – che contenga la rappresentazione letterale dello stesso. Stampa entrambi i vettori.

(LABORATORIO)

- Produrre una matrice 10*10 di numeri casuali in [0..500] e stamparla ben formattata. Successivamente calcolare il valore medio della matrice e stamparlo. Infine porre a zero tutti i valori della matrice che siano dispari e inferiori alla media. Stampare nuovamente la matrice. (300)