UNIVERSITÀ DEGLI STUDI DI SALERNO

Università di Salerno
Dipartimento di
Ingegneria Industriale

Fondamenti di Informatica

Problemi, Soluzioni ed Algoritmi

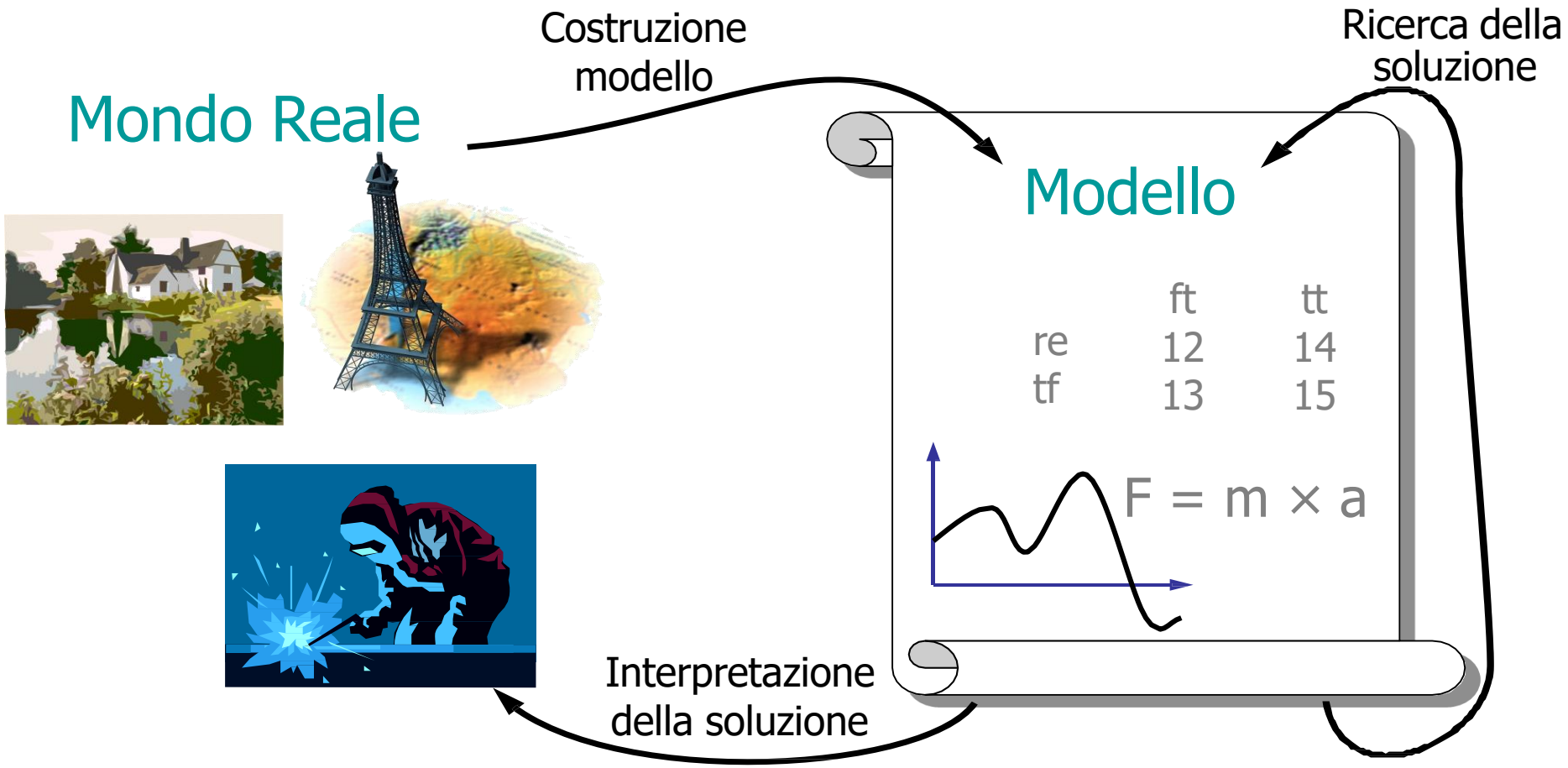
Prof. Raffaele Pizzolante

A.A. 2016/17

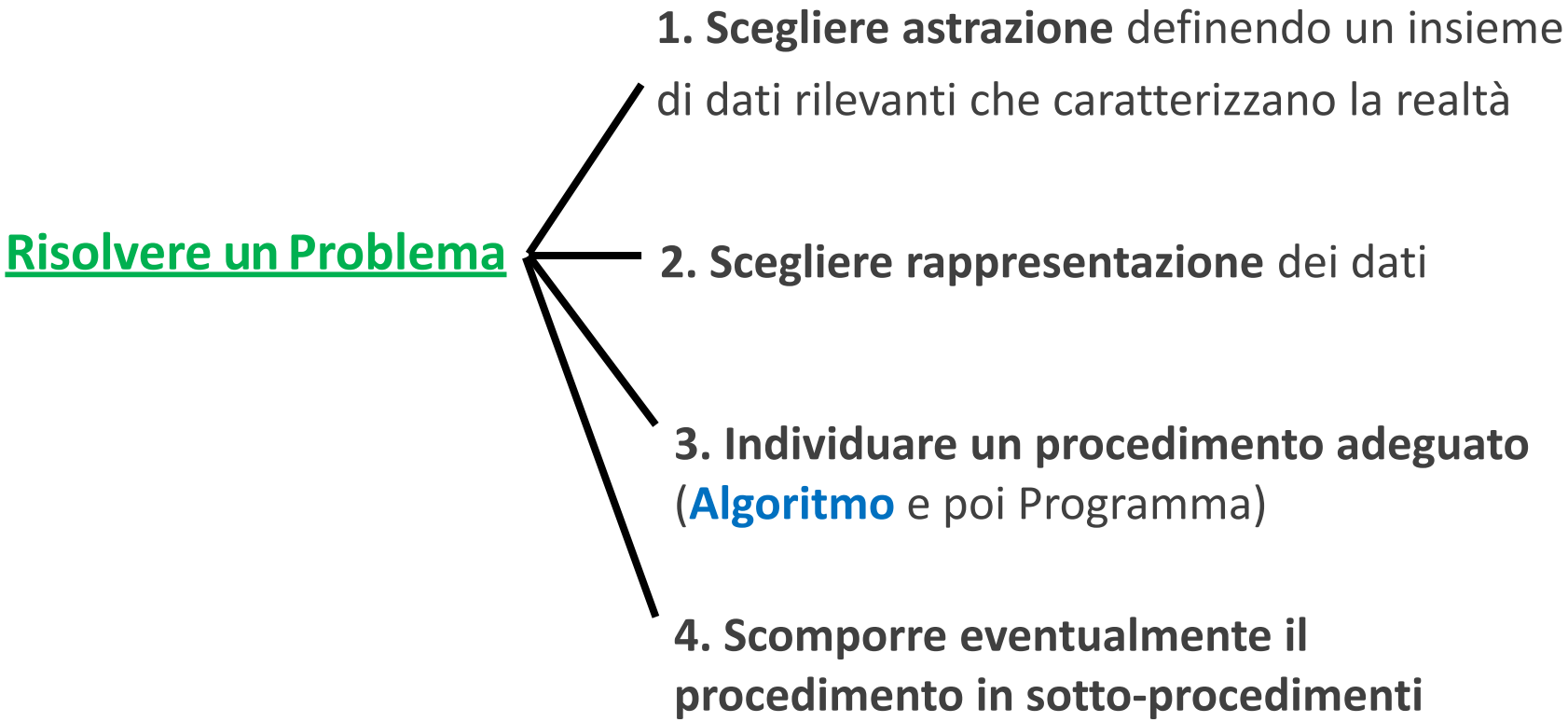
Come Instruire i Calcolatori a Risolvere Problemi

- Gli elaboratori sono strumenti per risolvere (o aiutare a risolvere) problemi basati su informazioni e dati
- Ma come ciò avviene?
  1. Abbiamo bisogno di codificare opportunamente informazioni e dati
  2. **Abbiamo bisogno di impartire le giuste istruzioni per risolvere correttamente i problemi**

Relazione tra Realtà e Modello: Come Modellare un Problema



Risolvere un Problema



Il Termine Algoritmo: Etimologia

- Deriva dal matematico Arabo Muḥammad ibn Mūsā **al-Khwārizmī** (c. 780-850), autore del testo Al-jabrw'al-muqabâla (da cui anche il termine Algebra)

Algoritmi: un po' di Storia

- Algoritmi di tipo numerico furono studiati da matematici babilonesi e indiani più di 3000 anni fa
- Algoritmi in uso fino a tempi recenti furono studiati dai matematici greci nel 500 a.C.
  - *Algoritmo di Euclide* per il *Massimo Comun Divisore*
  - Algoritmi geometrici (calcolo di tangenti, sezioni di angoli, etc)
  - Etc

Gli Algoritmi nel Trattamento dell'Informazione – 1/3

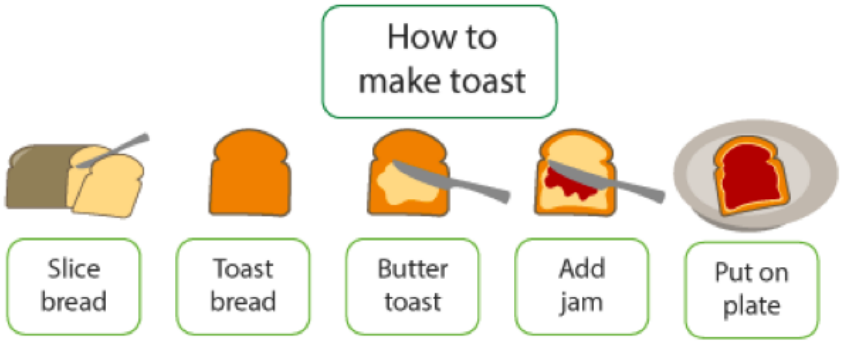
- Un **algoritmo consente di** realizzare un particolare trattamento dell'informazione o più in generale di **risolvere uno specifico problema**
  - Calcolare la somma di due numeri
  - Calcolare la lunghezza dell'ipotenusa di un triangolo rettangolo
  - Risolvere un'equazione di secondo grado
  - Ma anche…

Gli Algoritmi nel Trattamento dell'Informazione – 2/3

- **Trasmissione dati in Internet**
  - Come si gestisce in pratica il flusso di dati nella rete?
- **Ricerca nel WEB**
  - Come fa Google a trovare le informazioni nel WEB?
- **Bioinformatica**
  - Come il DNA determina le nostre caratteristiche?
- **Processi economici**
  - Come si gestiscono le aste on-line su Ebay?
  - Come si effettua la compravendita di azioni su Internet?
- **Organizzazione di risorse e servizi**
  - Come si schedulano i voli delle aerolinee?
  - Come si assegnano le frequenze nelle celle delle reti cellulari?
- **Etc**

Gli Algoritmi nel Trattamento dell'Informazione – 3/3

- L'esperienza quotidiana suggerisce infiniti esempi di algoritmi
- Molte azioni che svolgiamo abitualmente possono essere modellate mediante algoritmi
  - Preparazione del caffè
  - Prelievo bancomat
  - Preparazione di un ricetta
  - Indicazioni per un lavoro a maglia
  - Etc



Algoritmo: Definizioni

- **Definizione 1**
  - procedura di calcolo **ben definita**, che **prende** un insieme di valori come **input** e **produce** un insieme di valori come **output**
- **Definizione 2**
  - **sequenza di azioni elementari** che consente di **trasformare** i **dati iniziali nei risultati finali** attraverso **un numero finito di passi non ambigui**
- **Note**
  - Insieme di **dati iniziali ben definito**
  - Sequenza di passi può essere eseguita da un **esecutore** (ad es. calcolatore)

Esempio: Algoritmo per prendere l'Automobile

- **Algoritmo**
  1. Aprire l'auto
  2. Aprire la portiera
  3. Sedersi al posto di guida
  4. Allacciare la cintura
  5. Schiacciare la frizione
  6. Avviare il motore
  7. Inserire la prima marcia
  8. Togliere il freno a mano
  9. Rilasciare "delicatamente" la frizione per partire

Osservazione: i passi sono eseguiti in sequenza e l'**ordine delle istruzioni è essenziale per la correttezza**

Algoritmi: Esecutore e Linguaggio – 1/3

- Un algoritmo presuppone la presenza di qualcuno (o qualcosa) in grado di eseguirlo, chiamato **esecutore**
  - In informatica l'esecutore è il **calcolatore (computer)**
- L'algoritmo viene "letto" dall'esecutore
  - L'esecutore, partendo dai **dati in input**, esegue (in un **ordine ben preciso**) tutte **le istruzioni** dell'algoritmo stesso, ottenendo **al termine** della propria esecuzione i **dati in output**
- Per essere eseguito, l'**algoritmo deve essere formulato in un linguaggio comprensibile dall'esecutore**
  - Un **esecutore può anche eseguire un algoritmo formulato in un linguaggio che non conosce**, a patto che l'algoritmo stesso sia **preventivamente tradotto in un linguaggio** che invece gli è **noto**

Algoritmi: Esecutore e Linguaggio – 2/3

- L'algoritmo deve
  - Prevedere soltanto **istruzioni** che richiedono all'esecutore di effettuare **operazioni elementari**
    - Operazioni che egli sa compiere senza bisogno di ulteriori specificazioni
  - Essere **formulato in** un **linguaggio non ambiguo**, in cui cioè ogni istruzione caratterizzi univocamente una delle operazioni che l'esecutore è in grado di compiere
  - Specificare senza ambiguità l'**ordine di esecuzione** delle istruzioni, cui l'esecutore deve attenersi scrupolosamente

Algoritmi: Esecutore e Linguaggio – 3/3

- L'esecuzione di un algoritmo da parte di un "esecutore" (uomo o macchina) si traduce in una **successione di operazioni** che vengono effettuate nel tempo, evocando un **processo sequenziale**
  - Serie di eventi che occorrono uno dopo l'altro, ognuno con un inizio ed una fine ben identificabile
- Un **algoritmo può richiedere l'esecuzione di altri algoritmi** precedentemente specificati all'esecutore

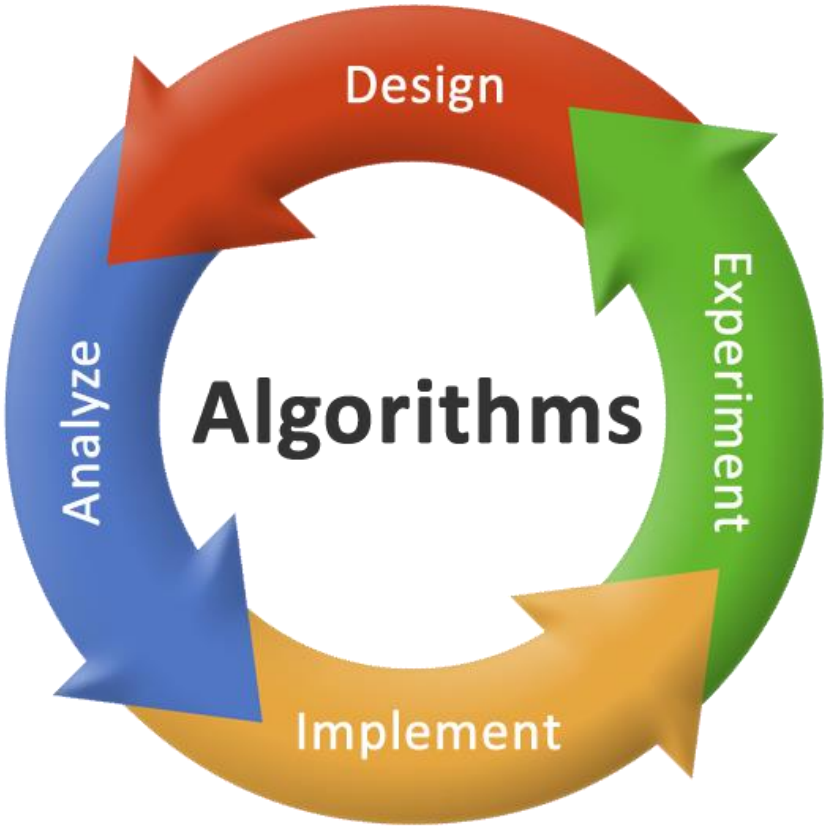
Algoritmi: Caratteristiche Principali – 1/2

- L'algoritmo deve essere formulato in un **numero finito di istruzioni** e **deve terminare,** fornendo i dati in output, **in un tempo finito**
- L'algoritmo può essere
  - **Deterministico:** eseguendo lo stesso algoritmo più volte sugli stessi dati di input, l'esecutore deve generare sempre gli stessi dati di output
  - **Probabilistico:** eseguendo lo stesso algoritmo più volte sugli stessi dati di input, l'esecutore deve generare dati di output sempre diversi
- **Noi ci occuperemo solo di algoritmi deterministici**
- **Se possiamo specificare un algoritmo allora possiamo automatizzare la soluzione**

Algoritmi: Caratteristiche Principali – 2/2

- Un algoritmo deve possedere le seguenti **proprietà**
  - **Correttezza:** produrre sempre una soluzione, a patto che i dati in input siano validi
  - **Determinatezza:** utilizzare le istruzioni di base fornite dall'esecutore
  - **Efficacia:** risolvere il problema tramite la combinazione di istruzioni di base
  - **Efficienza:** risolvere il problema usando la minor quantità possibile di risorse fisiche
    - Tempo di esecuzione, memoria, etc
- Un algoritmo può essere espresso in varie forme
  - Linguaggi umani, pseudo-codici, linguaggi grafici, linguaggi di programmazione, etc

Algoritmi: Fasi del Processo di Creazione



Come Rappresentare un Algoritmo

- Una soluzione algoritmica ad un problema, per essere eseguita da un calcolatore, deve essere rappresentata in maniera **formale**
  - **Formale:** descrizione in termini di sequenza di operazioni elementari
- Esistono numerosi strumenti per rappresentare una soluzione in modo formale, i più utilizzati sono
  - **Pseudocodici (testuale)**
  - **Diagrammi di flusso (grafico)**

Pseudocodici

- Nella rappresentazione degli algoritmi conviene **astrarsi dallo specifico linguaggio di programmazione**
- Per fare questo si usa un linguaggio detto **pseudocodice**
- Nello pseudocodice
  - Si impiegano metodi espressivi più chiari e concisi rispetto ai linguaggi di programmazione reali
  - Si possono usare frasi in linguaggio naturale per sintetizzare procedure talvolta complesse, ma non ambigue

Pseudocodici: Esempio 1

- Rappresentazione mediante pseudocodice di un possibile algoritmo per la preparazione del tè (N.B. **esecutore umano**)

INIZIO ALGORITMO prepararеUnaTazzaDiTè

1. Collegare il bollitore alla corrente elettrica
2. Mettere la bustina di tè in una tazza
3. Mettere l'acqua nel bollitore
4. Accendere il bollitore
5. Aspettare l'ebollizione dell'acqua
6. Aggiungere l'acqua alla tazza
7. Rimuovere la bustina di tè con il cucchiaio/forchetta
8. Aggiungere latte e/o zucchero
9. Servire

FINE ALGORITMO prepararеUnaTazzaDiTè

Pseudocodici: Esempio 1

- Rappresentazione mediante pseudocodice di un possibile algoritmo per la preparazione del tè (N.B. **esecutore umano**)

Notazione in camelCase

INIZIO ALGORITMO prepararеUnaTazzaDiTè

1. Collegare il bollitore alla corrente elettrica
2. Mettere la bustina di tè in una tazza
3. Mettere l'acqua nel bollitore
4. Accendere il bollitore
5. Aspettare l'ebollizione dell'acqua
6. Aggiungere l'acqua alla tazza
7. Rimuovere la bustina di tè con il cucchiaio/forchetta
8. Aggiungere latte e/o zucchero
9. Servire

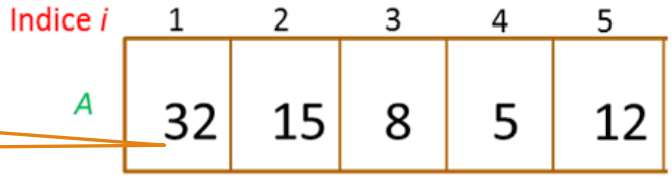
FINE ALGORITMO prepararеUnaTazzaDiTè

Pseudocodici: Esempio 2

- **Problema:** creare un algoritmo che trova il più grande elemento di un vettore di input *A* contenente *5* elementi
  - **Input:** *A* (un vettore di *5* numeri)
  - **Output:** *max* (l'elemento più grande in *A*)

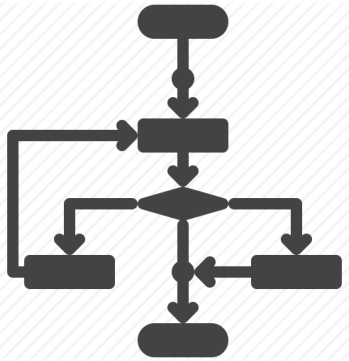
```
INIZIO ALGORITMO trovaMax
max = A[1]
Per i che va da 2 a 5
        Se A[i] > max
                max = A[i]
        Incrementa i
restituisci max
FINE ALGORITMO trovaMax
```

A[1] è uguale a 32



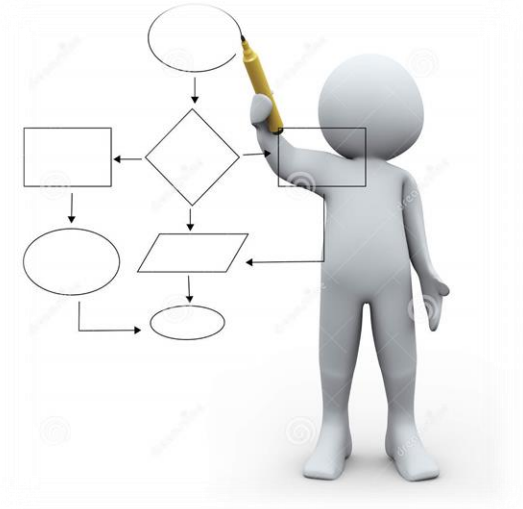
Diagrammi di Flusso – 1/3

- Per realizzare un algoritmo è necessario
  - Individuare i **dati** in **ingresso** ed in **uscita**
  - Individuare un **procedimento adeguato**
  - Scomporre il procedimento in una **sequenza** di **azioni elementari** e **non ambigue**
- Per meglio affrontare l'ultimo punto si fa spesso ricorso ai **diagrammi di flusso** (o **flow-chart**)
  - **Strumenti grafici** per **rappresentare** il **flusso logico** di **operazioni** che portano alla **risoluzione** di un **problema**
  - Costituiscono un **linguaggio** molto utile **per descrivere** gli **algoritmi**



Diagrammi di Flusso – 2/3

- Il **flusso** di **esecuzione** può essere **rappresentato graficamente** con un **diagramma di flusso**
  - Si sviluppa graficamente su **due dimensioni**
  - Si basa su **pochi simboli**
  - È un **linguaggio universale**
  - **Elimina** le **ambiguità**



Diagrammi di Flusso – 3/3

- Un diagramma di flusso è composto da
  - **Blocchi elementari**
    - Descrivono azioni e decisioni
  - **Archi orientati**
    - Collegano i vari blocchi e descrivono la sequenza di svolgimento delle azioni

Diagrammi di Flusso
Blocchi Elementari

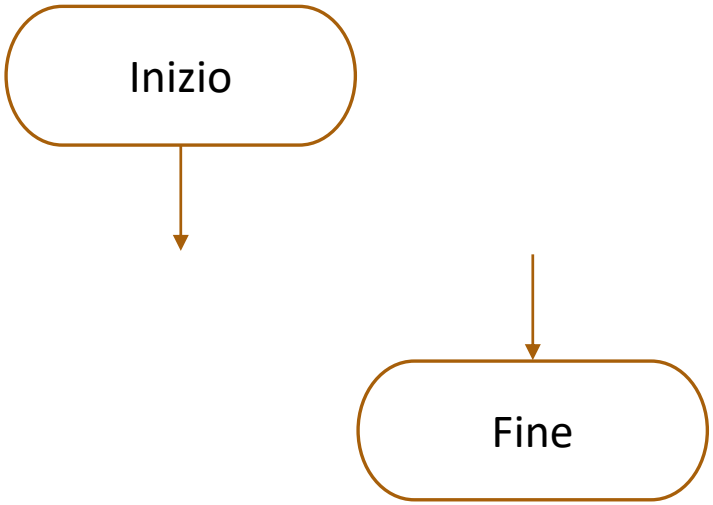
- **I blocchi elementari sono**
  - Blocco di **Inizio** e **Fine**
  - Blocco di **Connessione**
  - Blocco di **Azione Generica** e **Azione di I/O**
  - Blocco di **Decisione Binaria** (detta anche **Condizionale** oppure **A Due Vie**)

Diagrammi di Flusso
Blocchi Elementari

- **I blocchi elementari sono**
  - Blocco di **Inizio** e **Fine**
  - Blocco di **Connessione**
  - Blocco di **Azione Generica** e **Azione di I/O**
  - Blocco di **Decisione Binaria** (detta anche **Condizionale** oppure **A Due Vie**)

Diagrammi di Flusso: Blocchi di Inizio e Fine

- Un algoritmo (e di conseguenza una sua rappresentazione grafica) deve avere un **inizio** ed una **fine**



- Tra l'inizio e la fine ci deve sempre essere **almeno un'istruzione**

Diagrammi di Flusso
Blocchi Elementari

- **I blocchi elementari sono**
  - Blocco di Inizio e Fine
  - Blocco di **Connessione**
  - Blocco di **Azione Generica** e **Azione di I/O**
  - Blocco di **Decisione Binaria** (detta anche **Condizionale** oppure **A Due Vie**)

Diagrammi di Flusso: Blocco di Connessione

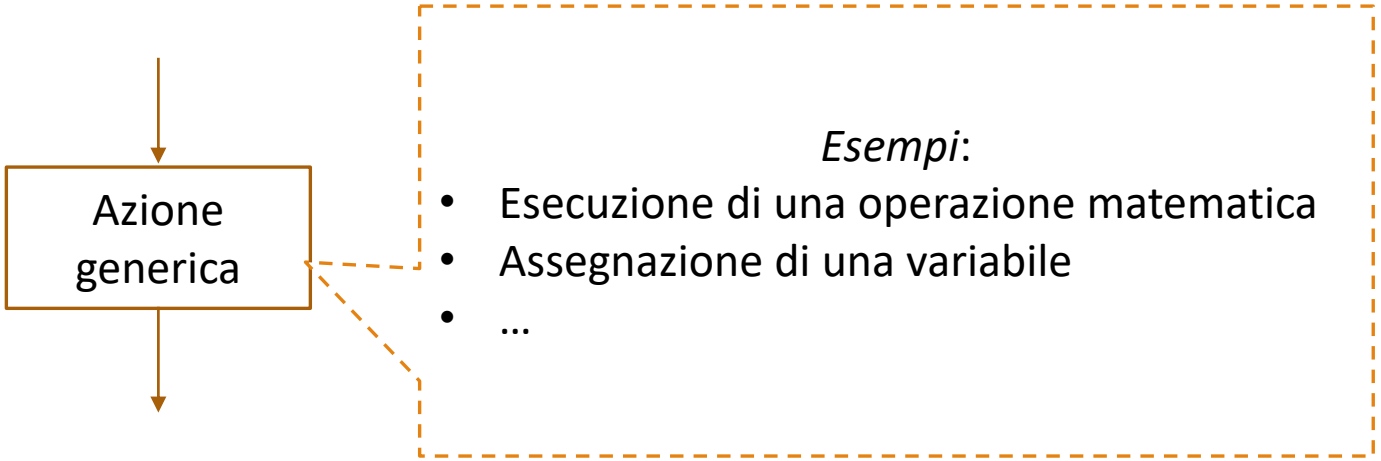
- La risoluzione di un problema consiste nell'esecuzione ordinata di una sequenza di operazioni
  - L'**ordine** nell'**esecuzione** delle istruzioni è fondamentale
  - Nei flow-chart è garantito dall'**orientamento** delle **frecce** che collegano i blocchi

Diagrammi di Flusso
Blocchi Elementari

- **I blocchi elementari sono**
  - Blocco di **Inizio** e **Fine**
  - Blocco di **Connessione**
  - Blocco di **Azione Generica** e **Azione di I/O**
  - Blocco di **Decisione Binaria** (detta anche **Condizionale** oppure **A Due Vie**)

Diagrammi di Flusso: Blocchi di Azione Generica ed I/O

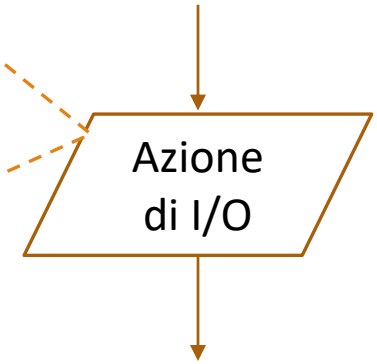
Diagrammi di Flusso: Blocchi di Azione Generica ed I/O



Diagrammi di Flusso: Blocchi di Azione Generica ed I/O

Esempi:

- Input/Output mediante periferiche di I/O:
  - Monitor
  - Tastiera
  - Ecc..
- Lettura di un dato da input (es. da tastiera)
- Mostrare un dato in output (es. su schermo)
- …

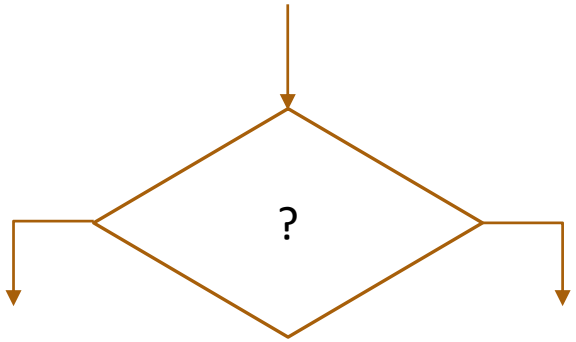


Diagrammi di Flusso
Blocchi Elementari

- **I blocchi elementari sono**
  - Blocco di **Inizio** e **Fine**
  - Blocco di **Connessione**
  - Blocco di **Azione Generica** e **Azione di I/O**
  - Blocco di **Decisione Binaria** (detta anche **Condizionale** oppure **A Due Vie**)

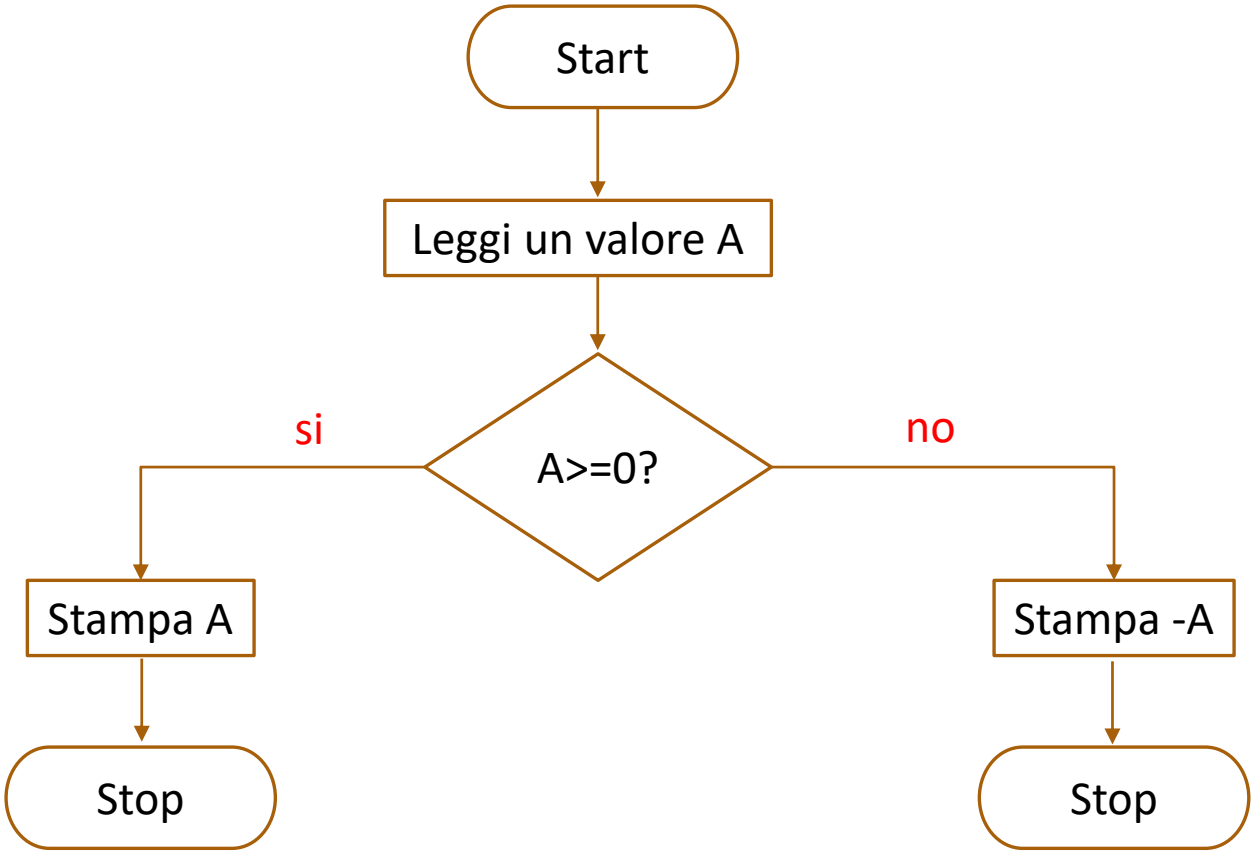
Diagrammi di Flusso: Blocco di Decisione Binaria (o Condizionale)

- Possono essere presenti **istruzioni condizionali**, la cui esecuzione dipende cioè da scelte effettuate in base ai dati
- Concettualmente, possiamo immaginare che il flusso di esecuzione si **ramifichi**
  - In base ad una **condizione** viene deciso se eseguire un'operazione **oppure** un'altra

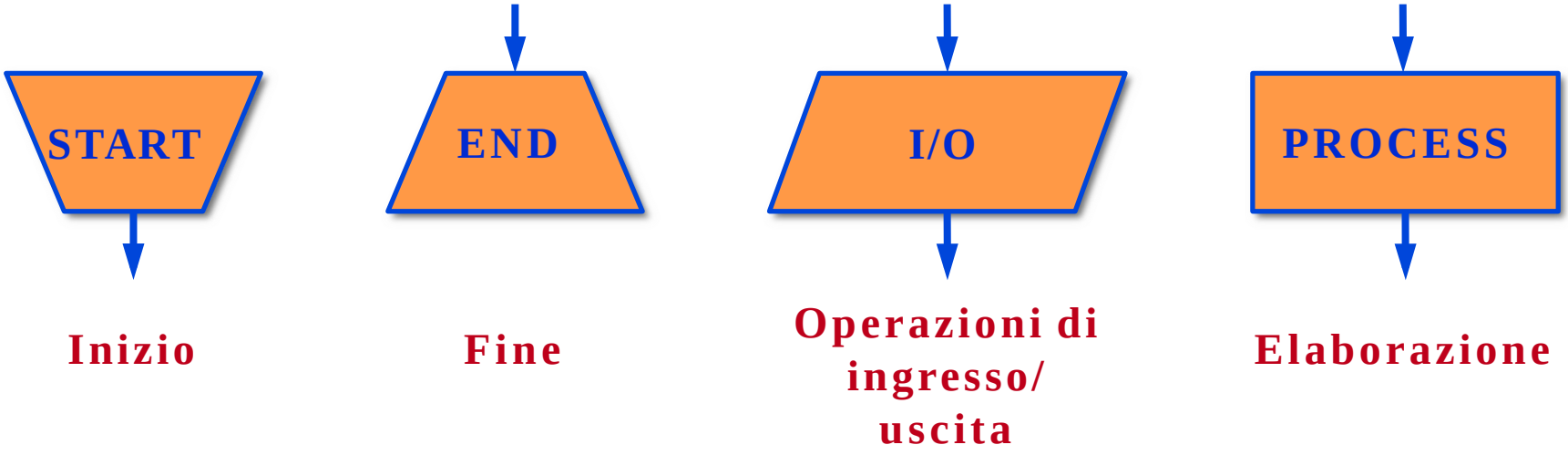


Diramazione
(condizionale)

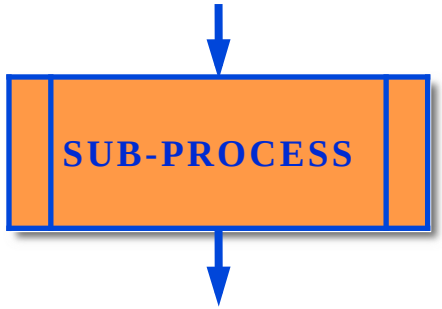
Esempio



Rappresentazione Alternativa dei Blocchi Elementari



Selezione a due vie



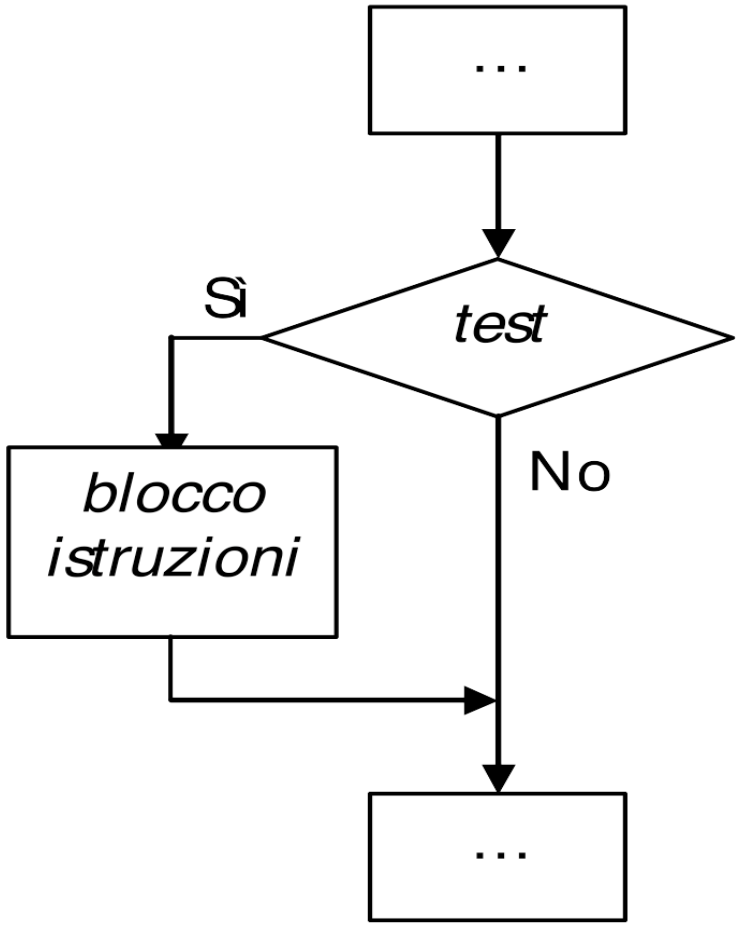
Sottoprogramma

Pseudocodice vs. Diagrammi di Flusso

- **Pseudocodice**
  - **Vantaggi**
    - Immediato
  - **Svantaggi**
    - Meno astratto
    - Interpretazione più complicata
- **Diagrammi di flusso**
  - **Vantaggi**
    - Più intuitivi perché grafici
    - Più astratti
  - **Svantaggi**
    - Richiedono apprendimento della funzione dei vari tipi di blocco

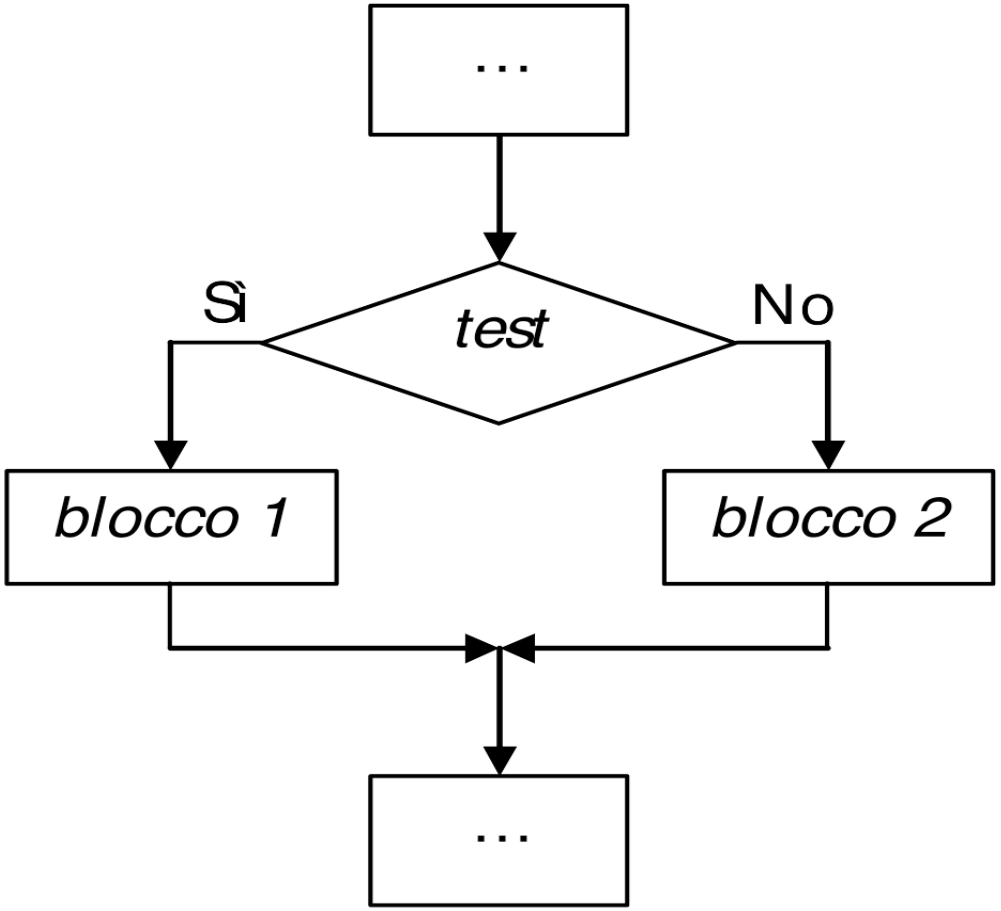
Strutture di Controllo: Controllare l'Ordine delle Istruzioni – 1/4

SELEZIONE SEMPLICE



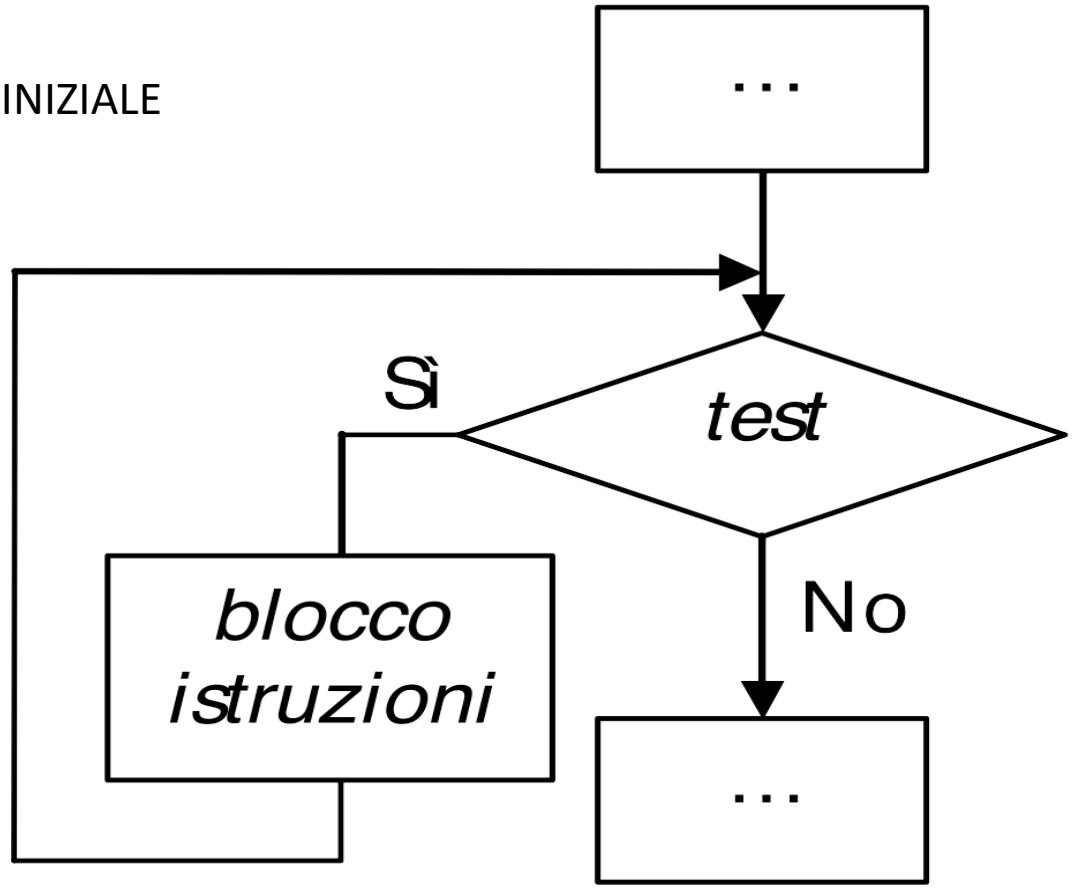
Strutture di Controllo: Controllare l'Ordine delle Istruzioni – 2/4

SELEZIONE A DUE VIE



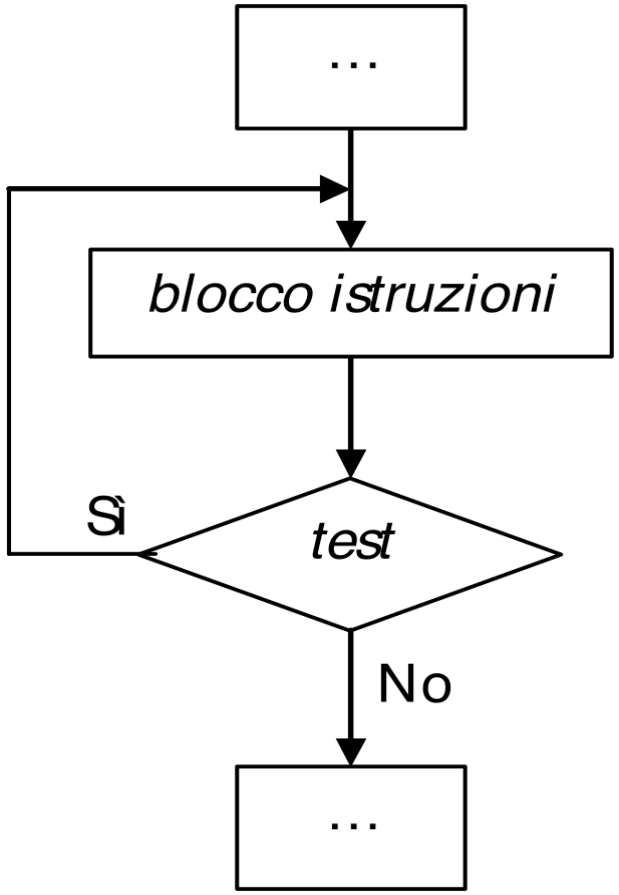
Strutture di Controllo: Controllare l'Ordine delle Istruzioni – 3/4

CICLO A CONDIZIONE INIZIALE



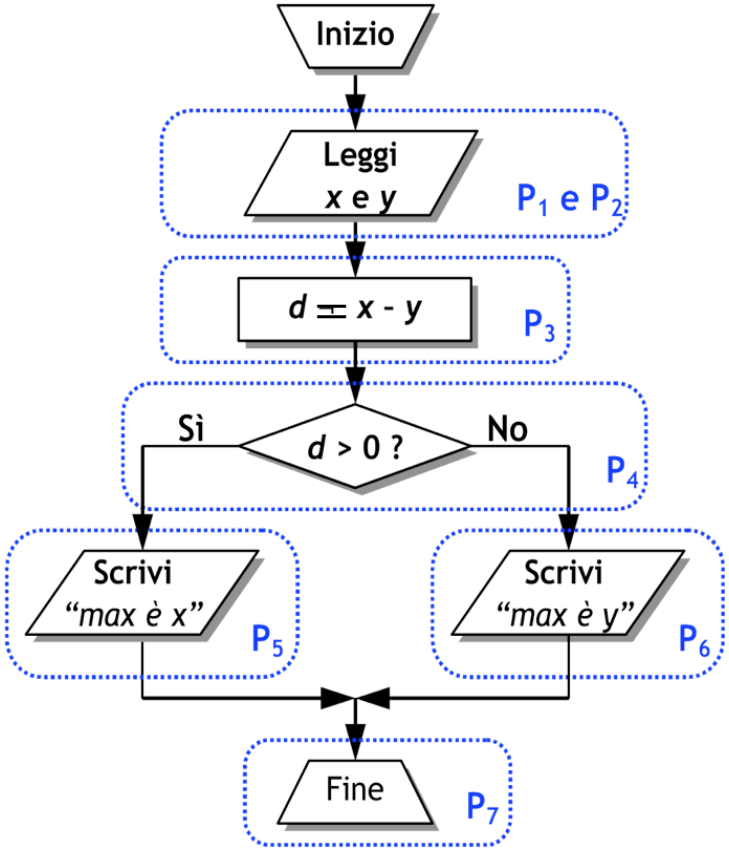
Strutture di Controllo: Controllare l'Ordine delle Istruzioni – 4/4

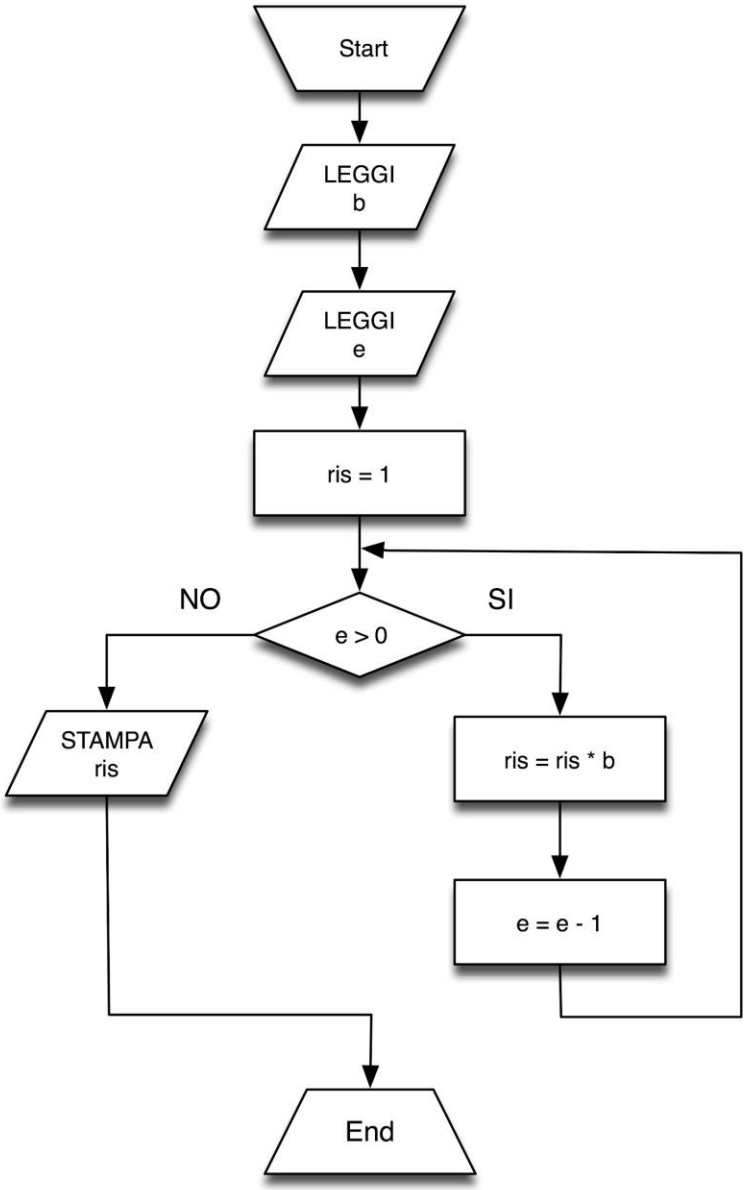
CICLO A CONDIZIONE FINALE



Esempio: Algoritmo per calcolare il massimo tra 2 numeri *x* e *y*

1. Leggi il valore di *x* dall'esterno
2. Leggi il valore di *y* dall'esterno
3. Calcola la differenza *d* fra *x* e *y* (*d=x-y*)
4. Se *d* è maggiore di *0* vai al passo *5.*, altrimenti vai al passo *6.*
5. Stampa "il massimo è …" seguito dal valore di *x* e vai al passo *7.*
6. Stampa "il massimo è …" seguito dal valore di *y*
7. Termina l'esecuzione

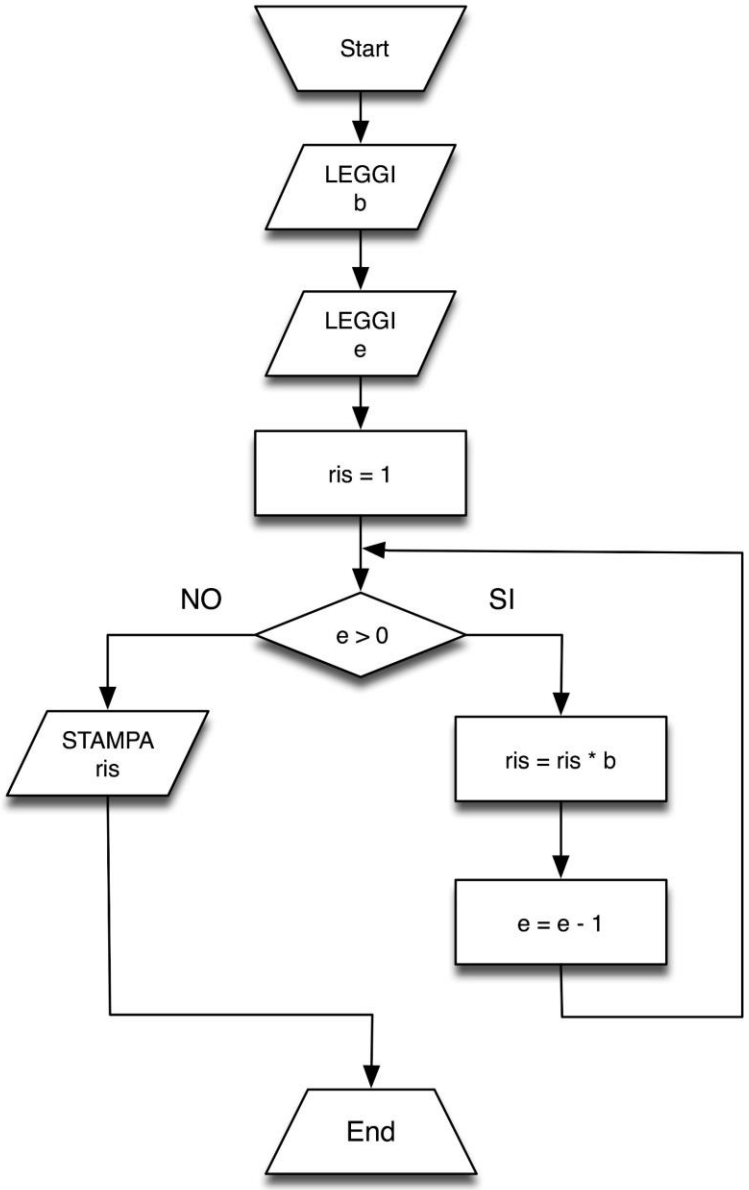




Algoritmo Elevazione a Potenza:
Base *b* elevata all'esponente *e*

Simulazione di esecuzione

b	e	ris	e > 0



Algoritmo Elevazione a Potenza:
Base *b* elevata all'esponente *e*

Simulazione di esecuzione

b	e	ris	e > 0
2	4	1	
			4 > 0 (SI)
2	3	2	
			3 > 0 (SI)
2	2	4	
			2 > 0 (SI)
2	1	8	
			1 > 0 (SI)
2	0	**16**	
			0 > 0 (NO)

Algoritmi e Programmi

- **Algoritmo:** sequenza di azioni per svolgere il calcolo
- **Programma:** algoritmo espresso in notazione formale (linguaggio di programmazione)
- **Creazione Programma**
  - **Fase 1** = Algoritmo
  - **Fase 2** = Implementazione dell'algoritmo in un dato linguaggio (di programmazione)

Processo per la Creazione di un Algoritmo – 1/5

- Talvolta il processo per la risoluzione di un problema è fisso
  - Sempre lo stesso ad ogni diversa esecuzione
- **Esempio:** algoritmo per calcolare l'importo di una fattura
  1. Cerca l'aliquota IVA sulla tabella
  2. Moltiplica l'importo netto per l'aliquota trovata
  3. Somma il risultato all'importo netto
- Questo algoritmo è composto da **tre istruzioni**
  - **Che devono essere eseguite in sequenza**

Processo per la Creazione di un Algoritmo – 2/5

- Altre volte lo stesso algoritmo può portare a più processi sequenziali differenti
  - A seconda delle condizioni iniziali
- **Esempio:** il precedente algoritmo, quando è richiesto di dover considerare la possibilità che la merce in esame non sia soggetta ad IVA, diventa

  - **SE la merce da fatturare è soggetta ad IVA ALLORA**
    1. Cerca la corretta aliquota IVA sulla tabella
    2. Moltiplica l'importo per l'aliquota trovata
    3. Somma il risultato all'importo netto
  - **ALTRIMENTI**
    4. Tieni conto solo dell'importo di partenza

Processo per la Creazione di un Algoritmo – 3/5

- **Osservazione:** nell'esempio visto in precedenza, il processo sequenziale non è fisso ma dipende dai dati da elaborare, in particolare dal tipo di merce da fatturare
  - L'algoritmo descrive un insieme costituito da due sequenze di esecuzione diverse

Processo per la Creazione di un Algoritmo – 4/5

- **Esempio:** algoritmo per effettuare una telefonata

1. Solleva il ricevitore
2. Componi il numero
3. **SE qualcuno risponde ALLORA**
   - Conduci la conversazione
4. **ALTRIMENTI**
   - Deponi il ricevitore e
   - Ripeti l'intero procedimento, a partire dal punto 1.

Processo per la Creazione di un Algoritmo – 5/5

- Un **algoritmo** può essere visto come un **testo in grado di descrivere** un insieme di **sequenze di esecuzione**
- L'algoritmo per effettuare una telefonata potrebbe avere un **processo ciclico che non termina mai**
  - L'interlocutore non risponde mai al telefono

Esempio: Prodotto di due Numeri mediante Addizioni Successive – 1/5

- **7** * **3** = **21**
  - **7** è detto **moltiplicando**
  - **3** è detto **moltiplicatore**
- Utilizzando il *metodo delle Addizioni Successive*, la moltiplicazione può essere espressa come
  - **7** + **7** + **7** = **21**
- L'algoritmo potrebbe essere costituito semplicemente dal seguente testo
  - **"Si sommi il moltiplicando a se stesso un numero di volte uguale al valore del moltiplicatore"**

Esempio: Prodotto di due Numeri mediante Addizioni Successive – 2/5

- Specifichiamo con **maggiore dettaglio** le operazioni richieste **evitando i costrutti ambigui**

- Ad esempio potremmo scrivere
  1. Si sommi il **moltiplicando** a se stesso, e si decrementi di uno il valore del **moltiplicatore**
  2. Si sommi ancora il **moltiplicando** al **valore** ottenuto dalla **precedente somma**, e si decrementi di nuovo il **valore** ottenuto dalla **precedente sottrazione**
  3. Si ripeta il procedimento fino a che, per decrementi successivi, il **moltiplicatore** non raggiunga il valore zero

Esempio: Prodotto di due Numeri mediante Addizioni Successive – 3/5

1. **7 + 0 = 7 (Val. prec. somma)** — **3 – 1 = 2 (Val. prec. sottrazione)**
2. **7 + 7 = 14 (Val. prec. somma)** — **2 – 1 = 1 (Val. prec. sottrazione)**
3. **7 + 14 = 21** — **1 – 1 = 0 (Val. prec. sottrazione)**

Sequenza di operazioni effettuate sul moltiplicando

Sequenza di operazioni effettuate sul moltiplicatore

Esempio: Prodotto di due Numeri mediante Addizioni Successive – 4/5

- **Osservazione:** per evitare ambiguità e specificare di volta in volta di quale valore si sta parlando, si è stati costretti ad usare le espressioni
  - **“valore ottenuto dalla precedente somma”**
  - **“valore ottenuto dalla precedente sottrazione”**
- Per distinguere tali valori si ricorre a simboli o identificatori (detti **variabili**) per riferirsi senza ambiguità, di volta in volta, al valore desiderato

Esempio: Prodotto di due Numeri mediante Addizioni Successive – 5/5

- **Algoritmo**
  1. Sia **M** il valore del moltiplicando, **N** il valore del moltiplicatore ed **M1** il risultato (inizialmente uguale a zero)
  2. Si ripetano le seguenti operazioni fino a che il valore di **N** non diventi uguale a 0
     - Si sommi il valore del moltiplicando **M** al valore di **M1** e si chiami il risultato ancora **M1**
     - Si sottragga 1 dal valore di **N**, e si chiami il risultato ancora **N**
  3. Alla fine il valore di **M1** è il risultato cercato
- I simboli **M**, **N** e **M1** sono detti identificatori di variabili
- **M1** è detto accumulatore di risultato

Esempio: Consultazione Libro in Biblioteca – 1/6

- I libri sono disposti sugli scaffali
- La posizione di ogni libro è fissa ed è individuata da due coordinate
  - Scaffale (numero dello scaffale)
  - Posizione nello scaffale
- La biblioteca è dotata di uno schedario (ordinato per autore/i e titolo), dove ogni scheda contiene nell'ordine
  - Cognome e nome dell'autore
  - Titolo del libro
  - Edizione e data di pubblicazione
  - Numero dello scaffale in cui si trova
  - Posizione attribuita al libro nello scaffale

Esempio: Consultazione Libro in Biblioteca – 2/6

Esempio di scheda

Autore: Sciuto Donatella, Buonanno Giacomo, Mari Luca

Titolo: Introduzione ai Sistemi Informatici

Edizione e data di pubblicazione: III Edizione, 2005

Scaffale: 42

Posizione: 81

Esempio: Consultazione Libro in Biblioteca – 3/6

- **Prima formulazione dell'algoritmo**
  1. Decidi il libro da richiedere
  2. Preleva il libro richiesto
- **Osservazione:** se un passo non è direttamente comprensibile ed eseguibile dall'esecutore (operazione semplice) allora occorre dettagliarlo a sua volta mediante un ulteriore algoritmo

Esempio: Consultazione Libro in Biblioteca – 4/6

- **Algoritmo per il Prelievo**
  1. Decidi il libro da richiedere
  2. **Cerca la scheda del libro richiesto**
  3. Segnati numero scaffale e posizione
  4. Cerca lo scaffale indicato
  5. Accedi alla posizione indicata e preleva il libro
  6. Scrivi i tuoi dati sulla “scheda prestito”

Esempio: Consultazione Libro in Biblioteca – 5/6

- **Il sotto-algoritmo di ricerca**
  1. Prendi la prima scheda
  2. Esamina se titolo e autore/i sono quelli cercati. In caso positivo la ricerca termina con successo, altrimenti passa alla scheda successiva e ripeti
  3. Se esaurisci le schede allora il libro cercato non esiste

Esempio: Consultazione Libro in Biblioteca – 6/6

- **Algoritmo per il Prelievo**
  1. Decidi il libro da richiedere
  2. **Cerca la scheda del libro richiesto come segue**
     1. Prendi la prima scheda
     2. Esamina se titolo e autore/i sono quelli cercati. In caso positivo la ricerca termina con successo, altrimenti passa alla scheda successiva e ripeti
     3. Se esaurisci le schede allora il libro cercato non esiste
  3. Segnati numero scaffale e posizione
  4. Cerca lo scaffale indicato
  5. Accedi alla posizione indicata e preleva il libro
  6. Scrivi i tuoi dati sulla “scheda prestito”

Esercizi

- Perché nell'esempio per il calcolo del massimo in una lista di 5 elementi non è stato posto *max=0*?
- Come generalizzare la procedura per liste di lunghezza *n*?
- Come trovare l'elemento minimo in *A*?
- Come trovare la posizione del più grande elemento?

Riferimenti

- **Libro di testo**
  - Capitolo 3
    - Paragrafi 1, 2, 3, 5